Draadloze Langeafstands-communicatie Voor Weerstations

Scriptie

Auteur:
Tommas Bakker
1611239

Organisatie:
M2M4ALL/Sodaq

Eerste examinator:
Henk van Nimwegen
Hogeschool Utrecht

Draadloze Langeafstands-communicatie Voor Weerstations

Scriptie

Afbeelding voorblad:
Kilimanjaro berg

Auteur:
Tommas Bakker

Studentennummer:
1611239

Versie:
1.4

Datum:
02-01-2016

Mentor:
Jan Willen Smeenk, M2M4ALL
Bedrijfsbegeleider

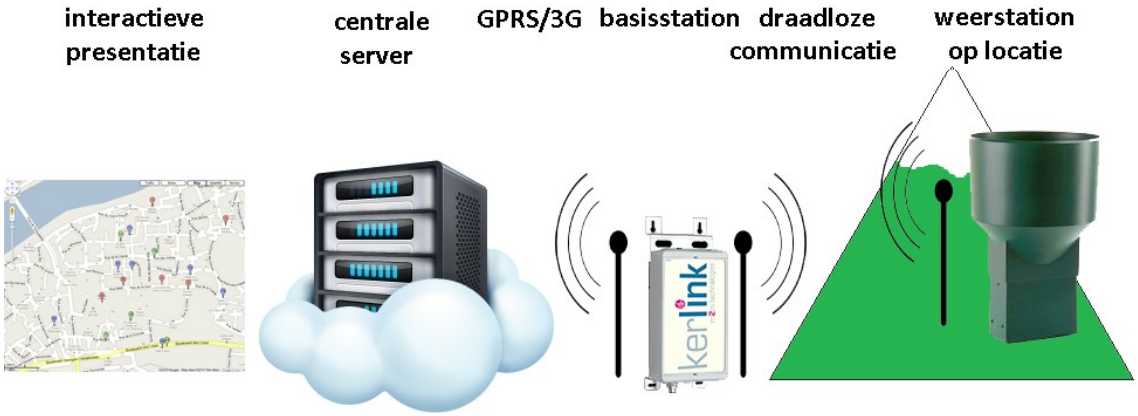
Examinatoren:
Henk van Nimwegen, Hogeschool Utrecht
Eerste examinator

Jan Zuurbier, Hogeschool Utrecht
Tweede examinator

i Management samenvatting

In dit document wordt een onderzoek naar de haalbaarheid van een draadloos netwerk voor weerstations op de Kilimanjaro berg beschreven. Deze opdracht is afgegeven door M2M4ALL en is bedoeld als *proof-of-concept* voor de onderzoeksgroep die op dit moment onderzoek doet naar klimaatverschijnselen op de Kilimanjaro berg.

Voor het onderzoek naar klimaatverschijnselen zijn al meerdere weerstations op de Kilimanjaro berg geïnstalleerd. Deze worden met de hand uitgelezen wat betekent dat meetgegevens pas over enkele dagen beschikbaar zijn. Het doel van het project is om, door middel van een *proof of concept*, aan te tonen dat het mogelijk is om meetgegevens energiezuinig, betrouwbaar en *semi-realtime* uit te lezen. Dit kan door de deze via een basisstation naar een centrale server te zenden en vanaf daar deze in een interactieve gebruikersinterface te tonen (zie afbeelding 1).



Afbeelding 1: Overzicht van alle componenten van het project

Hiervoor is een hoofdonderzoeksvraag en zijn enkele deelonderzoeksvragen opgesteld. Deze hoofdonderzoeksvraag luidt:

Hoe kunnen de gegevens van weerstations op de Kilimanjaro berg betrouwbaar en energie efficiënt uitgelezen worden door middel van draadloze communicatie en gepresenteerd worden in een interactieve gebruikersinterface zodat de gegevens geautomatiseerd en semi-realtime inzichtelijk zijn?

Om deze hoofdonderzoeksvraag te kunnen beantwoorden is allereerst gekeken hoe de communicatie tussen de weerstations en de basisstations gerealiseerd kan worden. Bij de keuze werden netwerktechnologieën, waaronder LoRaWAN en Sigfox, met elkaar vergeleken. Hieruit werd duidelijk dat dit kan met LoRaWAN en een eenvoudig eigen applicatieprotocol. Wel is het bereik van LoRaWAN een probleem, zo blijkt uit praktijktests. Hierdoor zijn de weerstations slechts op bepaalde plekken inzetbaar. Ook Sigfox was een goede kandidaat als netwerktechnologie maar is een te gesloten standaard en dus niet goed toepasbaar. Weerstations kunnen efficiënt met de beschikbare energie om gaan door waar mogelijk meerdere meetgegevens in één pakket te zenden. De betrouwbaarheid kan behaald worden door de meetgegevens meerdere keren te zenden.

Daarna is gekeken hoe de verbinding tussen de basisstations en de centrale server gerealiseerd kan worden. Uit vooronderzoek bleek dat dit via GPRS/3G gedaan kan worden omdat er voldoende dekking wordt geboden. Om dataverbruik te minimaliseren en tevens energie te besparen kan gebruik worden gemaakt van minificatie en datacompressie. Ook is onderzocht of hoe er om gegaan moet worden met langdurige netwerkstoring waardoor meetgegevens gebufferd moeten worden. Hieruit bleek dat SQLite het meest geschikt is voor deze functionaliteit.

Nadat de gegevens bij de centrale server zijn aangekomen moeten deze worden verwerkt en opgeslagen. Omdat het aannemelijk is dat de *proof of concept* later geïntegreerd zal worden in een ander systeem is niet verder onderzocht wat de meest geschikte implementatie is van de centrale server. Er is gekozen voor een RESTful webserver. Dit omdat dit een veel gebruikte implementatie is en deze waarschijnlijk makkelijker geïntegreerd kan worden. Om te valideren dat de database correct is ontworpen en er zich dus geen problemen voor gaan doen op het gebied van inconsistentie, is er gecontroleerd aan de hand van *functional dependencies* of deze voldoet aan de derde normaalvorm.

Naast de verbinding tussen de verscheidene componenten is er ook gekeken naar de energievoorziening van deze componenten. Omdat de weerstations en de basisstations op de berg staan en daar geen energievoorzieningen aanwezig zijn, moeten deze worden voorzien van een autonome energievoorziening. Zonnepanelen zijn het meest geschikt zijn voor deze situatie. Om de eisen aan de energievoorziening vast te stellen, is eerst het verbruik vastgesteld: Een weerstation heeft een zeer gering verbruik. Ook het versturen van een LoRaWAN bericht kost weinig energie. Het verbruik van een basisstation is aanzienlijk hoger dan dat van een weerstation.

Hierna is gekeken naar externe factoren die de opbrengst van een zonnepaneel kunnen beïnvloeden. Onder andere de effecten van sneeuw, temperatuur, bewolking, vuil en de daglengte zijn onderzocht en er is een simulatie ontwikkeld waar deze effecten worden meegewogen om zo de opbrengst van een zonnepaneel te kunnen inschatten. Hieruit blijkt dat om een basisstation te kunnen voeden, er een accu van 200 Wh in combinatie met een zonnepaneel van 200 Wp gebruikt dient te worden. Voor het weerstations is slechts een paneel van 6 Wh en een batterij van 4 Wp benodigd.

Vervolgens is er gekeken naar de betrouwbaarheid en veiligheid van meetgegevens. Uit onderzoek bleek de grootste oorzaak van storingen zogeheten single *event upsets* te zijn. Dit is een storing waarbij een bit “omvalt”. De kans op zo'n storing is onderzocht aan de hand van *single event upsets* in SRAM en daaruit bleek dat de kans klein is: Ongeveer elk jaar zal er een *single event upset* ontstaan bij één van de weerstations. Het voorkomen van een *single event upset* vergt mechanische aanpassingen en is niet verder onderzocht. Wel is onderzocht hoe *single event upsets* gedetecteerd kunnen worden. Door gebruik te maken van *Cyclic Redundancy Checks* en door berekeningen meerdere keren uit te voeren en de uitkomst onderling te controleren kunnen eventuele fouten gedetecteerd worden. Door de micro-controller te resetten wanneer een fout gedetecteerd wordt, wordt voorkomen dat de data verder verwerkt wordt.

Ook is er gekeken naar de beveiliging van de meetgegevens tegen kwaadwillenden. Op component niveau kan de beveiliging worden geregeld door gebruik te maken van veel gebruikte en goed onderhouden software waardoor de kans op beveiligingsgaten minimaal is. Door beheersinterfaces enkel via VPN aan te bieden wordt voorkomen dat deze worden aangevallen.

Als laatste is er gekeken naar de beveiliging van de verbindingen zelf. Door gebruik te maken van bijvoorbeeld VPN of LoRaWAN zelf, wordt voor deze verbindingen al genoeg veiligheid geboden. Met HTTPS wordt afluisteren voorkomen maar kan een *man-in-the-middle aanval* niet worden voorkomen. Om de gevolgen te beperken, is gebruik gemaakt van HMAC en *session tokens*. Zo kan worden voorkomen dat berichten worden aangepast en worden *replay aanvallen* voorkomen.

De conclusie van het onderzoek luidt dat het mogelijk is om een draadloos netwerk voor weerstations op de Kilimanjaro berg te realiseren door middel van LoRaWAN. De betrouwbaarheid wordt gegarandeerd door *single event upsets* te detecteren en berichten meerdere keren te zenden. De beschikbare energie wordt efficiënt gebruikt door slim om te gaan met buffers en waar mogelijk meerdere meetgegevens tegelijk te zenden. De GUI biedt een interactieve gebruikersinterface waar de gegevens *semi-realtime* op gepresenteerd worden.

ii Voorwoord

Dit document beschrijft het onderzoek dat ik het afgelopen half jaar heb verricht om de haalbaarheid van een draadloos netwerk voor weerstations te bepalen. Dit project vormt mijn afstudeerstage en is mijn laatste beproeving om mijn Bachelor diploma te halen.

Tijdens de uitvoering heb ik hulp van allerlei mensen gehad. Graag wil ik hen allen bedanken. Jan van Loenen wil ik in het bijzonder bedanken voor zijn hulp om onder andere mijn hoogtevrees te overwinnen. Daar waar ik de eerste keer letterlijk door mijn knieën zakte kan ik nu ongestoord werken.

Aanpassing Plan van Aanpak

Tijdens de uitvoering van dit project is het Plan van Aanpak (zie appendix A) aangepast. Dit omdat tijdens de uitvoering van dit project bleek dat er te veel tijd overbleef. Er is daarom gekozen om een extra deelvraag te behandelen (deelvraag 5) om zo de beschikbare tijd beter te benutten.

Inhoudsopgave

i Management samenvatting....iii
ii Voorwoord....v
Aanpassing Plan van Aanpak....v

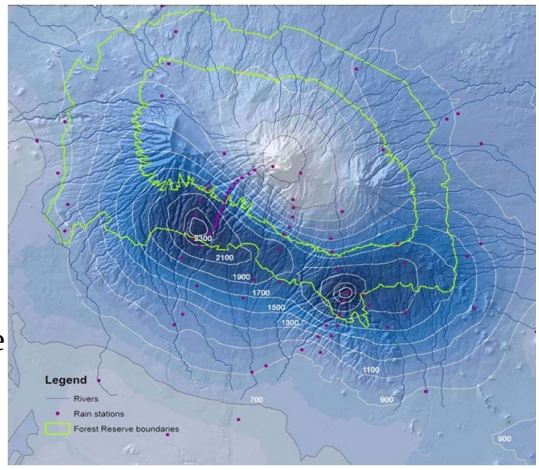
1 Inleiding....1
2 Project context....2
2.1 De organisatie....2
2.2 Bedrijfsbegeleider....2
2.3 Overige medewerkers....2
2.4 De stageopdracht....2
2.5 Kilimanjaro berg....3
3 Project definitie....4
3.1 De kwestie....4
3.2 De opdracht....4
3.3 De doelstelling....5
3.4 Onderzoeksvraag....6
3.5 Onderzoeksmethoden....7
3.6 (Deel)producten....8
4 Onderzoeksresultaten....9
4.1 Realisatie communicatie tussen de weerstations en de basisstations....9
4.2 Realisatie communicatie tussen de basisstations en de centrale server....19
4.3 Realisatie opslag van meetgegevens....25
4.4 Bepalen van eisen aan de autonome energievoorzieningen....27
4.5 De betrouwbaarheid van de meetgegevens waarborgen....35
5 Kwaliteitsbewaking....41
5.1 De GUI....41
5.2 De centrale server....41
5.3 De basisstations....41
5.4 De weerstations....41
6 Conclusie en aanbevelingen....42
6.1 Conclusie....42
6.2 Aanbevelingen....43
6.3 Integratie in data-analyse systemen....43
7 Evaluatie onderzoeksproces....44
7.1 Onvoldoende bereik....44
7.2 Planning....44
7.3 Literatuur van andere vakgebieden....44
7.4 Programmeerfouten in gebruikte libraries....45
8 Referenties....46

Appendix A – Plan van Aanpak....51
Appendix B – Bronnen voor specificaties netwerktechnologieën....70
Appendix C – Validatie van de database....71
Appendix D – Ontwerpverslag....74
Appendix E – Technische documentatie....95

1 Inleiding

Klimaatverandering heeft grote gevolgen voor ons bestaan. Zeker in Nederland zijn de gevolgen van een stijgende zeespiegel groot. Mede daarom is de politiek druk bezig om klimaatverandering tegen te gaan door onder andere windmolens te bouwen en door klimaatakkoorden te ondertekenen. Toch zijn de gevolgen van klimaatveranderingen onvoldoende bekend. Hier wordt momenteel op de Kilimanjaro berg onderzoek naar gedaan (“Project Summary”, 2010).

De Kilimanjaro berg is een berg in Tanzania, Afrika met een hoogte van 4877 meter (“Mount Kilimanjaro”, 2016). Op deze berg zijn 65 weerstations geplaatst (zie afbeelding 2) voor een onderzoek naar het klimaat op deze berg (Study Design. 2010).



Afbeelding 2: De Kilimanjaro berg, Tanzania, Afrika. De paarse stippen representeren weerstations (“Study area”, 2010)

Dit document beschrijft een onderzoek naar het draadloos uitleesbaar maken van weerstations op de Kilimanjaro berg. Het draadloos uitlezen van weerstations biedt verscheidene voordelen: Zo bespaart het kosten omdat er niet meer met de hand hoeft te worden uitgelezen. Omdat ze niet meer met de hand uitgelezen hoeven te worden kunnen de weerstations ook flexibeler worden ingezet. Zo kunnen ze op moeilijker bereikbare locaties worden geplaatst. Verder maakt draadloze communicatie het mogelijk om *semi-realtime* informatie van deze weerstations te verwerken. Dit maakt het mogelijk om sneller onderzoek te doen en doordat de gegevens *semi-realtime* beschikbaar zijn kunnen de weerstations bijvoorbeeld worden gebruikt voor regelsystemen van stuwdammen of alarmsystemen die waarschuwen voor modderlawines.

Om een beeld te geven van de context waarin dit onderzoek is uitgevoerd en omdat de context tevens invloed heeft op beslissingen die zijn gemaakt tijdens de uitvoering van het onderzoek, wordt deze in hoofdstuk 2 beschreven. In hoofdstuk 3 worden vervolgens het project definieert door de kwestie en de opdracht te beschrijven en de doelstelling te definieren. Deze zijn essentieel voor het opstellen van de hoofdonderzoeksvraag en de deelonderzoeksvragen. Ook deze zijn opgesteld in hoofdstuk 3. Nadat de onderzoeksvragen zijn opgesteld moeten deze worden beantwoord zodat later een conclusie kan worden getrokken. De antwoorden op de onderzoeksvragen en het verrichte onderzoek om tot deze antwoorden te komen zijn chronologisch beschreven in hoofdstuk 4. In hoofdstuk 5 wordt besproken hoe de kwaliteit van de deelproducten is bewaakt en in hoofdstuk 6 is aan de hand van de antwoorden op de onderzoeksvragen de uiteindelijke conclusie opgesteld en wordt de hoofdonderzoeksvraag beantwoord. Tevens worden in hoofdstuk 6 enkele aanbevelingen gedaan. Tenslotte wordt in hoofdstuk 7 teruggekeken op het onderzoekproces zelf.

Aan het eind van het document zijn tevens enkele appendices opgenomen. Deze zijn niet essentieel voor de inhoud van het document maar kunnen extra informatie bieden mocht dit wenselijk zijn. In appendix A is het Plan van Aanpak opgenomen. Hierin wordt het beheer van het project, waaronder de planning, de risicoanalyse en de afbakening beschreven. In appendix B zijn de bronnen die gebruikt zijn voor het vergelijken van de netwerktechnologieën opgenomen. In appendix C wordt het normalisatieproces van de database beschreven. Deze kan gebruikt worden om een indicatie te krijgen over de kwaliteit van de database. In appendix D wordt het ontwerp van de componenten/deelproducten beschreven. Deze kan worden gebruikt om meer inzicht in de werking van de componenten/deelproducten te krijgen. In appendix E wordt dieper hierop ingegaan en wordt de (code)technische werking van de componenten/deelproducten beschreven.

2 Project context

Dit hoofdstuk zal ingaan op de omgeving waarin het project wordt uitgevoerd en andere externe factoren waar het project mee te maken krijgt.

2.1 De organisatie

De organisatie heette M2M4ALL wat vertaald “machine naar machine voor iedereen” betekent. Met machine naar machine wordt M2M communicatie bedoeld. Veel producten van M2M4ALL zijn bekend geworden onder de merknaam SODAQ. Daarom is besloten om de bedrijfsnaam te veranderen naar SODAQ. De organisatie is op 01-09-2013 opgericht en telt elf medewerkers. De organisatie is gevestigd in Hilversum en houdt zich voornamelijk bezig met ontwikkelen van embedded hardware en software, consultancy en training.

SODAQ houdt zich ook bezig met het ontwikkelen van onderhoudsarme weerstations die autonoom op duurzame energiebronnen zoals zonnestroom kunnen draaien (“Verticals”, 2015). Deze worden voornamelijk ingezet in Afrika.

Omdat het een kleine en jonge organisatie is, is er praktisch geen sprake van een organisatiestructuur. Wel is gedurende de stage de organisatie gegroeid en veranderd. Zo is de kantooroppervlakte meer dan verdubbeld en wordt er tegenwoordig gezamenlijk geluncht. Ook wordt er meer gezamenlijk gepland en wordt er ingezet op betere samenwerking. Dit wordt onder andere gedaan door *scrum* toe te passen.

2.2 Bedrijfsbegeleider

De bedrijfsbegeleider is Jan Willem Smeenk. Tevens is hij de opdrachtgever. Hij heeft in het verleden zich bezig gehouden met onder andere het opzetten van netwerken en heeft kennis in IOT oplossingen. Ook op technisch gebied heeft hij brede kennis.

Naam:	Jan Willem Smeenk	Email adres:	janwillem@m2m4all.com
Telefoonnummer:	06-81517710		

2.3 Overige medewerkers

De meeste werknemers van SODAQ zijn elektrotechnici. Sommige zijn gespecialiseerd in bijvoorbeeld radiotechniek terwijl anderen erg vaardig zijn in het ontwerpen van PCBs. Twee werknemers hadden voor aanvang van de stage al met draadloze netwerktechnologieën gewerkt en hadden daar al enige ervaring in opgedaan.

Naast elektrotechnici zijn er ook enkele programmeurs. Deze zijn wel in de minderheid. Omdat enkele medewerkers geen Nederlands spreken is de voertaal over het algemeen Engels.

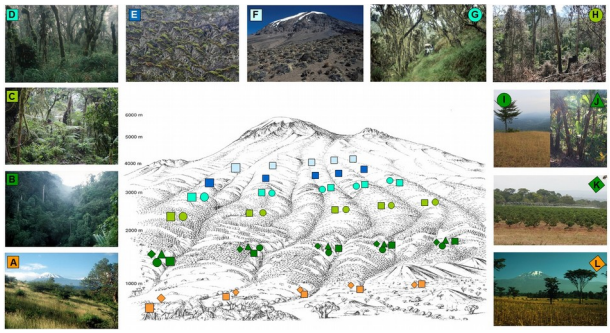
2.4 De stageopdracht

De stageopdracht is een, voor de organisatie, relatief lange opdracht. De uitvoering ervan vindt grotendeels onafhankelijk plaats naast andere lopende opdrachten. Daarom wordt bijvoorbeeld de uitvoering van de stageopdracht niet meegenomen in de *scrum meetings*.

2.5 Kilimanjaro berg

De Kilimanjaro berg is een berg met drie vulkanische kraters. Het hoogste punt ligt op 5895 meter NAP en is daarmee de hoogste berg in Afrika (“Mount Kilimanjaro”, 2016). De berg wordt door veel toeristen beklommen en er wordt ook onderzoek naar klimaat op de berg gedaan. Dit omdat er op de berg zich allerlei klimaatverschijnselen voordoen en er sprake is van meerdere klimaatzones, zoals te zien is op afbeelding 3 (“Mount Kilimanjaro Climate Zones”, 2016).

Het KiLi project, dat onderzoek doet naar het klimaat op de berg, is in februari 2010 gestart en bestaat uit meerdere deelprojecten (“Project Summary”. 2010). Deze doen allen onderzoek naar klimaat en ecosystemen op de Kilimanjaro berg. Er worden meerdere sensoren gebruikt, waaronder regensensoren, om het klimaat en de ecosystemen in kaart te brengen.



Afbeelding 3: Verscheidene klimaten op de Kilimanjaro berg (“Study area”, 2010)

Op en in de omgeving van de Kilimanjaro berg zijn slechts beperkte elektriciteitsvoorzieningen en is er geen vaste internet aansluiting aanwezig.

3 Project definitie

In dit hoofdstuk wordt de opdracht beschreven en worden de onderzoeksvragen gesteld. Ook worden de onderzoeksmethoden en de (deel)producten benoemd. Extra informatie betreffende de probleemstelling, zoals de afbakening, functionele tests en risicoanalyse zijn in hoofdstuk 3 van Appendix A terug te vinden.

3.1 De kwestie

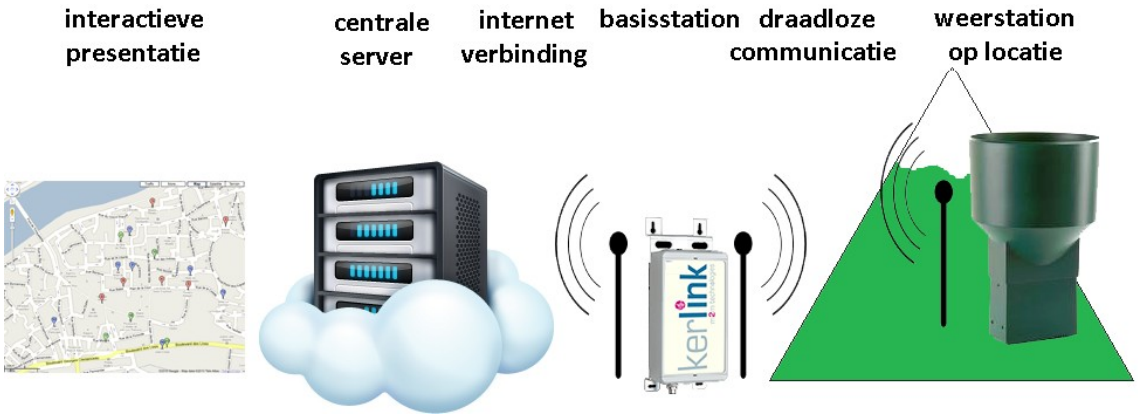
Weerstations worden al lang door bijvoorbeeld het KNMI gebruikt om het weer op een bepaalde locatie te meten. Dit wordt gedaan door bijvoorbeeld de neerslag, de luchtvochtigheid en de luchtdruk te meten. Deze gegevens worden verzameld door de sensoren uit te lezen en met deze gegevens kan, door middel van een complex rekenmodel, een weersvoorspelling worden gedaan. Deze weersvoorspelling wordt vervolgens bijvoorbeeld via het journaal gepubliceerd wat de bevolking de kans geeft om hun dag aan te passen aan het weer. Dit biedt maatschappelijke en economische voordelen.

In een dicht bevolkt en begaanbaar gebied zoals Nederland is het relatief eenvoudig om weerstations te plaatsen, te voorzien van elektriciteit, te onderhouden en uit te lezen. Op de Kilimanjaro berg is dat een ander verhaal: Elektriciteitsvoorzieningen en netwerkinfrastructuur ontbreken en ook begaanbare wegen zijn schaars. De weerstations kunnen alleen worden uitgelezen door te voet er naar toe te lopen, waardoor het uitlezen van sommige weerstations enkele dagen kan duren. Ook is het onderhouden en voeden van deze weerstations lastig.

M2M4ALL levert al onderhoudsarme en autonoom gevoede weerstations. Echter, een manier om de gegevens van een weerstation geautomatiseerd uit te lezen is er nog niet. Omdat het aanleggen van kabels op dit terrein niet haalbaar is, is een draadloze communicatie methode de enige oplossing. Maar omdat de energievoorziening zeer beperkt is en over langere afstanden moet worden gecommuniceerd zijn slechts een paar standaarden mogelijk geschikt.

3.2 De opdracht

De opdracht is om weerstations draadloos uitleesbaar te maken en de verkregen gegevens verder te verwerken. Dit omvat het gehele proces tot en met de uiteindelijke gebruikersinterface. De componenten hiervoor worden in afbeelding 4 getoond. De opdracht is dus een productopdracht (type 3).



Afbeelding 4: Overzicht van alle componenten van het project

Deze weerstations staan op de Kilimanjaro berg en hebben autonome energievoorziening. Aan de voet van de berg zullen ook enkele basisstations staan waar de weerstations hun gegevens door middel van draadloze communicatie naar toe zenden. De te bepalen locatie van deze weerstations hangt af van de mogelijkheden van de gekozen draadloze netwerktechnologie. Waar mogelijk zullen de basisstations aangesloten worden op het vaste elektriciteitsnetwerk. Waar dit niet mogelijk is, zullen deze een autonome energievoorziening moeten hebben. De basisstations zullen de gegevens van de weerstations via deze internetverbinding versturen naar een centrale server. Deze centrale server plaatst de gegevens in de database. Via een GUI kunnen de gegevens worden opgevraagd en worden getoond.

Omdat de gegevens voor wetenschappelijk onderzoek worden gebruikt is het essentieel dat deze betrouwbaar zijn. Er moet gekeken worden naar mogelijkheden om de betrouwbaarheid te verhogen door te voorkomen dat gegevens verloren gaan of corrupt raken.

Om de weerstations eventueel ook voor andere toepassingen, zoals alarmsystemen, te kunnen gebruiken moeten de gegevens binnen een bepaalde periode gepresenteerd worden aan de gebruiker. Echter, omdat er externe factoren, zoals de internetverbinding van de basisstations, aanwezig zijn die dit proces kunnen verstoren, is het onmogelijk om een daadwerkelijke realtime eis op te stellen. Daarom is er gekozen om een *semi-realtime* eis op te stellen: Onder normale omstandigheden dienen de meetgegevens van de weerstations binnen 10 minuten aan de eindgebruiker gepresenteerd te worden.

De opdracht is voldaan wanneer er een *proof of concept* kan worden getoond waarbij de werking van de infrastructuur wordt aangetoond. Wanneer de GUI de juiste gegevens van de weerstations toont en dit interactief kan laden is de opdracht succesvol voldaan. Dit dient gerealiseerd te worden binnen de periode van de stage.

3.3 De doelstelling

De doelstelling van het project is om, binnen de duur van de stageperiode, een oplossing te realiseren waarmee de weerstations op de Kilimanjaro berg draadloos uitgelezen kunnen worden om vervolgens de meetgegevens van deze weerstations op een centrale server op te slaan. Deze gegevens moeten tevens kunnen worden getoond in een interactieve gebruikersinterface.

3.4 Onderzoeksvraag

De opdracht zoals hierboven beschreven leidt tot de volgende onderzoeksvraag:

Hoe kunnen de gegevens van weerstations op de Kilimanjaro berg betrouwbaar en energie efficiënt uitgelezen worden door middel van draadloze communicatie en gepresenteerd worden in een interactieve gebruikersinterface zodat de gegevens geautomatiseerd en semi-realtime inzichtelijk zijn?

Daarvoor moeten de volgende deelonderzoeksvragen worden beantwoord:

1. Hoe kan de communicatie tussen de weerstations en de basisstations gerealiseerd worden?
    1. Welke netwerktechnologieën zijn geschikt?
        1. Wat is het bereik van deze netwerktechnologieën?
    2. Welk protocol moet er worden gebruikt?
2. Hoe kan de communicatie tussen de basisstations en de centrale server gerealiseerd worden?
    1. Hoe kan het basisstation garanderen dat de gegevens bij de centrale server aankomen?
        1. Hoe moet het basisstation omgaan met langdurige netwerkstoringen?
    2. Hoe kunnen de gegevens via GPRS/3G naar de centrale server worden verzonden?
3. Hoe kunnen de gegevens van de weerstations worden opgeslagen?
    1. Hoe moet de server qua software worden ingericht?
    2. Hoe moet de database worden ingericht?
4. Hoe moeten de weerstations en de basisstations worden gevoed?
    1. Hoeveel energie is er op de Kilimanjaro berg beschikbaar?
    2. Hoeveel energie verbruikt een weerstation?
    3. Hoeveel energie verbruikt een basisstation?
    4. Hoeveel capaciteit moet de batterij hebben?
5. Hoe kan de betrouwbaarheid van de gegevens worden gewaarborgd?
    1. Hoe kan datacorruptie worden gedetecteerd?
    2. Hoe kunnen de gegevens worden beveiligd zodat deze niet door derden gemanipuleerd kunnen worden?

3.5 Onderzoeksmethoden

In de volgende tabel worden per deelonderzoeksvraag de gebruikte onderzoeksmethode(n) en onderzoekstechniek(en) beschreven:

#	**Onderzoeksvraag**	**Type**	**Onderzoeks-methode(n)**	**Methode en techniek(en)**
1	Hoe kan de communicatie tussen de weerstations en de basisstations gerealiseerd worden?	Beschrijvend & vergelijkend	Deskresearch, experimenteren & veldonderzoek	
1.1	Welke netwerktechnologieën zijn mogelijk geschikt?	Beschrijvend & vergelijkend	Deskresearch & veldonderzoek	Door netwerktechnologieën te vergelijken en bereik in het veld te meten wordt een antwoord geformuleerd.
1.2	Welk protocol moet er worden gebruikt?	Beschrijvend	Deskresearch	Aan de hand van de te versturen gegevens wordt een protocol opgesteld.
2	Hoe kan de communicatie tussen de basisstations en de centrale server gerealiseerd worden?	Beschrijvend	Deskresearch & veldonderzoek	
2.1	Hoe kan het basisstation garanderen dat de gegevens bij de centrale server aankomen?	Beschrijvend & vergelijkend	Deskresearch	Door synchronisatie methoden te vergelijken wordt een antwoord geformuleerd.
2.2	Hoe kunnen de gegevens via GPRS/3G naar de centrale server worden verzonden?	Beschrijvend	Deskresearch & veldonderzoek	Door de werking van GPRS/3G te analyseren en de werking te testen wordt een antwoord geformuleerd.
3	Hoe kunnen de gegevens van de weerstations worden opgeslagen?	Beschrijvend	Deskresearch	
3.1	Hoe moet de server qua software worden ingericht?	Beschrijvend	Deskresearch	Aan de hand van de *functional requirements* wordt een antwoord geformuleerd.
3.2	Hoe moet de database worden ingericht?	Beschrijvend	Deskresearch	Aan de hand van de *functional requirements* en de normalisatie regels van relationele databases wordt een antwoord geformuleerd.
4	Hoe moeten de weerstations en de basisstations worden gevoed?	Beschrijvend	Deskresearch	

4.1	Hoeveel energie is er beschikbaar?	Beschrijvend	Deskresearch & veldonderzoek	Aan de hand van klimaateigenschappen en lokaal gemeten waarden wordt een antwoord geformuleerd.
4.2	Hoeveel energie verbruikt een weerstation?	Beschrijvend	Deskresearch & veldonderzoek	Aan de hand van datasheets van de gebruikte hardware en gemeten waarden wordt een antwoord geformuleerd.
4.3	Hoeveel energie verbruikt een basisstation?	Beschrijvend	Deskresearch & veldonderzoek	Aan de hand van datasheets van de gebruikte hardware en gemeten waarden wordt een antwoord geformuleerd.
4.4	Hoeveel capaciteit moet de batterij hebben?	Beschrijvend	Deskresearch	Aan de hand van klimaateigenschappen wordt een antwoord geformuleerd.
5	Hoe kan de betrouwbaarheid van de gegevens worden gewaarborgd?	Beschrijvend	Deskresearch	
5.1	Hoe kan datacorruptie worden gedetecteerd?	Beschrijvend & vergelijkend	Deskresearch	Aan de hand van bestaande methoden van foutdetectie wordt een antwoord geformuleerd.
5.2	Hoe kunnen de gegevens worden beveiligd zodat deze niet door derden gemanipuleerd kunnen worden?	Beschrijvend	Deskresearch	Aan de hand van bestaande beveiligingstechnieken wordt een antwoord geformuleerd.

3.6 (Deel)producten

Gedurende dit project zullen de volgende (deel)producten in chronologische volgorde worden opgeleverd:

1. Een stuk software dat op de micro-controllers, die in het weerstations worden gebruikt, kan draaien en in staat is om de regensensor uit te lezen en de gegevens draadloos te versturen.
2. Een stuk software dat op de basisstations kan draaien en in staat is om gegevens van de weerstations te ontvangen en door te sturen naar de centrale server.
3. Een ingerichte centrale server die in staat is om gegevens van de basisstations te ontvangen en op te slaan.
4. Een GUI die de gegevens die op de centrale server staan opgeslagen interactief kan presenteren.
5. Een uitbreiding op deelproduct 4 die de gegevens van de centrale server op een kaart kan presenteren.
6. Een uitbreiding op deelproduct 1 die in staat is om datacorruptie te detecteren.
7. Een uitbreiding op deelproduct 6 die in staat is om de werking van het weerstation te monitoren.
8. Een uitbreiding op deelproduct 7 die in staat is om meerdere sensoren uit te lezen.

4 Onderzoeksresultaten

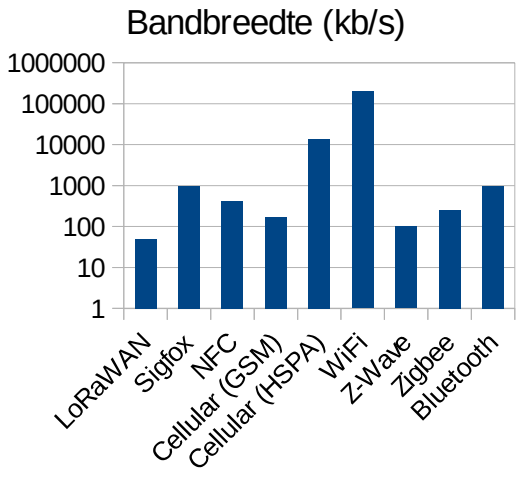
In dit hoofdstuk worden de resultaten van het verrichte onderzoek beschreven en wordt er antwoord gegeven op de in hoofdstuk 3 opgestelde deelvragen.

4.1 Realisatie communicatie tussen de weerstations en de basisstations

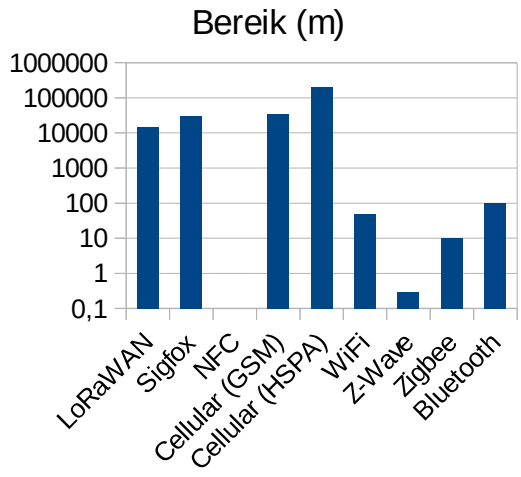
Om tot een realisatie van de communicatie tussen de weerstations en de basisstations te komen moet er eerst een technologie worden gekozen die voldoende bereik heeft, in het energiebudget past en genoeg bandbreedte heeft. Daarna moet worden geverifieerd of het geclaimde bereik wel bereikt wordt. Ook moet er een protocol voor de data worden gedefinieerd.

4.1.1 Geschikte netwerktechnologieën

De weerstations die op locatie staan moeten draadloos worden uitgelezen. Dit terwijl de stroomvoorzieningen beperkt zijn en er over langere afstanden moet worden gezonden. Om een geschikte netwerktechnologie te kiezen, moeten deze met elkaar worden vergeleken. De volgende figuren wegen de netwerktechnologieën tegen elkaar af op gebied van bereik, bandbreedte en verbruik[1].



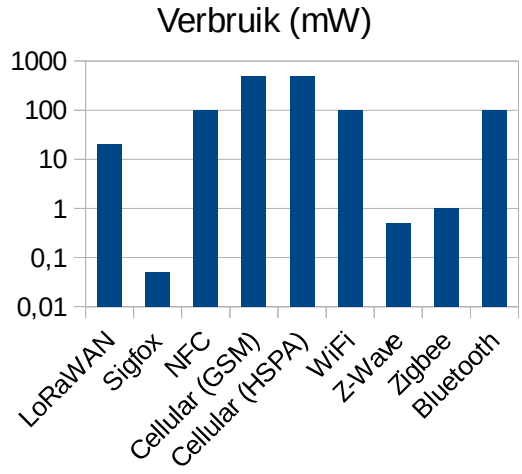
Afbeelding 5: Maximale bandbreedte van verscheidene netwerktechnologieën



Afbeelding 7: Maximale bereik van verscheidene netwerktechnologieën

De figuren maken duidelijk dat onderlinge verschillen groot zijn. Voor de toepassing in dit project is het verbruik en het bereik belangrijker dan de bandbreedte. Immers hoeft enkel de neerslag van een bepaalde periode verzonden worden.

Op gebied van bereik scoren HSPA, GSM, Sigfox en LoRaWAN goed. Op gebied van verbruik scoren Sigfox, Z-Wave, Zigbee en LoRaWAN goed. Voor dit project is de technologie die op beide gebieden goed scoort het meest geschikt. Dit zijn Sigfox en LoRaWAN.



Afbeelding 6: Maximale verbruik van verscheidene netwerktechnologieën

1 De gebruikte bronnen voor deze figuren zijn terug te vinden in Appendix B.

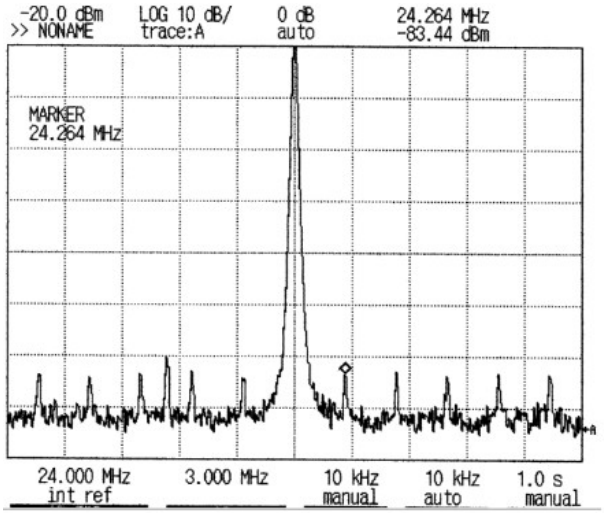
Sigfox versus LoRaWAN

Omdat beide netwerktechnologieën goed scoren is het belangrijk om deze technologieën nader te onderzoeken, om zo een goede keuze te kunnen maken.

Sigfox

Sigfox is een rond 2012 in Frankrijk ontwikkelde technologie gemaakt voor het uitlezen van kleine apparaten zoals slimme energiemeters en andere huishoudelijke apparaten. Het gebruikt *Ultra Narrow Band* (UNB) op een carrier frequentie van 868 MHz.

Bij *Ultra Narrow Band* wordt *Digital Phase Shift Keying* (DPSK) gebruikt om gegevens op de carrier frequentie te moduleren (zie afbeelding 8). Normaal resulteert dit tot een bredere band omdat de fase overgangen scherpe hoeken hebben. Echter, bij *Ultra Narrow Band* worden geavanceerde filters gebruikt die deze *side bands* kunnen wegfilteren (Walker, 2014).



Afbeelding 8: Een spectrum analyse van een Ultra Narrow Band signaal; (Walker, H. R. (2014). Ultra Narrow Band Modulation Textbook.)

Sigfox dekking wordt geboden door zogeheten *Sigfox Network Operators* (SNO's). In Nederland is dat Aerea. Echter, bij de Kilimanjaro berg is geen netwerkdekking van een SNO. Het is mogelijk om met Sigfox een contract aan te gaan om een nieuw netwerk op te zetten maar enkel met de intentie om een dekkend en openbaar netwerk op te zetten. Ook moet er gebruikt worden gemaakt van de clouddiensten die Sigfox biedt (“About SIGFOX”, 2015). Dit maakt het project afhankelijk van Sigfox.

LoRaWAN

LoRaWAN is een door Cycleo in 2009 ontwikkelde standaard voor *long range* (LoRa) netwerken zonder dat hiervoor grote zendvermogens zijn benodigd. In 2012 werd Cycleo overgenomen door Semtech. Beide bedrijven zijn gevestigd in Frankrijk.

LoRaWAN gebruikt LoRa als modulatie methode. Dit is een *proprietary* modulatie techniek die *Chirp Spread Spectrum* (CSS) en *Forward Error Correction* (FEC) combineert (“LoRa Modulation Basics”, 2015). LoRaWAN gebruikt dus de gehele bandbreedte van het kanaal en is daardoor beter bestand tegen verstoringen op een specifieke frequentie. De LoRaWAN ontvangers zijn in staat om tot 19,5 dB onder de *noise floor* data te ontvangen, wat detectie en locatiebepaling van LoRaWAN installaties erg lastig maakt (“SX1272/3/6/7/8; LoRa Modem”, 2013).

LoRaWAN wordt nog niet uitgebreid in Nederland aangeboden. KPN heeft enkele antennes geplaatst en ook een andere provider, The Things Network, heeft een netwerk in enkele steden operationeel. Bij LoRaWAN is het mogelijk om een *gateway* los aan te schaffen om zo een eigen netwerk op te zetten. Hierdoor zou het project niet afhankelijk zijn van diensten van derde partijen.

Afweging

Sigfox biedt betere mogelijkheden op gebied van technologie. Zo heeft het een groter bereik, een grotere bandbreedte en een lager verbruik. Toch is er gekozen voor LoRaWAN. Dat komt doordat de eisen die Sigfox stelt voor het opzetten van een eigen netwerk simpelweg niet haalbaar zijn binnen de scope van dit project.

Ook het feit dat LoRaWAN lastiger te traceren is dan Sigfox is een voordeel. Dit voorkomt diefstal van apparatuur en maakt het lastiger om personen of dieren te traceren. In Afrika, waar stroperij een probleem kan vormen, is dit een groot voordeel.

4.1.2 Het bereik van LoRaWAN

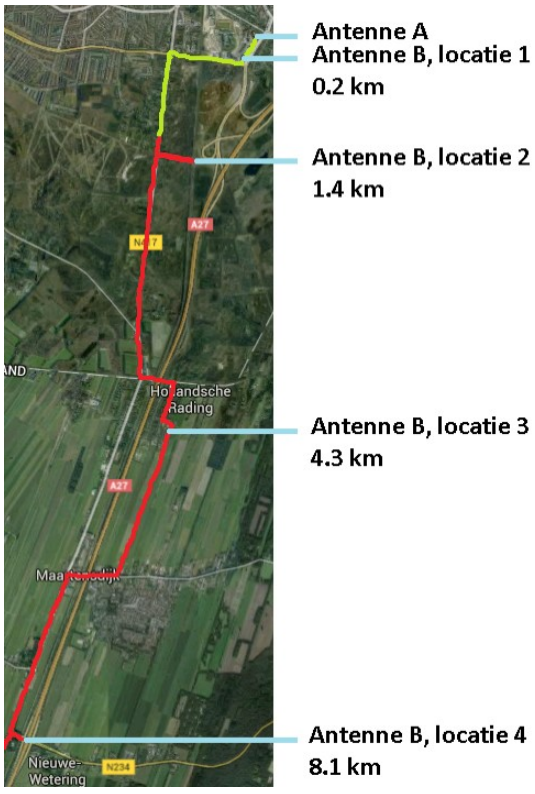
Om te bepalen wat het bereik van LoRaWAN, en daarmee de geschiktheid van deze technologie te bepalen, is er veldonderzoek verricht. Hiervoor zijn vier tests verricht.

Test 1

Bij test 1 werden op vaste locaties metingen verricht met zowel een LoRaWAN antenne als een WiFi richtantenne. Ook werd er met een tweede LoRaWAN antenne op de fiets rondgereden terwijl deze elke 30 seconden een bericht verstuurde. Omdat de kwaliteit van de verbinding significant afneemt wanneer er obstakels tussen beide antennes aanwezig zijn, is er gezocht naar locaties waar weinig obstakels aanwezig zijn. Op de gekozen route is slechts lichte bebouwing en bos aanwezig.

Antenne A werd op het dak van het kantoor geplaatst. Deze staat ongeveer op een hoogte van 25 meter NAP. Antenne B wordt op een afstand van 0,2 km, 1,4 km, 4,3 km en 8,1 km geplaatst. Op de laatste locatie staat antenne B op een brug met een hoogte van 9 meter NAP.

Enkel bij locatie 1 was nog ontvangst, bij alle andere locaties hadden zowel de LoRaWAN als de WiFi richtantenne geen bereik. De tweede LoRaWAN antenne had op enkele honderd meters afstand van locatie 2 geen bereik meer. Daarmee kan worden geconcludeerd dat de WiFi richtantenne een bereik heeft tussen de 0,2 en 1,4 km en de LoRaWAN antennes een bereik hebben van ongeveer 1,3 km.

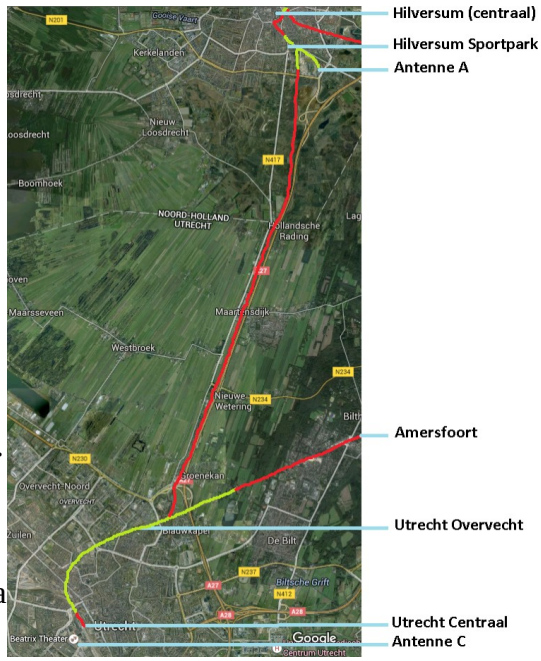


Afbeelding 9: Route van test 1. Groen is wel bereik, rood is geen bereik

Test 2

Bij test 2 werd elke 30 seconden een bericht verstuurd terwijl er met de zender (Antenne B) in de trein werd rondgereden. Het bericht werd tegelijkertijd naar antenne A in Hilversum en naar antenne C in Utrecht gestuurd. Antenne A staat op een hoogte van 25 meter NAP en antenne C staat op de Rabobank Bestuurscentrum toren (ook wel de Verrekijker genoemd), op een hoogte van 107 meter NAP.

Opvallend is dat tussen Hilversum Sportpark en Utrecht Overvecht bijna geen bereik is terwijl dit in test 1 wel was. Dit komt mogelijk omdat de trein recht van antenne A wegrijdt en deze daardoor als een kooi van Faraday functioneert. Tussen Utrecht Centraal en Amersfoort rijdt de trein in een boog van Antenne C weg waardoor deze via het treinraam zichtbaar blijft. Daardoor functioneert deze waarschijnlijk minder als een kooi van Faraday. Op deze route werd een bereik van 5.3 km behaald.



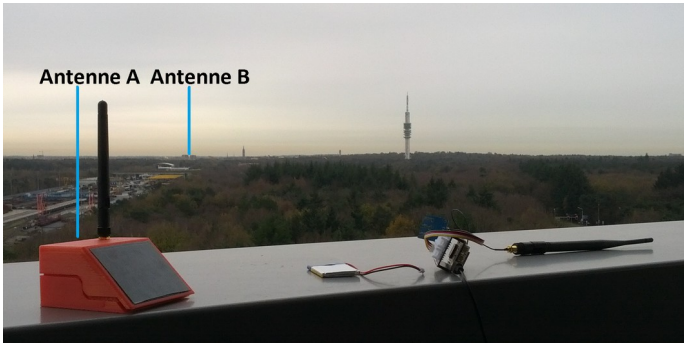
Afbeelding 10: Route van test 2. Groen is wel bereik, rood is geen bereik

Test 3

Bij test 3 werd er gekeken of LoRaWAN meer bereik heeft wanneer er minder obstakels, zoals gebouwen en bomen die de *line of sight* verstoren, aanwezig zijn. Hiervoor is gezocht naar twee hoger gelegen locaties en is gepoogd om een bericht over te brengen.

Antenne A stond op het balkon van de bovenste verdieping van de watertoren in Bussum op een hoogte van ongeveer 40 meter NAP. Antenne B stond op het dak van het kantoor op een hoogte van ongeveer 25 meter NAP. De afstand bedraagt ongeveer 6.3 kilometer. Met deze opstelling was het mogelijk om berichten te verzenden en te ontvangen.

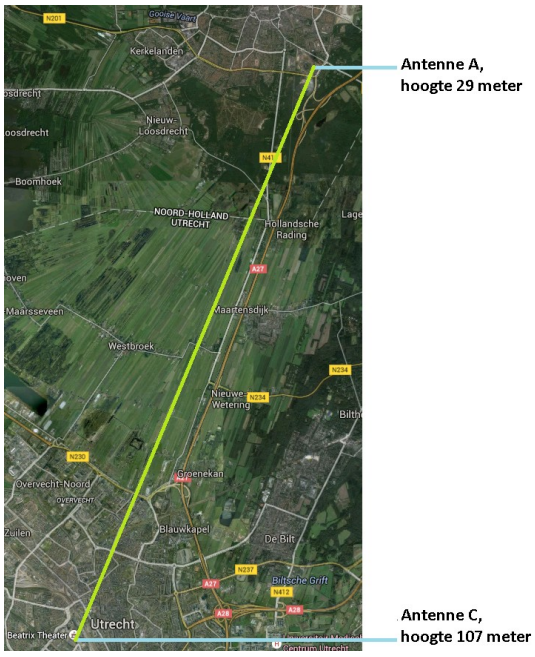
Afbeelding 11: Hemelsbrede verbinding van test 3



Afbeelding 12: Uitzicht vanaf locatie A met aan de horizon locatie B

Test 4

Bij test 4 werd er gekeken of er ook op langere afstand communicatie mogelijk is wanneer er minder obstakels aanwezig zijn. Hiervoor is gekeken of communicatie tussen Hilversum en Utrecht mogelijk is.



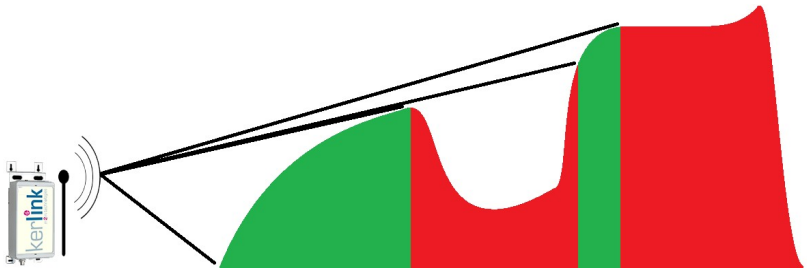
Afbeelding 13: Hemelsbrede verbinding van test 4

Antenne A staat op het dak van het kantoor op een paal van ongeveer 4 meter lengte, wat neerkomt op een hoogte van ongeveer 29 meter NAP. Antenne B staat op de Rabobank Bestuurscentrum toren, op een hoogte van 10 meter NAP. De afstand bedraagt ongeveer 15 kilometer. Met deze opstelling was het mogelijk om berichten te verzenden maar het was niet mogelijk om berichten te ontvangen. Dit komt mogelijk door de lagere gevoeligheid van de *transceiver.*

Conclusie

Uit de tests blijkt dat LoRaWAN enkel het geclaimde bereik behaald wanneer het op hogere hoogten, waar minder obstakels het signaal kunnen storen, wordt gebruikt. Zenden op lagere hoogten heeft drastische impact op het bereik en deze wordt gereduceerd tot slechts een kilometer. Dit komt redelijk overeen met tests van derden (“Extreme Range Links: LoRa 868 / 900MHz SX1272 Module for Arduino, Waspmote and Raspberry Pi”, 2015). Dit bereik is onvoldoende voor dit project.

Voor dit project is LoRaWAN enkel bruikbaar wanneer er voor wordt gezorgd dat er geen obstakels aanwezig zijn door op hogere hoogten te zenden. Dit kan onder andere worden gerealiseerd door gebruik te maken van antennemasten. Het plaatsen van allerlei antennemasten op de Kilimanjaro is kostbaar, diefstalgevoelig en onderhoudsgevoelig. Ook verstoren antennemasten het beeld van de berg wat, gezien het een toeristische trekpleisters is, niet wenselijk is. Als laatste doen antennemasten afbreuk aan de flexibele inzetmogelijkheden van de weerstations.



Afbeelding 14: Bereik van LoRaWAN op een berg. Groen is wel bereik, rood is geen bereik.

Het feit dat de weerstations op een berg staan kan echter voordelig worden gebruikt: Het hoogteverschil kan worden gebruikt om over obstakels heen te zenden. Door het basisstation op enige afstand van de berg te plaatsen en de weerstations op een helling te plaatsen is communicatie zonder obstakels mogelijk zonder gebruik te maken van antennemasten. Er zijn wel enkele nadelen. Zo is het niet mogelijk om vanuit een dal te zenden en is het niet mogelijk om te zenden wanneer er een groot plateau aan de voet van het weerstation aanwezig is (zie afbeelding 14). Het andere nadeel is dat het basisstation verder van de voet van de berg moet worden geplaatst waardoor effectief bereik wordt verspild.

Een ander bijkomend voordeel is het feit dat er op hogere hoogten geen plantgroei is (Körner, 1998). Hierdoor wordt het signaal minder verzwakt, wat resulteert in een meer uniforme signaalsterkte.

4.1.3 Positionering van basisstations

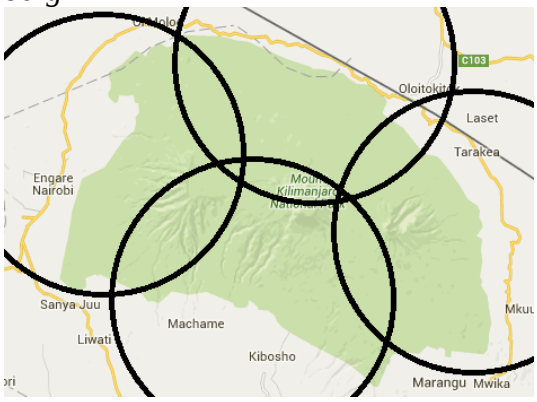
Om dekking op de Kilimanjaro berg te creëren is het noodzakelijk om basisstations te installeren. Om de installatiekosten en de complexiteit van het netwerk te beperken is het doel om het aantal basisstations tot een minimum te beperken

Omdat de grond van de weilanden en akkers om de Kilimanjaro berg relatief vlak is en omdat er over het algemeen geen hoge bomen staan, zijn deze uiterst geschikt voor het plaatsen van basisstations. Ook zijn deze door betere wegen makkelijker bereikbaar voor onderhoud.

Uit onderzoek blijkt dat de afstand tussen de top van de berg en de bosgrens maximaal ongeveer 34 kilometer bedraagt (zie afbeelding 15). Het is niet correct om te veronderstellen dat de te overbruggen afstand 34 kilometer is. Immers wordt hier uitgegaan van een *worst case scenario*. Door basisstations strategisch te plaatsen kan de te overbruggen afstand verkleind worden.

Afbeelding 15: Maximale hemelsbrede afstand tussen de top en de rand van de berg

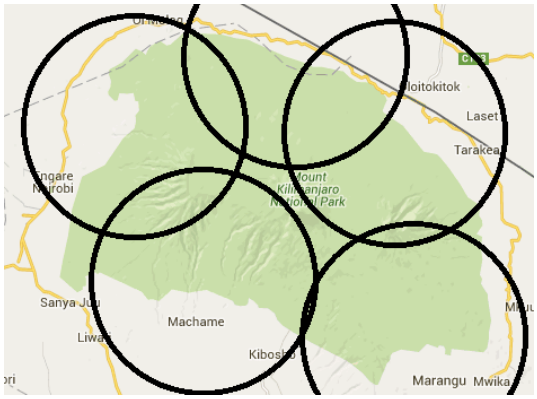
Wanneer aangenomen wordt dat het bereik op de Kilimanjaro berg ongeveer 20 kilometer bedraagt kan worden volstaan met het plaatsen van basisstations langs de rand van de bosgrens. Afbeelding 16 toont aan dat een dekkend netwerk met slechts vier basisstations mogelijk is.



Afbeelding 16: Een volledig dekkend netwerk met 4 basisstations met een bereik van 20 kilometer

Maar wanneer er aangenomen wordt dat het bereik op de Kilimanjaro berg slechts 15 kilometer bedraagt is het niet mogelijk om dekking te realiseren met slechts basisstations langs de bosgrens. Er zal ook een basisstation moeten worden geplaatst in het beboste gebied. Afbeelding 17 toont aan dat een dekkend netwerk met vijf basisstations (of meer) niet mogelijk is.

Anderzijds kan men ook accepteren dat op de top geen dekking is. Uit contact met de onderzoeksgroep blijkt dat er op dit moment geen weerstations op de top staan, waardoor dekking aan de top op dit moment geen voordelen biedt.



Afbeelding 17: Een volledig dekkend netwerk is niet mogelijk met een bereik van slechts 15 kilometer

Geconcludeerd kan worden dat wanneer het bereik 20 kilometer zou zijn, vier basisstations langs de bosgrens volstaan. Wanneer het bereik 15 kilometer zou zijn er met vier basisstations een dekking kan worden gerealiseerd die weliswaar niet de gehele berg dekt maar wel voldoende is voor het project.

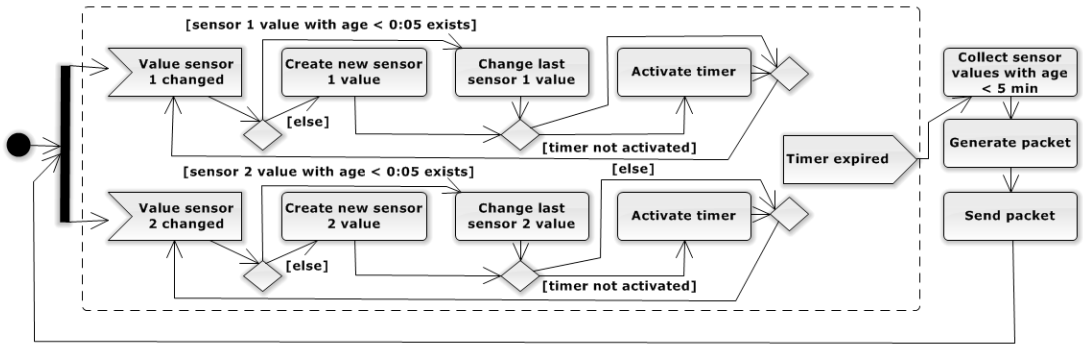
4.1.4 Het applicatieprotocol

Het protocol dat gebruikt wordt voor de communicatie tussen de weerstations en de basisstations functioneert met het LoRaWAN protocol. Het ontworpen protocol vult de APP component (het gedeelte waar de daadwerkelijke inhoud van het pakket in staat) van de LoRaWAN pakketten. De inhoud van de APP component wordt automatisch door de LoRaWAN chip geëncrypt.

Semi-realtime

De sensoren mogen volgens de *functional requirements*[2] een tijdsresolutie van maximaal 5 minuten hebben. Omdat het zenden van zoveel pakketten te veel energie zal verbruiken en de ether zal vervuilen, wordt er enkel een pakket verzonden wanneer de waarde van een sensor (significant) verandert.

Wanneer er veranderingen in de waarde van de sensor plaatsvinden zal deze eerst worden opgeslagen en pas na vijf minuten worden verzonden. Dit maakt het mogelijk om de waarde van andere sensoren die ook in die periode veranderen in hetzelfde pakket mee te sturen. Ook voorkomt het dat wanneer een sensor binnen bijvoorbeeld een minuut weer van waarde verandert er opnieuw een pakket wordt gestuurd. Het weerstation zal de eerder opgeslagen waarde, afhankelijk van het type sensor, overschrijven of een gemiddelde waarde berekenen. De functionele werking is in het volgende *activity diagram* weergegeven:



Afbeelding 18: Activity Diagram van het weerstation

Betrouwbaarheid

Omdat het project gebruikt wordt voor wetenschappelijke doeleinden waarbij betrouwbaarheid een grote rol speelt, wat zich onder andere manifesteert in de *functional requirement* dat slechts één procent van de meetgegevens verloren mag gaan, is het noodzakelijk om een verbinding te realiseren die hier aan kan voldoen. Omdat uit de voorgaande tests bleek dat op langere afstand het niet mogelijk was om vanaf het basisstation ontvangen berichten te bevestigen, is enkel simplex communicatie mogelijk. Dit betekent dat er geen bevestiging kan worden gegeven en er dus geen garantie kan zijn dat het pakket daadwerkelijk aankomt.

Om te bepalen of er aan de *functional requirement* wordt voldaan is gekeken naar de statistische kans dat meetgegevens verloren gaan. Hiervoor is gekeken naar de oorzaken van netwerkstoringen in draadloze netwerken en de kans hierop.

2 Zie hoofdstuk 3.4 in appendix A voor meer informatie.

Storingen in draadloze netwerken worden, naast defecten aan de zend- en ontvangstapparatuur zelf, veroorzaakt door elektromagnetische storingen in het spectrum waar het netwerk gebruikt van maakt (zie afbeelding 19). Vaak wordt dit “een storing in de ether” genoemd. Deze storingen worden veroorzaakt door andere bronnen die zelf ook signalen uitzenden in (ongeveer) dezelfde frequentieband. Vaak is dit (verouderde) apparatuur die niet of enkel aan verouderde standaarden voldoet op het gebied van toelaatbare *electromagnetic interference* (EMI). Hierbij moet worden gedacht aan apparatuur die gebruik maakt van hoogfrequente schakelingen zoals choppers, microprocessoren, CRT schermen, etc. (“Interference on AM and FM Radios”, 2016) (“Tandy Radio Shack TRS-80”, 2013) (Karar, Saini, Srinivas, & Kumar, 2003). Omdat de verwachting is dat in een dunbevolkt gebied met een lagere welvaart er minder sprake zal zijn van deze oorzaak en omdat van te voren niet bekend zal zijn door exact welk type apparaat deze storing dan wordt veroorzaakt (en dus de storingsfrequentie onbekend is), is deze bron van storing niet verder onderzocht.

Afbeelding 19: Een analoog televisiesignaal heeft last van storing (“Electromagnetic interference”, 2016)

Ook natuurlijke verschijnselen kunnen draadloze netwerken storen: Rain fading is het verschijnsel waarbij signalen worden gedempt door neerslag (Rockets, 2016). Echter, dit verschijnsel heeft alleen effect op hoogfrequente signalen (boven de 10 GHz), wat voor dit project niet van toepassing is.

Verder kunnen natuurlijke verschijnselen ook signalen uitzenden. Deze worden voornamelijk veroorzaakt door onweer (“Atmospheric noise”, 2016) (Krider, Weidman, & Noggle, 1977). Om de gevolgen van onweer in kaart te brengen is onderzocht hoe vaak en hoe lang het op de Kilimanjaro onweert.

Onweer

In Tanzania komt onweer op ongeveer 60 dagen per jaar voor (“Colour coded chart of number of storms per annum”, 2015) (“Thunderstorms: The “Stormiest” Places in The U.S.A. and the World”, 2012). Onweer komt in sommige maanden meer voor dan in andere, maar komt bijvoorbeeld in Australië, een gebied met een vergelijkbare hoeveelheid onweer, in de slechtste gevallen minder dan 60% van de dagen voor (Kuleshov, Hoedt, Wright, & Brewster, 2001). Op dagen met onweer onweert het 2 tot 2.6 uur per dag (Watson & Holle, 1996). Aangezien een onweersbui vaak slechts enkele uren duurt, is het aannemelijk om te stellen dat er vaak slechts één onweersbui op een dag voorkomt (Lombardo, Main, & Simiu, 2009). Dit betekent dat in deze perioden op maandbasis ongeveer 6% van de tijd het onweert. Met de volgende formule kan worden berekend hoe vaak een pakket moet worden verzonden om aan de *functional requirement* te voldoen dat 99% van de meetgegevens moet aankomen. Hierbij wordt uitgegaan van een *worst case scenario* waarbij tijdens een onweersbui draadloze communicatie volledig onmogelijk is.

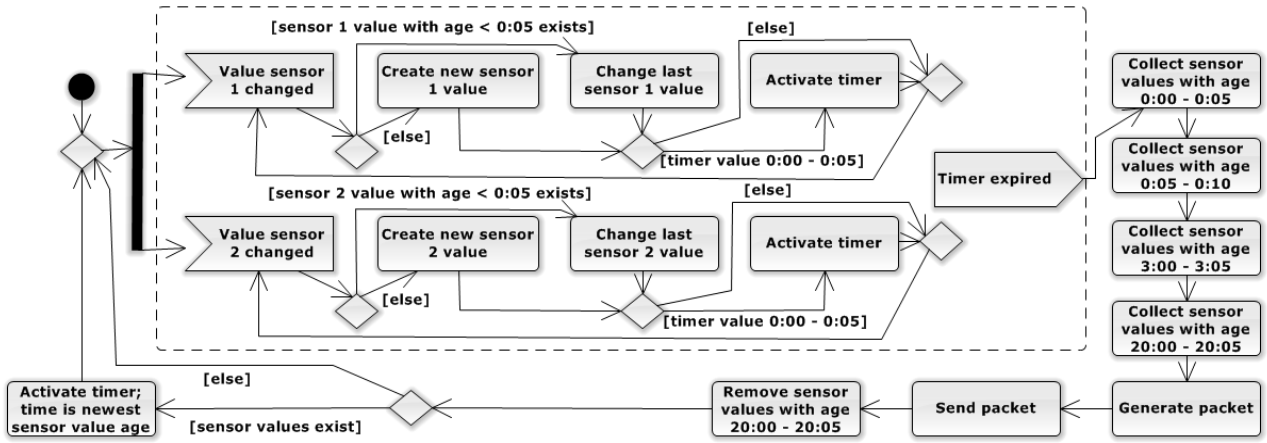
$$\log(0.01)/\log(0.06)=1.6$$

Omdat onweersbuien continue zijn, is het essentieel dat hertransmissietijden goed worden gekozen zodat meetgegevens “buiten de onweersbuien om” worden verzonden. Deze tijden staan in de volgende tabel:

Tijd (h:mm)	Motivatie
0:00	Eerste poging.
0:05	Tweede poging vlak na de eerste poging, om zo alsnog semi-realtime de data te ontvangen wanneer de eerste poging mislukte.
3:00	Derde poging na drie uur zodat het aannemelijk is dat eventuele onweersbuien zijn opgelost.
20:00	Vierde poging na 20 uur zodat langdurige storingen of onderhoud aan infrastructuur opgevangen kunnen worden zonder dat er dataverlies ontstaat. De waarde kon niet hoger i.v.m. beperkt geheugen op de gebruikte microcontroller.

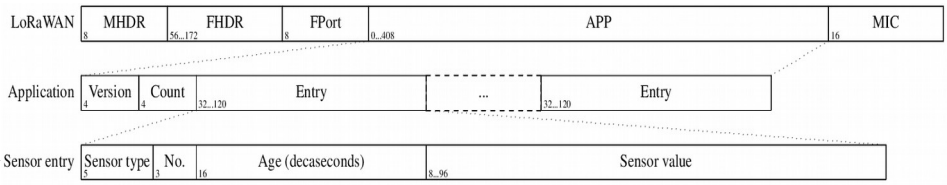
Er is gekozen voor vier transmissies, ook al zouden twee statistisch moeten voldoen. Dit omdat ook elders in de keten mogelijk pakketten verloren kunnen gaan en om, zoals hierboven beschreven, andere storingen te kunnen opvangen.

De functionele werking van het transmissiesysteem met de hertransmissies is in het volgende *activity diagram* weergegeven.



Afbeelding 20: Activity Diagram van het weerstation met hertransmissie

4.1.5 Protocol definitie



Afbeelding 21: Het applicatie protocol dat door de weerstations en basisstations worden gebruikt. De veldgrootten zijn in bits weergegeven

Het applicatieprotocol vult de APP component van het LoRaWAN pakket en stuurt, om overhead te beperken, verscheidene meetgegevens in één pakket door. Deze worden achtereen in het pakket geplaatst (zie afbeelding 21). Het protocol heeft een versie veld zodat er meerdere versies van het protocol naast elkaar gebruikt kunnen worden. Dit maakt het doorvoeren van eventuele aanpassingen makkelijker omdat er onderscheid kan worden gemaakt.

Hierna volgt een veld dat het aantal meetgegevens die in het pakket zitten aangeeft. Dit veld heeft een maximale waarde van 15, wat betekent dat per LoRaWAN pakket maximaal 15 meetgegevens verstuurd kunnen worden. Aangezien de lengte van een LoRaWAN pakket zeer beperkt is en de gemiddelde lengte van de meetgegevens meerdere bytes is, is dit voldoende om volle pakketten te kunnen genereren en zo efficiënt met de beschikbare bandbreedte en energie om te gaan.

Meetgegevens

Na het veld dat het aantal meetgegevens in het pakket aangeeft volgen de metingen. Om een meting te kunnen koppelen aan een sensor wordt er bij elke meting het type sensor en het nummer van de sensor doorgegeven. Het sensor type is een voorgedefinieerde waarde die het type sensor aangeeft. Hiermee wordt ook de manier waarop de waarde geïnterpreteerd moet worden aangegeven. Er kunnen maximaal 32 sensor typen gedefinieerd worden. Momenteel zijn de volgende sensor typen gedefinieerd[3]:

0. System measurement: De meting (zelfdiagnose) van het weerstation zelf. De spanning van de batterij die het weerstation voedt en de *uptime* worden opgestuurd. Hiermee kan de werking van de batterij, het zonnepaneel en de software worden gecontroleerd. Ook wordt de uitkomst van de zelfdiagnose opgestuurd.
1. GPS sensor: De GPS coördinaat van het weerstation, inclusief de hoogte. Hiermee kan de locatie van het weerstation worden bepaald en kan worden vastgesteld dat deze is verplaatst door bijvoorbeeld wind, grondverzakkingen of diefstal.
2. Temperature sensor: De gemeten temperatuur in 0.01 Kelvin.
3. Rain sensor: De gemeten regenval in micrometer per m^2/5 minutes.

Het nummer van de sensor wordt in het volgende veld geplaatst. Dit veld kan acht waarden aannemen wat betekent dat, om elke sensor nog te kunnen identificeren, per type sensor maximaal acht sensoren aangesloten kunnen worden. Als dit niet voldoende is zal er een extra sensor type moeten worden aangemaakt. De *unique identifier* van een sensor wordt daarmee [weerstation nr.] + [sensor type] + [sensor nr.].

Hierna volgt de datum en tijd van de meting. Omdat de weerstations door het ontbreken van een GPS sensor en duplex communicatie (waardoor zij geen berichten kunnen ontvangen) geen mogelijkheid hebben om hun tijd te synchroniseren is het onmogelijk de daadwerkelijke datum en tijd vast te stellen. Daarom wordt de leeftijd van de meting opgestuurd.

Als laatste wordt de daadwerkelijke sensor waarde zelf nog opgestuurd. De lengte en het formaat van dit veld is afhankelijk van het sensor type en wordt, om bandbreedte te besparen, niet expliciet opgegeven. De centrale server zal zelf de lengte en het formaat, aan de hand van de informatie in de database, moeten bepalen.

3 Zie hoofdstuk 2.1.3 van appendix D voor een meer technische definitie van de gedefinieerde sensor types.

4.2 Realisatie communicatie tussen de basisstations en de centrale server

Nadat de meetgegevens zijn aangekomen bij de basisstations moeten deze worden doorgestuurd naar de centrale server zodat daar de gegevens centraal kunnen worden verwerkt en kunnen worden gepresenteerd aan de eindgebruiker(s). Daarvoor moet een verbinding worden gedefinieerd.

4.2.1 Geschikte netwerktechnologieën

Voor de communicatie tussen de basisstations en de centrale server zijn drie netwerktechnologieën bekeken: een straalverbinding met WiFi, een satellietverbinding en een 3G/GPRS verbinding.

Straalverbinding

Afbeelding 22: Een WiFi richtantenne (“SXT 5”, 2015)

Straalverbinding zijn verbindingen waarbij twee (of meerdere) objecten op een vaste locatie draadloos worden verbonden door middel van richtantennes (zie afbeelding 22). Hiermee kunnen afstanden van meer dan drie kilometer worden overbrugd (“3Km Wifi linktest van Mikrotik Omnitik naar SXT met hindernis”, 2015). Dit is enkel mogelijk wanneer er weinig objecten in de weg staan omdat deze het signaal verstoren. De gevoeligheid hiervoor is tijdens het veldonderzoek aangetoond[4].

Doordat de basisstations voor een goede communicatie met de weerstations om de berg heen moeten staan, is een straalverbinding tussen de basisstations niet mogelijk. Immers blokkeert de berg de straalverbinding. Ook om de berg heen zenden, door meerdere straalantennes in serie te plaatsen is niet haalbaar en leidt tot meer complexiteit.

Satellietverbinding

Bij satellietverbindingen wordt de internetverbinding door middel van een satelliet tot stand gebracht. Deze verbindingen hebben vaak een hoge *latency* (“Satellite Internet faster than advertised, but latency still awful”, 2013). Voor dit project vormt dat niet een groot probleem.

Op het gebied van betrouwbaarheid scoren deze vergelijkbaar of, volgens een provider zelf, zelfs iets hoger dan bijvoorbeeld 3G/GPRS verbindingen (“Satellite links offer unparalled reliability”, 2013). Wel is een satellietverbinding gevoelig voor sneeuw en vorst omdat deze op de antenne afzetten en daardoor het signaal verstoren (“Can Weather Affect Satellite Internet?”, 2016). Een ander nadeel is dat de antenne installatie waarschijnlijk meer energie verbruikt dan bijvoorbeeld een 3G/GPRS verbinding. Tenslotte zijn de instapprijzen van satellietverbinding erg hoog (“Products & Services”, 2016). Dit maakt satellietverbinding slechts matig geschikt voor dit project.

3G/GPRS verbinding

Tijdens vooronderzoek is al vastgesteld dat er op en rond de Kilimanjaro berg voldoende 3G en GPRS dekking is[5]. Ook beschikt het basisstation over de benodigde hardware om een 3G of een GPRS verbinding te realiseren. Dit maakt deze oplossing uiterst geschikt voor het project.

4.2.2 Voorkomen van dataverlies tussen de basisstations en de centrale server

Ook tussen de basisstations en de centrale server mogen geen gegevens verloren gaan. Immers doet dat afbreuk aan de betrouwbaarheid van het systeem. Er zijn enkele scenario's opgesteld waarbij gegevens verloren kunnen gaan of corrupt kunnen raken.

4 Zie test 1 van hoofdstuk 4.1.2.
5 Zie hoofdstuk 2.5 van appendix A.

Netwerkstoringen

Doordat er gebruik wordt gemaakt van de 3G/GPRS netwerken van de telecomproviders in Tanzania, is de communicatie tussen de basisstations en de centrale server afhankelijk van een derde partij. Er kunnen daarom geen garanties worden gegeven over de betrouwbaarheid hiervan.

Afbeelding 23: Medewerkers van een Tanzaniaanse telecomprovider werken aan een GSM mast ("TTCL to enter mobile market by end of year", 2015)

Er wordt uitgegaan van een betrouwbaarheid die vele malen lager is dan providers in bijvoorbeeld Nederland bieden. Dit omdat volgens de opdrachtgever Afrikaanse telecomproviders een lager budget hebben en dus minder kwaliteit kunnen bieden. Afbeelding 23 bevestigt dit vermoeden.

Omdat er over een 3G/GPRS verbinding een IP verbinding kan worden opgezet, kan er tevens gebruik gemaakt worden van het *Transmission Control Protocol* (TCP). Dit protocol biedt garanties met betrekking tot de betrouwbaarheid van de gegevens die worden verzonden ("Transmission Control Protocol", 1981). Zo wordt gegarandeerd dat gegevens altijd in de juiste volgorde aankomen en er geen gaten ontstaan. Ook gebruikt TCP *Cyclic Redundancy Checks* (CRC) om datacorruptie te detecteren. TCP is in staat om, tot zekere hoogte, problemen zelfstandig op te lossen door de zender te verzoeken gegevens opnieuw te sturen. Dit maakt het protocol geschikt om de communicatie tussen de basisstations en de centrale server te realiseren.

Uitval door energievoorziening

Omdat het basisstation, net als de weerstations, op autonome energievoorziening draait en er dus niet gegarandeerd kan worden dat er ten alle tijden voldoende energie beschikbaar is, is het mogelijk dat het basisstation uitvalt. Uit analyse van de specificaties van het basisstation komt naar voren dat het basisstation voorzien is van een batterij die het basisstation korte tijd, ongeveer drie minuten, kan voorzien van stroom. Daarna zal het basisstation zichzelf *gracefully* uitschakelen.

Omdat het basisstation zichzelf *gracefully* uitschakelt en dus de programma's die draaien op het basisstation de tijd krijgen om af te sluiten, zijn deze in staat om de gegevens op te slaan zodat deze niet verloren gaan. Als later het basisstation zichzelf weer inschakelt kunnen deze gegevens weer worden uitgelezen en verder worden verwerkt. Naast de noodbatterij heeft het basisstation ook een *brown-out detector* waarmee verstoringen in de energielevering gedetecteerd kunnen worden.

Vastlopers en datacorruptie

Net als de weerstations zullen de basisstations last hebben van storingen die kunnen leiden tot vastlopers of datacorruptie. De gevolgen hiervan moeten tot een minimum worden beperkt.

De basisstations beschikken over een zogeheten "M2M Agent". Dit is een *deamon* die periodiek controleert of de gespecificeerde applicaties nog draaien. Indien dit niet het geval is, worden deze automatisch opnieuw opgestart. Dit garandeert dat de applicaties (bijna) ten alle tijden draaien. Ook beschikken de basisstations over een *watchdog timer* die, wanneer de embedded Linux *kernel* vastgelopen is, het gehele systeem reset.

Om datacorruptie tegen te gaan wordt de inhoud van de binnengekomen LoRaWAN pakketten bij binnenkomst voorzien van een CRC waarde. Wanneer de inhoud met het TCP protocol wordt verzonden naar de centrale server zal deze een eigen CRC genereren zodat ten alle tijden de inhoud beveiligd is tegen datacorruptie[6].

6 De functies van de Linux kernel ondersteunen geen CRC beveiligde data. Er kan dus geen absolute garantie worden gegeven op de beveiliging tegen datacorruptie.

4.2.3 Omgaan met langdurige netwerkstoringen

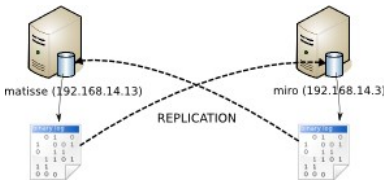
De verbinding tussen de centrale server en de basisstations wordt gerealiseerd over netwerken waarvan de *uptime* onzeker is. Door onder andere rolling black-outs, uitvallende routers, defecte kabels kan het netwerk over langere periode gestoord zijn. Doordat er slechts zeer beperkte voorzieningen op locatie aanwezig zijn is het niet mogelijk om eigen netwerken op te zetten of een andere netwerk provider te kiezen. Simpel gezegd zal er met deze instabiliteit gewerkt moeten worden.

Dit betekent dat het basisstation om moet kunnen gaan met een storing waardoor nieuwe meetgegevens niet naar de centrale server kunnen worden verzonden. Tegelijkertijd mogen deze gegevens niet verloren gaan. Dit betekent dat er een buffer moet worden aangelegd waar deze gegevens tijdelijk kunnen worden opgeslagen. Dit is op meerdere manieren te realiseren en de volgende zijn onderzocht:

- MySQL replication
- MySQL cluster
- CSV file
- SQLite

4.2.4 MySQL replication

MySQL replication is een feature van de MySQL DBMS om een *master* database te synchroniseren met meerdere s*lave* databases of door twee databases *bidirectional* te synchroniseren (zie afbeelding 24). Dit heet een master-master replication. Uit onderzoek blijkt dat MySQL replication slecht om kan gaan met netwerkstoringen: Wanneer de master een *record* met de zelfde *primairy key* aanmaakt de *slave* een fout geeft waarna de synchronisatie wordt beëindigd. Ook worden langdurige netwerk onderbrekingen niet altijd correct afgehandeld. Deze problemen spelen bij beide vormen van replication.

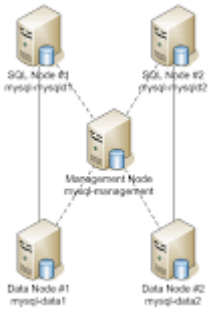


Afbeelding 24: Een MySQL master-master replication (“MySQL Master-Master Replication”, 2007)

Omdat er meerdere basisstations zijn en LoRaWAN pakketten door meerdere basisstations kunnen worden opgevangen is het mogelijk dat de records al bestaan. Dit maakt MySQL replication ongeschikt voor dit project.

4.2.5 MySQL cluster

MySQL cluster is een alternatieve versie van de MySQL DBMS. Hierbij worden gegevens over meerdere computers (hosts) verspreid om zo uitval van een host op te kunnen vangen (zie afbeelding 25). Ook kan de belasting van de database worden uitgespreid over meerdere hosts, wat de schaalbaarheid ten goede komt. In dit project zal echter de performance niet snel een belemmering vormen.



Afbeelding 25: Een MySQL cluster (“CentOS 6”, 2012)

Omdat de gegevens over meerdere computers verspreid worden, is het mogelijk om één computer op locatie te plaatsen en één computer in een datacenter te plaatsen. De computer in het datacenter zal altijd toegankelijk zijn voor de GUI en de computer op locatie zal altijd toegankelijk zijn voor de basisstations. Wanneer de verbinding verbroken is zullen de gegevens op beide zijden over tijd worden aangepast. Als de verbinding later dan weer wordt hersteld, zullen de aangepaste gegevens gemerged moeten worden.

Een MySQL cluster bestaat uit meerdere servers die onderling moeten communiceren. Dit maakt het netwerk relatief complex en dus minder geschikt. Ook is er onvoldoende geheugen op de basisstations aanwezig (“MySQL Memory Calculator”, 2015). Dit maakt deze oplossing ongeschikt voor dit project.

4.2.6 CSV file

Een CSV file is een zeer eenvoudige implementatie om gegevens langdurig op een hardeschijf op te slaan. Gegevens worden in een tabelvorm weggeschreven in een groot tekstbestand. Deze methode kent de volgende problemen:

- Datacorruptie kan snel ontstaan wanneer er fouten worden gemaakt bij het aanpassen van de CSV files. Wanneer er bijvoorbeeld records verwijderd of aangepast moeten worden kan dit snel leiden tot datacorruptie wanneer dit niet foutloos wordt gedaan. Omdat de software waarmee CSV bestanden worden bewerkt vaak zelf wordt geschreven is de kans daarop relatief groot[7].
- Concurrency, het gegeven dat meerdere processen hetzelfde bestand tegelijkertijd kunnen aanspreken wordt niet gegarandeerd afgevangen. Hiervoor moet vaak een eigen implementatie worden gerealiseerd.
- Het gegeven dat bepaalde tekens speciale betekenis hebben en dus niet in de inhoud mogen voorkomen of moeten worden geëscaped door bijvoorbeeld een backslash ervoor te plaatsen, leidt snel tot datacorruptie wanneer er hier fouten in worden gemaakt. Ook kan het voorkomen dat bepaalde data simpelweg niet opgeslagen kan worden.
- Lage prestaties die gepaard gaan met het gebruik van CSV bestanden kunnen problemen vormen wanneer de bestanden groter worden. Dit komt onder andere doordat er geen of moeilijk gebruik kan worden gemaakt van *indexes* zoals B+ trees of hashmaps, Deze indexes verhogen de prestaties aanzienlijk (Garcia-Molina, Ullman, & Widom, 2008).
- Non-atomaire aanpassen kunnen leiden tot inconsistentie in de gegevens wanneer bewerkingen slechts gedeeltelijk worden uitgevoerd. Dit kan voorkomen wanneer tijdens het uitvoeren van bewerkingen fouten optreden of het programma plotseling wordt afgesloten. Inconsistentie doet afbreuk aan de betrouwbaarheid van de gegevens en moeten ten alle tijden worden voorkomen (Garcia-Molina, Ullman, & Widom, 2008).

CSV biedt enkel een methode om gegevens op de hardeschrijf op te slaan, niet om deze gegevens te synchroniseren met de centrale server. Hier zou een eigen implementatie voor geschreven moeten worden. Al deze potentiële problemen doen afbreuk aan de betrouwbaarheid van CSV waardoor deze oplossing niet geschikt is voor dit project.

4.2.7 SQLite

SQLite lijkt erg veel op CSV files ware het niet dat het alle limitaties en problemen van CSV files oplost. Voor het schrijven in SQLite bestanden zijn zeer goed geteste *libraries* beschikbaar voor bijna alle platformen. Ook kan de *library* zelf speciale tekens escapen en biedt het mogelijkheden om te voorkomen dat meerdere processen tegelijkertijd hetzelfde bestand aanpassen. Door gebruik te maken van transacties worden complexe bewerkingen atomair en kunnen deze niet leiden tot inconsistentie.

Ook SQLite biedt geen methode om te synchroniseren met de centrale server. Hiervoor moet een eigen implementatie voor geschreven worden. Desondanks is SQLite de meest geschikte oplossing om gegevens langdurig te kunnen opslaan wanneer er een netwerkstoring zich voordoet.

7 Verscheidene fora draaiend op software die CSV of soortgelijke opslagmethoden gebruiken zijn door datacorruptie verloren gegaan (Tweakers.net 10 jaar: De hostinggeschiedenis 1998 – 2001, 2008).

4.2.8 Communiceren over een GPRS/3G verbinding

Omdat er, wegens gebrek aan een vaste internetaansluitingen op de Kilimanjaro berg, voor gekozen is om de verbinding tussen de basisstations en de centrale server door middel van GPRS/3G te implementeren, is het noodzakelijk dat deze verbinding wordt opgezet. De basisstations beschikken al over de hardware waarmee een SIM kaart kan worden uitgelezen en over een *transceiver* waarmee berichten kunnen worden verzonden. Wel moeten deze nog geconfigureerd worden.

Het opzetten van een GPRS/3G verbinding werkt iets anders dan een standaard internetaansluitingen omdat er ingelogd moet worden op de simkaart en daarna ook op het netwerk alvorens een verbinding tot stand kan worden gebracht (Access Point Name, 2016). Het inloggen op de simkaart en het netwerk alsmede het configureren van de hardware wordt grotendeels gedaan door software van de Linux omgeving op het basisstation. Tevens bewaakt de software de conditie van de verbinding door elke tien minuten te pingen. Indien er geen antwoord ontvangen wordt, wordt de verbinding opnieuw opgezet.

In verband met veiligheid worden de meetgegevens over een VPN verbinding verstuurd. Deze encrypt alle gegevens zodat bijvoorbeeld de netwerk operator deze niet kan afluisteren of manipuleren[8]. Door gegevens niet naar het fysieke IP-adres van de centrale server maar naar het virtuele IP-adres te sturen wordt er automatisch voor gezorgd dat de meetgegevens via de VPN verbinding in plaats van de standaard verbinding worden verstuurd.

Dataverbruik

Dataverbruik is bij GPRS/3G verbindingen een grote kostenpost. Tien megabyte aan dataverbruik kost al snel enkele euro's, zeker bij internationale providers (WorldSIM Data SIM Card. 2016). Om de kosten te beperken is er gekeken naar methoden om het dataverbruik te beperken.

De verbinding tussen de basisstations en de centrale server gebruikt op applicatie niveau JSON over HTTP. Dit is een eenvoudig en, in tegenstelling tot bijvoorbeeld XML, een compact formaat wat het geschikt maakt voor gebruik over GPRS/3G verbindingen.

Een veel gebruikte techniek in de *web development* om dataverbruik te verminderen, en daarmee de laadtijden te verbeteren, is het toepassen van geminificeerde bestanden. Bij geminificeerde bestanden zijn alle tekens die de leesbaarheid van de code ten goede komen maar geen invloed hebben op de daadwerkelijke code verwijderd. Hierbij moet worden gedacht aan *white-space*, commentaar en onnodige haakjes (Minification (programming). 2015). Ook worden bijvoorbeeld variabelenamen vervangen met kortere. Hierdoor worden de bestanden kleiner en worden ze sneller geladen. De snelheidswinst van geminificeerde Javascript is ongeveer 25% (How Does Reducing JavaScript Requests & Minifying JavaScript Impact Site Performance? 2013).

De verwachting is dat de effectiviteit van minificatie van JSON klein is: Omdat het protocol in beginsel redelijk compact is, in JSON over het algemeen geen commentaar aanwezig is en omdat variabelenamen niet aangepast kunnen worden (immers zal dan de definitie worden aangepast), zal minificatie alleen de *white-space* kunnen verwijderen. Omdat de gebruikte JSON *library* standaard ondersteuning biedt voor minificatie, is deze desondanks toegepast in de basisstations.

8 Zie hoofdstuk 4.5.2 voor meer informatie.

Naast minificatie is compressie ook een veel gebruikte techniek om dataverbruik tegen te gaan. De HTTP standaard biedt ondersteuning voor meerdere compressiestandaarden waaronder GZIP en Deflate (Leach, P. J. et al. 1999. Hypertext Transfer Protocol -- HTTP/1.1). Dit maakt de implementatie van compressie eenvoudiger deze kan worden toegepast zonder van de standaard af te wijken en omdat webservers deze compressiestandaarden vaak al standaard ondersteunen.

Plain-text formaten zoals XML en JSON zijn, in tegenstelling tot enkele binaire formaten, erg geschikt voor compressie: compressieratio's groter dan 7 zijn haalbaar. (Augeri, C. J. et al. 2007. An Analysis of XML Compression Efficiency) (Podlasek, T. et al. 2010. Efficiency of compression techniques in SOAP).

Naast significante reductie in dataverbruik biedt compressie ook andere voordelen. Doordat de grootte van het over te sturen bericht door het toepassen van compressie afneemt, neemt ook de zendtijd af. Hierdoor wordt de IO van een apparaat minder belast wat energie bespaart. Hier tegenover staat natuurlijk dat door het toepassen van compressie de CPU meer wordt belast. Toch bespaart compressie energie wanneer deze wordt toegepast op geschikte data (Chen, Y., et al. 2010. To Compress or Not to Compress - Compute vs. IO Tradeoffs for Mapreduce Energy Efficiency). Omdat de basisstations gegevens via GPRS/3G verzenden en deze *tranceivers* een hoger verbruik dan bijvoorbeeld WiFi *tranceivers* hebben, is de potentie voor energiebesparing significant groter. Zo is de energiebesparing door toepassing van compressie bij smartphones ruim vijf keer zo groot wanneer de data via 3G wordt verzonden dan wanneer de data via WiFI wordt verzonden (Gil, B., & Trezentos, P. 2011. Impacts of Data Interchange Formats on Energy Consumption and Performance in Smartphones).

4.3 Realisatie opslag van meetgegevens

Wanneer de meetgegevens eenmaal zijn aangekomen bij de centrale server moeten deze worden opgeslagen zodat later onder andere data-analyses kunnen worden uitgevoerd. Hiervoor moet de centrale server worden ingericht. In dit hoofdstuk wordt de inrichting van de server belicht op het gebied van software alsmede op het gebied van de database.

4.3.1 Inrichting van de server

Om te bepalen welke software op de server moet worden geïnstalleerd, is het essentieel om eerst vast te stellen welke functionaliteit de server moet bieden. Er is eerder vastgesteld dat er basisstations zijn die meetgegevens van weerstations aanleveren en er een interactieve GUI is waarmee eindgebruikers gegevens kunnen opvragen en eventuele beheerstaken kunnen uitvoeren.

Omdat de onderzoeksgroep op de Kilimanjaro hoogstwaarschijnlijk al een server gebruikt voor de opslag van de meetgegevens is het aannemelijk dat deze geïntegreerd moeten worden. Ook wordt er op het moment door collega's een webinterface en een bijhorende backend omgeving ontwikkeld voor het presenteren van meetgegevens van weerstations in Kenia. Tijdens de planningsfase van het project was onduidelijk hoe deze omgeving zou functioneren en of deze bijvoorbeeld aan de *functional requirements* zou voldoen. Ook was het onduidelijk hoe de integratie exact gerealiseerd moest worden.

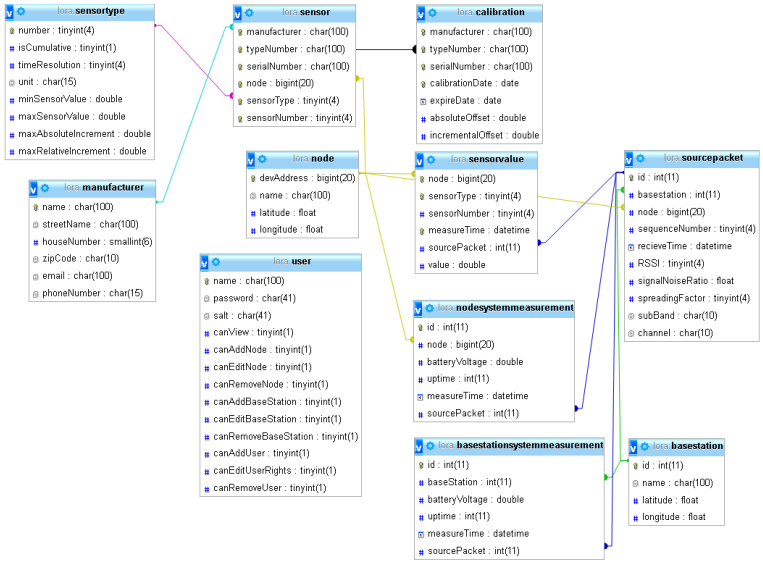
Om het project zonder vertraging te kunnen voortzetten is er gekozen om een eenvoudige en breed inzetbare server in te richten. Deze dient enkel om de *proof of concept* te realiseren en zal hoogstwaarschijnlijk (gedeeltelijk) vervangen worden in een later stadium. Tijdens aanvang van het project is door de opdrachtgever de wens geuit dat deze op zowel een Raspberry Pi moet kunnen draaien alsmede op een *cloud platform*. De reden dat de Raspberry Pi moet worden ondersteund is dat deze een kleine, goedkope, zuinige en relatief krachtige computer is. Daarvoor kan deze goed op locatie worden gebruikt. Immers zijn daar de voorzieningen beperkt. Effectief betekent dit dat er relatief lichte software die op veel platformen kan draaien moet worden gekozen.

Er is gekozen voor een *RESTful* webserver op een PHP op Apache platform. *RESTful* is een wijdgebruikte standaard en wordt bijvoorbeeld door KPN gebruikt voor Thingpark, de LoRaWAN implementatie van KPN (Lora Connect; Developer Manual. 2015). De PHP taal en de Apache webserver zijn beide zeer veel gebruikt en zijn daarmee erg geschikt voor gebruik op cloud platformen (Apache HTTP Server. 2016) (Usage Statistics and Market Share of PHP for Websites. 2016). Ook zijn beide beschikbaar voor Rasbian (een Linux variant die op de Raspberry Pi draait). Het feit dat PHP veel gebruikt wordt maakt software geschreven in deze taal makkelijker te onderhouden dan bijvoorbeeld een eigen in C++ geschreven webserver.

Nadat de gegevens zijn verwerkt op de centrale server dienen deze ook nog ergens te worden opgeslagen. Dit wordt in een database gedaan. Voor de centrale server is gekozen om MySQL als DBMS (DataBase Management System) te kiezen. MySQL is één van de meest gebruikte DBMS voor relationele databases wat het, net als Apache en PHP, geschikt maakt voor dit project (DB-Engines Ranking. 2016).

4.3.2 Inrichting van de database

De meetgegevens worden in een relationele database opgeslagen. Deze moet, net als iedere andere relationele database, een van te voren vastgestelde structuur hebben. Deze structuur wordt weergegeven in het *entity relation diagram* dat afbeelding 26 weergeeft.



Afbeelding 26: Het Entity Relation Diagram van de database van de centrale server

Om de juistheid van de database te kunnen aantonen kan worden gecontroleerd of de database aan de normaalvormen voldoet (Garcia-Molina, Ullman, & Widom, 2008). De database wordt tot de derde normaalvorm (3NF) gecontroleerd. Bij elke normaalvorm hoort een criterium waaraan de database moet voldoen. Indien de database aan de derde normaalvorm, en dus aan alle criteria, voldoet kan redelijkerwijs worden gesteld dat er geen fouten in het ontwerp van de database aanwezig zijn en dat de data zonder mogelijke inconsistentie kan worden opgeslagen en kan worden bewerkt.

Normaliseren met Functional Dependencies

Functional dependencies zijn één van de eigenschappen van data en hoe deze aanwezig zijn in een tabel is essentieel in het normalisatieproces van een database ontwerp. Er is sprake van een functionele dependency wanneer er geen verschillende waarden B mogelijk zijn voor een bepaalde waarde A (Garcia-Molina, Ullman, & Widom, 2008). In formulevorm wordt dit als volgt opgeschreven:

$$A_{1,}A_{2,}...A_{n} \rightarrow B_{1,}B_{2,}...B_{n}$$

De naam functional dependency wordt afgeleid uit het feit dat de waarde van B effectief te herleiden is uit de waarde van A. Immers kan er voor een bepaalde waarde van A altijd slechts één waarde B zijn. De functional dependency vormt een *superkey* wanneer de elementen in het linker en in het rechter lid alle attributen van de relatie (wetenschappelijke term voor een tabel) beschrijft. Een *superkey* is een *key* wanneer het linker lid minimalistisch is.

Met functionele dependencies kan worden gecontroleerd of het ontwerp in de derde normaalvorm is. De derde normaalvorm is als volgt gedefinieerd:

> *A relation R is in third normal form (3NF) if (…) for each non trivial FD, either the left side is a superkey, or the right side consists of prime attributes only* (Garcia-Molina, Ullman, & Widom, 2008)

Dus, door alle functional dependencies te achterhalen en te controleren of deze dan wel een *superkey* vormen of de attributen in het rechter lid allen onderdeel vormen van een *key* is te controleren of het ontwerp voldoet aan de derde normaalvorm.

4.4 Bepalen van eisen aan de autonome energievoorzieningen

Omdat de weerstations en de basisstations op de Kilimanjaro berg staan, waar geen energievoorzieningen zijn, moeten de weerstations en de basisstations zelf hun energie opwekken. Dit kan op verscheidene manieren waaronder het gebruiken van generatoren, windturbines of zonnepanelen.

Generatoren kunnen worden gebruikt voor het opwekken van elektriciteit wanneer er geen energievoorziening aanwezig is. Een verbrandingsmotor wordt gekoppeld aan een generator waardoor de kinetische energie van de verbrandingsmotor wordt omgezet naar elektriciteit (zie afbeelding 27). Omdat de verbrandingsmotor brandstof verbruikt moet deze periodiek worden bijgevuld. Dit betekent dat personeel periodiek naar de weerstations moet om deze bij te vullen. Dit ondermijnt het hele doel van dit project: het realiseren van draadloze autonome weerstations, die dus niet periodiek bezocht hoeven worden. Ook zijn de kosten van generatoren op langere termijn hoger en zijn deze minder betrouwbaar dan zonnepanelen (Kolhe M., Kolhe S., & Joshi, 2002).

Afbeelding 27: Een kleine generator met een vermogen van 2.8 kW (“Aggregaat Honda ECM 2800 - Open frame”, 2014)

Ook windturbines kunnen worden gebruikt voor het opwekken van elektriciteit. Deze worden vaak op grote schaal (windmolenparken met grote molens) ingezet om grotere gebieden van elektriciteit te kunnen voorzien of om de energievoorziening te verduurzamen. Voor het project zijn deze windturbines niet relevant aangezien het niet haalbaar is om windturbines van tientallen meters hoog te plaatsen.

Echter, er zijn ook kleine windturbines. Deze worden vaak op geringe hoogten (ongeveer tien meter) geplaatst (zie afbeelding 28). Deze zijn veel minder efficiënt omdat de wind op hogere hoogtes regelmatiger, langer en harder waait (Woofenden, 2008). Ook hebben windturbines op lagere hoogten last van onder andere bomen die de wind afvangen waardoor ze slechter presteren dan zonnepanelen (“Wind Turbines vs Solar Panels”, 2016).

Afbeelding 28: Een kleine windturbine op het dak van een woning; (Wind Turbines vs Solar Panels. 2016)

Zonnepanelen zetten elektriciteit direct om in elektriciteit door het fotovoltaïsche effect (“Solar Cells”, 2015). Zonnepanelen zijn uiterst betrouwbaar en gaan vaak langer dan 30 jaar mee (Skoczek, Sample, & Dunlop, 2009). Ook zijn ze goed op kleine schaal toe te passen (Taneja, Jeong, & Culler, 2008). Tenslotte zijn zonnepanelen ook op kleine schaal goedkoop te verkrijgen waar generatoren en windturbines een hogere instapprijs hebben (“Solar Vs Wind”, 2016).

4.4.1 Verwachte opbrengst van het zonnepaneel

Bij het bepalen van de eisen aan de autonome energievoorziening moet allereerst de opbrengst van een zonnepaneel bepaald worden. Hierbij moet rekening gehouden worden met het klimaat ter plaatse. Het aantal zonuren, de temperatuur en de breedtegraad hebben allen invloed op de (dagelijkse) opbrengst van een zonnepaneel.

Uit PVGIS, een online tool gepubliceerd door het Intitute for Energy and Transport (onderdeel van de Europese Commissie), blijkt dat de minimale gemiddelde dagelijkse instraling 3.16 kWh/m^2 is (zie afbeelding 29).

Het vermogen van een zonnepaneel wordt uitgedrukt in wattpiek (Wp). Dit is het vermogen wat een zonnepaneel opwerkt onder de standaard testcondities (STC). Bij de STC test wordt een zonnepaneel blootgesteld aan 1 kW/m^2 bij een temperatuur van 25 graden Celsius.

Fixed system: inclination=30°, orientation=157°				
Month	E_d	E_m	H_d	H_m
Jan	2.78	86.2	3.57	111
Feb	3.01	84.3	3.88	109
Mar	3.10	96.0	4.02	125
Apr	2.75	82.6	3.59	108
May	3.33	103	4.28	133
Jun	3.91	117	4.94	148
Jul	4.03	125	5.09	158
Aug	3.76	117	4.77	148
Sep	3.64	109	4.62	139
Oct	3.08	95.6	3.94	122
Nov	2.49	74.6	3.19	95.8
Dec	2.47	76.5	3.16	98.1
Yearly average	**3.20**	**97.2**	**4.09**	**124**
Total for year		**1170**		**1490**

Afbeelding 29: Verwachte opbrengsten op de Kilimanjaro berg (PVGIS)

Omdat er de minimale gemiddelde dagelijkse instraling 3.16 kWh/m^2 is, levert een zonnepaneel van 2 Wattpiek, in een omgevingstemperatuur van 25 graden Celsius, gemiddeld 6.32 Wattuur per dag op. Met een zonnepaneel van 60 Wattpiek zou dat resulteren in een gemiddelde van ongeveer 190 Wattuur per dag.

4.4.2 Verbruik van een weerstation

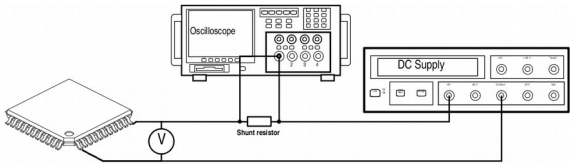
Omdat het weerstation op een autonome stroomvoorzieningen draait, bestaande uit een zonnepaneel en een accu, is het belangrijk om het verbruik van dit weerstation te bepalen. Dit heeft immers invloed op de eisen die gesteld worden aan het zonnepaneel en de accu. Om het verbruik te bepalen zijn enkele metingen verricht.

Het weerstation met een regensensor bestaat uit de volgende elektronica:

- De regensensor, bestaande uit een kiepbakje en een *Reed-contact*.
- Een LoRaWAN zendmodule om berichten te kunnen versturen.
- De microcontroller die sensoren aanstuurt, gegevens verzamelt en de zendmodule aanstuurt.

Omdat de regensensor uit een *Reed-contact* bestaat zal deze geen stroom verbruiken wanneer deze geen *tick* afgeeft. Dit omdat een *Reed-contact* een mechanische schakelaar is en er dus simpelweg geen gesloten circuit is. Wanneer er een *tick* wordt afgegeven zal er een gesloten circuit zijn, maar het verbruik ervan is afhankelijk van de digitale *IO pin* van de microcontroller. Bij de gebruikte microcontroller heeft deze een erg hoge weerstand waardoor het verbruik te verwaarlozen is.

Meetopstelling



Afbeelding 30: Schema van de meetopstelling

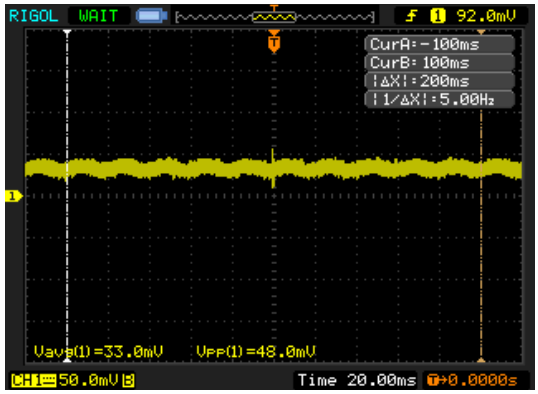
Omdat er hier gewerkt wordt met kleine stromen welke soms slechts enkele milliseconden aanwezig zijn, zijn conventionele meetmethoden, zoals multimeters en wattmeters, niet geschikt. Zij hebben simpelweg niet de resolutie en de meetsnelheid.
Daarom is er gekozen voor een opstelling waarbij een oscilloscoop de spanning over een *shunt resistor* meet (zie afbeelding 30). Uit deze spanning is zeer eenvoudig de stroomsterkte te herleiden met de wet van Ohm:

$$U = I \cdot R$$

Door de herleide stroomsterkte te vermenigvuldigen met de spanning die door de voltmeter gemeten wordt kan het vermogen worden vastgesteld. Door de duur van de verhoogde stroomsterkte van de oscilloscoop af te lezen kan kan ook de tijdsperiode van de handeling bepaald worden. Door het vermogen te vermenigvuldigen met de tijdsperiode kan de benodigde energie voor de handeling worden bepaald.

Verbruik microcontroller

De microcontroller heeft volgens de specificaties een verbruik van 3.37 mA wanneer deze actief is (Atmel, 2016). De gemeten waarde is 33.0 mA (zie afbeelding 31). Dit is aanzienlijk hoger dan de gespecificeerde waarde. Dit is te verklaren door het feit dat de meetopstelling het verbruik van de gehele *PCB* meet en niet enkel van de microcontroller zelf. Andere *ICs* zoals *LiPo-chargers* verbruiken ook elektriciteit. Ook is er een *voltage regulator* aanwezig. Verder wijkt de configuratie van de microcontroller af en worden niet exact dezelfde instructies uitgevoerd tijdens het meten.

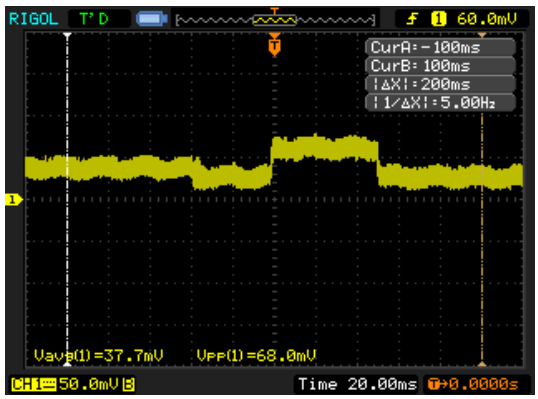


Afbeelding 31: Gemeten spanning over de shunt resistor wanneer de microcontroller actief is

Tijdens het meten was de *Vcc* spanning 3.72 V wat het verbruik van de microcontroller *PCB* op 123 mW brengt.

Verbruik LoRaWAN module

De LoRaWAN module verbruikt een bepaalde hoeveelheid energie per bericht dat moet worden verstuurd. Door het verbruik per bericht te meten kan het verbruik van de LoRaWAN module worden bepaald wanneer berichten op een bepaalde frequentie worden verzonden.

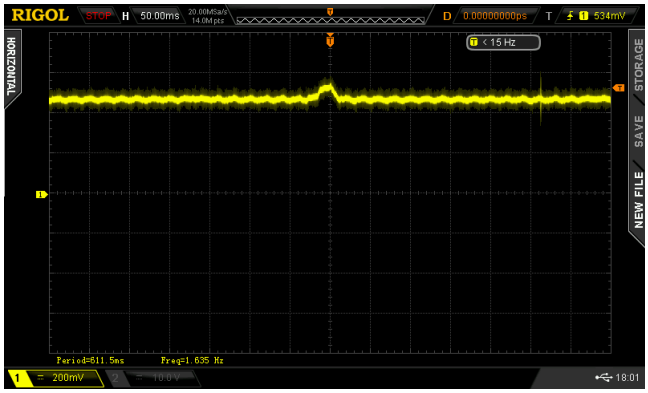


Afbeelding 32: Gemeten spanning over de shunt resistor wanneer de microcontroller een bericht zendt

Uit de meting blijkt dat het versturen van en pakket ongeveer 50 milliseconden duurt en gedurende die periode de stroomsterkte ongeveer 75 mA is (zie afbeelding 32). Dit is een toename van 42 mA. Met een spanning van 3.72 V komt de toename van het verbruik uit op 156 mW. Per bericht wordt er dus 7.80 J verbruikt.

4.4.3 Verbruik van een basisstation

Ook het basisstation moet mogelijk op autonome stroomvoorzieningen kunnen werken. Om de eisen aan deze stroomvoorzieningen vast te stellen is het noodzakelijk om het verbruik van het basisstation vast te stellen. Hiervoor is een zelfde soort meetopstelling gebruikt als voor de microcontroller en de LoRaWAN module.



Afbeelding 33: Gemeten spanning over de shunt resistor wanneer het basisstation een ACK bericht zendt

Wanneer het basisstation ingeschakeld is maar het modem, dat de *RF* signaal verwerking doet, uit staat is, het gemeten verbruik 1.1 W. Wanneer het modem ingeschakeld wordt is het verbruik 3.0 W. Het verbruik stijgt niet wanneer het basisstation berichten ontvangt en verwerkt zolang deze niet bevestigd hoeven te worden.

Wanneer het basisstation wel een bericht bevestigt, bedraagt het verbruik 3.57 W. Het verzenden van het bericht duurt ongeveer 25 milliseconden wat betekend dat 0.089 J wordt verbruikt (zie afbeelding 33). Dat het verbruik van het basisstation minder stijgt dan het verbruik van de LoRaWAN module komt waarschijnlijk doordat de *tranceiver* van het basisstation, in tegenstelling tot de LoRaWAN module, ook buiten de zendtijden ingeschakeld is.

Als laatste is er ook nog gekeken naar het verbruik van het basisstation wanneer de CPU van deze gestresst wordt. Hiervoor is een programma gestart met een oneindige loop. Het verbruik was toen ongeveer 4.05 W.

4.4.4 Benodigde batterijcapaciteit

Om de benodigde batterijcapaciteit te bepalen is het noodzakelijk om de maximaal te overbruggen periode te berekenen. Deze periode is voornamelijk afhankelijk van het klimaat op de Kilimanjaro berg.

Om de opbrengst van het zonnepaneel en daarmee de benodigde batterij capaciteit te bepalen is er gekozen om een eenvoudige simulatie te programmeren waarmee externe invloeden op het zonnepaneel, zoals sneeuw, temperatuur en bewolking, worden meegerekend.

Daglengte

De lengte van de dag bepaalt direct hoeveel zonlicht er op het zonnepaneel valt. Op bijvoorbeeld de noordpool is er geen dag meer in de winter, de zon komt in de winter niet boven de horizon. Dit komt doordat de noordpool erg ver van de evenaar is (“Position of the Sun”, 2016). Een zonnepaneel zal daar daarom ook niet tot nauwelijks energie leveren. Tanzania is erg dichtbij de evenaar en heeft hier minder last van. De lengte van een dag is daar 12:10 ± 20 min (“Sunrise and sunset times in Dar es Salaam”, 2016). In de simulatie wordt gewerkt met de gemeten zonnestraling wat al inherent rekening houdt met de daglengte.

Bewolking

Wanneer er bewolking is, reduceert de opbrengst vaak tot 20 procent (“Do solar panels work in cloudy weather?”, 2012). Dit is vaak niet genoeg om het systeem operationeel te houden en er zal dus aanspraak moeten worden gedaan op de batterij. De benodigde capaciteit is dus afhankelijk van de duur van de bewolking. Wederom wordt hier rekening mee gehouden door in de simulatie te werken met de gemeten zonnestraling die afhankelijk is van de aanwezigheid van bewolking.

Sneeuw

Wanneer sneeuw op een zonnepaneel valt zal deze bedekt worden en zolang deze sneeuw blijft liggen nauwelijks tot geen energie meer opwekken. Sneeuw zal echter niet altijd een zonnepaneel bedekken:

> *At ambient temperatures of -3°C or less, snow particles tend to hit a surface and bounce off, which leads to the snow sliding off a surface, providing that it is inclined at an adequate tilt angle. For example a roof with a tilt of 30° from horizontal is an adequate angle.* (“How Snow Affects Solar Panel Production”, 2015)

In de simulatie wordt er van uitgegaan dat zonnepanelen onder een hoek van 30 graden of meer staan om zo gebruik te kunnen maken van het bovenstaande gegeven.

Wanneer sneeuw wel op het zonnepaneel blijft liggen zal het daar blijven liggen totdat het gesmolten is. Hiervoor moet de sneeuw eerst verwarmd worden om vervolgens van fase te veranderen. Beide processen kosten energie: Het kost 2009 J om één kg één K te verwarmen en het kost maar liefst 33 kJ om één kg te laten smelten (Schuler, 2006).

De temperatuur van sneeuw is afhankelijk van de dikte ervan; dit is de zogeheten cold content (Schuler, 2006). Omdat de sneeuw in dit geval op een zonnepaneel ligt dat boven de grond bevestigd is en dus zowel van boven als van onder blootgesteld wordt aan warmte en omdat de dikte van de sneeuw onbekend is, wordt deze factor niet meegerekend in de simulatie.

Sneeuw smelt omdat het warmte opneemt. De snelheid van het opwarm proces is van veel factoren afhankelijk (Schuler, 2006). In de simulatie wordt de zonnestraling en de omgevingstemperatuur mee genomen. De invloeden hiervan op het smeltproces zijn beschreven in het Coup Model ("Empirical Melting/Freezing Function", 2015). Deze waarden zullen worden gebruikt in de simulatie.

Vuil

Wanneer zonnepanelen buiten in de open lucht staan, verzameld er na verloop van tijd vuil zoals zand op de panelen (zie afbeelding 34). Dit beïnvloedt de opbrengst van zonnepanelen naarmate er meer vuil op de panelen verzameld. Dit kan, afhankelijk van de gekozen hoek, oplopen tot 64%. De vorming van vuil kan worden terugbracht door onder andere regen en wind (Elminir et al., 2006).

In de simulatie wordt aangenomen dat bij een regenval van meer dan of gelijk aan 1.0 mm alle vervuiling van het zonnepaneel verwijderd is. Ook wordt er van uitgegaan de zonnepanelen onder een hoek van 30 graden staan en er 4 gram per vierkante meter per maand aan vuil neerslaat. Voor de verlies in opbrengst wordt de volgende formule aangehouden (Elminir et al., 2006).:

$$y = +0.0381 * x^4 + 0.8626 * x^3 - 6.4143 * x^2 + 15.051 * x - 16.769$$

Afbeelding 34: Vuil op een zonnepaneel (Solar Panel Cleaning, Clean Solar Panels, Solar Panel Maintenance. 2012)

Temperatuur

De temperatuur van het zonnepaneel heeft invloed op de opbrengst ervan. Vaak vermindert de efficiëntie naarmate het paneel warmer wordt. De exacte vermindering staat vaak gespecificeerd in de specificaties van het paneel.

De exacte temperatuur van het paneel is afhankelijk van meerdere factoren. Doordat het zonnepaneel in de zon staat en zonlicht opneemt wordt het warmer dan de omgeving. Tegelijkertijd zal het meer warmte afgeven aan de omgeving naarmate het warmer wordt. Dit zal resulteren in een balans tussen de opgenomen energie en de afgegeven energie wat uiteindelijk de temperatuur bepaalt. De hoeveelheid opgenomen en afgegeven energie hangt af van vele factoren zoals materiaal soort, luchtvochtigheid en de kleur van het paneel (Howell, Menguc, & Siegel, 2010) (Tsilingiris, 2008) ("Which colors absorb the most heat? Why is this? Does a bright color like yellow absorb a lot of heat?", 2015). Al deze factoren afwegen en meenemen in de simulatie valt buiten de scope van dit project. Er wordt voor de simulatie aangenomen dat de temperatuur 1 Kelvin per 100 Watt / m^2 boven de omgevingstemperatuur zal stijgen.

De simulatie

De simulatie zal per tijdsperiode in de weergegevens de opbrengst van de zonnepanelen berekenen en het verbruik bepalen. Het verschil tussen de twee wordt bepaald en de uitkomst wordt verrekend met de energie in de batterij.

Er worden twee waarden van de batterij gesimuleerd: De reële resterende energie en de theoretische resterende energie. Het verschil tussen de twee is dat de theoretische resterende energie negatief kan zijn, om zo het capaciteitstekort te kunnen bepalen. Met de reële resterende energie kan de *downtime* worden bepaald.

De simulatie stopt wanneer alle weergegevens verwerkt zijn. Aan het eind wordt een grafiek met daarop de theoretische en reële resterende energie van de batterij gegenereerd. Met deze grafiek kan worden bepaald of de configuratie voldoende presteerde en wat de down time van het systeem was.

Resultaten

Omdat de kwaliteit van de resultaten afhankelijk is van de gebruikte weergegevens, is gezocht naar goede bron. Zo is er gepoogd om de weergegevens van de onderzoeksgroep te verkrijgen. Dit is helaas niet gelukt.

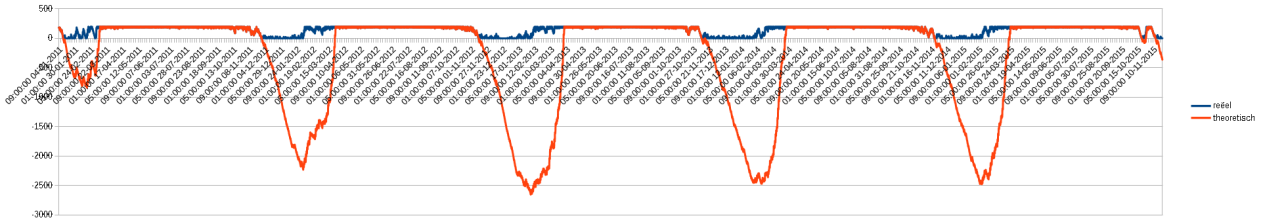
Ook is er gepoogd om de weergegevens te verkrijgen door weersvoorspellingen van het weer op de Kilimanjaro te verzamelen en te analyseren. Helaas konden deze niet worden gebruikt voor de simulatie omdat er te groten gaten in de weergegevens aanwezig waren. Hierdoor kon het voorkomen dat een nachtelijke meting werd geëxtrapoleerd over meerdere dagen waardoor alle energie in de batterij werd verbruikt. Dit is uiterst onrealistisch omdat er op een dag altijd zonlicht zal zijn. De resultaten met die weergegevens zijn daardoor onbruikbaar.

Daarna is gepoogd om de meetgegevens van een weerstation van de organisatie zelf die aan de voet van de Kilimanjaro berg staat te gebruiken. Echter, ook hier waren grote gaten in de weergegevens aanwezig waardoor de resultaten onbruikbaar zijn.

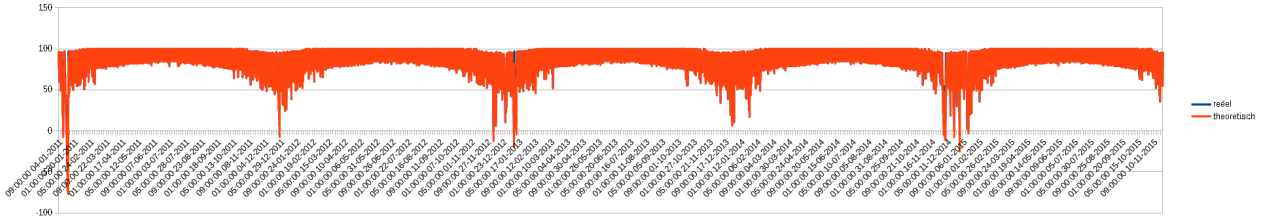
Daarom is besloten om de weergegevens vanaf 01-01-2011 van het weerstation in De Bilt te gebruiken. Uit PVGIS blijkt dat de omstandigheden op de Kilimanjaro berg gunstiger zijn dan de omstandigheden in De Bilt. Het is daarom aannemelijk dat wanneer er voldoende energie geleverd kan worden in De Bilt dit ook op de Kilimanjaro het geval zal zijn.

Basisstation

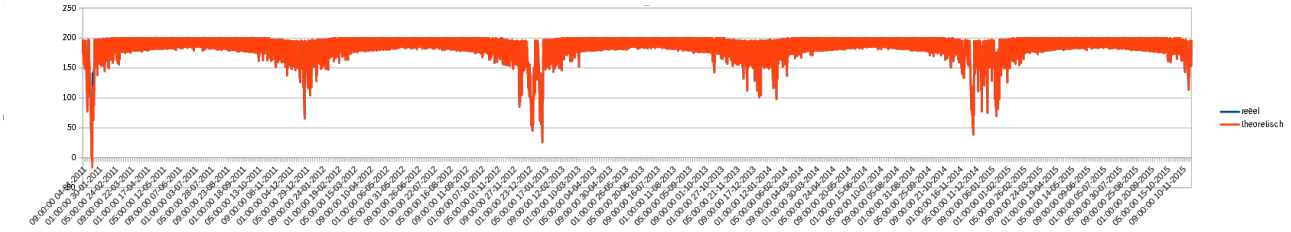
De reële en theoretische resterende energie zijn uitgezet in de onderstaande grafiek:



In de eerste scenario is een paneel van 60 Wattpiek en een accu van 100 Wattuur gebruikt. Het is duidelijk zichtbaar dat in de winter er te weinig energie wordt opgewekt en dat de accu nauwelijks wordt opgeladen. Daardoor loopt hij al snel leeg en valt het basisstation over langere tijd uit. Het tekort aan energie loopt ongeveer op tot 2500 Wattuur in de winter van 2012-2013. Ook is zichtbaar dat mocht er een accu zijn met voldoende capaciteit, in die winter het tot april zal duren totdat de accu weer volledig geladen is.



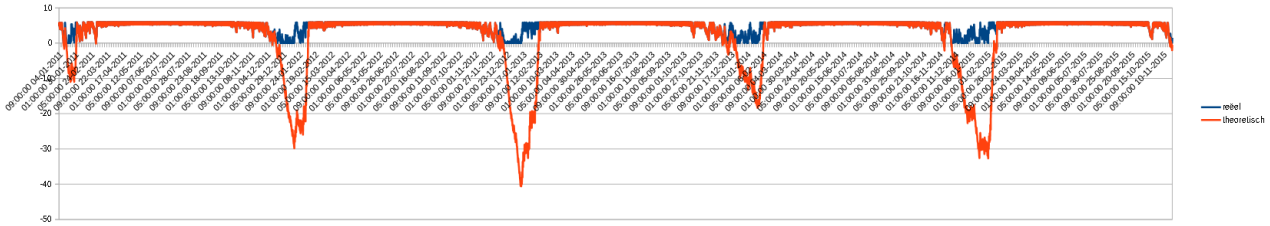
In de tweede scenario is een paneel van 240 Wattpiek en een accu van 100 Wattuur gebruikt. Het is duidelijk zichtbaar dat het zonnepaneel ook in de winter vaak er in slaagt om de accu weer (volledig) op te laden. Toch zijn er perioden waar er simpelweg geen tot nauwelijks zonnestraling is en de batterij toch volledig wordt ontladen, met down time als gevolg. De enige remedie hiertegen is een grotere batterij.



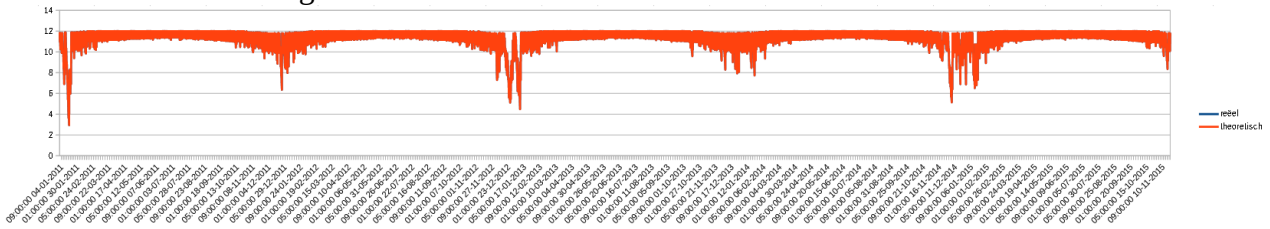
In de derde scenario is een paneel van 200 Wattpiek en een accu van 200 Wattuur gebruikt. Het is duidelijk zichtbaar dat er bijna geen *downtime* is. Enkel in januari 2011 is er sprake van een zeer korte *downtime*. Het is dan ook redelijk om te concluderen dat een zonnepaneel van 200 Wattpiek en een accu van 200 Wattuur voldoende is om het basisstation te voeden.

Weerstation

De reële en theoretische resterende energie zijn uitgezet in de onderstaande grafiek:



In de eerste scenario is een paneel van 4 Wattpiek en een accu van 6 Wattuur gebruikt. Het is zichtbaar dat er erg veel *downtime* is. Om de *downtime* te reduceren is zowel een groter paneel als een grotere accu benodigd. Toch kan worden gesteld dat een paneel van 4 Wattpiek en een accu van 6 Wattuur voldoende is. Dit omdat het verbruik van de microcontroller gemeten is terwijl deze actief was. Normaliter is de microcontroller slechts zeer zelden actief en slaapt deze. Hierdoor wordt het verbruik sterk gereduceerd.



In de tweede scenario is een paneel van 8 Wattpiek en een accu van 6 Wattuur gebruikt. Zoals te zien is het met deze configuratie zelfs mogelijk om de microcontroller te voeden wanneer deze continue actief is.

4.5 De betrouwbaarheid van de meetgegevens waarborgen

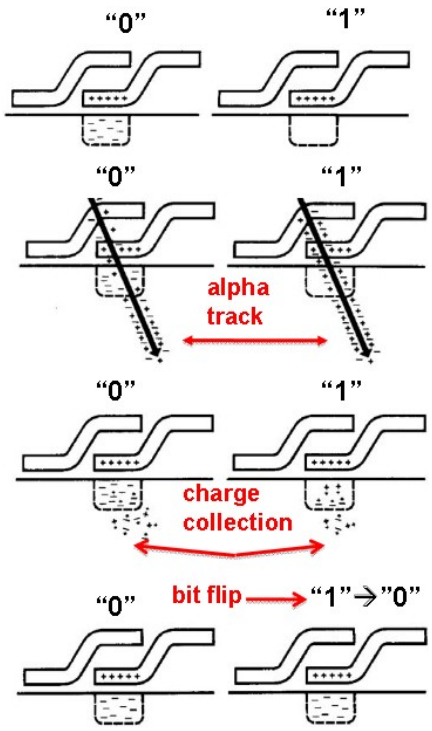
Omdat de meetgegevens van de weerstations voor wetenschappelijke doeleinden worden gebruikt is het essentieel dat deze betrouwbaar zijn. Immers kunnen foutieve gegevens leiden tot foutieve conclusies met alle mogelijke gevolgen van dien. Om de betrouwbaarheid te waarborgen is gekeken naar twee aspecten: datacorruptie en databeveiliging. Er is expliciet niet naar de betrouwbaarheid van sensoren gekeken daar dit buiten de scope van dit project valt.

4.5.1 Detectie van datacorruptie

Datacorruptie, een term voor data die niet langer de juiste waarde heeft, kan door meerdere zaken worden veroorzaakt. Vaak wordt datacorruptie veroorzaakt door fouten in de software waardoor gegevens onbedoeld (gedeeltelijk) worden overschreven. Dit kan worden bestreden door de software grondig te testen.

Ook defecte hardware of hardware dat buiten de operationele specificaties functioneert (bijvoorbeeld op een te lage voedingsspanning) kan datacorruptie veroorzaken. Dit kan worden bestreden door hardware van te voren te testen, defecte hardware te vervangen en door te voorkomen dat hardware buiten de operationele specificaties moet functioneren.

Echter, datacorruptie kan ook door externe bronnen worden veroorzaakt (Elliott, Mueller, Stoyanov, & Webster, 2013). Bij stabielere systemen zijn deze zogeheten *soft errors* de grootste oorzaak van systeemuitval (Schroeder & Gibson, 2010). Deze *soft errors* worden onder andere veroorzaakt door ruimtestraling (Ziegler & Lanford, 1979). De hoeveelheid straling waaraan men wordt blootgesteld neemt toe naarmate men een hogere hoogte opzoekt (O Sullivan, 2000). *Soft errors* kunnen in allerlei componenten van een microcontroller voorkomen, waaronder ook het SRAM geheugen (Nguyen, Yagil, Seifert, & Reitsma, 2005). In afbeelding 35 ontstaat een *soft error* door ruimte straling.



Afbeelding 35: Ruimtestraling veroorzaakt een soft error in het geheugen (Slayman, C. 2011. Soft Error Trends and Mitigation Techniques in Memory Devices)

Op normale hoogten ontstaan er bij SRAM 0.001 *soft errors* per bit per 10^9 uur (Slayman, 2011). Aangezien de gebruikte microcontroller 32 kibibyte geheugen heeft, zullen er 262 *soft errors* per 10^9 uur voorkomen. Dit betekent dat op normale hoogte ongeveer 1 *soft error* per 435 jaar zal voorkomen. Echter, de straling op de top van de Kilimanjaro is ongeveer 7,5 keer zo hoog ("Rads on a Plane: Hot Seats in First Class", 2015)). Dit betekent dat ongeveer elke 58 jaar per weerstation zich een *soft error* zal voordoen. Aangenomen dat er 65 weerstations op de berg staan die elk aan dezelfde straling worden blootgesteld zal er ongeveer elk jaar in een weerstation een *soft error* in het SRAM plaatsvinden.

Dit is een relatief laag aantal en zou als acceptabel gezien kunnen worden. Hierbij moet wel rekening gehouden worden met het feit dat ook in andere componenten van een microcontroller datacorruptie kan ontstaan. Oftewel, het aantal *soft errors* in het SRAM is de ondergrens, het kan enkel meer zijn. Tijdens het onderzoek is ook gezocht naar statistieken voor andere componenten. Deze zijn helaas niet gevonden. Er wordt aangenomen dat de kans aanzienlijk hoger is omdat tijdens bijvoorbeeld het bewerken van data deze over verscheidene databussen wordt getransporteerd. Door het grotere oppervlakte is de verwachting dat deze meer kans op *soft errors* hebben.

Immunity-aware programming

Er zijn verscheidene methoden om op hardware niveau de kans op *soft errors* te verkleinen. Deze hebben vaak betrekking op het *PCB* ontwerp, aanbrengen van filters en voldoende *shielding* (Carlton, Racino, & Suchyta, 2005). Het onderzoeken en eventueel toepassen van deze methoden vergt mechanische aanpassingen en valt buiten de scoop van dit project.

Ook op software niveau kunnen verscheidene aanpassingen worden gedaan om *soft errors* te detecteren en de gevolgen van *soft errors* te verkleinen door *immunity-aware programming* toe te passen (Atmel, 2007). Een van deze methoden is het valideren van waarden door middel van pariteit, *checksums*, *Cyclic Redundancy Check* of *Hamming Code*.

Hamming Code is, in tegenstelling tot alle andere codes, een code die in staat is om fouten zowel te detecteren als te repareren (van Moergestel, 2007). Dit betekend dat wanneer er door bijvoorbeeld een *soft error* datacorruptie is ontstaan deze ongedaan kan worden gemaakt. Dit voorkomt verlies van meetgegevens. Een nadeel van Hamming Code is dat het slechts data met enkele *soft errors* aankan (“Hamming Code”, 2016). Data met meerdere *soft errors* wordt mogelijk niet gedetecteerd door Hamming Code wat betekent dat er ongemerkt foutieve meetgegevens kunnen worden verzonden. Dit is niet wenselijk en daarom is Hamming Code ook niet geschikt.

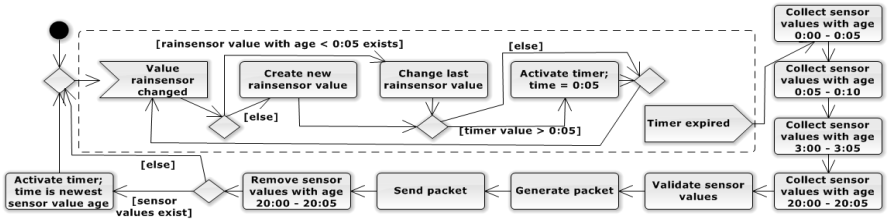
Een veel toegepaste *error detection code* is CRC. CRC staat voor *cyclic redundancy check* en gebruikt een polynoom die over het bericht wordt “heen geschoven”. Hier volgt een bepaald resultaat uit dat aan het oorspronkelijke bericht wordt toegevoegd. Wanneer de ontvanger dezelfde polynoom over het bericht heen schuift zal het resultaat 0 zijn wanneer alle bits correct zijn. Een ander antwoord geeft aan dat er ergens een bit is omgevallen (van Moergestel, 2007).

CRC biedt een hoge zekerheidsgarantie met betrekking tot foutdetectie. Een polynoom van 16 bits zal altijd een fout detecteren zolang er niet meer dan 16 bits achtereenvolgend fout zijn (van Moergestel, 2007). De kans dat dit gebeurt is uitermate klein en kan met de volgende formule worden berekend:

$$(errorBits/totalBits)^{errorStreak}$$

Hieruit blijkt dat de kans bij een bericht van 128 bits met 32 *soft errors* ongeveer 0.000000002% is en dus simpelweg verwaarloosbaar is.

De validatie van de meetgegevens kan kort voor het verzenden worden gedaan, zoals weergegeven in afbeelding 36.



Afbeelding 36: Activity Diagram van het weerstation met validatie

Berekeningsfouten

Immunity-aware programming schrijft naast het toepassen van CRC's ook voor om elke berekening te verifiëren. Dit kan worden gedaan door een berekening meerdere keren uit te voeren en de uitkomsten te vergelijken. Wanneer deze overeenkomen is het aannemelijk dat de berekening correct is. Dit heeft uiteraard wel als consequentie dat de prestaties significant achteruit gaan.

Om te voorkomen dat dit in de software op allerlei locaties geïmplementeerd moet worden, is er gekozen voor een zogeheten class template. In deze class template worden alle standaard berekeningen overschreven (*overriding*) met aangepaste berekeningen die de standaard berekening twee keer uitvoert en de uitkomsten vergelijkt (zie codefragment 1). Dit maakt het implementeren in de software relatief eenvoudig.

```
SecuredPrimitive<T> operator+(const T& y) {
  volatile T firstAttempt = x + y;
  validateCrc();
  T secondAttempt = x + y;
  SecuredPrimitive<T> output = secondAttempt;
  Assert(firstAttempt == secondAttempt);
  return output;
}
```

Codefragment 1: Een operator die controleert op berekeningsfouten

4.5.2 Preventie van manipulatie van data door derden

Beveiliging is bij elk computersysteem een essentieel onderwerp waar maar al te vaak onvoldoende aandacht aan wordt besteed. Met enige regelmaat worden grote beveiligingsgaten ontdekt en worden er persoonlijke gegevens, zoals creditcard-nummers, ontvreemd. Omdat er in dit project niet met persoonlijke of anderzijds gevoelige informatie wordt gewerkt, wordt er niet zozeer aandacht besteed aan het verborgen houden van informatie, maar meer aan het voorkomen dat deze informatie kan worden gemanipuleerd.

Beveiliging van componenten

De logische stap in beveiligen van een systeem is voorkomen dat er ingebroken kan worden op de apparatuur van het systeem. In dit project gaat het dan om de centrale server, de basisstations en de weerstations. De software configuratie van de centrale server is al eerder in hoofdstuk 4.3.1 beschreven en zal dus niet verder worden behandeld.

De basisstations draaien op een embedded computer met een volwaardige Linux omgeving. Deze omgeving is fysiek toegankelijk via een speciale terminal interface en op afstand toegankelijk via een *SSH interface*. Voor beide interfaces moet men inloggen met een wachtwoord alvorens er toegang wordt verleend. Door een wachtwoord te kiezen met voldoende lengte kan theoretisch worden gesteld dat het kraken van het wachtwoord zodanig lang duurt, dat het veilig is. Hierbij is de beveiliging wel volledig afhankelijk van de beheerder die een sterk wachtwoord moet gebruiken en deze geheim moet houden. Dit blijkt in de praktijk niet te werken:

> *Vorig jaar werd een database van softwaremaker Adobe gestolen, met daarin de privégegevens van 150 miljoen gebruikers en ongehashte wachtwoorden. Beveiligingsonderzoekers konden uit de database afleiden dat het wachtwoord '123456' onder Adobe-gebruikers het populairst was: dat wachtwoord werd maar liefst 1,9 miljoen keer gebruikt. Ook '123456789', 'password' en 'adobe123' werden honderdduizenden keren gebruikt.* (“De zwakte van wachtwoorden - Wat uitgelekte naaktfoto's ons leren”, 2014)

Er is daarom voor gekozen om beheer via een VPN verbinding uit te laten voeren. Om in te loggen op de interface zal eerst een verbinding via VPN moeten worden opgezet. Hiervoor moet de beheerder beschikken over de juiste sleutels. Daarna zal hij ook nog moeten inloggen op de SSH interface zelf.

De weerstations draaien op een zodanig platform dat softwarematige beveiliging geen rol speelt. Immers is er geen communicatie naar de weerstations mogelijk. Ook zullen kwaadwillenden ten alle tijden via een *programmer* het geheugen kunnen uitlezen of eventueel de chip kunnen herprogrammeren. Ook door de kleine rol van een weerstation binnen het geheel van het project, maakt een weerstation ook een veel minder interessant doelwit voor aanvallen. De enige manier om een weerstation te beveiligen is het gebruikmaken van mechanische beveiliging zoals met een sleutel te openen kast. Soortgelijke beveiliging valt buiten de scope van dit project en zal niet verder behandeld worden.

Beveiliging van verbindingen

Niet alleen componenten zelf, maar ook de verbindingen er tussen kunnen doelwit zijn van kwaadwillenden en moeten dus voldoende worden beveiligd. Deze verbindingen kunnen worden afgeluisterd waardoor informatie ontvreemd kan worden. Ook kan informatie door een kwaadwillende worden onderschept en kan deze gemanipuleerd worden, waardoor (kritische) processen kunnen worden verstoord (Whalen, Engle, & Romeo, 2001).

Verbinding tussen weerstations en basisstations

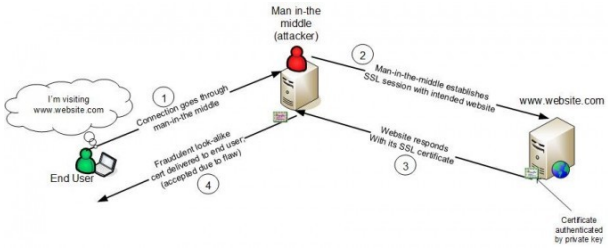
Bij de verbinding tussen de weerstations en de basisstations wordt gebruik gemaakt van LoRaWAN. De LoRaWAN standaard schrijft voor dat de inhoud van een LoRaWAN (het APP component) pakket geëncrypt dient te zijn met een 128 bits *AES Rijndeal* encryptiesleutel om afluisteren te voorkomen. De *AES Rijndeal* is een wijdgebruikte en erkende encryptietechniek die eenvoudig toegepast kan worden op hardware niveau (Satoh, Morioka, Takano, & Munetoh, 2001). De in het weerstation gebruikte LoRaWAN chip biedt de mogelijkheid om berichten hardwarematig te versleutelen waardoor het afluisteren van berichten onmogelijk wordt. De encryptiesleutels worden of van te voren verspreid of eenmalig over het netwerk verstuurd (Sornin, Luis, Eirich, Kramp, & Hersent, 2015).

Het encrypten van LoRaWAN berichten biedt geen bescherming tegen datamanipulatie. Ook al weet een kwaadwillende niet wat de inhoud is, door willekeurige bits te veranderen kan hij wel de inhoud wijzigen. Ook hier biedt de LoRaWAN standaard oplossing: Aan elk bericht wordt een *Message Integrity Code* (MIC) toegevoegd. Deze MIC kan gezien worden als een soort versleutelde CRC *checksum*. Wanneer de inhoud van een bericht gemanipuleerd is, valideert de MIC niet en zal het bericht worden genegeerd. Omdat de MIC een geheime sleutel gebruikt kan een kwaadwillende niet zelf zijn eigen MIC genereren (“The AES-CMAC Algorithm”, 2005).

Als laatste biedt de LoRaWAN standaard ook beperkte bescherming tegen *replay aanvallen* door elk bericht te voorzien van een volgnummer. Dit volgnummer kan 256 waarden aannemen en telt sequentieel op. Dit is een beperkt aantal berichten en werkt enkel in combinatie met het feit dat apparaten voorzien van LoRaWAN over het algemeen slechts enkele berichten op een dag zenden waardoor het opvangen van 256 berichten (waarna pas een succesvolle *replay aanvallen* kan worden uitgevoerd) alsnog enkele dagen tot weken in beslag neemt.

Verbindingen tussen basisstations en de centrale server

Ook de verbindingen tussen de basisstations en de centrale server moeten worden voorzien van de nodige beveiligingsmiddelen. Ook al biedt GPRS en 3G encryptie, de implementatie hiervan is afhankelijk van een derde partij (Vodafone, 2000). Er is daarom gekozen om de gegevens over een VPN tunnel tussen de basisstations en de centrale server te sturen. VPN encrypt en decrypt alle gegevens tussen de beide componenten zonder dat hiervoor op applicatieniveau voorzieningen voor moeten worden getroffen. Dit maakt implementatie relatief eenvoudig. Ook biedt VPN bescherming tegen *man in the middle aanvallen*. Dit is een aanval waarbij een kwaadwillende zich voordoet als één van de twee componenten, door gebruik te maken van certificaten (Pitts, 2003). Hierdoor kan hij alle communicatie onderscheppen en naar gelieve aanpassen (zie afbeelding 37).



Afbeelding 37: Een man in the middle attack bij een beveiligde website (“Apple’s SSL Bug”, 2014)

Verbinding tussen de centrale server en de GUI

De laatste verbinding, de verbinding tussen de centrale server en de GUI, benodigd ook beveiliging. Dit mede doordat gebruikers met de GUI van alle locaties verbinding kunnen maken met de centrale server en het dus niet bekend is met welk netwerk, en de staat ervan, zij dat doen. Mogelijk zijn deze netwerken erg vatbaar voor kwaadwillenden of mogelijk worden deze zelfs beheerd door kwaadwillenden. Hier moet rekening mee gehouden worden.

Allereerst moet de verbinding worden geëncrypt. Hiervoor is de HTTPS, die gebruik maakt van de SSL standaard, ontwikkeld. HTTPS biedt effectieve bescherming tegen afluisteren, maar niet tegen *man in the middle aanvallen* (Callegati, Cerroni, & Ramilli, 2009). M*an in the middle aanvallen* worden afgevangen door gebruik te maken van certificaten die zijn uitgegeven door gespecialiseerde bedrijven (zogeheten *Root Certificate Authorities*, CA). Het systeem werkt op het principe dat beide partijen een CA vertrouwen en certificaten die erdoor zijn ondertekend vertrouwen. Dit systeem blijkt in de praktijk niet waterdicht te zijn[9].

Voor de beveiliging van een verbinding met een *RESTful* server, wat de centrale server is, vormt deze tot zekere zin een dilemma: Een eigenschap van een *RESTful* server is dat deze *stateless* is. Er kan dus niet op worden ingelogd met een sessie die de server bijhoudt (“I want users to login into my RESTful API so only they can see (protected) resources. What is the correct way to do this?”, 2015). In plaats daarvan zal de cliënt zich bij ieder verzoek moeten authenticeren.

Daarvoor is gekozen voor een beveiliging gebaseerd op HMAC. HMAC is, net als CMAC, een *Message Integrity Code*. HMAC werkt, in tegenstelling tot CMAC, op *hashes* en geheime *salts* (“HMAC: Keyed-Hashing for Message Authentication”, 1997). Een bericht wordt (gedeeltelijk) gehashed met een geheime *salt*. Deze *hash* wordt toegevoegd aan het bericht en kan door de ontvanger worden geverifieerd door opnieuw het bericht met de geheime *salt* te *hashen*. Voor kwaadwillenden is het onmogelijk om een geldige *hash* te genereren voor een bericht omdat zij de *salt* niet kennen. HMAC biedt daarmee effectieve bescherming tegen datamanipulatie (“Salted Password Hashing - Doing it Right”, 2016).

De geheime *salt* die hierbij wordt gebruikt is een sha256 *hash* van het wachtwoord. Dit is ook de waarde die op de centrale server wordt opgeslagen. De centrale server kent dus niet het daadwerkelijke wachtwoord, wat de gevolgen van een eventuele database inbraak beperkt.

Naast datamanipulatie zijn ook *replay aanvallen* mogelijk. Hierbij onderschept een kwaadwillende alle berichten tussen de cliënt en de server om deze op een ander moment opnieuw naar de server te sturen (Mo & Sinopoli, 2009). Hierbij kan de kwaadwillende een eerdere authentieke handeling van een gebruiker herhalen. Dit kan een systeem verstoren en is onwenselijk. *Replay aanvallen* kunnen worden tegengewerkt door gebruik te maken van *session tokens* die voor slechts korte duur geldig zijn (Aura, 1997). Echter, doordat een *RESTful* server *stateless* dient te zijn, is het niet wenselijk om op de server een lijst met geldige *session tokens* bij te houden. Daarvoor is gekozen voor *session tokens* die zichzelf valideren (“CSRF protection with “self-validating” tokens”, 2010).

Afbeelding 38: Beveiligingscamera's zijn in films vaak vatbaar voor replay attacks. Hierdoor kunnen kwaadwillenden ongezien hun slag slaan. (Ocean's Eleven, 2001)

9 In 2010 is één van de grote CA in Nederland gehacked waardoor er certificaten in de handen van kwaadwillenden terecht zijn gekomen (DigiNotar Removal Follow Up, 2010)

5 Kwaliteitsbewaking

De kwaliteit van de verscheidene deelproducten is, zeker gezien het feit dat de meetgegevens voor wetenschappelijke doeleinden worden gebruikt, een belangrijk aspect. Tijdens de planningsfase is besloten om de kwaliteit van de deelproducten te bewaken door enkele tests te definiëren. Hierbij is de gedachte dat de deelproducten van voldoende kwaliteit zijn wanneer zij de tests doorstaan. De tests staan beschreven in hoofdstuk 3.5 van appendix A. Het ontwerp en de technische implementatie van de deelproducten staan respectievelijk in appendix D en E beschreven.

5.1 De GUI

Voor de GUI was geen functionele test gespecificeerd. Wel zijn alle mogelijke functies door middel van een functionele test getest. Zo zijn alle mogelijke handelingen in de GUI uitgevoerd en zijn deze uitgevoerd met invoer die lastig te verwerken is (zoals spaties die in URLs moeten worden geëncoded).

5.2 De centrale server

Tijdens het programmeren van de centrale server zijn *unit tests* gedefinieerd. Deze testen de werking van de software op zowel model niveau (door functies van classes aan te spreken) als op controller niveau (door webpagina's van de server aan te spreken). Hierbij wordt getest op corrupte invoer. Hiermee is dus voldaan aan de gespecificeerde functionele test. Wel moet rekening gehouden worden met het feit dat invoer die onrealistisch is niet perse corrupt is. Deze wordt dan ook niet afgevangen. Een voorbeeld hiervan is een locatie van een weerstation dat buiten bereik van alle basisstations is.

5.3 De basisstations

Voor de basisstations is gespecificeerd dat deze pakketten moet bufferen wanneer de netwerkverbinding wegvalt. Deze test is succesvol verlopen. Er is niet getest of dit ook onder extreme omstandigheden werkt omdat de code van het basisstation nog veel *debugging* bevat wat de prestaties van de code negatief beïnvloedt.

5.4 De weerstations

Voor de weerstations zijn naast twee functionele tests tijdens het programmeren ook enkele *unit tests* gedefinieerd. Deze *unit tests* testen onder andere de werking van de gebruikte circulaire buffer en tijdsberekeningen waarbij de correcte werking bij integer overflows wordt getest.

De functionele test om te bepalen of de weerstations correct datacorruptie detecteren is getest door de waarde van de variabele aan te passen en te controleren of datacorruptie werd gedetecteerd. Dit werkte ten alle tijden[10]. Er is besloten om de eis dat dit onder extreme omstandigheden wordt gegarandeerd los te laten. Bij de meeste variabelen die worden bewaakt wordt wel voldaan aan de eis omdat deze variabelen klein zijn. Maar bij de circulaire buffer waarin de meetgegevens worden bewaard, wordt niet aan de eis voldaan omdat dit teveel geheugen zal kosten. Echter, omdat er alsnog erg hoge kans is dat datacorruptie wordt gedetecteerd vormt dit in praktijk geen groot probleem[11].

De tests dat alle *ticks* van de regensensor gedetecteerd moeten worden is, in overleg met de opdrachtgever, niet uitgevoerd. Echter, doordat de weerstations gebruik maken van een *realtime operating system* is het zeer aannemelijk dat aan deze test wordt voldaan.

10 Er is zelfs een fout in de software gedetecteerd waarbij de CRC waarde niet correct werd geüpdate.
11 Zie hoofdstuk 4.5.1

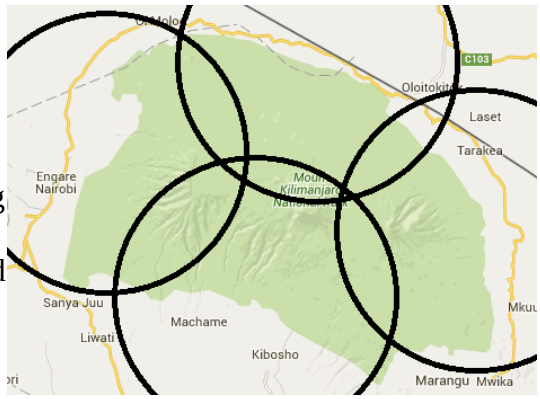
6 Conclusie en aanbevelingen

In dit hoofdstuk wordt aan de hand van de onderzoeksresultaten zoals beschreven in het vorige hoofdstuk een antwoord geformuleerd op de hoofdvraag waarna nog enkele aanbevelingen worden gedaan. De hoofdvraag luidt:

Hoe kunnen de gegevens van weerstations op de Kilimanjaro berg betrouwbaar en energie efficiënt uitgelezen worden door middel van draadloze communicatie en gepresenteerd worden in een interactieve gebruikersinterface zodat de gegevens geautomatiseerd en semi-realtime inzichtelijk zijn?

6.1 Conclusie

Het antwoord op deze vraag luidt dat het met LoRaWAN mogelijk is om weerstations op de Kilimanjaro berg geautomatiseerd uit te lezen. Door vier basisstations strategisch te plaatsen kan een volledig dekkend netwerk worden gerealiseerd wanneer wordt aangenomen dat het bereik van LoRaWAN twintig kilometer is (zie afbeelding 39). Wanneer aangenomen wordt dat het bereik vijftien kilometer is, is het niet mogelijk om een volledig dekkend netwerk te realiseren zonder basisstations op de berg te plaatsen. Wel kan er een netwerk worden gerealiseerd dat voldoende dekking biedt voor de huidige opstelling van de weerstations. Het daadwerkelijke bereik van LoRaWAN kon niet volledig worden vastgesteld. Wel is vastgesteld dat deze onder goede omstandigheden groter dan vijftien kilometer is.



Afbeelding 39: Een volledig dekkend netwerk met 4 basisstations met een bereik van 20 kilometer

Doordat LoRaWAN energiezuinig is en het verzenden van een bericht slechts 7.80 J kost, kunnen de weerstations autonoom functioneren op de Kilimanjaro. Omdat de microcontroller slechts 0.123 W verbruikt wanneer deze actief is, kan deze worden gevoed met een zonnepaneel van 4 Wp en een batterij van 6 Wh, zo blijkt uit de simulatie. Ook de basisstations, die een verbruik hebben van ongeveer 3 W, kunnen op autonome energievoorziening draaien wanneer men tenminste 200 Wh aan batterijcapaciteit en 200 Wp aan zonnepanelen installeert, zo blijkt uit de simulatie.

Door slim om te gaan met buffers in de weerstations en de basisstations, is er een balans tussen energieverbruik, dataverbruik en realtime prestaties. Hierdoor is het mogelijk om de sensoren *semi-realtime* uit te lezen: Omdat de meetgegevens door het weerstation vijf minuten worden gebufferd en door het basisstation 1 minuut worden gebufferd zullen onder normale omstandigheden de meetgegevens binnen tien minuten gepresenteerd worden aan de eindgebruiker. Hiermee wordt voldaan aan de *semi-realtime* eis. Echter, door externe storingen kan er geen *realtime* garantie worden afgegeven.

Betrouwbaarheid is ook uitgebreid onderzocht. Doordat de weerstations alle meetgegevens vier keer, uitgespreid over 20 uur, te zenden wordt ruim voldaan aan de functionele eis dat 99% van de meetgegevens moet aankomen. Doordat er tussen de basisstations en de centrale server gebruik wordt gemaakt van TCP over een VPN tunnel kan daar geen verlies van meetgegevens ontstaan.

Ook langdurige netwerkstoringen kunnen worden opgevangen door de gegevens in een SQLite database te bufferen. Ook de integriteit van de gegevens is een belangrijk onderdeel van de betrouwbaarheid, zeker omdat de meetgegevens voor wetenschappelijke doeleinden worden gebruikt. De integriteit van de gegevens kan worden gewaarborgd door deze te voorzien van *CRC checksums* en alle berekeningen twee maal uit te voeren. Alleen wanneer de uitkomst overeenkomt is de berekening correct uitgevoerd.

Beveiliging is bij elk computersysteem een essentieel onderwerp en kan grote gevolgen hebben. Daarom is er zowel naar de beveiliging van de componenten zelf als naar de verbindingen ertussen gekeken.

De basisstations zijn op afstand benaderbaar via een *SSH interface*. Deze is beveiligd met een wachtwoord. Omdat in praktijk blijkt dat een beveiliging met enkel een wachtwoord niet voldoende is, is gekozen om de *SSH interface* via een VPN verbinding aan te bieden. Dit biedt een betere beveiliging. Bij de weerstations speelt beveiliging bijna geen rol omdat deze niet van afstand benaderbaar zijn.. Bij de centrale server is gekozen om gebruik te maken van veel gebruikte en goed onderhouden software om zo de kans op een beveiligingsgat tot een minimum te beperken.

Omdat LoRaWAN gebruikt maakt van AES encryptie en de berichten tevens beveiligd met CMAC is de verbinding tussen de weerstations en de basisstations voldoende beschermd. De communicatie tussen de basisstations en de centrale server is voldoende beschermd omdat deze via een VPN verbinding loopt. Tussen de centrale server en de GUI is gebruik gemaakt van HTTPS. Echter, HTTPS biedt onvoldoende bescherming tegen *man in the middle aanvallen*. Door gebruikt te maken van HMAC en zelf validerende *sessions tokens* worden de effecten van *man in the middle aanvallen* en *replay aanvallen* tot een minimum beperkt.

6.2 Aanbevelingen

Aan de hand van de onderzoeksresultaten en de opgeleverde (deel)producten worden de volgende aanbevelingen gedaan:

6.2.1 Ondersteuning ADR en netwerkbeheer

Omdat de basisstations ontworpen zijn om met langdurige netwerkstoringen om te kunnen gaan functioneren zij zelfstandig. Voor het afhandelen van pakketten wordt niet direct gecommuniceerd met een centrale server. Dit heeft als gevolg dat netwerkbeheer niet mogelijk is.

Onder netwerkbeheer wordt het aansturen van de basisstations verstaan. Wanneer een pakket door meerdere basisstations wordt ontvangen moet slechts één antwoorden. Hiervoor is centrale aansturing noodzakelijk. Ook voor het uitvoeren van speciale *MAC commands* waaronder *Automatic Data Rate commands* (ADR) is centrale aansturing noodzakelijk. *Automatic Data Rate* maakt het mogelijk voor weerstations om automatisch hun zendvermogen te regelen. Dit biedt meer mogelijkheden om energie te besparen. Omdat de weerstations voorzien zijn van een zonnepaneel speelt dit een beperkte rol. Wanneer de software voor andere toepassingen wordt gebruikt, waar minder energie beschikbaar is, zal het praktisch nut toenemen.

6.3 Integratie in data-analyse systemen

De meetgegevens verder te verwerken moet de software worden geïntegreerd met externe systemen. Hierdoor kunnen de externe systemen weersvoorspellingen of andere analyses genereren. De uitkomsten van deze externe systemen kunnen mogelijk weer worden weergegeven in de ontwikkelde GUI.

7 Evaluatie onderzoeksproces

Tijdens het uitvoeren van het onderzoek hebben zich verscheidene noemenswaardige gebeurtenissen zich voorgedaan. Deze gebeurtenissen worden in dit hoofdstuk beschreven alsmede hoe deze zijn aangepakt.

7.1 Onvoldoende bereik

Tijdens het veldonderzoek bleek al snel dat het bereik van LoRaWAN bij lange na niet de geclaimde specificaties behaalde. Doordat uit het veldonderzoek bleek dat het bereik zo beperkt was, kon worden geconcludeerd dat het niet mogelijk was om draadloze langeafstands-communicatie te realiseren. Uit overleg met de opdrachtgever bleek dat dit niet een wenselijk onderzoeksresultaat was en dat er gekeken moest worden naar gunstigere omstandigheden.

Daarop is besloten om in een later stadia opnieuw veldonderzoek te verrichten en te kijken naar gunstigere omstandigheden. Hieruit bleek dat wanneer er op hogere hoogten, zoals daken, wordt gewerkt wel het gespecificeerde bereik wordt behaald. Met deze resultaten is vastgesteld onder welke omstandigheden er voldoende bereik is en welke consequenties dat heeft op de positionering van de weerstations. Zo kan er onder deze gunstigere omstandigheden alsnog draadloze langeafstands-communicatie gerealiseerd worden.

7.2 Planning

Tijdens het opstellen van het Plan van Aanpak is er voor gekozen om een zeer uitgebreide en gedetailleerde planning op te stellen[12]. Hierbij is het nooit de doelstelling geweest om exact deze planning aan te houden daar dit door externe factoren simpelweg niet mogelijk zal zijn. De planning wordt naast planningsdoeleinden ook gebruikt om overzicht te houden in de te verrichten werkzaamheden. Doordat de planning alle onderdelen van het project benoemd en daar een tijdsinschatting aan wordt gegeven is het mogelijk om een inschatting te maken tot hoeverre het project voor of achter op schema loopt. Door de voortgang per onderdeel bij te houden kan dit nog preciezer worden bepaald. Dit maakt het mogelijk om extra werk, zoals extra literatuuronderzoek, in te plannen om zo de beschikbare tijd beter te besteden.

Hierdoor is bijvoorbeeld er voor gekozen om het Plan van Aanpak aan te passen omdat volgens de planning het erg duidelijk werd dat de nog openstaande taken simpelweg te snel klaar zouden zijn en er dus tijd over zou blijven.

7.3 Literatuur van andere vakgebieden

Tijdens het onderzoek is veel gebruik gemaakt van literatuur van verscheidene vakgebieden. Dit leidde soms tot problemen omdat vakjargon en een typische manier van schrijven de tekst vaak moeilijk te begrijpen maakte.

Door literatuur te gebruiken van auteurs waarvan Engels niet hun eerste taal is kon vaak om de typische manier van schrijven heen gewerkt worden omdat deze auteurs simpelweg niet de taalvaardigheden hebben om zulke teksten te schrijven. Ook is er gezocht naar en gebruik gemaakt van literatuur die voldoende illustraties bevatte die het verhaal duidelijker maakten. Immers valt uit een illustratie veel sneller en eenvoudiger af te leiden wat de daadwerkelijke onderzoeksresultaten zijn dan uit een stuk tekst waarin deze worden beschreven.

12 Zie hoofdstuk 4.3 in Appendix A voor meer informatie.

In sommige gevallen is er voor gekozen on een bepaald onderwerp niet uit te werken. Zo is er bijvoorbeeld gezocht naar de theoretische voordelen van het gebruik van een richtantenne ten opzichte van een normale antenne. Hierbij moest worden gerekend met allerlei factoren. Deze waren vaak lastig te begrijpen en werden onvoldoende uitgelegd. Dit leidde tot antwoorden waarvan onvoldoende zekerheid bestond of deze antwoorden klopten. Daarom is besloten om deze niet op te nemen in het onderzoek.

7.4 Programmeerfouten in gebruikte libraries

Tijdens het programmeren is meermaals gebruik gemaakt van *libraries*. Het gebruik van *libraries* voorkomt dat functies opnieuw moeten worden geprogrammeerd en bespoedigt derhalve ook het programmeren. Bijkomend voordeel is dat *libraries* doorgaans goed gebruikt en getest worden en daarom weinig programmeerfouten bevatten. Echter, dit bleek voor enkele gebruikte *libraries* niet op te gaan:

Zo veroorzaakte de gebruikte r*ealtime operating system* in het weerstation dat interrupts van de externe interrupt controller niet langer werden geregistreerd. Deze fout werd, zo bleek na twee dagen *debuggen*, veroorzaakt door het feit dat de variabele, waarmee wordt aangegeven of er een *critical section* (een stuk code die niet door een interrupt onderbroken mag worden) wordt uitgevoerd, incorrect werd geïnitialiseerd. Dit gebeurde enkel wanneer er functies van deze *library* werden aangesproken in een declaratie van een globale of statische variabele. Deze programmeerfout is verholpen door de variabele correct te initialiseren alvorens er functies van de *library* worden aangesproken.

Een andere fout werd geconstateerd in de *library* waarmee het modem van het basisstation wordt uitgelezen. Om ontvangen berichten uit te lezen dient deze gepolled te worden. Tijdens het programmeren van het basisstation is besloten dat een interval van 100 milliseconden voldoende zou moeten zijn om ontvangen berichten tijdig uit te lezen. Echter, dit had als gevolg dat bijna geen enkel bericht nog werd ontvangen.

Omdat andere software die ook gebruik maakte van de *library* wel alle berichten ontving is er gekeken naar de verschillen tussen de twee. Hieruit bleek uiteindelijk dat de andere software een interval van 10 milliseconde hanteerde. Nadat de interval in de eigen software was aangepast werkte ook hier de *library* correct. Toch zou dit niet mogen: het gebruiken van een grotere interval zou niet als gevolg mogen hebben dat er berichten verloren gaan. Zeker niet als de berichtenbuffer niet vol zit. Omdat de gebruikte *library* relatief complex is en het probleem door de wijziging effectief verholpen is, is er voor gekozen om het daadwerkelijke probleem niet verder te onderzoeken.

Ook een andere *library* die gebruikt wordt in de basisstations bevat mogelijk een fout. Deze *library* wordt gebruikt voor het berekenen, en dus valideren, van de *message integrity codes* (MIC). Hierdoor worden de MICs incorrect berekend waardoor het onmogelijk is om datamanipulatie te detecteren. Ook al is dit een beveiligingslek, er zijn geen directe gevolgen voor de *proof of concept*. Daarom is besloten, na twee dagen debuggen, om deze fout een lage prioriteit toe te kennen en pas in een later stadium te verhelpen.

8 Referenties

3Km Wifi linktest van Mikrotik Omnitik naar SXT met hindernis. (2015). Geraadpleegd 1 maart 2016, van http://www.wirelessinfo.be/index.php/linktesten/pages/3km-obstr-omnitik-sxt.php

About SIGFOX. (2015). Geraadpleegd 1 maart 2016, van http://makers.sigfox.com/

Access Point Name. (2016, februari 19). In Wikipedia, the free encyclopedia. Geraadpleegd van https://en.wikipedia.org/w/index.php?title=Access_Point_Name&oldid=705771885

Aggregaat Honda ECM 2800 - Open frame. (2014). Geraadpleegd van https://wegenaer.nl/product/aggregaat/aggregaat-honda-ecm2800-frame-generator/

Apache HTTP Server. (2016, februari 29). In Wikipedia, the free encyclopedia. Geraadpleegd van https://en.wikipedia.org/w/index.php?title=Apache_HTTP_Server&oldid=707505951

Apple's SSL Bug: Another Man-in-the-Middle Attack. (2014, februari 22). Geraadpleegd 14 maart 2016, van https://blog.css-security.com/blog/apples-ssl-flaw-another-man-middle-attack

Atmel. (2007). Microcontroller in a Harsh Environment. Geraadpleegd van https://web.archive.org/web/20141222100142/http://www.atmel.com/images/doc9108.pdf

Atmel. (2016, januari). Atmel SAM D21E/SAMD21G/SAMD21J SMART ARM-Based Microcontroller DATASHEET. Geraadpleegd van http://www.atmel.com/images/atmel-42181-sam-d21_datasheet.pdf

Atmospheric noise. (2016, januari 5). In Wikipedia, the free encyclopedia. Geraadpleegd van https://en.wikipedia.org/w/index.php?title=Atmospheric_noise&oldid=698346751

Augeri, C. J., Bulutoglu, D. A., Mullins, B. E., Baldwin, R. O., & Baird, L. C., III. (2007). An Analysis of XML Compression Efficiency. In Proceedings of the 2007 Workshop on Experimental Computer Science. New York, NY, USA: ACM. http://doi.org/10.1145/1281700.1281707

Aura, T. (1997). Strategies against replay attacks. In Computer Security Foundations Workshop, 1997. Proceedings., 10th (pp. 59–68). http://doi.org/10.1109/CSFW.1997.596787

Callegati, F., Cerroni, W., & Ramilli, M. (2009). Man-in-the-Middle Attack to the HTTPS Protocol. IEEE Security & Privacy Magazine, 7(1), 78–81. http://doi.org/10.1109/MSP.2009.12

Can Weather Affect Satellite Internet? (2016). Geraadpleegd 11 maart 2016, van http://smallbusiness.chron.com/can-weather-affect-satellite-internet-26822.html

Carlton, R., Racino, G., & Suchyta, J. (2005). Improving the Transient Immunity Performance of Microcontroller-Based Applications. Geraadpleegd van http://www.nxp.com/files/microcontrollers/doc/app_note/AN2764.pdf

CentOS 6: Install MySQL Cluster – The Simple Way. (2012, november 22). Geraadpleegd van http://blog.secaserver.com/2012/11/centos-6-install-mysql-cluster-the-simple-way/

Chen, Y., Ganapathi, A., & Katz, R. H. (2010). To Compress or Not to Compress - Compute vs. IO Tradeoffs for Mapreduce Energy Efficiency. In Proceedings of the First ACM SIGCOMM Workshop on Green Networking (pp. 23–28). New York, NY, USA: ACM. http://doi.org/10.1145/1851290.1851296

Colour coded chart of number of storms per annum. (2015). Geraadpleegd van http://www.liveline.co.za/images/gfd-4.jpg

CSRF protection with "self-validating" tokens. (2010, april 17). Geraadpleegd 1 maart 2016, van http://crisp.tweakblogs.net/blog/3928/csrf-protection-with-self-validating-tokens.html

CSRF protection with "self-validating" tokens - Crisp's blog - Tweakblogs - Tweakers. (z.d.). Geraadpleegd van zotero://attachment/313/

DB-ENGINES. (2016). DB-Engines Ranking. Geraadpleegd 1 maart 2016, van http://db-engines.com/en/ranking

De zwakte van wachtwoorden - Wat uitgelekte naaktfoto's ons leren. (2014, september 3). Geraadpleegd 1 maart 2016, van http://tweakers.net/reviews/3687/wat-naaktselfies-ons-leren-over-wachtwoorden.html

Do solar panels work in cloudy weather? (2012, mei 30). Geraadpleegd van https://solarpowerrocks.com/solar-basics/how-do-solar-panels-work-in-cloudy-weather/

Electromagnetic interference. (2016, maart 12). In Wikipedia, the free encyclopedia. Geraadpleegd van https://en.wikipedia.org/w/index.php?title=Electromagnetic_interference&oldid=709682532

Elliott, J. J., Mueller, F., Stoyanov, M. K., & Webster, C. G. (2013). Quantifying the Impact of Single Bit Flips on Floating Point Arithmetic (No. ORNL/TM-2013/282). Oak Ridge National Laboratory (ORNL). Geraadpleegd van http://www.osti.gov/scitech/biblio/1089338

Elminir, H. K., Ghitas, A. E., Hamid, R. H., El-Hussainy, F., Beheary, M. M., & Abdel-Moneim, K. M. (2006). Effect of dust on the transparent cover of solar collectors. Energy Conversion and Management, 47(18–19), 3192–3203. http://doi.org/10.1016/j.enconman.2006.02.014

Empirical Melting/Freezing Function. (2015). Geraadpleegd 13 maart 2016, van http://www.coupmodel.com/default.htm?turl=WordDocuments%2Fempiricalmeltingfreezingfunction1.htm

Extreme Range Links: LoRa 868 / 900MHz SX1272 Module for Arduino, Waspmote and Raspberry Pi. (2015). Geraadpleegd 1 maart 2016, van https://www.cooking-hacks.com/documentation/tutorials/extreme-range-lora-sx1272-module-shield-arduino-raspberry-pi-intel-galileo/

Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). Database Systems: The Complete Book (2 edition). Upper Saddle River, N.J: Pearson.

Gil, B., & Trezentos, P. (2011). Impacts of Data Interchange Formats on Energy Consumption and Performance in Smartphones. In Proceedings of the 2011 Workshop on Open Source and Design of Communication (pp. 1–6). New York, NY, USA: ACM. http://doi.org/10.1145/2016716.2016718

Hamming code. (2016, februari 24). In Wikipedia, the free encyclopedia. Geraadpleegd van https://en.wikipedia.org/w/index.php?title=Hamming_code&oldid=706626307

HMAC: Keyed-Hashing for Message Authentication. (1997, februari). Geraadpleegd 1 maart 2016, van http://tools.ietf.org/html/rfc2104

Howell, J. R., Menguc, M. P., & Siegel, R. (2010). Thermal Radiation Heat Transfer, 5th Edition. CRC Press.

How Snow Affects Solar Panel Production. (2015, februari 19). Geraadpleegd 1 maart 2016, van http://www.oneroofenergy.com/snow/

Interference on AM and FM Radios. (2016). Geraadpleegd 1 maart 2016, van http://www.selfhelpandmore.com/interference/am-fm-radio-interference.php

I want users to login into my RESTful API so only they can see (protected) resources. What is the correct way to do this? (2015). Geraadpleegd 1 maart 2016, van http://restcookbook.com/Basics/loggingin/

Karar, V., Saini, S. S., Srinivas, M. S. N., & Kumar, V. (2003). Reducing EMI in CRT based display systems through grounding and shielding. In 8th International Conference on Electromagnetic Interference and Compatibility, 2003. INCEMIC 2003 (pp. 191–198). http://doi.org/10.1109/ICEMIC.2003.1287810

Kolhe, M., Kolhe, S., & Joshi, J. C. (2002). Economic viability of stand-alone solar photovoltaic system in comparison with diesel-powered system for India. Energy Economics, 24(2), 155–165. http://doi.org/10.1016/S0140-9883(01)00095-0

Körner, C. (1998). A re-assessment of high elevation treeline positions and their explanation. Oecologia, 115(4), 445–459. http://doi.org/10.1007/s004420050540

KPN. (2015). Lora Connect; Developer Manual. Geraadpleegd van http://zakelijke-community.kpn.com/etgtr99956/attachments/etgtr99956/lora_nieuws_zm/16/2/LoRa%20Connect%20-%20Developer%20Manual-%20v.0.9.1.pdf

Krider, E. P., Weidman, C. D., & Noggle, R. C. (1977). The electric fields produced by lightning stepped leaders. Journal of Geophysical Research, 82(6), 951–960. http://doi.org/10.1029/JC082i006p00951

Kuleshov, Y., Hoedt, G. D., Wright, W., & Brewster, A. (2001). Thunderstorm distribution and frequency in Australia.

Leach, P. J., Berners-Lee, T., Mogul, J. C., Masinter, L., Fielding, R. T., & Gettys, J. (1999, juli). Hypertext Transfer Protocol -- HTTP/1.1. Geraadpleegd 6 maart 2016, van https://tools.ietf.org/html/rfc2616#section-3.5

Lombardo, F. T., Main, J. A., & Simiu, E. (2009). Automated extraction and classification of thunderstorm and non-thunderstorm wind data for extreme-value analysis. Journal of Wind Engineering and Industrial Aerodynamics, 97(3–4), 120–131. http://doi.org/10.1016/j.jweia.2009.03.001

LoRa Modulation Basics. (2015, mei). Geraadpleegd van http://www.semtech.com/images/datasheet/an1200.22.pdf

Minification (programming). (2015, december 10). In Wikipedia, the free encyclopedia. Geraadpleegd van https://en.wikipedia.org/w/index.php?title=Minification_(programming)&oldid=694572777

Moergestel, L. J. M. van. (2007). Computersystemen en embedded systemen (2de ed.). Sdu Uitgevers BV.

Mount Kilimanjaro. (2016, februari 27). In Wikipedia, the free encyclopedia. Geraadpleegd van https://en.wikipedia.org/w/index.php?title=Mount_Kilimanjaro&oldid=707223229

Mount Kilimanjaro Climate Zones. (z.d.). Geraadpleegd van http://www.thomsontreks.com/kili-climate-zones/

Mo, Y., & Sinopoli, B. (2009). Secure control against replay attacks. In 47th Annual Allerton Conference on Communication, Control, and Computing, 2009. Allerton 2009 (pp. 911–918). http://doi.org/10.1109/ALLERTON.2009.5394956

MySQL Master-Master Replication. (2007, april 24). Geraadpleegd 14 maart 2016, van http://www.lefred.be/node/45

MySQL Memory Calculator. (2015). Geraadpleegd 1 maart 2016, van http://www.mysqlcalculator.com/

Nguyen, H. T., Yagil, Y., Seifert, N., & Reitsma, M. (2005). Chip-level soft error estimation method. IEEE Transactions on Device and Materials Reliability, 5(3), 365–381. http://doi.org/10.1109/TDMR.2005.858334

Nieuw tabblad. (z.d.). Geraadpleegd 13 maart 2016, van about:newtab

Nightingale, J. (2011, september 2). DigiNotar Removal Follow Up. Geraadpleegd 1 maart 2016, van https://blog.mozilla.org/security/2011/09/02/diginotar-removal-follow-up/

Norman, F. (2012, april 17). Solar Panel Cleaning, Clean Solar Panels, Solar Panel Maintenance. Geraadpleegd 6 maart 2016, van https://www.easyecoblog.com/225/solar-panel-maintenance-and-cleaning/

O Sullivan, D. (2000). Evaluation of the Cosmic Radiation Exposure of Aircraft Crew. Geraadpleegd van http://cordis.europa.eu/documents/documentlibrary/75331981EN6.pdf

Pitts, S. (2003, maart 17). An Overview of Digital Certificates and How They Are Used in VPN Authentication. Geraadpleegd van https://www.giac.org/paper/gsec/2711/overview-digital-certificates-vpn-authentication/104625

Podlasek, T., Sliwa, J., & Amanowicz, M. (2010). Efficiency of compression techniques in SOAP.

Position of the Sun. (2016, februari 20). In Wikipedia, the free encyclopedia. Geraadpleegd van https://en.wikipedia.org/w/index.php?title=Position_of_the_Sun&oldid=706001532

Products & Services. (2016). Geraadpleegd 11 maart 2016, van http://www.wafa.ae/en/vsat/products/hx/satellite-internet-hughes-hx50.aspx

Project Summary. (2010). Geraadpleegd 1 maart 2016, van https://www.kilimanjaro.biozentrum.uni-wuerzburg.de/project/Summary.aspx

Rads on a Plane: Hot Seats in First Class. (2015, juli 30). Geraadpleegd van http://news.spaceweather.com/category/cosmicrays/

Rockets, P. (2016). Rain Fade Calculations. Geraadpleegd van http://www.philsrockets.org.uk/Rain%20Fades.pdf

Salted Password Hashing - Doing it Right. (19-02-12016). Geraadpleegd 1 maart 2016, van https://crackstation.net/hashing-security.htm

Satellite Internet faster than advertised, but latency still awful. (2013, februari 15). Geraadpleegd 11 maart 2016, van http://arstechnica.com/information-technology/2013/02/satellite-internet-faster-than-advertised-but-latency-still-awful/

Satellite links offer unparalled reliability. (2013). Geraadpleegd 11 maart 2016, van http://www.vsat-systems.com/Education/Satellite-Internet-Explained/Performance/reliability/

Satoh, A., Morioka, S., Takano, K., & Munetoh, S. (2001). A Compact Rijndael Hardware Architecture with S-Box Optimization. In C. Boyd (Red.), Advances in Cryptology — ASIACRYPT 2001 (pp. 239–254). Springer Berlin Heidelberg. Geraadpleegd van http://link.springer.com/chapter/10.1007/3-540-45682-1_15

Schroeder, B., & Gibson, G. (2010). A Large-Scale Study of Failures in High-Performance Computing Systems. IEEE Transactions on Dependable and Secure Computing, 7(4), 337–350. http://doi.org/10.1109/TDSC.2009.4

Schuler, T. V. (2006, maart 21). Principles of snow melt. Geraadpleegd van http://www.uio.no/studier/emner/matnat/geofag/GEO4430/v06/undervisningsmateriale/geo4430_ebalanse_tvs.pdf

Skoczek, A., Sample, T., & Dunlop, E. D. (2009). The results of performance measurements of field-aged crystalline silicon photovoltaic modules. Progress in Photovoltaics: Research and Applications, 17(4), 227–240. http://doi.org/10.1002/pip.874

Slayman, C. (2011, januari). Soft Error Trends and Mitigation Techniques in Memory Devices. Geraadpleegd van http://www.opsalacarte.com/pdfs/Tech_Papers/Soft_Error_Trends_and_Mitigation_Techniques_in_Memory_Devices_Presentation_by_Charlie_Slayman,Opsalacarte.pdf

Solar Cells. (2015). Geraadpleegd 1 maart 2016, van http://www.chemistryexplained.com/Ru-Sp/Solar-Cells.html

Solar Vs Wind. (2016). Geraadpleegd 1 maart 2016, van http://www.iowawindandsolar.com/solar-vs-wind.html

Sornin, N., Luis, M., Eirich, T., Kramp, T., & Hersent, O. (2015). LoRa Specification. Geraadpleegd van https://www.lora-alliance.org/What-Is-LoRa/Technology

Study area. (2010). Geraadpleegd 14 maart 2016, van https://www.kilimanjaro.biozentrum.uni-wuerzburg.de/project/Studysite.aspx

Study Design. (2010). Geraadpleegd 1 maart 2016, van https://www.kilimanjaro.biozentrum.uni-wuerzburg.de/project/Methods.aspx

Sunrise and sunset times in Dar es Salaam. (2016). Geraadpleegd 1 maart 2016, van http://www.timeanddate.com/sun/tanzania/dar-es-salaam

SX1272/3/6/7/8; LoRa Modem. (2013, juni). Geraadpleegd van http://www.semtech.com/images/datasheet/LoraDesignGuide_STD.pdf

SXT 5. (2015). Geraadpleegd 14 maart 2016, van http://routerboard.com/RBSXT5HPnDr2

Taken, F. (2008, september 21). Tweakers.net 10 jaar: De hostinggeschiedenis 1998 - 2001. Geraadpleegd 6 maart 2016, van http://tweakers.net/reviews/994/tweakers-punt-net-10-jaar-de-hostinggeschiedenis-1998-2001.html

Tandy Radio Shack TRS-80 model I computer. (2013, december 10). Geraadpleegd 1 maart 2016, van http://www.oldcomputers.net/trs80i.html

Taneja, J., Jeong, J., & Culler, D. (2008). Design, Modeling, and Capacity Planning for Micro-solar Power Sensor Networks. In Proceedings of the 7th International Conference on Information Processing in Sensor Networks (pp. 407–418). Washington, DC, USA: IEEE Computer Society. http://doi.org/10.1109/IPSN.2008.67

The AES-CMAC Algorithm. (2005, juni). Geraadpleegd 1 maart 2016, van https://tools.ietf.org/html/rfc4493

Thunderstorms: The “Stormiest” Places in The U.S.A. and the World. (2012, juni 21). Geraadpleegd 1 maart 2016, van http://www.wunderground.com/blog/weatherhistorian/thunderstorms-the-stormiest-places-in-the-usa-and-the-world

Transmission Control Protocol. (1981, september). Geraadpleegd 1 maart 2016, van https://tools.ietf.org/html/rfc793#page-9

Tsilingiris, P. T. (2008). Thermophysical and transport properties of humid air at temperature range between 0 and 100 °C. Energy Conversion and Management, 49(5), 1098–1110. http://doi.org/10.1016/j.enconman.2007.09.015

TTCL to enter mobile market by end of year. (2015, maart 7). Geraadpleegd 14 maart 2016, van http://www.www.theeastafrican.co.ke/business/TTCL-to-enter-mobile-market-by-end-of-year/-/2560/2645400/-/fsry7pz/-/index.html

Verticals. (2015). Geraadpleegd 1 maart 2016, van http://www.m2m4all.com/?page_id=1223

Vodafone. (2000, februari 23). GPRS encryption. Geraadpleegd van ftp://www.3gpp.org/tsg_sa/WG3_Security/TSGS3_11_Mainz/Docs/PDF/S3-000201.pdf

W3 Techs. (2016). Usage Statistics and Market Share of PHP for Websites. Geraadpleegd 1 maart 2016, van http://w3techs.com/technologies/details/pl-php/all/all

Walker, H. R. (2014). Ultra Narrow Band Modulation Textbook. Geraadpleegd van http://www.vmsk.org/Textbook.pdf

Watson, A. I., & Holle, R. L. (1996). An Eight-Year Lightning Climatology of the Southeast United States Prepared for the 1996 Summer Olympics. Bulletin of the American Meteorological Society, 77(5), 883–890. http://doi.org/10.1175/1520-0477(1996)077<0883:AEYLCO>2.0.CO;2

Weil, A. (2013, januari 16). How Does Reducing JavaScript Requests & Minifying JavaScript Impact Site Performance? Geraadpleegd van http://www.yottaa.com/company/blog/application-optimization/how-does-reducing-javascript-requests-minifying-javascript/

Whalen, S., Engle, S., & Romeo, D. (2001, april). An introduction to ARP spoofing. Geraadpleegd van http://www.leetupload.com/database/Misc/Papers/arp_spoofing_slides.pdf

Which colors absorb the most heat? Why is this? Does a bright color like yellow absorb a lot of heat? (2015). Geraadpleegd 1 maart 2016, van http://scienceline.ucsb.edu/getkey.php?key=1464

Wind Turbines vs Solar Panels. (2015). Geraadpleegd 1 maart 2016, van http://solarelectricityhandbook.com/Solar-Articles/wind-turbines.html

Woofenden, I. (2008). Wind Power Curves; Whats Wrong, Whats Better. Geraadpleegd van https://web.archive.org/web/20081209175049/http://www.abundantre.com/windpowercurves.pdf

WORLDSIM. (2016). WorldSIM Data SIM Card. Geraadpleegd 6 maart 2016, van https://www.worldsim.com/data-sim-card?___store=eu

Ziegler, J. F., & Lanford, W. A. (1979). Effect of Cosmic Rays on Computer Memories. Science, 206(4420), 776–788. http://doi.org/10.1126/science.206.4420.776

Appendix A – Plan van Aanpak

Draadloze Langeafstands-communicatie Voor Weerstations

Plan van Aanpak

Draadloze Langeafstands-communicatie Voor Weerstations

Plan van Aanpak

Cover afbeelding:
Kilimanjaro berg

Auteur:
Tommas Bakker

Studentennummer:
1611239

Versie:
1.5

Datum:
11-11-2015

Mentors:
Jan Willen Smeenk, M2M4ALL
Bedrijfsbegeleider

Examinators:
Henk van Nimwegen, Hogeschool Utrecht
Eerste examinator

Jan Zuurbier, Hogeschool Utrecht
Tweede examinator

Inhoudsopgave

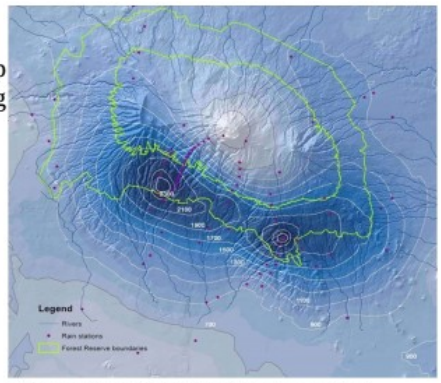
1Inleiding....4
2Project context....5
2.1De organisatie....5
2.2Bedrijfsbegeleider....5
2.3Overige medewerkers....5
2.4Theoretisch kader....5
2.5Kilimanjaro berg....6
Netwerkdekking....6
3Project definitie....7
3.1De opdracht....7
3.2Onderzoeksvraag....8
3.3Afbakening....9
3.4Functionele requirements....9
3.5Tests....10
3.6Onderzoeksmethoden....11
3.7(Deel)producten....12
3.8Risico's....13
4Project beheer....14
4.1Ontwikkelmethode....14
4.2Milestones....14
4.3Planning....15
5Referenties....17
6Appendix A – Axiomatic Design Matrix....18
7Appendix B – Gantt Chart....19

1 Inleiding

Weerstations worden al lang door bijvoorbeeld het KNMI gebruikt om het weer op een bepaalde locatie te meten. Dit wordt gedaan door bijvoorbeeld de neerslag, de luchtvochtigheid en de luchtdruk te meten. Deze gegevens worden verzameld door de sensoren af te lezen en met deze gegevens kan, door middel van een complex rekenmodel, een weersvoorspelling worden gedaan. Deze weersvoorspelling wordt vervolgens bijvoorbeeld via het journaal gepubliceerd wat de bevolking de kans geeft om hun dag aan te passen aan het weer. Dit geeft maatschappelijke en economische voordelen.

Weerstations kunnen ook voor andere doeleinden dan weersvoorspellingen worden gebruikt: Ze kunnen ook worden gebruikt voor onderzoek naar klimaat of verandering daarin. Zo is het mogelijk om de effecten van het broeikaseffect te analyseren of om de effecten van de aanwezigheid van mensen te analyseren (DFG Research Group Kilimanjaro, Project Summary).

In een dicht bevolkt en begaanbaar gebied zoals Nederland is het relatief eenvoudig om weerstations te plaatsen, te voorzien van elektriciteit, te onderhouden en uit te lezen. Op andere locaties is dit een ander verhaal. De Kilimanjaro berg is een berg in Tanzania, Afrika met een hoogte van 4877 meter (*Wikipedia, the free encyclopedia*. Mount Kilimanjaro). Op deze berg zijn 65 weerstations geplaatst voor een onderzoek naar het klimaat op deze berg (DFG Research Group Kilimanjaro, Study Design). Elektriciteitsvoorzieningen en netwerkinfrastructuur ontbreken en ook begaanbare wegen zijn schaars. De weerstations kunnen alleen worden uitgelezen door te voet er naar toe te lopen, waardoor het uitlezen van sommige weerstations enkele dagen kan duren. Ook is het onderhouden en frequent uitlezen van deze weerstations daarom lastig.



Afbeelding 1: De Kilimanjaro berg, Tanzania, Afrika. De paarse stippen representeren weerstations

M2M4ALL biedt hiervoor een oplossing. Het levert onderhoudsarme weerstations die autonoom op duurzame energiebronnen zoals zonnestroom kunnen draaien (M2M4ALL, Verticals). Een manier om de gegevens van een weerstation naar een basisstation te transporteren is er echter nog niet. Omdat het aanleggen van kabels op dit terrein niet haalbaar is, is een draadloze communicatie methode de enige oplossing. Maar omdat de energievoorziening zeer beperkt is en over langere afstanden moet worden gecommuniceerd zijn slechts een paar standaarden mogelijk geschikt.

Het draadloos uitlezen van weerstations biedt verscheidene voordelen: Zo bespaart het kosten omdat er niet meer met de hand hoeft te worden uitgelezen. Omdat ze niet meer met de hand uitgelezen hoeven te worden kunnen de weerstations ook flexibeler worden ingezet. Zo kunnen ze op moeilijkere bereikbare locaties worden geplaatst. Verder maakt draadloze communicatie het mogelijk om semi-realtime informatie van deze weerstations te verwerken. Dit maakt het mogelijk om sneller onderzoek te doen en doordat de gegevens semi-realtime beschikbaar zijn kunnen de weerstations bijvoorbeeld worden gebruikt voor regelsystemen van stuwdammen of alarmsystemen die waarschuwen voor verhoogde kans op modderlawines.

2 Project context

Dit hoofdstuk zal ingaan op de omgeving waarin het project wordt uitgevoerd en andere externe factoren waar het project mee te maken krijgt.

2.1 De organisatie

De organisatie heet M2M4ALL wat vertaald “machine naar machine voor iedereen” betekend. Met machine naar machine wordt M2M communicatie bedoelt. Het bedrijf is op 01-09-2013 opgericht ent telt elf medewerkers. Het is gevestigd in Hilversum en houdt zich voornamelijk bezig met ontwikkelen van embedded hardware en software, consultancy en training.

2.2 Bedrijfsbegeleider

De bedrijfsbegeleider zal Jan Willem Smeenk worden. Hij heeft voldoende kennis om deze taak te kunnen volbrengen, zo bleek uit het sollicitatiegesprek. Zo had hij kennis van Raspberry Pies en Banana Pies en de verschillen ertussen. Ook had hij kennis van Arduino en PIC micro-controllers. Verder had hij kennis over hardware en de werking daarvan en heeft hij kennis over en ervaring met het opzetten van draadloze netwerken.

De bedrijfsbegeleider zal de functie van opdrachtgever vervullen en zal tevens milestones en deelproducten beoordelen en daar eventueel feedback op geven.

Naam:	Jan Willem Smeenk	Email adres:	janwillem@m2m4all.com
Telefoonnummer:	06-81517710		

2.3 Overige medewerkers

Bij M2M4ALL werken ook andere medewerkers die ondersteuning kunnen bieden tijdens de uitvoering van dit project. De meeste werknemers zijn elektrotechnici en hebben bijvoorbeeld kennis van PCB ontwerp, radiotechniek en solderen. Twee werknemers hebben al met de netwerktechnologie LoRaWAN gewerkt en daar al enige ervaring in opgedaan.

Er is één werknemer die programmeur is. Hij zou bijvoorbeeld ontwerpen kunnen toetsen om zo de kwaliteit te verbeteren.

2.4 Theoretisch kader

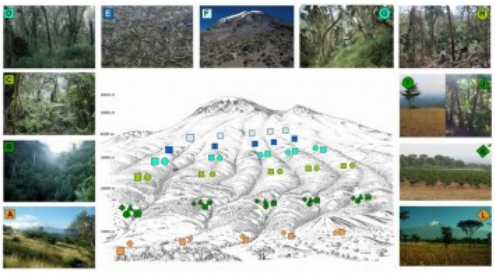
De volgende informatiebronnen behoren tot het theoretische kader van dit project:

- LoRaWAN Alliance. (2015). LoRaWAN specification. Van https://www.lora-alliance.org/For-Developers/LoRaWANDevelopers
- Atmel. (2015). ATmega1284P Datasheet. Van http://www.atmel.com/images/doc8059.pdf
- Semtech Kerlink. (2015). Long Range IoT Station. Van http://wikikerlink.fr/
- Rolf H. Weber, Romana Weber. (2010). Internet of Things.
- Wu Geng, S. Talwar, K. Johnsson. (2011). M2M: From mobile to embedded internet.
- Robert Davis, Alan Burns. (2005). Hierarchical Fixed Priority Pre-emptive Scheduling.

2.5 Kilimanjaro berg

De Kilimanjaro berg is een berg met drie vulkanische kraters. Het hoogste punt ligt 4877 meter en is daarmee de hoogste berg in Afrika. De berg wordt door veel toeristen beklommen en er wordt ook onderzoek naar klimaat op de berg gedaan. Dit omdat er op de berg zich allerlei klimaatverschijnselen voordoen (*Wikipedia, the free encyclopedia*. Mount Kilimanjaro).

Het KiLi project, dat onderzoek doet naar het klimaat op de berg, is in februari 2010 gestart en bestaat uit meerdere deelprojecten (DFG Research Group Kilimanjaro, Project Summary). Deze doen allen onderzoek naar klimaat en ecosystemen op de Kilimanjaro berg. Hiervoor worden meerdere sensoren gebruikt, waaronder regensensoren, om het klimaat en de ecosystemen in kaart te brengen.



Afbeelding 2: Verscheidene klimaten op de Kilimanjaro berg

Op en in de omgeving van de Kilimanjaro berg zijn slechts beperkte elektriciteitsvoorzieningen en is het internet enkel via een satellietverbinding toegankelijk. Satellietverbindingen hebben vaak een lage bandbreedte en een hoge latency (Brodkin, J. Satellite Internet faster than advertised, but latency still awful). Deze satellietverbinding is slechts op twee locaties toegankelijk en zal waarschijnlijk gedeeld moeten worden met andere gebruikers.

Netwerkdekking

Voor het opstellen van dit document is er vooronderzoek verricht om te bepalen of er GPRS of 3G dekking is bij de Kilimanjaro berg. De netwerkdekking van telecombedrijven kan online ingekeken worden (GSM Coverage Maps).

Afbeelding 3: Netwerkdekking Airtel Tanzania

Afbeelding 4: Netwerkdekking Tanzania Limited (Tigo)

Afbeelding 5: Netwerkdekking Vodacom Tanzania Limited

Hieruit blijkt dat alle drie telecombedrijven aan de voet van de berg dekking bieden. Vodacom Tanzania Limited biedt ook deels op de berg dekking. Tanzania Limited (Tigo) biedt bijna op de gehele berg dekking.

3 Project definitie

In dit hoofdstuk wordt de opdracht beschreven en worden de onderzoeksvragen gesteld. Daarna wordt de afbakening beschreven en worden de functionele requirements benoemd. Vervolgens worden er enkele tests beschreven waarmee functionele requirements gevalideerd kunnen worden. Daarna worden de onderzoeksmethoden genoemd en als laatste zullen de (deel)producten die tijdens dit project worden opgeleverd en de risico's worden benoemd.

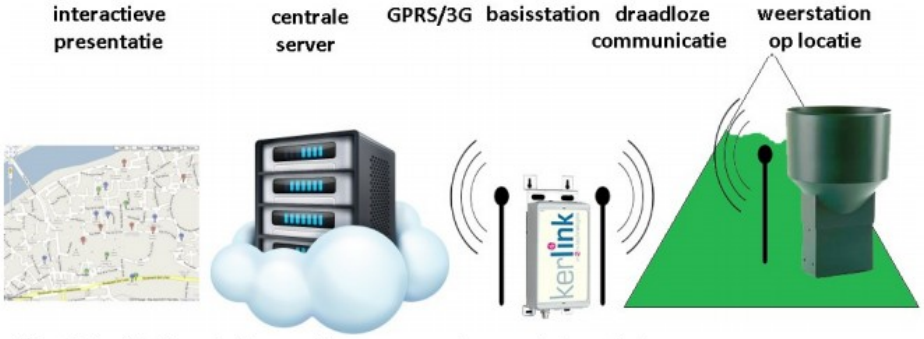
3.1 De opdracht

De opdracht is om weerstations draadloos uitleesbaar te maken en de verkregen gegevens verder te verwerken. Dit omvat het gehele proces tot en met de uiteindelijke gebruikersinterface. Dit moet worden gerealiseerd door middel van een proof of concept. De opdracht is dus een productopdracht (type 3).

Deze weerstations staan op de Kilimanjaro berg en hebben autonome energievoorziening. Aan de voet van de berg, of mogelijk ook op de berg, zullen ook enkele basisstations staan waar de weerstations hun gegevens door middel van draadloze communicatie naar toe zenden. De locatie van deze weerstations hangt af van de mogelijkheden van de gekozen draadloze netwerktechnologie. Waar mogelijk zullen de basisstations aangesloten worden op het vaste elektriciteitsnetwerk. Waar dit niet mogelijk is, zullen deze een autonome energievoorziening moeten hebben.

Omdat de gegevens voor wetenschappelijk onderzoek worden gebruikt is het essentieel dat deze betrouwbaar zijn. Er moet gekeken worden naar mogelijkheden om de betrouwbaarheid te verhogen door te voorkomen dat gegevens verloren gaan of corrupt raken.

De opdracht is voldaan wanneer er een proof of concept kan worden getoond waarbij de werking van de infrastructuur wordt aangetoond. Wanneer de GUI de juiste gegevens van de weerstations toont en dit interactief kan laden is de opdracht succesvol voldaan. Dit dient gerealiseerd te worden binnen de periode van de stage.



Afbeelding 6: Overzicht van alle componenten van het project

3.2 Onderzoeksvraag

De opdracht zoals hierboven beschreven leidt tot de volgende onderzoeksvraag:

Hoe kunnen de gegevens van weerstations op de Kilimanjaro berg betrouwbaar en energie efficiënt uitgelezen worden door middel van draadloze communicatie en gepresenteerd worden in een interactieve gebruikersinterface zodat de gegevens geautomatiseerd en semi-realtime inzichtelijk zijn?

Daarvoor moeten de volgende deelonderzoeksvragen worden beantwoord:

1. Hoe kan de communicatie tussen de weerstations en de basisstations gerealiseerd worden?
    1. Welke netwerktechnologieën zijn geschikt?
        1. Wat is het bereik van deze netwerktechnologieën?
    2. Welk protocol moet er worden gebruikt?
2. Hoe kan de communicatie tussen de basisstations en de centrale server gerealiseerd worden?
    1. Hoe kan het basisstation garanderen dat de gegevens bij de centrale server aankomen?
        1. Hoe moet het basisstation omgaan met langdurige netwerkstoringen?
    2. Hoe kunnen de gegevens via GPRS/3G naar de centrale server worden verzonden?
3. Hoe kunnen de gegevens van de weerstations worden opgeslagen?
    1. Hoe moet de server qua software worden ingericht?
    2. Hoe moet de database worden ingericht?
4. Hoe moeten de weerstations en de basisstations worden gevoed?
    1. Hoeveel energie is er op de Kilimanjaro berg beschikbaar?
    2. Hoeveel energie verbruikt een weerstation?
    3. Hoeveel energie verbruikt een basisstation?
    4. Hoeveel capaciteit moet de batterij hebben?
5. Hoe kan de betrouwbaarheid van de gegevens worden gewaarborgd?
    1. Hoe kan datacorruptie worden gedetecteerd?
    2. Hoe kunnen de gegevens worden beveiligd zodat deze niet door derden gemanipuleerd kunnen worden?

3.3 Afbakening

Voor dit project zijn de volgende afbakeningen gesteld:

- De infrastructuur richt zich op laag frequente gegevens overdracht.
- De infrastructuur richt zich op toepassingen met lage bandbreedte.
- De infrastructuur moet geschikt zijn voor gebruik op de Kilimanjaro berg. Andere omgevingen met minder gunstigere omgevingsfactoren zullen simpelweg niet ondersteund worden.
- Het project wordt enkel in Nederland getest. Er wordt zoveel mogelijk rekening gehouden met omstandigheden op de Kilimanjaro berg, maar er wordt niet daadwerkelijk ter plaatse getest.
- Het project houdt zich enkel bezig met elektronische componenten en software. Mechanische zaken, zoals geschikte ophanging voor antennes of een omhulsel dat aan de benodigde IP rating voldoet, is niet onderdeel van dit project.
- Onderzoek naar de geschiktheid en precisie van sensoren is geen onderdeel van dit project.
- Analyse van data om hier hogere informatie uit te halen (zoals een weersvoorspelling) is geen onderdeel van dit project.
- Onderzoek naar emissierechten, zendrechten en vrije frequentiebanden of andere juridische zaken is geen onderdeel van dit project.
- Onderzoek naar kosten en andere zakelijke aspecten is geen onderdeel van dit project

3.4 Functionele requirements

In het vooronderzoek zijn de functionele requirements van het project onderzocht. Deze zijn opgesteld en ingedeeld met de MoSCoW methode:

Prioriteit	Functionele requirement
Must have	Meet de regenval
	Verstuur de gegevens van de weerstations draadloos
	Toon de regenval per sensor
Should have	Versleutel de gegevens end-to-end (om zo afluisteren tegen te gaan)
	Toon de weerstations op een kaart
	Toon de basisstations op een kaart
	Het weerstation moet onderhoudsarm zijn
	Monitor de werking van de weerstations
Could have	Ondersteuning voor andere sensoren moet aanwezig zijn
Won't have	De weerstations moeten zelf hun locatie bepalen en rapporteren

De functionele requirements zijn verder uitgesplitst en opgesteld in de design matrix, zoals de Axiomatic Design methode dat voorschrijft (Suh, N. P., & Do, S.-H. Axiomatic Design of Software Systems). Deze staat in Appendix A.

3.5 Tests

Van de functionele requirements kunnen voor enkele een test worden gedefinieerd. Hiermee kan de requirement gevalideerd worden. De FR nummers in de onderstaande tabel refereren naar de functionele requirements zoals opgesteld in Appendix A. Mogelijk worden in een later stadium tests toegevoegd.

Test #	FR #	Beschrijving
1	1.1	De regensensor werken door regen in een bakje op te vangen. Wanneer deze vol is, kiepert het bakje waardoor de deze leeg loopt. Ook wordt er een Reed-contact geactiveerd wat een tick geeft. Controleer of de microcontroller altijd de ticks van de regensensor detecteert door het een gedefinieerd aantal ticks te geven en dit aantal vergelijken met het aantal gedetecteerde ticks. Ook onder extreme omstandigheden (100 ticks / seconde) moet dit blijven functioneren. Deze test controleert of de microcontroller instaat is om 100 ticks / seconde te verwerken (performance) en of de microcontroller altijd correct een tick detecteert (of een interrupt of iets dergelijks altijd wordt afgevuurd). Het is onwaarschijnlijk dan een regensensor 100 ticks / seconde genereert, maar met meerdere (regen)sensoren is dit aantal reëel.
2	9	Controleer of het basisstation correct buffert wanneer de netwerkverbinding tussen het basisstation en de centrale server wordt verbroken en controleer of het basisstation de gebufferde berichten correct verwerkt wanneer de netwerkverbinding hersteld. Ook onder extreme omstandigheden (100 weerstations die elke 5 minuten een bericht sturen, resulterend in 3 berichten / seconde) moet dit blijven functioneren. Deze test controleert of het basisstation instaat is om te detecteren dat de netwerkverbinding verbroken is, en daar correct op reageert (door de berichten te gaan bufferen). Tevens wordt er test of het basisstation instaat is om berichten op een hoge frequentie te verwerken. Als laatste wordt ook de netwerkverbinding tussen het basisstation en de centrale server getest door, wanneer de netwerkverbinding herstelt is, er in grote hoeveelheden berichten over te sturen.
3	3.1	Controleer of de microcontroller een fout in een variabele detecteert door softwarematig (met een speciale test function/class) deze aan te passen. De foutdetectie moet blijven werken onder extreme omstandigheden (50% van de bits corrupt) Deze test controleert of de microcontroller instaat is om data corruptie door hardwarestoringen te detecteren.
4	4	Controleer of de centrale server corrupte invoer correct verwerkt (door deze niet op te slaan) en dat dubbele berichten correct worden afgehandeld (door slechts één exemplaar op te slaan). Deze test controleert de invoer protectie van de centrale server.

3.6 Onderzoeksmethoden

In de volgende tabel wordt per deelonderzoeksvraag de gebruikte onderzoeksmethode en onderzoekstechnieken beschreven:

#	Onderzoeksvraag	Type	Onderzoeks-methode(n)	Methode en techniek
1	Hoe kan de communicatie tussen de weerstations en de basisstations gerealiseerd worden?	Beschrijvend & vergelijkend	Deskresearch, experimenteren & veldonderzoek	
1.1	Welke netwerktechnologieën zijn mogelijk geschikt?	Beschrijvend & vergelijkend	Deskresearch & veldonderzoek	Door netwerktechnologieën te vergelijken en bereik in het veld te meten wordt een antwoord geformuleerd.
1.2	Welk protocol moet er worden gebruikt?	Beschrijvend	Deskresearch	Aan de hand van de te versturen gegevens wordt een protocol opgesteld.
2	Hoe kan de communicatie tussen de basisstations en de centrale server gerealiseerd worden?	Beschrijvend	Deskresearch & veldonderzoek	
2.1	Hoe kan het basisstation garanderen dat de gegevens bij de centrale server aankomen?	Beschrijvend & vergelijkend	Deskresearch	Door synchronisatie methoden te vergelijken wordt een antwoord geformuleerd.
2.2	Hoe kunnen de gegevens naar via GPRS/3G naar de centrale server worden verzonden?	Beschrijvend	Deskresearch & veldonderzoek	Door de werking van GPRS/3G te analyseren en de werking te testen wordt een antwoord geformuleerd.
3	Hoe kunnen de gegevens van de weerstations worden opgeslagen?	Beschrijvend	Deskresearch	
3.1	Hoe moet de server qua software worden ingericht?	Beschrijvend	Deskresearch	Aan de hand van de functionele requirements wordt een antwoord geformuleerd.
3.2	Hoe moet de database worden ingericht?	Beschrijvend	Deskresearch	Aan de hand van de functionele requirements en de normalisatie regels van relationele databases wordt een antwoord geformuleerd.
4	Hoe moeten de weerstations en de basisstations worden gevoed?	Beschrijvend	Deskresearch	

4.1	Hoeveel energie is er beschikbaar?	Beschrijvend	Deskresearch & veldonderzoek	Aan de hand van klimaateigenschappen en lokaal gemeten waarden wordt een antwoord geformuleerd.
4.2	Hoeveel energie verbruikt een weerstation?	Beschrijvend	Deskresearch & veldonderzoek	Aan de hand van datasheets van de gebruikte hardware en gemeten waarden wordt een antwoord geformuleerd.
4.3	Hoeveel energie verbruikt een basisstation?	Beschrijvend	Deskresearch & veldonderzoek	Aan de hand van datasheets van de gebruikte hardware en gemeten waarden wordt een antwoord geformuleerd.
4.4	Hoeveel capaciteit moet de batterij hebben?	Beschrijvend	Deskresearch	Aan de hand van klimaateigenschappen wordt een antwoord geformuleerd.
5	Hoe kan de betrouwbaarheid van de gegevens worden gewaarborgd?	Beschrijvend	Deskresearch	
5.1	Hoe kan datacorruptie worden gedetecteerd?	Beschrijvend & vergelijkend	Deskresearch	Aan de hand van bestaande methoden van foutdetectie wordt een antwoord geformuleerd.
5.2	Hoe kunnen de gegevens worden beveiligd zodat deze niet door derden gemanipuleerd kunnen worden?	Beschrijvend	Deskresearch	Aan de hand van bestaande beveiligingstechnieken wordt een antwoord geformuleerd.

3.7 (Deel)producten

Gedurende dit project zullen de volgende (deel)producten op chronologische volgorde worden opgeleverd. De kwaliteit van deze (deel)producten wordt bepaald door deze te onderwerpen aan de eerder beschreven tests. De uitkomsten van deze tests zullen in een testrapport worden opgenomen.

1. Een stuk software dat op de micro-controllers, die in het weerstations wordt gebruikt, kan draaien en instaat is om de regensensor uit te lezen en de gegevens draadloos te versturen.
2. Een stuk software dat op de basisstations kan draaien en instaat is om gegevens van de weerstations te ontvangen en door te sturen naar de centrale server.
3. Een ingerichte centrale server die instaat is om gegevens van de basisstations te ontvangen en op te slaan.
4. Een GUI die de gegevens op de centrale server staan opgeslagen interactief kan presenteren.
5. Een uitbreiding op deelproduct 4 die de gegevens van de centrale server op een kaart kan presenteren.
6. Een uitbreiding op deelproduct 1 die instaat is om datacorruptie te detecteren.
7. Een uitbreiding op deelproduct 6 die instaat is om de werking van het weerstation te monitoren.
8. Een uitbreiding op deelproduct 7 die instaat is om meerdere sensoren uit te lezen.

3.8 Risico's

De volgende risico's zijn voor dit project gedefinieerd:

#	Risico	Kans (1 – 5)	Ernst (1 – 5)	Consequentie	Tegenmaatregelen
1	De draadloze netwerk-technologieën hebben onvoldoende bereik	3	3	Stabiele communicatie is niet langer mogelijk	Richtantennes gebruiken om bereik te vergroten, heeft als gevolg dat er inzetbaarheid verkleind wordt. Ook kunnen de afstanden verkleint worden door meerdere basisstations te gebruiken die ook op de berg zelf geïnstalleerd staan.
2	Gebruik van draadloze netwerk-technologieën overschrijdt het energiebudget	2	2	Het weerstation valt uit wanneer er over langere tijd geen zonlicht is.	Richtantenne en zendvermogen (adaptief) regelen of grotere/efficiëntere zonnepanelen of een grotere accu gebruiken.
3	De internetverbinding heeft onvoldoende bandbreedte.	2	3	De basisstations zullen niet instaat zijn om hun berichten te verwerken en de buffers zullen vollopen met mogelijk verlies van gegevens.	Een andere communicatie techniek met meer bandbreedte gebruiken zoals een satellietverbinding of datacompressie toepassen.

4 Project beheer

In dit hoofdstuk wordt de ontwikkelmethode, de milestones en de planning verder toegelicht.

4.1 Ontwikkelmethode

De gebruikte ontwikkelmethode voor dit project is een aangepaste versie van het waterval model. Deze bestaat uit de volgende fasen:

1. **Requirements**: In deze fase worden de externe factoren van het project geanalyseerd en worden de functionele requirements vastgesteld.
2. **Onderzoek**: In deze fase wordt onderzoek gedaan om de onderzoeksvragen te kunnen beantwoorden
3. **Ontwerp**: In deze fase worden software componenten ontworpen en worden zaken zoals concurrency diagrammen en class diagrammen uitgewerkt.
4. **Programmeren**: In deze fase wordt de software daadwerkelijk geprogrammeerd.
5. **Validatie**: In deze fase wordt de software getest. Hierbij wordt bijvoorbeeld de software getest door de hardware spontaan te resetten.
6. **Afronding**: In deze fase worden de scriptie en de presentatie geschreven.

4.2 Milestones

De volgende milestones zijn voor dit project gedefinieerd:

Fase	Milestone	Beschrijving
Requirements	Functionele requirements	De functionele requirements opstellen door externe factoren, beschikbare hardware en wensen van de opdrachtgever te analyseren.
	Plan van aanpak	Een plan van aanpak opleveren en deze goed laten keuren.
Onderzoek	Rapport van bevindingen	Een rapport over de bevindingen die zijn opgedaan tijdens het (veld)onderzoek.
Ontwerp	Software ontwerp	Een rapport met daarin ontwerp diagrammen van de op te leveren software.
Programmeren	Deelproducten	Het programmeren van de beschreven deelproduct.
Validatie	Testrapporten	De opgestelde tests uitvoeren en de resultaten in een testrapport beschrijven.
Afronding	Scriptie	De scriptie afronden en deze inleveren.
	Eindpresentatie	De eindpresentatie schrijven en geven.

De milestones “Rapport van bevindingen” en “Software ontwerp” zullen een document opleveren dat tevens zal dienen als een concept versie van de scriptie. Immers kan de inhoud en de grotendeels ook de structuur van het document in de uiteindelijke versie van de scriptie worden overgenomen.

4.3 Planning

De gehele planning van dit project staat in de tabel hieronder. De concept versies van de scriptie zullen na taak 12 en na taak 18 worden ingediend. In de planning is een vakantie van drie weken opgenomen i.v.m. met verhuizen en verbouwen. Deze drie weken zijn gecompenseerd door eerder te starten met stage lopen. De bijhorende Gantt chart staat in Appendix B.

#	Beschrijving/Milestone	Start datum	Eind datum	Duur
1	**Functionele requirements**	19-10-15	02-11-15	11 dagen
2	Analyse microcontroller hardware (SODAQ tatu)	19-10-15	20-10-15	2 dagen
3	Analyse LoRaWAN module en server	21-10-15	23-10-15	3 dagen
4	Analyse basisstation (Kerlink LoRa IOT station)	26-10-15	28-10-15	3 dagen
5	Omgeving van de Kilimanjaro berg analyseren	29-10-15	29-10-15	1 dag
6	Eerste MoSCoW opstellen	30-10-15	30-10-15	1 dag
7	Functionele requirements uitbreiden	02-11-15	02-11-15	1 dag
8	**Plan van aanpak**	03-11-15	12-11-15	8 dagen
9	Eerste versie	03-11-15	05-11-15	3 dagen
10	Feedback van docent- en bedrijfsbegeleider	06-11-15	10-11-15	3 dagen
11	Feedback verwerken en nieuwe versie	11-11-15	12-11-15	2 dagen
12	**Rapport van bevindingen**	13-11-15	25-11-15	9 dagen
13	Veldonderzoek voor richtantenne en LoRaWAN voorbereiden	13-11-15	13-11-15	1 dag
14	Veldonderzoek voor richtantenne en LoRaWAN uitvoeren en beschrijven	16-11-15	17-11-15	2 dagen
15	Veldonderzoek voor energievoorziening voorbereiden	18-11-15	18-11-15	1 dag
16	Veldonderzoek voor energiebronnen uitvoeren en beschrijven	19-11-15	20-11-15	2 dagen
17	Rapport van bevindingen opstellen	23-11-15	25-11-15	3 dagen
18	**Software ontwerp**	26-11-15	12-01-16	22 dagen
19	Sequence diagrammen systeem	26-11-15	27-11-15	2 dagen
20	Activity diagram microcontroller software	30-11-15	30-11-15	1 dag
21	Class diagram microcontroller software	01-12-15	02-12-15	2 dagen
22	Concurrency diagram microcontroller software	03-12-15	04-12-15	2 dagen
23	Activity diagram basisstation software	07-12-15	07-12-15	1 dag
24	Class diagram basisstation software	08-12-15	09-12-15	2 dagen
25	Concurrency diagram basisstation software	10-12-15	11-12-15	2 dagen
26	Entity relation diagram centrale server	14-12-15	15-12-15	2 dagen

#	Beschrijving/Milestone	Start datum	Eind datum	Duur
27	Entity relation diagram valideren	16-12-15	16-12-15	1 dag
Vakantie				
28	Activity diagram centrale server software	04-01-16	04-01-16	1 dag
29	Class diagram centrale server software	05-01-16	06-01-16	2 dagen
30	Concurrency diagram centrale server software	07-01-16	08-01-16	2 dagen
31	Ontwerp GUI	11-01-16	11-01-16	1 dag
32	Activity diagram GUI	12-01-16	12-01-16	1 dag
33	**Deelproducten**	13-01-16	24-03-16	52 dagen
34	Microcontroller programmeren (JTAG)	13-01-16	14-01-16	2 dagen
35	Deelproduct 1 programmeren	15-01-16	18-01-16	2 dagen
36	Crosscompilen voor het basisstation	19-01-16	19-01-16	1 dag
37	Deelproduct 2 programmeren	20-01-16	01-02-16	8 dagen
38	Deelproduct 3 programmeren	02-02-16	04-02-16	3 dagen
39	Deelproduct 4 programmeren	05-02-16	09-02-16	3 dagen
40	Deelproduct 5 programmeren	10-02-16	12-02-16	3 dagen
41	Deelproduct 6 programmeren	15-02-16	16-02-16	2 dagen
42	Deelproduct 7 programmeren	17-02-16	19-02-16	3 dagen
58	Deelproduct 8 programmeren (optioneel)	16-03-16	24-03-16	7 dagen
43	**Testrapport**	22-02-16	26-02-16	5 dagen
44	De performance van de microcontrollers meten	22-02-16	22-02-16	1 dag
45	De buffer van de basisstations testen	23-02-16	23-02-16	1 dag
46	De datacorruptie-detectie testen	24-02-16	24-02-16	1 dag
47	De data integriteit op de server testen	25-02-16	25-02-16	1 dag
48	Testrapport opstellen	26-02-16	26-02-16	1 dag
49	**Scriptie**	01-03-16	15-03-16	10 dagen
50	Inhoud andere documenten in scriptie plaatsen	01-03-16	01-03-16	1 dag
51	Extra stukken schrijven	02-03-16	04-03-16	3 dagen
52	Laten controleren door bedrijfsbegeleider	07-03-16	09-03-16	3 dagen
53	Feedback verwerken	10-03-16	11-03-16	2 dagen
54	Scriptie printen en binden	14-03-16	14-03-16	1 dag
55	Scriptie inleveren	15-03-16	15-03-16	1 dag
55	**Eindpresentatie**	25-03-16	31-03-16	4 dagen
56	Presentatie voorbereiden	25-03-16	25-03-16	1 dag
57	Presentatie geven	29-03-16	31-03-16	3 dagen

5 Referenties

Brodkin, J. (2013, februari 15). Satellite Internet faster than advertised, but latency still awful. Geraadpleegd 11 november 2015, van http://arstechnica.com/information-technology/2013/02/satellite-internet-faster-than-advertised-but-latency-still-awful/

DFG Research Group Kilimanjaro, Project Summary. (2010). Geraadpleegd 11 november 2015, van https://www.kilimanjaro.biozentrum.uni-wuerzburg.de/project/Summary.aspx

DFG Research Group Kilimanjaro, Study Design. (2010). Geraadpleegd 11 november 2015, van https://www.kilimanjaro.biozentrum.uni-wuerzburg.de/project/Methods.aspx

GSM Coverage Maps | Tanzania. (2015). Geraadpleegd 11 november 2015, van http://maps.mobileworldlive.com/network.php?cid=89&cname=Tanzania

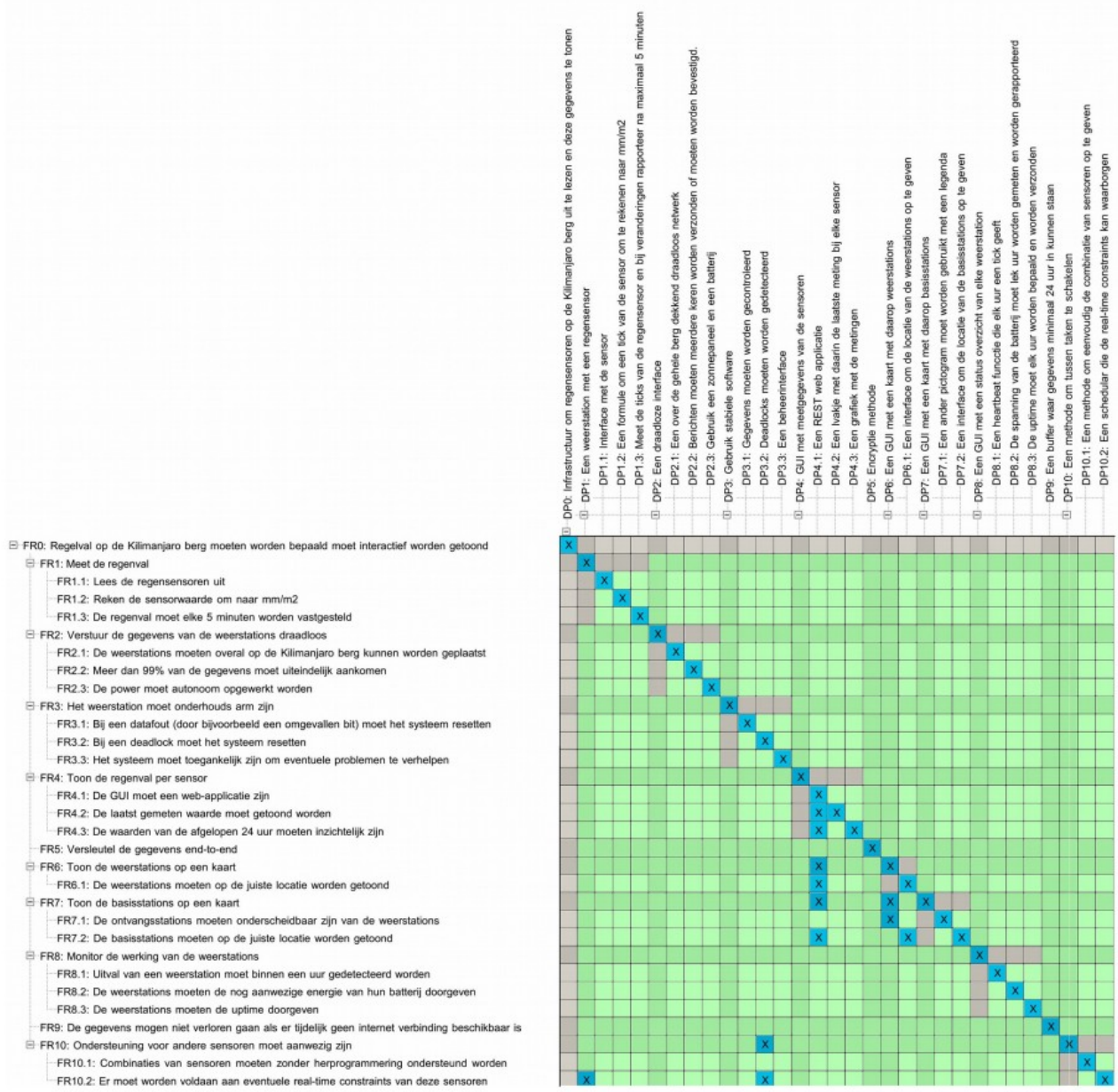
M2M4ALL. (2015). Verticals. Geraadpleegd 11 november 2015, van http://www.m2m4all.com/?page_id=1223

Mount Kilimanjaro. (2015, november 9). In *Wikipedia, the free encyclopedia*. Geraadpleegd van https://en.wikipedia.org/w/index.php?title=Mount_Kilimanjaro&oldid=689771647

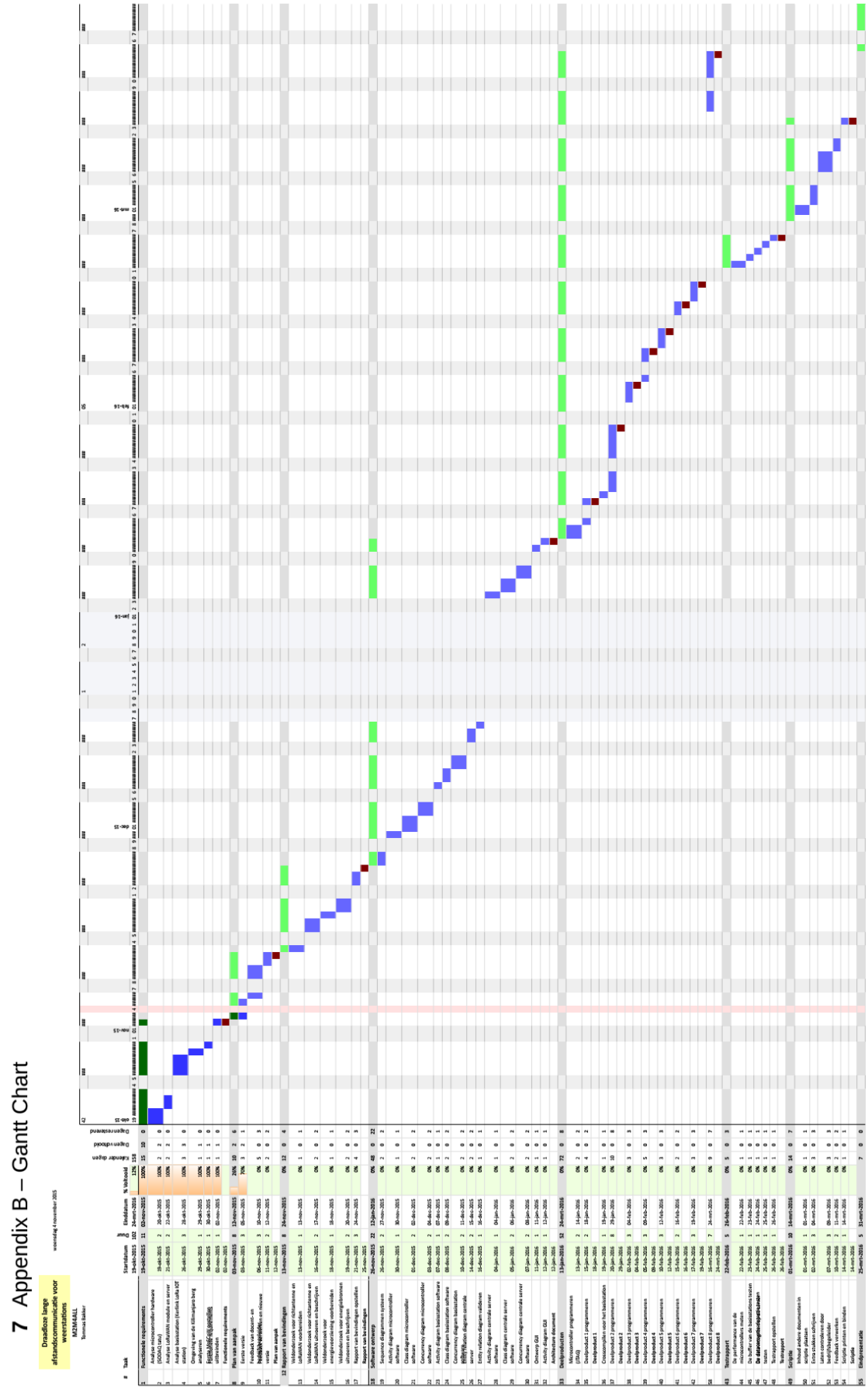
Suh, N. P., & Do, S.-H. (2000). Axiomatic Design of Software Systems. *CIRP Annals - Manufacturing Technology*, *49*(1), 95–100. http://doi.org/10.1016/S0007-8506(07)62904-7

6 Appendix A – Axiomatic Design Matrix

De matrix hieronder zet de functionele requirements (FR) tegen de design parameters (DP) en geeft de onderlinge afhankelijkheid aan.



7 Appendix B – Gantt Chart



Appendix B – Bronnen voor specificaties netwerktechnologieën

De volgende bronnen zijn gebruik voor het vergelijken van de netwerktechnologieën:

Business Wire. (2014, december 2). AXSEM and SIGFOX Announce “Ultra-low-power” System-on-Chip Solution for Long-range Internet of Things Connectivity. Geraadpleegd 13 maart 2016, van http://www.businesswire.com/news/home/20141202005485/en/AXSEM-SIGFOX-Announce-%E2%80%98Ultra-low-power%E2%80%99-System-on-Chip-Solution-Long-range

dBm. (2016, februari 29). In Wikipedia, the free encyclopedia. Geraadpleegd van https://en.wikipedia.org/w/index.php?title=DBm&oldid=707457860

Frenzel, L. (2012, maart 29). What’s The Difference Between ZigBee And Z-Wave? Geraadpleegd 13 maart 2016, van http://electronicdesign.com/communications/what-s-difference-between-zigbee-and-z-wave

High Speed Packet Access. (2016, januari 31). In Wikipedia, the free encyclopedia. Geraadpleegd van https://en.wikipedia.org/w/index.php?title=High_Speed_Packet_Access&oldid=702571531

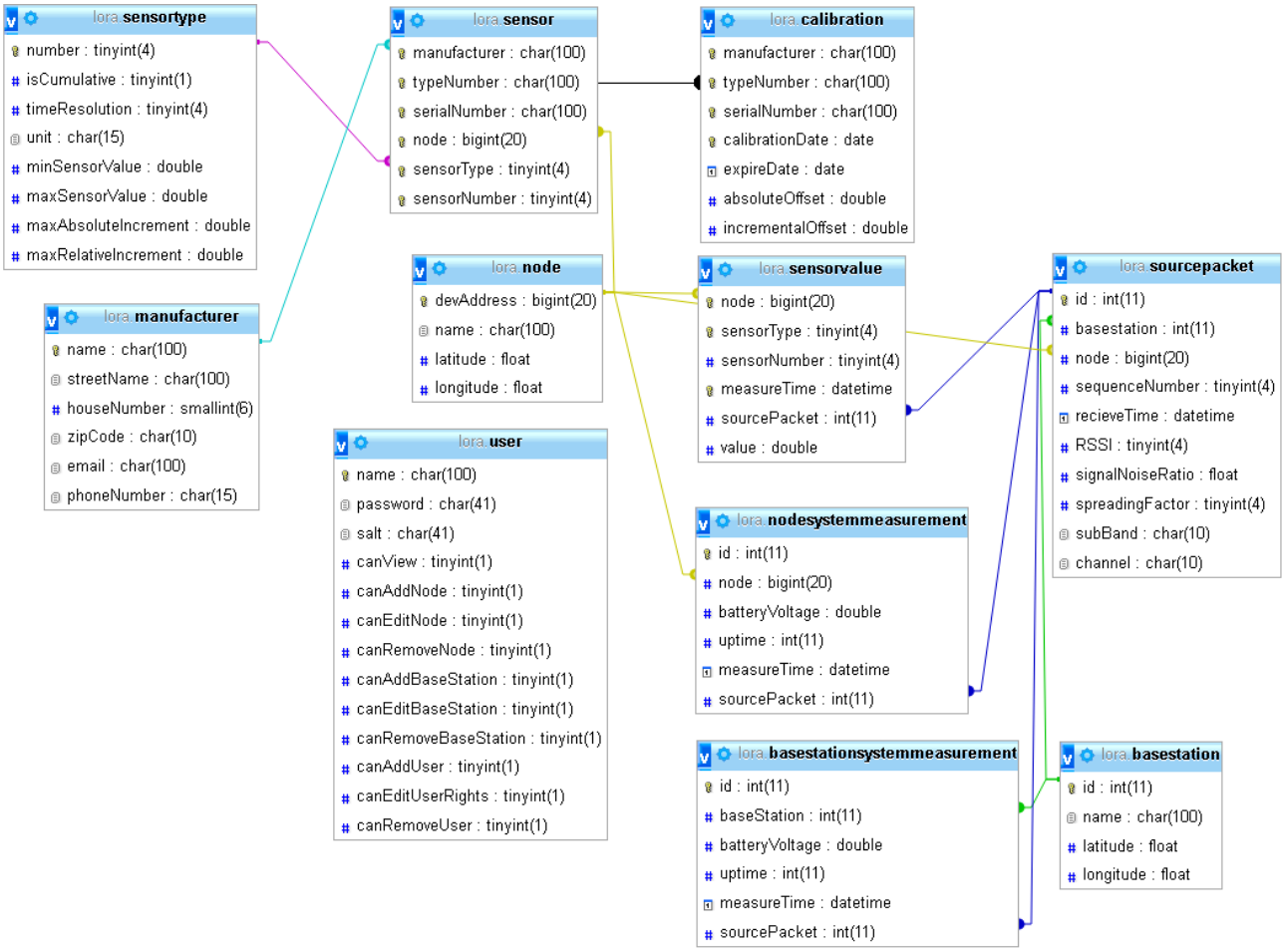
Panasonic. (2015). Z-Wave - Modem PAN8550. Geraadpleegd van http://www.mouser.com/ds/2/315/PAN8550-180434.pdf

RS Components. (2015). 11 Internet of Things (IoT) Protocols You Need to Know About. Geraadpleegd 13 maart 2016, van http://www.rs-online.com/designspark/electronics/knowledge-item/eleven-internet-of-things-iot-protocols-you-need-to-know-about

Wi-Fi. (2016, maart 13). In Wikipedia, the free encyclopedia. Geraadpleegd van https://en.wikipedia.org/w/index.php?title=Wi-Fi&oldid=709816721

Appendix C – Validatie van de database

Het valideren van de database wordt aan de hand van functional dependencies gedaan. Hiermee kan de database tot de derde normaalvorm (3NF) worden genormaliseerd. De “met de hand” ontworpen database heeft de volgende ERD:



Afbeelding 40: Een "met de hand" gemaakte ERD

Om deze te normaliseren moeten eerst alle functionele dependencies worden opgesteld:

`Manufacturer.name → Manufacturer.streetName + Manufacturer.houseNumber + Manufacturer.zipCode + Manufacturer.email + Manufacturer.phoneNumber`[13]

`Manufacturer.zipCode → Manufacturer.streetName + Manufacturer.houseNumber`

`SensorType.number → SensorType.isCumulative + SensorType.timeResolution + SensorType.unit + SensorType.minSensorValue + SensorType.maxSensorValue + SensorType.maxAbsoluteIncrement + SensorType.maxRelativeIncrement`

`Sensor.manufacturer + Sensor.typeNumber + Sensor.serialNumber → Sensor.node + Sensor.sensorType + Sensor.sensorNumber`

`Sensor.node + Sensor.sensorType + Sensor.sensorNumber → Sensor.manufacturer + Sensor.typeNumber + Sensor.serialNumber`

`Calibration.manufacturer + Calibration.typeNumber + Calibration.serialNumber + Calibration.calibrationDate → Calibration.expireDate + Calibration.absoluteOffset + Calibration.incrementalOffset`

`Node.devAddress → Node.name + Node.latitude + Node.longitude`[14]

`SensorValue.node + SensorValue.sensorType + SensorValue.sensorNumber +`

13 Hierbij wordt er van uitgegaan dat de bedrijfsnaam uniek is

14 De FD `Node.latitude + Node.longitude → Node.devAddress + Node.name` gaat niet op omdat mogelijk weerstations zo dicht op elkaar staan dat er geen onderscheid kan worden gemaakt.

```
    SensorValue.measureTime → SensorValue.sourcePacket + SensorValue.value
SensorValue.node + SensorValue.sensorType + SensorValue.sensorNumber +
    SensorValue.sourcePacket →  SensorValue.measureTime + SensorValue.value
NodeSystemMeasurement.id → NodeSystemMeasurement.node +
    NodeSystemMeasurement.batteryVoltage + NodeSystemMeasurement.uptime +
    NodeSystemMeasurement.measureTime + NodeSystemMeasurement.sourcePacket
NodeSystemMeasurement.sourcePacket + NodeSystemMeasurement.node →
    NodeSystemMeasurement.id + NodeSystemMeasurement.batteryVoltage +
    NodeSystemMeasurement.uptime + NodeSystemMeasurement.measureTime
NodeSystemMeasurement.node + NodeSystemMeasurement.measureTime →
    NodeSystemMeasurement.id + NodeSystemMeasurement.batteryVoltage +
    NodeSystemMeasurement.uptime + NodeSystemMeasurement.sourcePacket
BaseStationSystemMeasurement.id → BaseStationSystemMeasurement.baseStation +
    BaseStationSystemMeasurement.batteryVoltage +
    BaseStationSystemMeasurement.uptime +
    BaseStationSystemMeasurement.measureTime +
    BaseStationSystemMeasurement.sourcePacket
BaseStationSystemMeasurement.sourcePacket +
    BaseStationSystemMeasurement.baseStation → BaseStationSystemMeasurement.id
    + BaseStationSystemMeasurement.batteryVoltage +
    BaseStationSystemMeasurement.uptime +
    BaseStationSystemMeasurement.measureTime
NodeSystemMeasurement.baseStation + BaseStationSystemMeasurement.measureTime →
    BaseStationSystemMeasurement.id +
    BaseStationSystemMeasurement.batteryVoltage +
    BaseStationSystemMeasurement.uptime +
    BaseStationSystemMeasurement.sourcePacket
BaseStation.id → BaseStation.name + BaseStation.latitude +
    BaseStation.longitude[15]
SourcePacket.id → SourcePacket.baseStation + SourcePacket.node +
    SourcePacket.sequenceNumber + SourcePacket.receiveTime + SourcePacket.RSSI
    + SourcePacket.signalNoiseRatio + SourcePacket.spreadingFactor +
    SourcePacket.subBand + SourcePacket.channel
SourcePacket.id → SourcePacket.baseStation + SourcePacket.node +
    SourcePacket.sequenceNumber + SourcePacket.receiveTime + SourcePacket.RSSI
    + SourcePacket.signalNoiseRatio + SourcePacket.spreadingFactor +
    SourcePacket.subBand + SourcePacket.channel
SourcePacket.baseStation + SourcePacket.receiveTime → SourcePacket.id +
    SourcePacket.node + SourcePacket.sequenceNumber + SourcePacket.RSSI +
    SourcePacket.signalNoiseRatio + SourcePacket.spreadingFactor +
    SourcePacket.subBand + SourcePacket.channel
SourcePacket.node +  SourcePacket.receiveTime → SourcePacket.id +
    SourcePacket.baseStation + SourcePacket.sequenceNumber + SourcePacket.RSSI
    + SourcePacket.signalNoiseRatio + SourcePacket.spreadingFactor +
    SourcePacket.subBand + SourcePacket.channel
User.name → User.password + User.salt + User.canView + User.canAddNode +
    User.canEditNode + User.canRemoveNode + User.canAddBaseStation +
    User.canEditBaseStation + User.canRemoveBaseStation + User.canAddUser +
    User.canEditUserRights + User.canRemoveUser
```

15 De FD `BaseStation.latitude + BaseStation.longitude → BaseStation.id + BaseStation.name` gaat niet op omdat mogelijk weerstations zo dicht op elkaar staan dat er geen onderscheid kan worden gemaakt.

Nu alle functionele dependencies zijn vast vastgesteld moet er worden gekeken of één van deze de criteria van de 3NF niet haalt. Hierbij zijn de functionele dependencies die een relatie (wetenschappelijke term voor tabel) delen verdacht. Er wordt voldaan aan de 3NF als:

> *A relation R is in third normal form (3NF) if (…) for each non trivial FD, either the left side is a superkey, or the right side consists of prime attributes only* (Garcia-Molina, Ullman, & Widom, 2008)

Uit controle blijkt dat volgende functionele dependency de enige is die niet voldoet aan de 3NF:
`Manufacturer.zipCode → Manufacturer.streetName + Manufacturer.houseNumber`
Dit kan worden opgelost door een relatie Address te defineren en in de relatie Manufacturer de attributen streetName en houseNumber te verwijderen. De zipCode vormt nu een foreign key naar de relatie Address.

Omdat er geen verdere functional dependencies zijn die niet aan de criteria van 3NF voldoen, kan worden gesteld dat database voldoet aan de 3NF.

Appendix D – Ontwerpverslag

Wireless Long Range Communication for Weather Stations

Design rapport

Wireless Long Range Communication for Weather Stations

Design rapport

Wireless Long Range Communication for Weather Stations

Design rapport

Cover image:
Kilimanjaro mountain

Author:
Tommas Bakker

Student number:
1611239

Version:
1.0

Date:
25-11-2015

Mentors:
Jan Willen Smeenk, M2M4ALL
Company mentor

Examinators:
Henk van Nimwegen, Hogeschool Utrecht
First examinator

Jan Zuurbier, Hogeschool Utrecht
Second examinator

Table of Contents

1 The network....4
1.1 The LoRaWAN standard....4
1.1.1 Limitations....5
1.2 Proposed network....5
2 Protocol....6
2.1 Weather station – base station....6
2.1.1 Semi-realtime....6
2.1.2 Reliability....7
2.1.3 Protocol definition....8
2.2 Base station – central server....9
2.3 Central server – GUI....9
3 The weather station....10
3.1 First version....10
3.2 Second version....11
3.3 Third version....12
3.4 Fourth version....13
3.4.1 Concurrency....14
3.4.2 Classes....14
4 The base station....15
4.1 Activity diagram....15
4.2 Class diagram....16
5 The central server....17
5.1 Entity relation diagram....17
5.2 Activity diagram....18
5.3 Class diagram....19
6 GUI....20

1 The network

This chapter describes the design of the network that will be used for this project. First, the existing standard network is examined. Then the own design is described.

1.1 The LoRaWAN standard

The measurements of the weather station are transmitted to the base station via LoRaWAN. The LoRaWAN standard on its own has no high OSI layers in the protocol. This is simply done by the "application segment". However, the standard has given a configuration for a typical network.

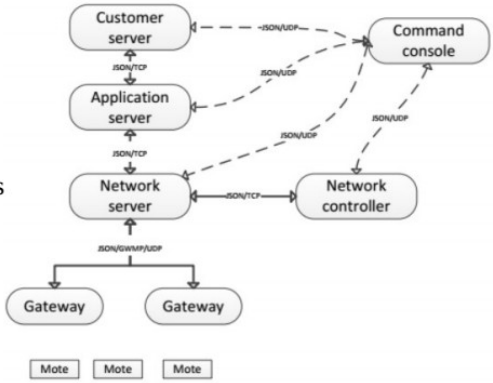


Illustration 1: The LoRaWAN standard network configuration

In the configuration weather stations are called nodes and base stations are called gateways. The gateways have a very limited task: forwarding packets. Effectively the gateway acts as a bridge.

The network server receives all incoming packets of the gateways and will process them. This is done by decrypting the MAC component of the packet. The MAC component is all data of the packet except the APP component. This APP component holds the actual content. The actual content is decrypted by the application server as this allows safe transfer of data over a third party network. To be able to decrypt the MAC component of a packet the keys of the sender and receiver must match. Therefore, an encryption key is sent to the module when it connects to the network or the encryption key is preprogrammed onto the device.

The application server process the APP component of the packet. The server decrypts the component and verifies the authentication of the module. When this is the case, the application server will forward the packet content to the customer server. The application server also distributes encryption keys for the encryption and decryption of the APP component.

The customer server is the server that actually processes the content of the packet by, for example, storing it in a database. The customer server is usually managed by the customer of the network provider.

The network controller controls the network and monitors the quality of the network. This is accomplished by monitoring (and if required, adjusting) various parameters such as the RSSI and the spreading factor.

1.1.1 Limitations

A limitation of the network configuration as described is that it is very susceptible to disruptions in the network. Also during tests this seemed to be the case. This forms a risk as the quality of the network being used depends on a third party and is expected to be quite low. This forms a major problem in the connection between the gateways and the network servers.

Another limitation is that the configuration is composed of many components. This makes the network quite complex and thus more susceptible to disruptions. The network is designed to function as a universal network, with multiple parties using the same network and each wanting to keep their data private. This is simply not the case for this project: the network isn't managed by a third party and no other users exist. This means that the various components can be merged.

1.2 Proposed network

The proposed network is considerably more simple and combines various logical components to one physical component.

The base station (the Kerlink IOT station) is an embedded computer with a bare Linux environment. It has 128 MiB of RAM and has 8 GB storage capacity. It also has a battery that can cover power disruptions for three minutes, after which the base station is shutdown gracefully. This makes the hardware of the base station suitable to run a small server.

It has been decided to house all LoRaWAN specific components in the base station. This allows a more universal interface on higher levels. This means that the receiving, decrypting and encrypting of packets is done on the base station.

The central server runs a universal application server (and thus does not depend on any LoRaWAN specific components). This means that optionally a different network technology (such as Zigbee) can be used without requiring any major modifications. The interface of the server will be a simple RESTful interface. The central server will check the data and store it in the database.

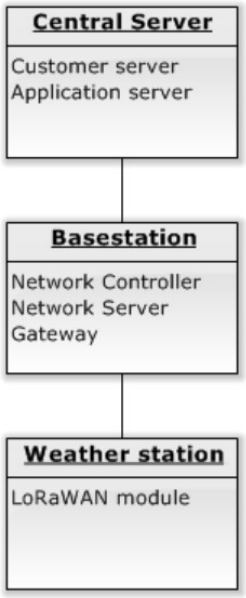


Illustration 2: The proposed network with the physical components and their functional components

2 Protocol

To be able to eventually store the measurements of the weather stations in the database of the central server, the measurements have to be transferred one or multiple times. Therefore one or multiple protocols have to be defined.

Because the measurements are first transferred from the weather station to the base station via LoRaWAN, then are transferred to the central server via a IP protocol, and finally are transferred from the central server to a GUI, it has been decided to define three protocols. These protocols should not deviate from existing protocols.

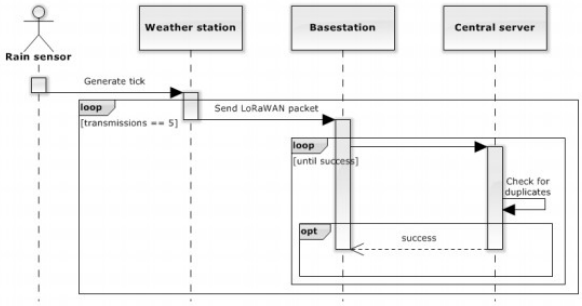


Illustration 3: Transmission of the measurements to the central server

2.1 Weather station – base station

The protocol being used for the communicatio between the weather stations and the base stations functions with the LoRaWAN protocol. The designed protocol fills the content component of the LoRaWAN packets. The content component is automatically encrypted by the LoRaWAN chip.

2.1.1 Semi-realtime

The functional requirements specify that the sensors must have a time resolution of five minutes max. Because sending this many packets will consume to much energy and will also pollute the ether, the measurements are only sent when the value of the sensor changes (significantly).

When a change in the sensor value occurs, it will first be buffered and sent after five minutes. This allows the weather station to also sent other sensor values that also have changed in the five minutes. This also prevents that when the sensor value, for example, changes after a minute a new packet has to be sent. Instead, the weather station will overwrite of calculate a new average sensor value. The mechanism is shown in the following activity diagram:

2.1.2 Reliability

There also has to be a high reliability which among other things means that it is important that the measurements arrive on the other side. Because only simplex communication is possible on long ranges, it is impossible to receive any confirmations. This means that there is no guarantee that packets actually arrive.

It has been determined that external influences, such as thunderstorms, cause a statistical six percent loss of packets. To meet the functional requirement that at least 99 percent of the packets should arive, the packets must be sent 1.6 times. However, to cover any additional disruptions such as maintenance it has been decided to sent all measurements four times.

The following transmission times have been chosen:

Time (h:mm)	Motivation
0:00	First attempt.
0:05	Second attempt right after the first attempt so that the measurements are still received semi-realtime if the first attempt failed.
3:00	Third attempt after three hours so that any thunderstorms have dissipated.
20:00	Fourth attempt after 20 hours so that any long term disruptions or maintenance can be covered without any data los. The value can't be higher due to limits of the used data field.

The retransmission mechanism is shown in the following activity diagram:

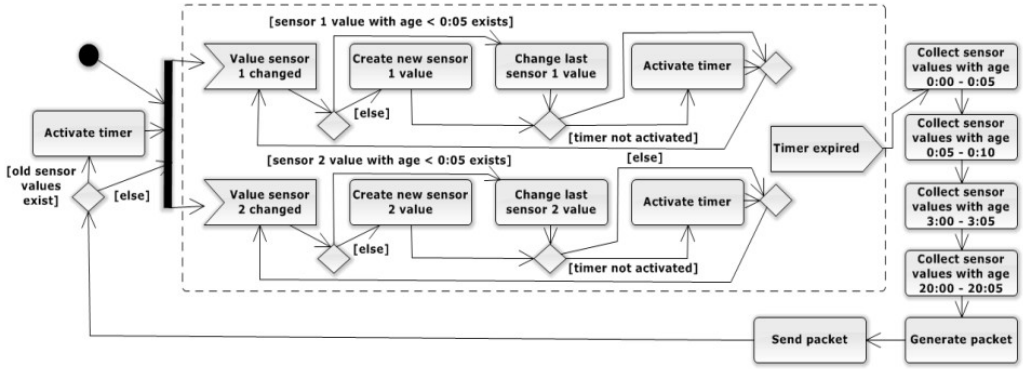


Illustration 4: The activity of the retransmission system

2.1.3 Protocol definition

The protocol handles various sensor values. These are concatenated in the packet. The protocol also has a version field which enables multiple versions to be used simultaneously. This makes making any adjustments easier.

The field that indicates the number of measurements in the packets follows after this. This field has a max value of 15 which means that a LoRaWAN packet can hold 15 measurements at max. Because the max size of a LoRaWAN is very limited and the average size of a sensor value is multiple bytes, this should be sufficient to generate full packets and thus be efficient with the available bandwidth and energy.

After this field the actual measurements follow. To couple a measurement to a sensor, every measurement is supplemented with a sensor type and a sensor number. The sensor type is predefined sensor type which also influences the way the measurement should be interpreted. Thirty two sensor types can be defined at max. Currently the following sensor types are defined:

Sensor Type	Name	Size	Field name	Field format	Field size	Unit	Description
0	System measurement	7	Battery voltage	float	4	Volts	The voltage of the battery powering the weather station
			Uptime	uint	2	decaSeconds	The uptime of the station in decaseconds
			Self diagnostic	bool	1		True if errors were detected, false otherwise
1	GPS sensor	12	Longitude	float	4	degrees	The longitude of the weather station
			Latitude	float	4	degrees	The latitude of the weather station
			Altitude	int	4	meters	The altitude of the weather station in meters
2	Temperature sensor	2	Temperature	int	2	centiKelvin	The temperature in centikelvin. This means that the max temperature 327.67 Kelvin is
3	Rain sensor	2	Rainfall	int	2	microMeter/ m^2/5 minutes	The rainfall in micrometer per square meter per five minutes

The sensor number is located in the next field. This field can hold 8 different values which means that of every sensor type only 8 can be connected to the weather station. If this is not sufficient, an additional sensor type has to be defined. The unique identifier of a sensor therefore is [weather station id] + [sensor type] + [sensor number].

The measurement time is stored in the next field. Because the weather stations lack a GPS sensor and duplex communication they cannot synchronize their time and thus cannot determine the actual time and date. Therefore the age of the measurements is sent.

Finally the actual sensor value itself is sent. The length and format of the field depends on the sensor type and, to save bandwidth, it not explicitly defined in the packet. The central server will have to determine the length and format of the field by retrieving this information from the database.

2.2 Base station – central server

The connection between the base station and the central server is realized using 3G, GPRS, satellite or a similar connection. Al these connections support the IP protocol. The central server offers a RESTful interface for storage of measurements. Here a simple and expendable protocol is desired. The standard LoRaWAN network configuration also uses the IP protocol. The data is then sent in JSON format to the application server. To keep some compatibility it has been decided to use the same protocol. However, because the central server is universal and thus does not depend on any LoRaWAN specific components, many fields are optional and not interpreted by the central server.

2.3 Central server – GUI

The communication between is central server and the GUI is realized using HTTPS. This because the GUI is a webpage and the central server is a RESTful webserver and thus both are very web oriented. The HTTPS standard provides protection against eavesdropping but not against man in the middle attacks or replay attacks. Therefore it has been decided that every message is accompanied by a HMAC hash and a self validating session token. The session tokes expires within 10 minutes and thus makes replay attacks ineffective. The HMAC hash is generated using a secret salt which is only known by the server and the GUI. This provides effective protection against data manipulation.

3 The weather station

The weather station runs on a Atmel SAMD21J18A micro controller. This micro controller is based on the ARM Cortex M0 processor and runs at 48 MHz max. With 32 kiB SRAM and 256 kiB flash memory the micro controller has high specifications and can be used in a wide range. The weather station uses the Microcip RN2483 LoRaWAN module for communication.

The weather station is designed in multiple phases. The firest phase will only be able to measure rain and transmit the data. The second version will be able to detect data corruption. The third version will be able to monitor the weather station and the final version will be able to use multiple sensors.

3.1 First version

The first version will run a simple program that performs the required tasks by using interrupts and timers. The functionality is shown in the following activity diagram:

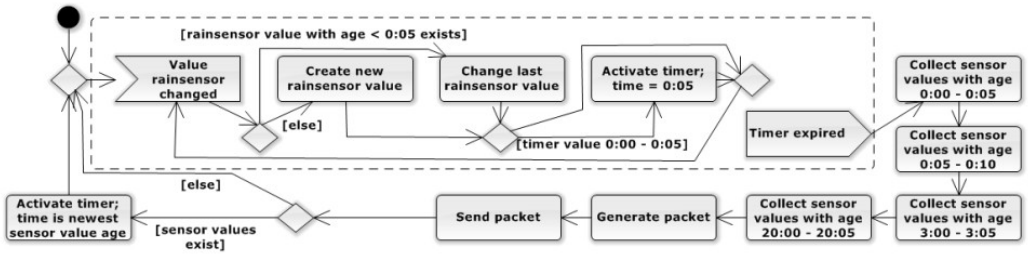


Illustration 5: The activity diagram of the weather station

If the weather station waits for a tick from the sensor, it will go into sleep mode. This save a considerable amount of energy. Both an interrupt from the rain sensor as a timer will wake the weather station.

The measurements are stored in a circular buffer which has sufficient capacity. The required capacity can be determined using the following formula:

$$sensorCount * (hours * 12) * (valueSize + 2)$$

To store twenty hours of measurements from ten sensors which generate a new sensor value with a size of ten bytes every five minutes, 28.8 kB of memory is required. Because the micro controller has 32 kiB of memory, this should be sufficient. Also, the scenario described above is a worst case scenario and the capacity will be required rarely. Therefore a circular buffer with less capacity should also suffice.

The operation of the code and the related classes are shown in the following class diagram:

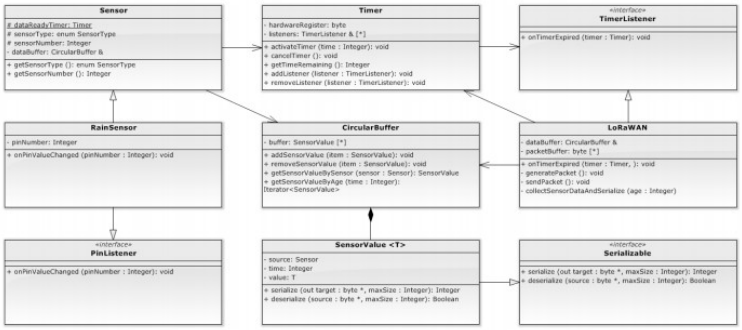


Illustration 6: The class diagram of the weather station

3.2 Second version

Data corruption can be caused by external influences such as space radiation. It has been determined that space radiation causes data corruption in SRAM every 59 years. This number might seem very low, but as data corruption may occur in other components aswell this is the min value.

To detect data corruption, the software uses CRC. CRC has a very high reliability of detecting data corruption and guarantees detection data corruption if the size of the data corruption is lower than the size of the checksum. If that is not the case, the change that data corruption may go unnoticed is still very low, as is determined with the following formula:

$$(errorBits/totalBits)^{errorStreak}$$

If a value of 128 bits has 32 corrupted bits and the checksum size is 16 bits, the change that the data corruption goes unnoticed is 0.000000002 percent. This value is simply neglectable.

The validation of the sensor values is performed just before transmitting the packet, as shown in the following activity diagram:

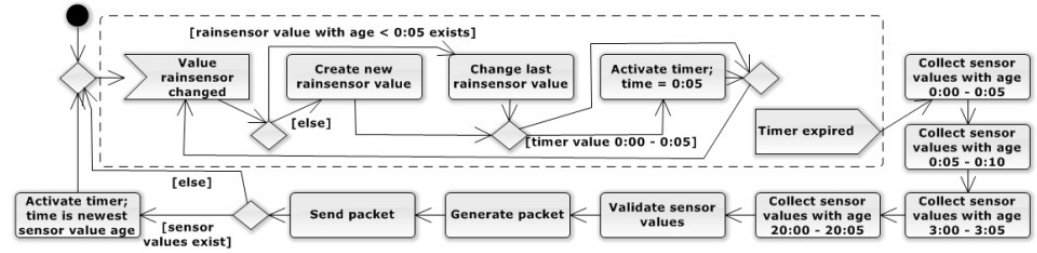


Illustration 7: The activity of the weather station including the data corruption detection

3.3 Third version

The third version will monitor the condition of the weather station This means that the battery voltage and uptime is monitored. Also, by applying immunity-aware programming, the pin registers, the interrupt registers, the interrupt vectors, the watchdog timers, the stackpointer and unused memory is validated periodically. The condition of the weather station is reported every 30 minutes.

The following activity diagram shows the operation of the weather station with the inclusion of the periodically self diagnostic:

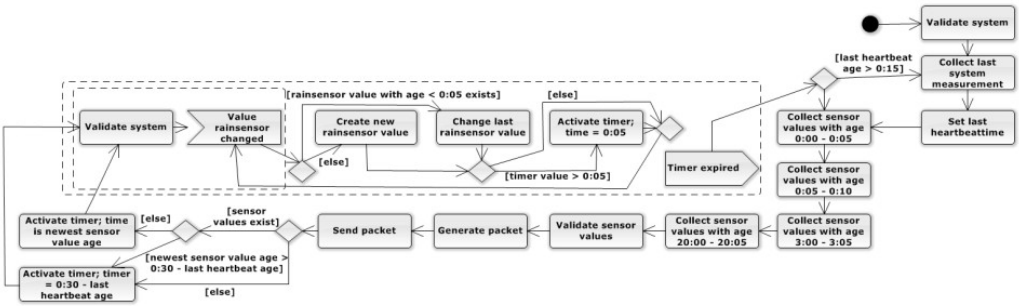


Illustration 8: The activity of the weather station including the self diagnostic

The operation of the self diagnostic is shown in the activity diagram on the right.

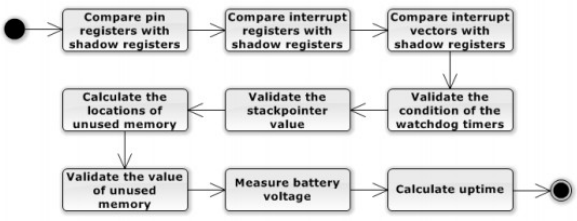


Illustration 9: The activity of the self diagnostic

3.4 Fourth version

The fourth version will support multiple sensors which can be added or remove online. This requires dynamic behavior from the software and also should comply with the deadlines of the sensors as well as support multi tasking and mutual exclusion. Realtime operating systems offer a solution for these demands and there it has been decided to use a RTOS for the weather station.

There are many realtime operating system implementations available, all with different possibilities and targeted applications. Multiple implementations have been compared for usage in the weather station. It has been decided to use FreeRTOS because it is actively maintained, is open source and is compatible with both the AVR- as the ARM instruction set. FreeRTOS also supports a tickless mode which should save energy.

By using a RTOS the software components of the weather station can be divided into multiple tasks as shown in the activity diagram below:

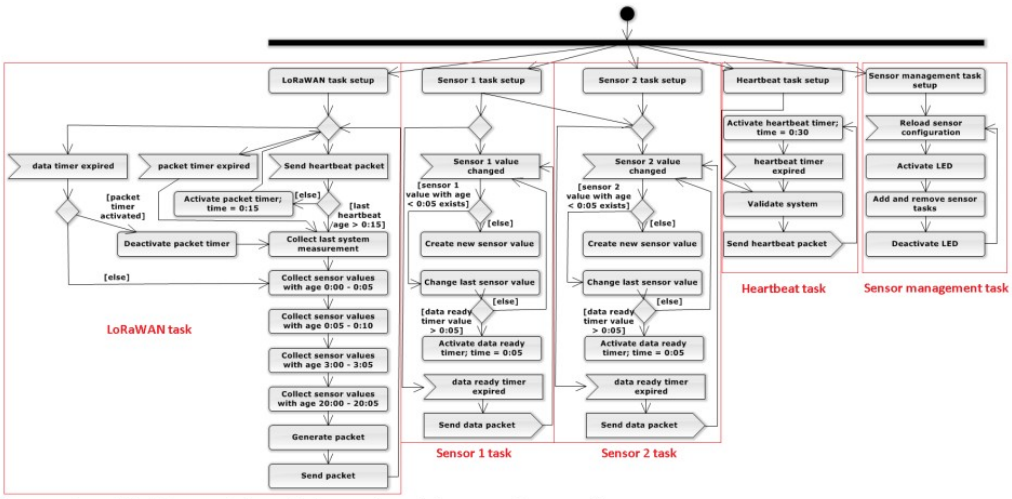


Illustration 10: The activity of the tasks of the weather station

The ability to add or remove sensors dynamically is accomplished by the sensor management task. By using an interface the sensor configuration can be adjusted. A LED will be lit to attend the user the change was processed. The sensors can be started or stopped by adding the task to or removing the task from the schedular. After the task has been removed, the used stack can also be removed

The LoRaWAN task will sent packets on request of the sensor tasks or the heartbeat task. To prevent that unnecessary packets are sent, system measurements (self diagnostics) are buffered for 15 minutes. If a measurement has to be sent in the meantime, the system measurement is added to that packet. A separate packet for the system measurement is then no longer required.

3.4.1 Concurrency

The communication between the various tasks is shown in the concurrency diagram on the right. The heartbeat task communicates with the LoRaWAN task via a channel. Because the LoRaWAN task will always process messages in the channel in fifteen minutes and the heartbeat task generates a message every 30 minutes, a capacity of one message should suffice.

The sensors communicate with the LoRaWAN task via the data ready flag. After the data ready timer expires, the flag will be set. This will cause the LoRaWAN to collect all sensor values and store them in the sensor values buffer. The data ready timer is activated by a sensor when it detects its value has (significantly) changed.

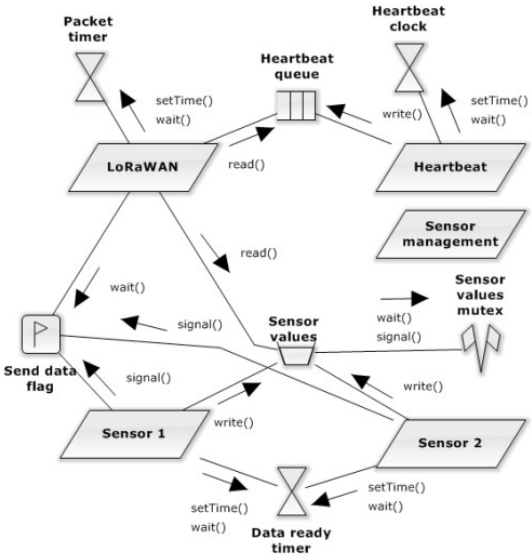


Illustration 11: The concurrency of the taks of the weather station

3.4.2 Classes

The classes and their interaction is shown in the class diagram below:

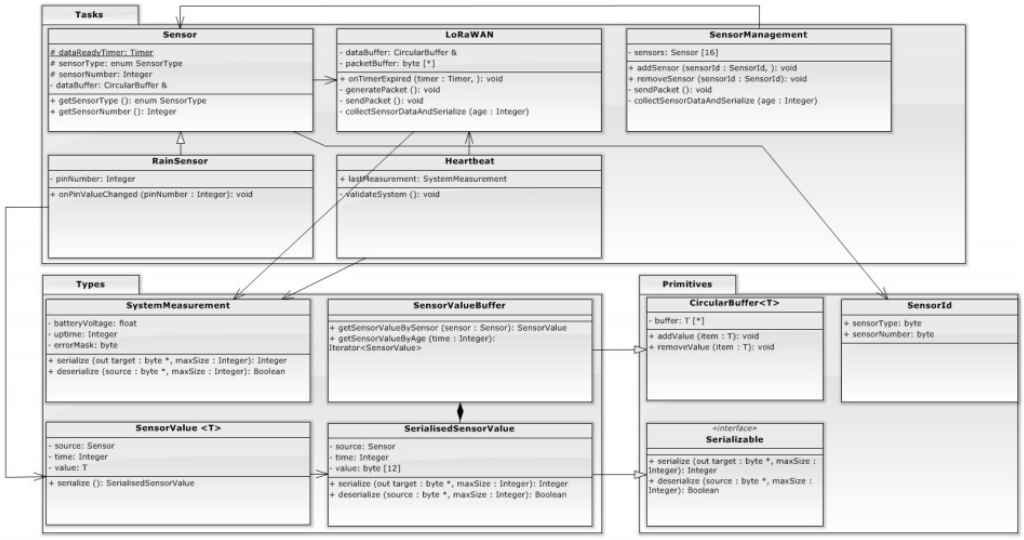


Illustration 12: The class diagram of the weather station

Currently the RainSensor is the only implementation of the Sensor class. In a later stage other implementations can be defined.

4 The base station

The base station must receive the packets from the weather stations and (eventually) transmit them to the central server via 3G/GPRS. The 3G/GPRS network is managed by a third party and thus there are no guarantees regarding the reliability. Therefore it has been decided that the network is unstable, unsafe and can have long downtimes.

Therefore the connection has been realized with the TCP protocol. This protocol guarantees the condition of the data when it arrives and also has some method to counteract dataloss. For security reasons all data is encrypted by sending it via a VPN tunnel. The VPN tunnel also prevents so called man in the middle attacks.

Because the network could have long downtimes, the data has to be buffered. It has been decided to buffer the data using SQLite because:

- There is limited RAM memory available but there is sufficient harddisk space available. This allows storage of large amounts of data.
- The base station could shutdown unexpectedly if the power supply fails. If the data were to be stored in RAM, it would be lost.
- SQLite guarantees the integrity of the data. If the base station were to shutdown unexpectedly the file, acting as buffer, could get corrupted. SQLite offers, like any ACID complaint databases, guarantees regarding data integrity.
- By using SQLite it is possible to use SQL queries. This make retrieving data from the buffer considerable easier.

4.1 Activity diagram

The activity diagram of the base station is shown below:

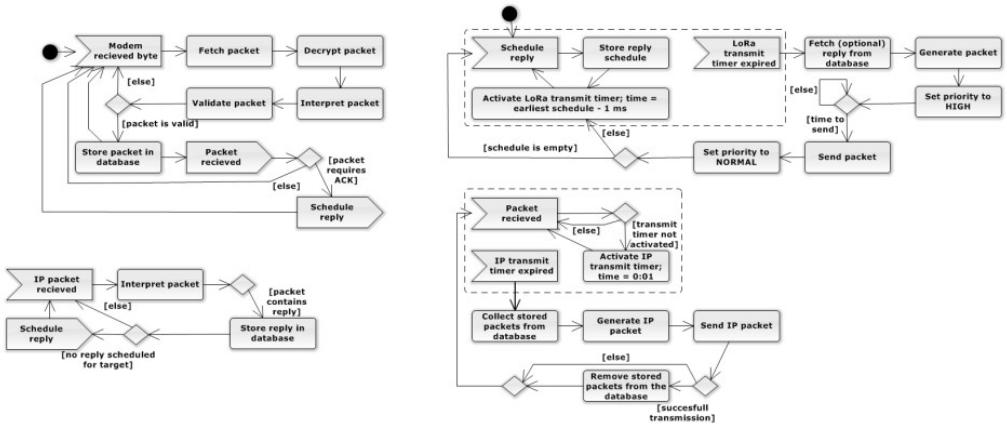


Illustration 13: The four tasks of the base station

The base station performs four tasks simultaneously:

- Receiving LoRaWAN packets from the weather station via the modem
- Transmitting LoRaWAN packets to the weather stations via the modem
- Receiving IP packets from the central server via the HTTP connection
- Transmitting IP packets to the central server via the HTTP connection.

Because the receive window of a LoRaWAN class A device (like the weather station) is only 20 microseconds long, it has been decided to expire the timer a few milliseconds before the actual transmit time. This allows the modem hardware to “warm up” and allows compensation of jitter in the timers caused by the nature of the schedular of the Linux kernel.

4.2 Class diagram

The class diagram below shows the classes and the interaction between the classes of the base station:

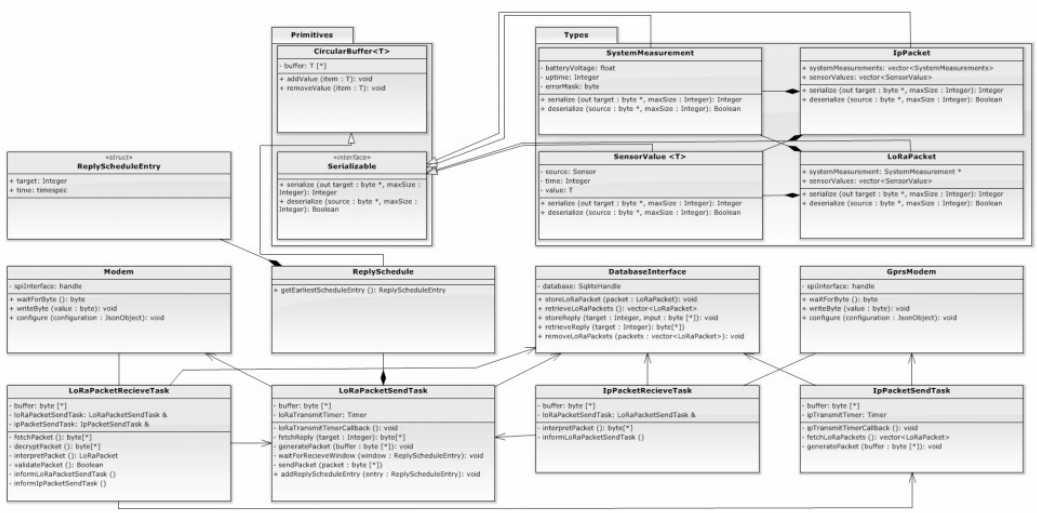


Illustration 14: The class diagram of the base station

5 The central server

The central server is the component that actually processes, stores and manages the measurements. The measurements are stored in the database and the server provides interfaces to allow the base stations to add measurements and the GUI to retrieve measurements.

5.1 Entity relation diagram

The measurements are stored in a relational database. This database must have a predefined structure. This structure is shown in the diagram below.

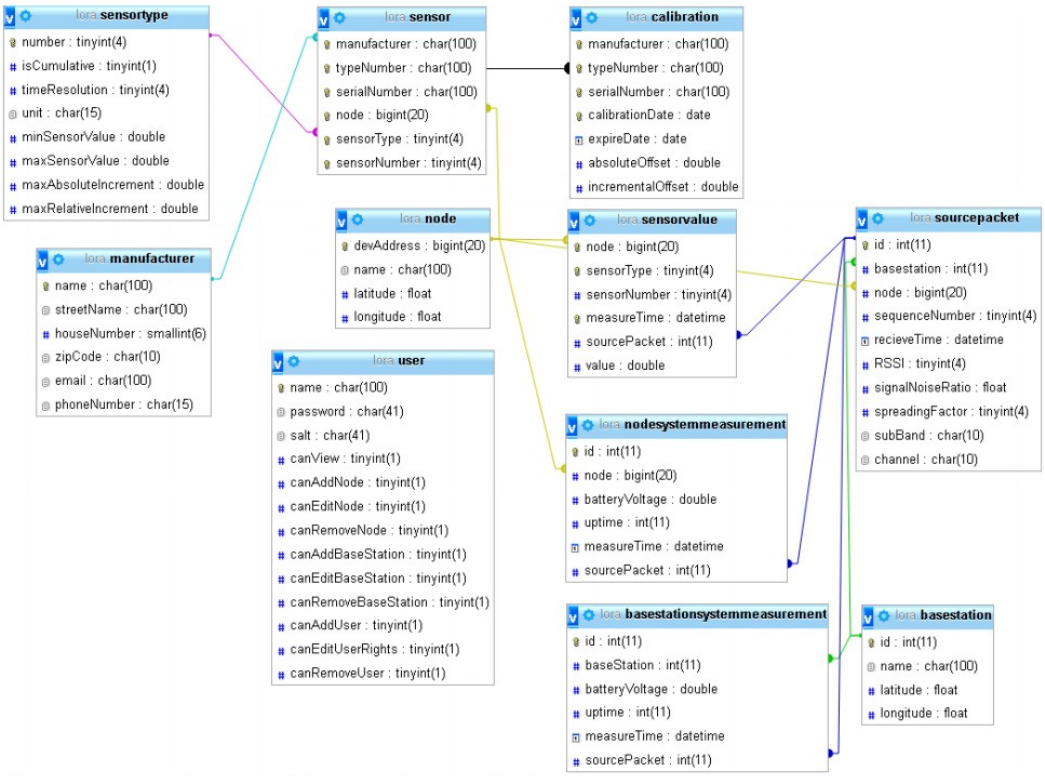


Illustration 15: The ERD of the central server database

While designing the structure of the database the fact that the database could also be used for different project has been taken in to account. Therefore there are no entities such as “WeatherStation” or “LoRaWanPacket”. This also allows the usage of different network technologies such as Zigbee.

5.2 Activity diagram

The actions that the server has to perform are relatively simple as the server uses the RESTful priciple. This simple prescribes that every action is autonomous and should not use any session variables. This means there are no dependencies between the actions. Both the base station and the GUI are designed to comply with this principle.

The base station collect the available LoRaWAN packets and sent them to the central server. These packets are sent in the JSON format.

To prevent unauthorized base station to insert measurements into the server, every base station must identify itself. This is done using the HMAC principle where the entire message is hashes using a secret salt. HMAC is a bettery alternative to for example sha256 as they are vulnerable to length extension attacks.

The log in procedure is also stateless and thus cookieless. This means that the client will have to authenticate itself every time is performs an action on the server. Because password should remain secret at all times and thus should not be stolen if the server were to be hacked, it has been decided to hash the passwords using sha256. This means that if the hash has been stolen, the actual password could not be determined from the hash without using a brute force attack.

When the username is entered, the salt of the user is retrieved from the server. The salt is then used to hash the password twice. The first hash is the private secret and the hash of the hash is the public secret. The public secret is sent to the server for authentication while the private secret is used for the message integrity code (MIC). The MIC prevents anyone from modifying the message sent by the user. This is done by hashing the message using HMAC. The private secret is used as the salt for the HMAC hash. The server will verify the MIC using the private secret stored in the database.

To prevent replay attacks, a session token is added to every message. The session tokens are self validating and expire in ten minutes. This limits the effectiveness of a replay attack considerably. To retrieve session tokens, the correct username and public secret must be provided. If they are invallid, the server will not provide a new session token. This system is shown in the activity diagram on the right.

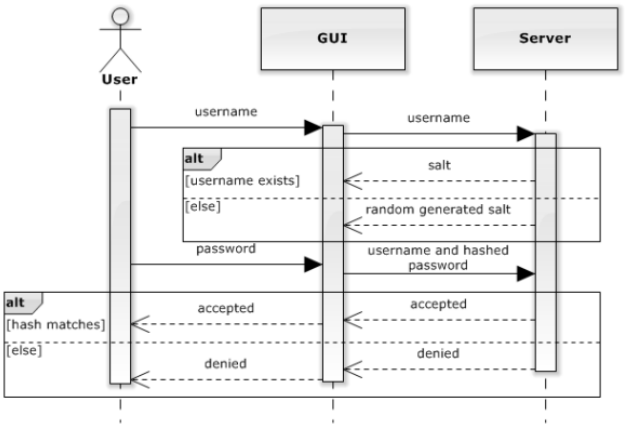


Illustration 16: Retrieving a token from the server (with the user entering username and password for the first time)

5.3 Class diagram

The class diagram below represent the model component of the server. It effectively represents the database and thus is very similar to the entity relation diagram shown above.

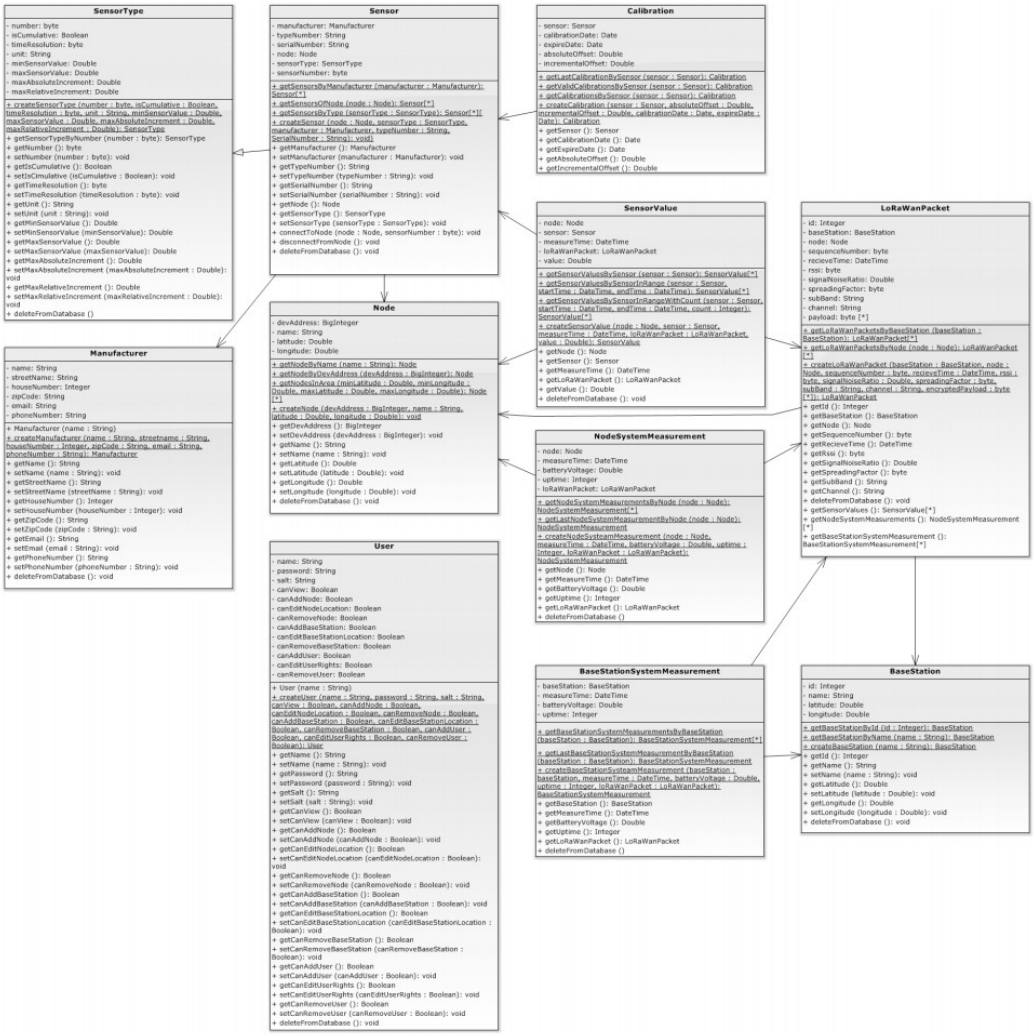


Illustration 17: Class diagram of the central server

6 GUI

When using the GUI the user has to be able to retrieve sensor values. The user should also be able to modify the configuration of the network and add or remove weather stations, base stations, sensors, sensor types and/or users.

It has been decided that the GUI will be an interactive webpage where on the right the data is shown in the form of a map and on the left in the form of a list. The design is shown below:

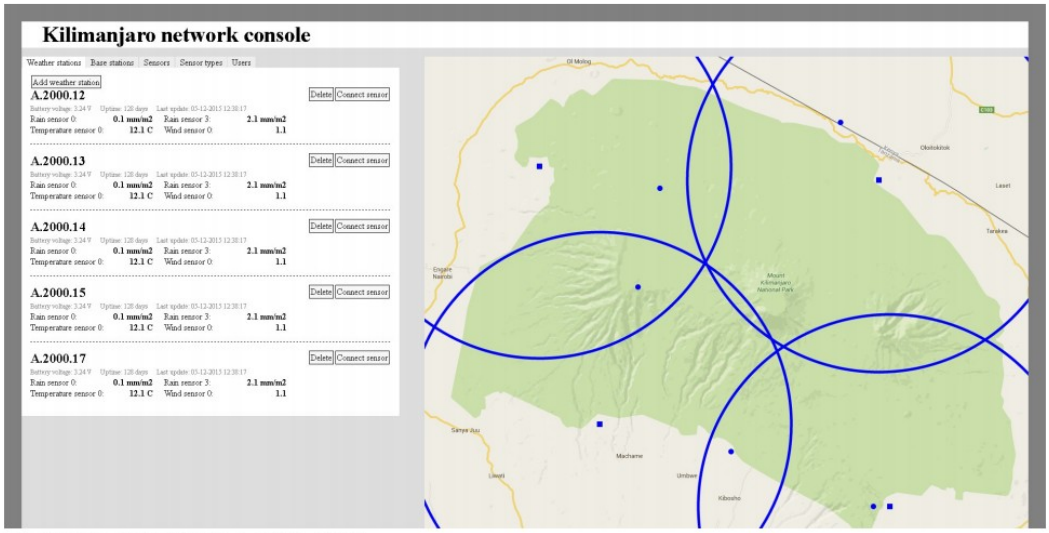


Illustration 18: Design of the GUI

Appendix E – Technische documentatie

Wireless Long Range Communication for Weather Stations

Technical documentation

Wireless Long Range Communication for Weather Stations

Technical documentation

Cover image:
Kilimanjaro mountain

Author:
Tommas Bakker

Student number:
1611239

Version:
1.0
(work in progress)

Date:
25-02-2016

Mentors:
Jan Willen Smeenk, M2M4ALL
Company mentor

Examinators:
Henk van Nimwegen, Hogeschool Utrecht
First examinator

Jan Zuurbier, Hogeschool Utrecht
Second examinator

Table of Contents

1 Weather station....4
1.1 Creating a C++ project with ASF in Atmel Studio....4
1.2 C++ and limitations in the environment....5
1.2.1 Pure virtual methods....5
1.2.2 Error detection and debugging....5
1.2.3 Dynamic Memory Allocation....5
1.3 Bugs in Atmel Studio and ASF....6
1.3.1 Breakpoints firing on the wrong location....6
1.3.2 FreeRTOS disabling all interrupts....6
1.4 Configuring the weather station software....7
1.5 Settings....7
1.5.1 HeartBeat.h....7
1.5.2 LoRaWan.h....8
1.5.3 Sensor.h....8
1.5.4 SerializedSensorValue.h....9
1.5.5 SerializedSensorValueBuffer.h....9
1.5.6 modems/Modem.h....9
1.6 modems/LoRaWanModem.h....9
1.6.1 modems/RN2483.h....10
1.6.2 sensors/RainSensor.h....10
1.7 Adding additional sensor types....11
2 Base station....12
2.1 The Linux environment....12
2.1.1 GPRS/3G connection management....12
2.1.2 Application management....13
2.2 Developing for the base station environment....14
2.3 The enhanced packet forwarder....15
2.3.1 Configuration....15
3 The central server....16
3.1 Configuration....16
3.2 Tests....16
4 The GUI....17
4.1 Files....17
4.2 Configuration....17

1 Weather station

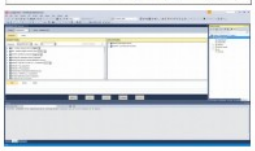
This chapter describes the programming of the weather station as well as some generic information. To get insight into the design of the weather station code, please consult the design document.

1.1 Creating a C++ project with ASF in Atmel Studio

Atmel studio offers a large library containing code of using various peripherals of the micro controller including WDT, ADC, RTC, etc. Having to reprogram all these functions is a waste of time and therefore it is strongly recommended to use the ASF library.

Unfortunately ASF by default is incompatible with a C++ project as the ASF wizards incorrectly reconfigures your project. Therefore some manual actions have to be performed:

1. Create a new C++ executable project and give it a fancy name.
2. Select the micro controller you are using. For the Autonomo this is the ATSAMD21J18A. You can limit the number of micro controllers by selecting the family.
3. Atmel studio will now create your project. This project is an empty C++ project and does not contain any ASF components.
4. Open the ASF wizard in the menu: ASF → ASF wizard
5. The ASF wizard will give a warning that ASF modules are not supported for C++. This is exactly what we are trying to fix. The actual code is compatible with C++ (for the most part atleast).
6. The ASF wizard will give a warning that no board has been selected. To continue, press OK.
7. Select the User Template Board and press Next.
8. Press Finish and let the ASF wizard reconfigure your project.
9. The ASF wizard will give a warning regarding multiple main functions. This is part of the problem and will be fixed later. Press Apply. The wizard will apply the changes to your project. This might take a while.
10. You can now add various components of ASF to your project by selecting them on the left panel and pressing Add on the bottom. When you are finished, press Apply.
11. Try to compile your project by pressing F7 or in the menu: Build → Build Solution. Compiling might take a while. When it finishes it should give various errors regarding duplicate definitions.
12. In the project explorer on the right hand side, open the Device_Startup directory.
13. Delete the startup_samd21.c and system_samd21.c files from the disk (removing them from the project is not sufficient). Do NOT remove or delete the samd21j18a_flash.ld or the samd21j18a_sram.ld file.
14. In the project explorer on the right hand side, open the src directory.
15. Remove the main.c file from the disk (removing them from the project is not sufficient).
16. Try to compile your project by pressing F7 or in the menu: Build → Build Solution. Compiling might take a while. This time the build should finish successfully.



You should now be able to use the ASF library with C++. Please note that the ASF library itself is compiled using C. Therefore any includes should be surrounded with extern statements. Furthermore, as the library uses function pointers for callbacks, you can only use static methods of functions and not normal methods.

```
extern "C" {
        #include "src/asf.h"
}
```

1.2 C++ and limitations in the environment

The environment has some limitations which might become a problem. This chapter describes workarounds and fixes for some of the limitations.

1.2.1 Pure virtual methods

A common programming technique in C++ is using pure virtual methods. This technique is often used when programming interfaces or other abstract classes of which a virtual method is declared but not defined (as the subclass must implement the method and thus define it).

As it is theoretically possible a pure virtual method is called (eg, the vtable of the object has no implementation of the method currently assigned), the C++ compiler will define an error handler. The default error handler will issue an error via the std::err stream. As the micro controller has no streams, this will result in a linker error. The solution is to redefine the error handler.

The error handler is called __cxa_pure_virtual(). To be properly notified in case of a pure virtual method being called, a breakpoint could be issued. The example automatically issues a breakpoint whenever the method is called.

```
extern "C" void __cxa_pure_virtual() { asm("BKPT #0"); }
```

1.2.2 Error detection and debugging

Error detection and debugging are essential for properly testing any application. Because no std::cout or std::err streams are defined, using debug lines is not possible. Using a debugger such as the Atmel ICE is essential for proper application development. With the Atmel ICE the usage of breakpoints is possible. Also manually stepping through code is possible.

To detect errors, the ASF library offers a macro called Assert(x). The macro is defined in the ASF/sam0/utils/compiler.h file. The macro will automatically issue a breakpoint when the condition x equals false. By properly using sanity checking, many possible undefined behaviors can be prevented by using this macro.

```
Assert(inputPtr != NULL);
```

1.2.3 Dynamic Memory Allocation

The micro controller by default does not support dynamic memory allocation. This is often not a problem as dynamic memory allocation in C or C++ is rarely required. However, many STL classes such as the vector require dynamic memory allocation. These are therefore not supported.

A solution could be to implement your own new and delete operator. Another possible solution is to use the heap managed by FreeRTOS. Both solutions have not been further researched and thus no further information is available.

1.3 Bugs in Atmel Studio and ASF

Like any other piece of software both Atmel Studio and the ASF library contain bugs and other related issues.

1.3.1 Breakpoints firing on the wrong location

Sometimes breakpoints are fired on the wrong location. Often this is the result of the compiler optimizing the code and thus shuffling some statements. This can be counteracted by disabling the optimization of the compiler, although this is not recommended. An other solution is changing the location of the actual breakpoint.

If breakpoints are being fired on completely different locations or in assertions of which it is known that the assertion should not occur, it is most likely that Atmel Studio is not correctly firing the breakpoints. The best solution is to restart Atmel Studio and remove and then reinstate all the breakpoints in the project.

1.3.2 FreeRTOS disabling all interrupts

The ASF library includes various version of FreeRTOS. This is an open source Real Time Operating System and can be used to process multiple tasks parallel. The FreeRTOS port for the Cortex-M0 (on which the ATSAMD21J18A is based) has a bug which comes into effect whenever FreeRTOS functions are used before the scheduler is started.

When a FreeRTOS function, such as creating a timer, is called, the FreeRTOS will attempt to allocate memory on the heap. To prevent any interrupts which might also attempt to allocate memory on the heap, all interrupts are disabled. As functions which disable interrupts may call other functions which also disable interrupts, FreeRTOS keeps track how many times the interrupts are disabled and enabled using a nesting variable called uxCriticalNesting. This variable is incorrectly initialized causing the nesting mechanism to fail. When the scheduler starts, the variable is correctly initialized.

Because the nesting mechanism fails the micro controller will fail to detect any interrupt and will become completely deaf. To fix this problem, open the FreeRTOS port file (ASF/thirdparty/freertos/freertos-8.0.1/Source/portable/GCC/ARM_CM0/port.c) and modify initialization of the uxCriticalNesting variable so that it initializes to 0 instead of 0xaaaaaaaa.

1.4 Configuring the weather station software

The behavior of the weather station software can be modified by modifying the definitions in the code. This chapter will summarize all the definitions and describe them. Also the procedure for adding additional sensor types is explained.

1.5 Settings

Various settings can be set in various files. This chapter will summarize all the definitions and describe them.

1.5.1 HeartBeat.h

Name	Default Value	Description
HEART_BEAT_INTERVAL	30 * 60	This definition determines the frequency at which a system measurement is performed (and sent). The value is in seconds.
BATVOLT_R1	4.7	This definition determines the value of R1 used when measuring the battery voltage. R1 is used by the voltage divider.
BATVOLT_R2	10	This definition determines the value of R2 used when measuring the battery voltage. R2 is used by the voltage divider.
ADC_SAMPLES	1	This definition determines how many samples are taken to measure the battery voltage. Do not change this definition as the software will always only use the first value.
ADC_AREF	3.3	This definition determines the voltage of the reference used by the ADC to measure the battery voltage. This voltage should equal the voltage the micro is powered with. The ADC will properly divide the voltage of the input to make sure it is within range.

1.5.2 LoRaWan.h

Name	Default Value	Description
RETRANSMIT_DURATION	10	This definition determines the waiting time whenever a LoRaWAN packet could not be sent because the modem was unable to complete the request (because the device is busy, the network does not respond (only applies when using acknowledged messages), etc). The value is in seconds
RETRANSMISSION_ATTEMPTS	3	This definition determines the number of times sensor values are retransmitted. The number of actual transmissions is RETRANSMISSION_ATTEMPTS + 1.
RETRANSMISSION_TIMES	{ 5, 180, 1200 }	This definition determines the times a sensor value is retransmitted. The values are in minutes.
SYSTEM_MEASUREMENTS_QUEUE_LENGTH	1	This definition determines size of the system measurements queue. Do not change this definition as you should always choose a buffer time that is smaller than the frequency.
APPLICATION_PROTOCOL_VERSION	0	This definition determines the version of the application protocol used for the communication between the weather station and the base station. This value is used in the header.
MAX_APPLICATION_ENTRY_COUNT	`(1 << 4) - 1`	This definition determines the max number of sensor value entries in a single LoRaWAN packet. This number should not exceed the max value of the segment in the LoRaWAN packet header that holds the number of sensor values.

1.5.3 Sensor.h

Name	Default Value	Description
LOW_SENSOR_PRIORITY	1	This definition determines the task priority of sensor taks with low priority.
MEDIUM_SENSOR_PRIORITY	2	This definition determines the task priority of sensor taks with low priority.
HIGH_SENSOR_PRIORITY	3	This definition determines the task priority of sensor taks with low priority.
SENSOR_DATA_PACKET_TIMER_PERIOD	5 * 60	This definition determines the maximum buffer time of sensor values. The value is in seconds.

1.5.4 SerializedSensorValue.h

Name	Default Value	Description
MAX_VALUE_SIZE	12	This definition determines the max size of a sensor value when it is serialized. The size is chosen so that three floats can be stored. GPS sensors for example uses three floats. The value should be kept to a minimum as increasing it will have significant impact on memory usage.

1.5.5 SerializedSensorValueBuffer.h

Name	Default Value	Description
SERIALIZED_SENSOR_VALUE_BUFFER_CAPACITY	600	This definition determines the capacity of the serialized sensor buffer. This number determines number of sensor values that can be stored for retransmission. If the weather station uses many sensors which often change their value, this number must be high. However, the value should be kept to a minimum as increasing it will have significant impact on memory usage.

1.5.6 modems/Modem.h

Name	Default Value	Description
MAX_MESSAGE_SIZE	50	This definition determines the maximum size of a (LoRaWAN) packet. This number should match the actual maximum size because else the software will generate invalid packets and will continue attempting to send them

1.6 modems/LoRaWanModem.h

Name	Default Value	Description
DEFAULT_NETWORK_SESSION_KEY	{0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE, 0xD2, 0xA6, 0xAB, 0xF7, 0x15, 0x88, 0x09, 0xCF, 0x4F, 0x3C}	This definition determines the network session key used by the modem.
DEFAULT_APPLICATION_SESSION_KEY	{0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE, 0xD2, 0xA6, 0xAB, 0xF7, 0x15, 0x88, 0x09, 0xCF, 0x4F, 0x3C}	This definition determines the application session key used by the modem.

1.6.1 modems/RN2483.h

Name	Default Value	Description
RN2483_SERCOM_INTERFACE	SERCOM5	This definition determines the hardware interface used by the modem. Because on the Autonomo the RN2483 is used on the bee socket and the pints are connected to PAD0 and PAD1 of SERCOM5, this value should be used.
READ_TIMEOUT_DURATION	10	This definition determines the timeout duration for the modem. This value is in seconds. If the timeout occurs, the method will return a TIMEOUT. The software will then reset the micro controller (and thus the modem)
MAX_RX_BUFFER_LENGTH	16	The definition determines the capacity of the input buffer. This value should be large enough to hold any responses from the modem.
MAX_TX_BUFFER_LENGTH	16 + MAX_MESSAGE_SIZE * 2	The definition determines the capacity of the output buffer. This value should be large enough to hold any possible output. Because for every byte in a LoRaWan packet two characters are required (because of the conversion to hex) and the size of the actual command, the default value should suffice.

1.6.2 sensors/RainSensor.h

Name	Default Value	Description
AMOUNT_OF_RAIN_PER_TICK	1	This definition determines the amount of rain that is collected per tick. The value depends on the hardware used by the sensor.

1.7 Adding additional sensor types

To add additional sensor types, first create the files in the sensors directory. The sensor type must derive from the Sensor base class. The class name must match the file name and all virtual methods must be implemented. Also make sure the initialize of the base class is called when the sub class is initializing. The sub class should not perform any tasks or construct objects in the constructor. Instead these should be performed in the initialize method.

After adding the necessary files, the sensor type must be registered to the system. The sensors/Sensors.h file determines which sensors are registered and with what sensor number. To add a new sensor type simple add the definition to the file. The file will make sure that the header file is included and that memory is allocated for the instances of the sensor type.

```
#define SENSOR_TYPE_3 RainSensor
```

For each sensor type a number of sensor is created. The system will select an instance whenever a sensor of that type is created. The instances of all the sensors are stored in the sensorTypes variable.

2 Base station

The base station runs an embedded Linux kernel on which an application interfaces with the modem via a SPI interface. This chapter will describe some features offered by the Linux environment, how to develop for this environment, and will describe the software that has been programmed for the base station.

2.1 The Linux environment

The Linux environment offers various features for device management and application management. These features are described in this chapter.

2.1.1 GPRS/3G connection management

To establish a GPRS/3G connection the base station has to communicate with the SIM card and the GPRS/3G hardware. Both have to be configured with various settings such as the APN and the PIN. The procedure to establish a GPRS/3G connection is described below.

1. Shutdown the base station and insert a micro SIM card. It is essential that the base station is shut down to prevent possible damage and the base station will only detect SIM cards during boot. To remove the bracket from the base station, push on the left of the bracket. Place the micro SIM card in the bracket and reinsert it.
2. Turn on the base station and log in via either the debug probe or via SSH.
3. Modify the /etc/sysconfig/network file with vi. Set the GPRSAPN to the APN of the provider. Set the GPRSPIN to the PIN of the SIM card. Set the GPRSUSER and GPRSPASSWORD to the username and password specified by the provider.

```
# Selector operator APN
GPRSAPN=m2minternet
# Enter pin code if activated
GPRSPIN=
# Update /etc/resolv.conf to get dns facilities
GPSDNS=yes
# PAP authentication
GPRSUSER=kerlink
GPRSPASSWORD=password
```

4. Save the file.
5. Modify the /knet/knetd.xml file with vi. Set the CONNECT auto_connection property to YES. This will issue the KNET deamon to automatically connect to the network. By default the metric of the eth0 interface is higher and thus all traffic will be diverted to that interface. If the eth0 interface goes down, the traffic is automatically routed to the GPRS/3G interface.

```
<!-- ############## connection parameters ############## -->
<!-- enable the autoconnect feature (YES/NO) -->
<CONNECT auto_connection="YES" />
<!-- frequency of connection monitoring -ping- (in seconds) -->
<CONNECT link_timeout="30"/>
<!-- DNS servers will be pinged if commented or deleted. Some operators can block the ping on there DNS servers -->
<CONNECT ip_link="192.168.4.90"/>
<!-- ############## default area for connection policy ############## -->
<AREA id="default">
<ACCESS_POINT bearer="gprs" />
</AREA>
```

6. Set the ACCESS_POINT bearer to gprs to allow the system to manage the interface.
7. Reboot the base station to make the changes take effect. After booting, log in to the system.
8. Execute the ifconfig command. An interface called ppoa should be present. This command also allows monitoring of bandwidth consumption.

2.1.2 Application management

The Linux environment offers a management agent that can manage applications by starting, restarting and stopping them. This agent also allows the monitoring of applications and automatic restarting if the application crashes. This guarantees that the application always runs whenever the base station is online.

The agent manages applications by searching for files with the name “manifest.xml” in the /mnt/fsuser-1/ directory. This file must contain the startup command, the startup parameters and the signal to be used to stop the program.

```
<?xml version="1.0"?>
<manifest>
        <app name="my_app" appid="1"
binary="executable.bin" >
                <start param="-f /mnt/fsuser-
1/my_app/config.txt" autostart="y"/>
        <stop kill="9"/>
        </app>
</manifest>
```

2.2 Developing for the base station environment

The Linux environment runs on a ARM based processor using a somewhat old version of Linux. The base station itself is not capable of compiling code and thus has to be cross compiled. It is therefore required that the toolchain for the base station is installed and configured on the host computer. To install and configure the toolchain, perform the following steps.

1. Retrieve the username and the password for the Kerlink Wiki. These can be found on the LoRa Google Drive directory.
2. Log in on the Wiki server
3. Download the following file: ftp://wiki-ftp-wirgrid:uBHWPzEzaHy3Ac6dEn7W@ftp.kerlinkm2mtechnologies.fr/_kerlink/arm-2011.03-wirgrid_r79.tar.xz via http://wikikerlink.fr/lora-station/doku.php?id=wiki:ressources
4. Extract the file at a suitable location (by typing tar -xvf <filename>)
5. To use the toolchain, the shell environment must be configured. This can be accomplished in multiple ways:
   - The shell can be configured manually by manually settings the required environment variables. These can be set using the export command. For example, to set the LDFLAGS command to include the myLibDir/ directory, issue the following command: “export LDFLAGS=-LmyLibDir ${LDFLAGS}”.
   - The shell can be automatically configured by using a CMake toolchain file. This requires you to use CMake as your buildtool aswell. The toolchain file specifies the include and lib paths aswell as the compiler and linker that should be used. The file must be included into the CmakeLists.txt file. This can be accomplished by issuing the following command: “ cmake -DCMAKE_TOOLCHAIN_FILE=toolchain-arm-linux-gnueabi.cmake”.

```
# CMake toolchain file for building ARM software on OI environment

# this one is important
SET(CMAKE_SYSTEM_NAME Linux)
#this one not so much
SET(CMAKE_SYSTEM_VERSION 1)

# specify the cross compiler
SET(CMAKE_C_COMPILER   /opt/toolchains/arm-2011.03-wirgrid/bin/arm-none-linux-gnueabi-gcc)
SET(CMAKE_CXX_COMPILER /opt/toolchains/arm-2011.03-wirgrid/bin/arm-none-linux-gnueabi-g++)
SET(CMAKE_STRIP /opt/toolchains/arm-2011.03-wirgrid/)

# where is the target environment
SET(CMAKE_FIND_ROOT_PATH  /opt/toolchains/arm-2011.03-wirgrid/)

# search for programs in the build host directories
SET(CMAKE_FIND_ROOT_PATH_MODE_PROGRAM NEVER)
# for libraries and headers in the target directories
SET(CMAKE_FIND_ROOT_PATH_MODE_LIBRARY ONLY)
SET(CMAKE_FIND_ROOT_PATH_MODE_INCLUDE ONLY)
```

6. After compilation the binary file has to be uploaded to the base station itself. The best way to do this is to use the scp command. To, for example, upload the file app.bin to the gateway at 192.168.1.70, issue the following command: “scp app.bin root@192.168.1.70:/mnt/fsuser-1/”. You will be asked to enter the password if required.

2.3 The enhanced packet forwarder

The enhanced packet forwarder is partly based on the basic packet forwarder provided by semtech/kerlink. Both packet forwarders use the by semtech/kerlink provided hardware abstraction layer (HAL) library.

The enhanced packet forwarder differs from the basic packet forwarder in that it is designed to deal with long term network failure between the base station and the central server. The enhanced packet forwarder will process and temporarily store packets and can schedule acknowledgments independently while the basic packet forwarder entirely depends on the central server to manage those tasks.

Further differences are that the enhanced packet forwarder can forward packets to multiple central servers and receive commands from multiple central server while the basic packet forwarder cannot.

The enhanced packet forwarder uses HTTP RESTful JSON interfaces to communicate to the central server (when it is available). For the communication the default HTTP port 80 is used.

2.3.1 Configuration

The configuration of the enhanced packet forwarder is done by using two configuration files. These are called local_configuration.json and global_configuration.json. These files are similar to the files used by the basic packet forwarder but are NOT compatible as the target servers and the radio configuration is stored in a different format.

The global_configuration.json file is the configuration file that applies to every base station in the network. It contains the configuration for the radio (modem) and contains the server configuration. The local_configuration.json file is the configuration file that applies only to this base station. It contains the ID of the base station as well as the secret with which the base station authenticates itself when communicating with the central server.

3 The central server

The central server is a RESTful webserver developed in PHP. This chapter will describe the configuration options of the server as well as the unit tests defined for the webserver.

3.1 Configuration

The configuration of the central server is managed by the config.json file in the model directory. It is therefore very important that users cannot access these files as that would allow them to steal the database username and password with all posible consequences. The configuration file specifies the database username, the database password, the database name and the database URL. It also specifies the tokenDuration in minuts. This is the time a the self validation tokes are valid. The server secret is also stored in the configuration file. This value is used as salt for the generation of self validating session tokes. This value must remain secret as otherwise other can generate valid session tokens making the central server susceptible to replay attacks.

3.2 Tests

During the development of the central server, various unit tests have been developed. These test both the model component of the server as well as the controller component. The unit tests reside in the test directory.

The following tests are available:

- controller add and remove.php: This test will communicate with the controller component and will add and remove various things. If the result of the test is an empty page, the test was succesfull. You have to specify the URL of the server at the top of the file.
- controller add testcase.php: This test will communicate with the controller component and will add a testcase of 4 base stations at a specified location and 65 weather stations all with a rain sensor at random locations. This testcase can be used for demonstration purposes. If the result of the test is an empty page, the test was successful. You have to specify the URL of the server at the top of the file.
- controller get.php: This test will communicate with the controller component and will retrieve various things. If the result of the test is an empty page, the test was successful. You have to specify the URL of the server at the top of the file.
- model create and select.php: This test will communicate with the model component and will add and then retrieve various things. If the result of the test is an empty page, the test was successful. You have to specify the URL of the server at the top of the file.
- model create and set.php: This test will communicate with the model component and will add and then modify various things. If the result of the test is an empty page, the test was successful. You have to specify the URL of the server at the top of the file.

4 The GUI

The GUI is an interactive webpage developed using JQuery and various other libraries for map generation, graph generation and cryptic functionality.

4.1 Files

The GUI exists, besides various library files and image files of six files: an index file and a file for every tab. To increase loading speeds the files are merged on the server and then sent as one single large file to the client. The merging is done via php inclusions.

4.2 Configuration

The GUI has a single configuration parameter: The token refresh interval. This property determines how often the GUI retrieves a new token fron the server. This value must be smaller than the tokenDuration value in the central server as otherwise the tokens will expire before the GUI retrieves a new token, resulting in access denied errors.