LoWPAN

Draadloze communicatie voor Amigo

Ralph Zuidinga

30-3-2012

Dit document is gepubliceerd door:
TASS BV
Eindhoven, Nederland

Commentaar en suggesties kunnen worden gestuurd naar:
TASS B.V.
Postbus 80060
5600 KA Eindhoven
Nederland
Tel: +31 40 2503200
Fax: +31 40 2503201

AFSTUDEERVERSLAG VOOR FONTYS HOGESCHOOL ICT

Gegevens student

Naam + voorletters student	Zuidinga, R.W.
Studentnummer	1338235
Opleiding	Informatica (deeltijd)
Afstudeerperiode	8-10-2010 tot en met 30-3-2012

Gegevens schoolbegeleider

Naam	Jos Boonen

Gegevens bedrijf

Naam bedrijf / instelling	TASS Technology Solutions
Plaats	Eindhoven

Gegevens bedrijfsbegeleider

Naam	Peter Vink
Functie	Domotica Expert

Gegevens verslag

Titel	6LoWPAN: Draadloze communicatie voor Amigo
Datum uitgifte	30-3-2012

Getekend voor gezien door de bedrijfsbegeleider, Peter Vink:

Datum:

Voorwoord

Dit verslag heb ik geschreven in het kader van mijn afstudeeropdracht als student van de opleiding HBO Informatica aan de Fontys Hogeschool. Binnen het bedrijf TASS Technology Solutions te Eindhoven is er zelfstandig een onderzoek- en ontwikkelopdracht uitgevoerd naar de mogelijkheden om het bestaande Amigo netwerk van TASS uit te breiden met low-power draadloze sensoren.

Via deze weg wil ik mijn dank uitbrengen aan iedereen die een bijdrage heeft geleverd aan mijn afstuderen en de totstandkoming van dit verslag. Speciale dank gaat uit naar mijn bedrijfsbegeleider Peter Vink en schoolbegeleider Jos Boonen voor de uitstekende begeleiding. Daarnaast wil ik Piet Boonen en Herman Roebbers bedanken voor de technische ondersteuning en Tiny Henst voor het bewaken van de voortgang. Vanaf deze plaats wil ik tenslotte mijn vrouw Kirsten bedanken, die mij zo gesteund heeft en gedurende mijn studie heel veel extra taken op zich heeft genomen, zodat ik ongestoord mijn opleiding kon voltooien.

Ralph Zuidinga
Maart 2012

Inhoudsopgave

Samenvatting

De in dit document beschreven afstudeeropdracht is uitgevoerd bij TASS Technology Solutions te Eindhoven. TASS is een top-3 dienstverlener op het gebied van technische en embedded software. Uit eerdere onderzoeken is het zogenaamde Amigo platform ontwikkeld waar diverse aspecten van een toekomstig thuisnetwerk hun plaats hebben gekregen. Het Amigo platform maakt gebruik van webservices (SOAP, WS-Discovery).

Het bestaande Amigo platform moet uitgebreid worden met low-power draadloze sensoren. De onderzoeksvraag luidt: “Welke draadloze technologie is het beste geschikt voor uitbreiding van het Amigo platform?”. De technologie moet aan twee criteria voldoen. Ten eerste moeten de sensoren meerdere jaren kunnen werken zonder vervanging van de batterijen. En ten tweede moet de technologie compatibel zijn met het bestaande Amigo platform. Tijdens het onderzoek heb ik 6LoWPAN en ZigBee met elkaar vergeleken. Beide technologieën voldoen aan het eerste criteria. 6LoWPAN heeft als grote voordeel dat het gebaseerd is op IPv6. De sensor, hoe klein dan ook, krijgt zijn eigen IPv6 adres. 6LoWPAN voldoet hiermee ook aan het tweede criteria, terwijl bij ZigBee nog een vertaalslag nodig is. Nadelen van zowel 6LoWPAN als ZigBee zijn een kleine pakketgrootte en een lage bandbreedte (127 bytes / 250 kbps).

Het was erg lastig om geschikte hardware te vinden. Ik heb maar vier bedrijven gevonden met een 6LoWPAN development kit. Twee van de vier waren uiteindelijk niet meer verkrijgbaar doordat het bedrijf recentelijk was overgenomen. Van de twee overgebleven fabrikanten is op basis van de prijs gekozen voor de development kit van Atmel. Deze kit bestaat uit twee temperatuursensoren en een USB stick met een radiochip. De USB stick is verbonden met een Plug Computer. Op deze manier kan de Plug Computer het draadloze 6LoWPAN netwerk verbinden met het TASS netwerk. Nadeel van de gekozen development kit is dat de sensoren aangestuurd worden door twee microcontrollers. Nu moet ook de onderlinge communicatie tussen de microcontrollers geïmplementeerd worden.

De software van de temperatuursensor heb ik geschreven in de programmeertaal C en maakt gebruik van het Contiki besturingssysteem. Om via het netwerk met de temperatuursensor te communiceren heb ik een PC applicatie geschreven in C#. Amigo maakt gebruik van WS-Discovery om te ontdekken wat voor services beschikbaar zijn in het lokale netwerk. De WS-Discovery standaard beschrijft twee modes: Ad Hoc op basis van IPv6 multicast, en Managed Mode op basis van IPv6 unicast en een Discovery Proxy. Het bestaande Amigo platform maakt gebruik van Ad Hoc. Tijdens de implementatiefase ben ik tegen het probleem aangelopen dat er geen IPv6 multicast berichten bij de sensor aankomen. Als oplossing heb ik ervoor gekozen om over te stappen op Managed Mode en zelf een Discovery Proxy te implementeren.

De belangrijkste conclusie is dat 6LoWPAN geschikt is voor het beoogde doel! Daarnaast is het gelukt om de door Amigo gebruikte standaarden te implementeren op de temperatuursensor. Dit ondanks het feit dat de sensor maar voorzien is van een beperkte hoeveelheid flash en geheugen. Als laatste is gebleken dat de nadelen van 6LoWPAN geen rol spelen bij het versturen van een meetwaarde. De belangrijkste aanbeveling is om WS-Eventing te implementeren zodat de sensor bij temperatuurveranderingen zelf een meetwaarde kan opsturen.

Verklarende woordenlijst

6LoWPAN	Standaard voor draadloze verbindingen tussen apparaten op korte afstand.
Daemon	Programma dat op de achtergrond draait om bepaalde taken uit te voeren.
Domotica	De integratie van technologie en diensten binnen de woning, ten behoeve van een betere kwaliteit van wonen en leven. Het woord domotica is een samenvoeging van het Latijnse woord voor huis 'domus' en informatica, telematica en robotica.
IPv4	Internet Protocol versie 4.
IPv6	Internet Protocol versie 6 is de opvolger van het huidige IPv4.
Kernel	De kern van een besturingssysteem.
KiB	Kibibyte (KiB), een eenheid van informatie gelijk aan 1024 bytes.
Middleware	Software die de communicatie tussen twee verschillende systemen regelt.
Multicast	Multicast is een techniek waarbij je één pakket verstuurt en dat het netwerk er automatisch voor zorgt dat dit bij meerdere ontvangers aankomt.
Multitasking	Multitasking is een methode om één processor schijnbaar meerdere taken tegelijkertijd te laten uitvoeren.
Non-preemptive multitasking	Bij non-preemptive multitasking blijft de processor met één taak bezig totdat de taak klaar is, of zelf besluit om de microcontroller vrij te geven.
Open source	Bij open source mag je de broncode (de 'kern' van de software) inzien, gebruiken, verbeteren, aanvullen en verder verspreiden.
Proces	Eén of meer reeksen opdrachten, die door een besturingsprogramma worden beschouwd als een werkeenheid.
Router	Een apparaat dat twee of meer verschillende netwerken aan elkaar verbindt.
SOAP	Simple Object Access Protocol (SOAP) is een protocol voor het uitwisselen van gestructureerde informatie middels HTTP en XML tussen twee verschillende systemen.
Thread	Met behulp van threads kan een computerprogramma verschillende taken tegelijkertijd uitvoeren net zoals in een besturingssysteem met multitasking meerdere processen tegelijk kunnen draaien.
Thuisnetwerk	Een thuisnetwerk is een datanetwerk dat gebruikt wordt in huizen. Oorspronkelijk vooral gebruikt voor het verbinden van computers onderling en computers met Internet. Tegenwoordig worden ook andere apparaten, zoals spelcomputers, IP-telefoons en IPTV-settopboxen verbonden.
Unicast	Het verzenden van datapakketten van één bron naar één enkele bestemming.
XML	Extensible Markup Language (XML) is een manier om gegevens gestructureerd weer te geven in tekstvorm.
ZigBee	Standaard voor draadloze verbindingen tussen apparaten op korte afstand.

1. Inleiding

Het huis van de toekomst. Fictie of realiteit? Lang wordt al gedroomd over het huis dat zichzelf schoonhoudt, de koelkast die zelf boodschappen doet en de gordijnen die automatisch sluiten wanneer het donker wordt. Alles wordt geregeld en er hoeft eigenlijk niets meer gedaan te worden.

Een complete samenwerking tussen alle verschillende apparaten in een huishouden is nog steeds niet gelukt. Om de ontwikkeling van het huis van de toekomst te versnellen, hebben vijftien van Europa's leidende bedrijven en onderzoeksinstituten van 2004 tot en met begin 2008 samengewerkt aan het Amigo project. Doel van het Amigo project is om de apparaten die we gebruiken voor de verschillende domeinen in huis, zoals het huishouden, entertainment, telecommunicatie en beveiliging, met elkaar te laten communiceren. Het Amigo project heeft hiervoor een open source, gestandaardiseerde middleware ontwikkeld.

TASS Technology Solutions volgt het Amigo project al vanaf het begin. TASS is nog dagelijks bezig om het huis van de toekomst te creëren door, met nieuwe of aangepaste services, het bestaande Amigo platform uit te breiden met nieuwe mogelijkheden. Mijn afstudeeropdracht richt zich op het uitbreiden van het bestaande Amigo platform met low-power draadloze sensoren. Hiervoor ga ik eerst onderzoeken welke draadloze technologie het beste geschikt is voor uitbreiding van het Amigo platform.

In Hoofdstuk 2 wordt informatie gegeven over TASS Technology Solutions, het bedrijf waar het afstuderen heeft plaatsgevonden. In Hoofdstuk 3 wordt eerst behandeld wat Amigo inhoudt. Vervolgens wordt de beginsituatie, de probleemstelling en het doel van de opdracht uitgelegd. Tenslotte volgt de volledige opdrachtomschrijving. Hoofdstuk 4 beschrijft de gekozen projectaanpak. In Hoofdstuk 5 worden de resultaten van verschillende deelonderzoeken besproken. In Hoofdstuk 6 geef ik de knelpunten aan waar ik op vastliep en wat ik bedacht heb om de problemen op te lossen. Hoofdstuk 7 beschrijft de eindproducten, het resultaat van mijn afstuderen. Tot slot worden er in Hoofdstuk 8 en 9 conclusies en aanbevelingen gedaan naar aanleiding van dit project.

2. Het bedrijf

TASS Technology Solutions is een dienstverlener op het gebied van technische en embedded software. TASS heeft vestigingen in Eindhoven, Apeldoorn en Brussel en levert met meer dan 200 medewerkers een grote expertise in de vorm van detachering en projecten. Als top-3 dienstverlener voor technische en embedded software levert TASS een belangrijke bijdrage aan de ontwikkeling van intelligente en innovatieve producten.

2.1. Geschiedenis

In september 1978 besloten werknemers van de voormalige automatiseringsdivisie van Philips genaamd “Information Systems and Automation (ISA)”, een nieuwe software afdeling te beginnen onder de naam “Special Projects Group (SPG)”. SPG specialiseerde zich in het ontwikkelen van technische software voor verschillende productdivisies van Philips. Na verschillende naamswijzigingen ging vanaf 1990 deze afdeling door het leven als Philips TASS.

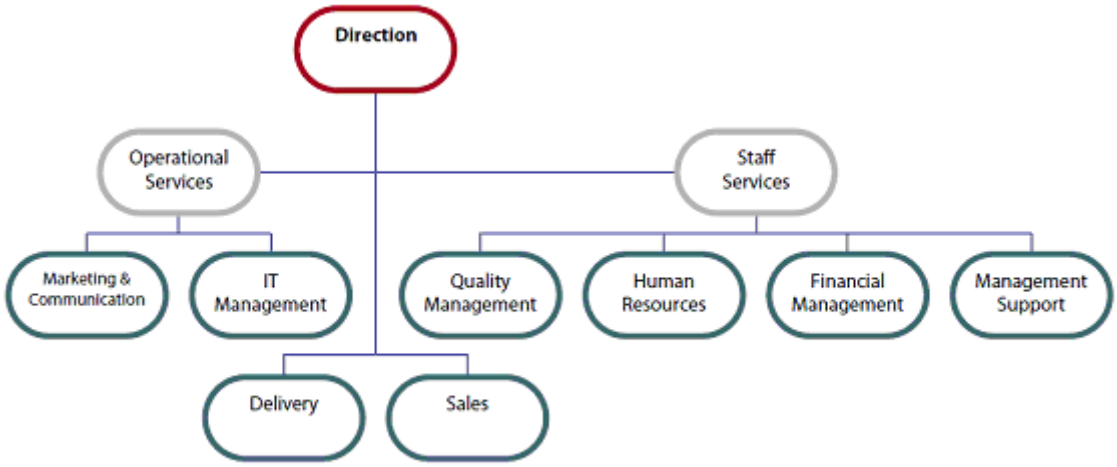
Figuur 1 TASS Kantoor Eindhoven

In 1995 werd Philips TASS een onafhankelijke B.V. en verleende vanaf dat moment diensten op het gebied van technische en embedded software aan externe klanten. Philips TASS groeide in 3 jaar tijd zo hard dat in 1998 TASS International werd opgericht wat bestond uit een Nederlandse en een Belgische divisie.

Philips besloot zich in 2007 meer te gaan richten op de core business van het bedrijf zijn core-business, zodoende werd Philips TASS verkocht en werd het onderdeel van Total Specific Solutions (TSS). Philips TASS werd TASS Technology Solutions maar het duurde echter tot januari 2008 voordat TASS geen onderdeel meer van Philips was en ook een eigen kantoor had op het Larixplein te Eindhoven.

2.2. Organisatie

Het onderstaande organogram beschrijft de organisatie van TASS:



Figuur 2 Organogram TASS

Omdat de kernactiviteit van TASS bestaat uit het detacheren van medewerkers, speelt de afdeling 'Sales' een belangrijke rol. Zij zorgen ervoor dat de klantenportefeuille gevuld blijft. Als een medewerker akkoord gaat met een nieuwe opdracht, zal deze gekoppeld worden aan een people manager. De people manager vervult een begeleidende rol; hij ziet toe op de opdracht, beoordeelt de medewerker periodiek en bepaalt ook een eventuele beloning bij goede prestaties. People managers worden ook zeer actief betrokken bij het selecteren van nieuwe medewerkers. Alle afstudeerders, werknemers en people managers zijn onderdeel van 'Delivery' en staan onder leiding van algemeen directeur Edwin Manten.

De afdelingen 'Operational Services' en 'Staff Services' zijn ondersteunende afdelingen. 'Operational Services' houdt zich bezig met zowel interne als externe communicatie (Marketing & Communication) en verzorgt de voorzieningen op IT gebied (IT Management).

'Staff Services' regelt de overige ondersteunende taken als 'Financial Management', kwaliteitsbewaking (Quality Management), ondersteunen van het management (Management Support) en alle personeelszaken (Human Resources).

Binnen TASS wordt er gewerkt aan het speerpunt Embedded Services: "Everything becomes connected". Dit speerpunt houdt zich onder andere bezig met de ontwikkeling van het Amigo platform. Het speerpunt valt onder de afdelingen 'Delivery' en 'Sales'. Sales draagt de gebieden aan en de afdeling Delivery zorgt voor de technische invulling / uitwerking. Mijn afstudeeropdracht valt onder het interessegebied van dit speerpunt. Peter Vink, mijn bedrijfsbegeleider, is een van de leden van dit speerpunt. De leden van het speerpunt zijn de uiteindelijke gebruikers van de eindproducten van mijn afstudeeropdracht.

3. De opdracht

Uit eerdere onderzoeken is het zogenaamde Amigo platform ontwikkeld. Het Amigo platform is het thuisnetwerk van de toekomst. De opdracht bestaat enerzijds uit een onderzoek en anderzijds uit het maken van nieuwe software die aansluit op het bestaande Amigo platform. In dit hoofdstuk wordt eerst behandeld wat Amigo inhoudt. Vervolgens wordt de beginsituatie, de probleemstelling en het doel van de opdracht uitgelegd. Tenslotte volgt de volledige omschrijving van de opdracht.

3.1. Amigo

Van september 2004 tot en met eind februari 2008 hebben vijftien van Europa's leidende bedrijven en onderzoeksinstituten op het gebied van mobiele- en thuisnetwerken, software ontwikkeling, consumentenelektronica en huishoudelijke apparaten samengewerkt aan het Amigo project. Het Amigo project was deels gesubsidieerd door de Europese Unie.

Het doel van het Amigo project is om het dagelijks leven van de burger te veraangenamen. Het probleem is dat de apparaten, die we gebruiken voor de verschillende domeinen in huis, zoals het huishouden, entertainment, telecommunicatie en beveiliging, niet met elkaar kunnen communiceren. Door de apparatuur in huis met elkaar te laten communiceren, worden werelden met elkaar verbonden, die tot nu toe gescheiden waren.

Samenwerking tussen verschillende elektronica apparaten in een thuisnetwerk is steeds meer in opkomst. Denk bijvoorbeeld aan AirPlay waarmee je foto's en video's van een iPad kunt streamen naar een AppleTV, of het beluisteren van je iTunes muziekcollectie op externe luidsprekers. Dit zijn echter voorbeelden binnen één domein. Een complete samenwerking tussen alle verschillende apparaten in een huishouden is nog niet gelukt. Redenen hiervoor zijn complexe installatieprocedures en het gebrek aan samenwerking / afspraken tussen de verschillende fabrikanten.

Amigo is gericht op het zoeken naar en het maken van oplossingen voor deze problemen. Het project heeft een open source, gestandaardiseerde middleware en aantrekkelijke gebruikersservices ontwikkeld om de bruikbaarheid van een thuisnetwerk te verbeteren. Amigo zorgt voor interoperabiliteit tussen de diverse bestaande oplossingen. Alle populaire protocollen worden daarbij ondersteund en waar nodig vertaald. Hiermee wordt de installatieprocedure gemakkelijker gemaakt en wordt het mogelijk voor apparatuur in huis om onderling te communiceren. Zelfs voor apparaten van verschillende fabrikanten of uit verschillende domeinen.

3.2. Beginsituatie

Het Amigo platform is servicegeoriënteerd. Zo bestaat de huidige demonstratie opstelling van het Amigo platform bijvoorbeeld uit een lamp die als service de mogelijkheid aanbiedt om het licht aan of uit te zetten. Andere apparaten binnen het Amigo netwerk kunnen van deze service gebruik maken en op die manier daadwerkelijk de lamp bedienen. Daarnaast bestaat de demonstratie opstelling onder andere uit lampen die gedimd kunnen worden of van kleur kunnen veranderen, uit gordijnrails, een fotolijstje, een pokergame, en uit een positioning service.

3.3. Opdrachtomschrijving

Het bestaande Amigo platform moet uitgebreid worden met low-power draadloze sensoren. De sensoren moeten meerdere jaren kunnen werken zonder dat de batterijen moeten worden vervangen. Er moet onderzocht worden welke technologie hiervoor het beste geschikt is. Tijdens het onderzoek moeten er minimaal twee draadloze technologieën vergeleken worden. Daarnaast moet er een PC applicatie gemaakt worden (met nieuwe of aangepaste Amigo middleware services), die gebruik maakt van de sensoren. Bovenstaande moet verwerkt worden in een (nieuwe) demonstratie omgeving.

De volgende subdoelen zijn te onderkennen:

- Onderzoek naar draadloze technologieën voor uitbreiding van het Amigo systeem.
- Het bestaande Amigo systeem uitbreiden met low-power draadloze sensoren op basis van een van de onderzochte technologieën.
- Het maken van een PC applicatie die gebruik maakt van de low-power draadloze sensoren.
- Eindresultaat tonen met behulp van een demonstratie.

De opdrachtomschrijving is bewust vaag. Zo staat er alleen dat het bestaande Amigo platform uitgebreid moet worden met low-power draadloze sensoren, maar er wordt niet gespecificeerd wat voor een type sensoren (lichtsensor, temperatuursensor, etc.). De reden hiervoor is dat er eerst onderzocht moet worden welke draadloze technologie het beste geschikt is voor uitbreiding van het Amigo platform. Daarna moet er onderzocht worden met welke hardware het mogelijk is om via de geselecteerde technologie te communiceren. Pas op dat moment is bekend over welk type sensoren de hardware beschikt.

Ik heb zelf mijn afstudeeropdracht samengesteld. Dit in samenspraak met mijn bedrijfsbegeleider Peter Vink. De oorspronkelijke opdracht staat beschreven in Bijlage A Project Management Plan. Als onderdeel van het Project Management Plan heb ik een eerste opzet gemaakt voor de planning. Daaruit bleek dat de opdracht veel te groot was. Op dat moment is de opdrachtbeschrijving nog niet aangepast. De Software Requirements Specificatie is ook gebaseerd op de oorspronkelijke opdracht. Pas gaandeweg de implementatie zijn we stapsgewijs gekomen tot de bovenstaande opdrachtomschrijving.

3.4. Doel

TASS heeft een drietal doelen met betrekking tot mijn afstudeeropdracht:

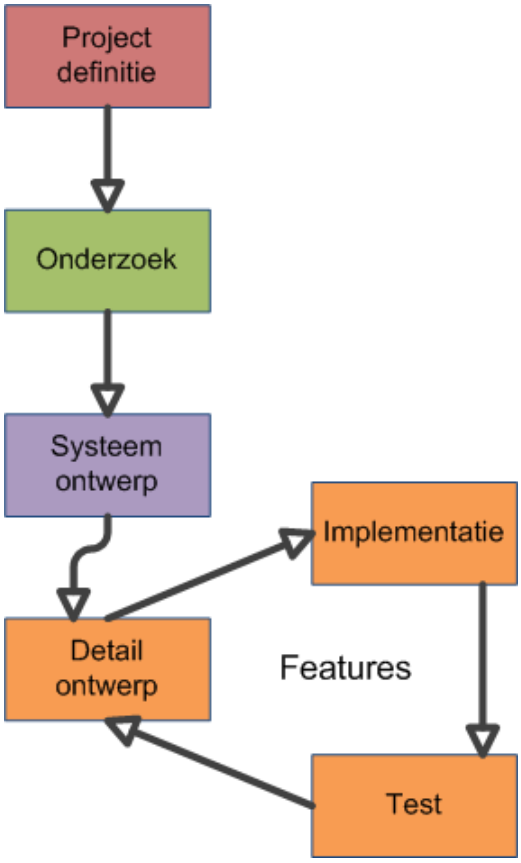
- Het opbouwen / verbreden van kennis op het gebied van draadloze technologieën.
- De mogelijkheid voor TASS om, met behulp van de demonstratie van het eindproduct, op beurzen en evenementen aan potentiële klanten te tonen dat zij bekend is met de nieuwste draadloze technologieën.
- Het bestaande systeem uitbreiden met dynamisch gedrag.

Met dynamisch gedrag wordt bedoeld dat de gebruiker de mogelijkheid moet krijgen om binnen een bepaalde context acties te kunnen koppelen aan gebeurtenissen (bijvoorbeeld als het donker is en je komt de kamer binnen dan moet het licht aan). De gebruiker moet in staat zijn om het dynamische gedrag vast te leggen (bijvoorbeeld door definitie van regels).

Het laatste doel, dynamisch gedrag, wordt niet gerealiseerd met behulp van deze afstudeeropdracht. Het bestaande Amigo platform (zie paragraaf 3.2) heeft momenteel alleen de positioning service als input. Eén type input is erg weinig als basis voor een volwaardige implementatie van dynamisch gedrag. Vandaar dat als opdracht gekozen is om het bestaande Amigo platform uit te breiden met low-power draadloze sensoren. Na afloop van mijn afstuderen zijn er dus wel meerdere vormen van input beschikbaar, maar is het nog niet mogelijk om acties aan gebeurtenissen te koppelen. Hiermee is in ieder geval wel een stap in de goede richting gezet.

4. Methode

Ik heb het project opgedeeld in 3 fasen: project definitie, onderzoek en design & implementatie. Voor de realisatie van het project heb ik gebruik gemaakt van het W-model (zie [10]). Het W-model is gebaseerd op het V-model. Het grote verschil met het V-model zit aan de onderkant: het detailontwerp, implementatie en testen wordt cyclisch en iteratief uitgevoerd. Het voordeel van deze werkwijze is dat er aan het einde van het project altijd een werkend eindproduct is.



Figuur 3 Project fasering

4.1. Fasering

De eerste fase van het project is de project definitie, in deze fase is het Project Management Plan (PMP) en de Software Requirements Specificatie (SRS) gemaakt. Als onderdeel van het PMP is ook een eerste opzet gemaakt voor de planning. Vervolgens zijn in de onderzoeksfase de mogelijkheden onderzocht om het bestaande Amigo platform uit te breiden met een draadloze technologie. Daarnaast is er gekeken naar hoe de huidige Amigo apparaten onderling communiceren. In de design & implementatie fase heb ik eerst een systeem ontwerp gemaakt wat geschikt is voor de complete gewenste functionaliteit. Daarna heb ik, op volgorde van prioriteit, per feature een detailontwerp gemaakt, de bijbehorende requirements geïmplementeerd en de code getest.

5. Onderzoek

De onderzoeksvraag luidt: “Welke draadloze technologie is het beste geschikt voor uitbreiding van het Amigo platform?”. De criteria waar de technologie aan moet voldoen om in aanmerking te komen zijn:

1. De sensoren, op basis van de geselecteerde technologie, moeten meerdere jaren kunnen werken zonder vervanging van de batterijen.
2. De technologie moet compatibel zijn met het bestaande Amigo platform.

Er bestaan verschillende draadloze technologieën zoals: Wi-Fi, Bluetooth, Z-Wave, ZigBee, 6LoWPAN, Wavenis, etc. Tijdens het onderzoek heb ik twee draadloze technologieën met elkaar vergeleken, namelijk 6LoWPAN en ZigBee.

6LoWPAN is het beste geschikt voor uitbreiding van het Amigo platform, omdat 6LoWPAN net als Amigo op Internet protocollen gebaseerd is. 6LoWPAN voldoet hiermee ook aan het tweede criteria, terwijl bij ZigBee nog een extra vertaalslag nodig is. Zowel 6LoWPAN als ZigBee voldoen aan het eerste criteria.

Het onderzoek bestaat uit de volgende deelonderzoeken:

1. Verdiepen in de werking van 6LoWPAN.
2. Kennis en ervaring opdoen met de werking van IPv6.
3. Verdiepen in de werking van ZigBee.
4. Met welke hardware is het mogelijk om via 6LoWPAN te communiceren?
5. Welk besturingssysteem heeft 6LoWPAN ondersteuning?
6. Proof-of-concept.

In Hoofdstuk 5.1 6LoWPAN worden de werking en de voor- en nadelen van 6LoWPAN uitgelegd. De resultaten van het deelonderzoek naar de werking van ZigBee staan in Bijlage C Onderzoeksrapport.

Ik heb, in overleg met Peter Vink, ervoor gekozen om maar twee draadloze technologieën met elkaar te vergelijken. Anders zou er teveel tijd in het onderzoek gaan zitten en te weinig tijd overblijven voor de design & implementatiefase. Ik heb voor 6LoWPAN en ZigBee gekozen omdat dit nieuwe technologieën zijn. TASS heeft bijvoorbeeld in het verleden al kennis opgedaan van Bluetooth. Ik heb Wi-Fi, de bekendste draadloze technologie, niet meegenomen in het onderzoek, omdat Wi-Fi niet voldoet aan het eerste criteria. Wi-Fi gebruikt teveel zendvermogen om een bericht te versturen. Bij gebruik van Wi-Fi zouden de batterijen te snel vervangen moeten worden.

5.1. 6LoWPAN

6LoWPAN staat voor: “IPv6 over Low-Power Wireless Personal Area Networks”. De 6LoWPAN standaard is opgesteld door de gelijknamige werkgroep van de Internet Engineering Task Force (IETF). De werkgroep is in 2004 opgericht met als doel om het mogelijk te maken via IPv6 te communiceren over de IEEE 802.15.4 standaard. Dit wordt gerealiseerd door een zogenaamde “adaptation layer”, een laag die IPv6 compatibel maakt met IEEE 802.15.4. Zie onderstaande afbeelding ter verduidelijking. 6LoWPAN is in september 2007 een officiële Internet standaard geworden (RFC4944).

Figuur 4 OSI-model

Kenmerken van IEEE 802.15.4 zijn:

- Kleine pakketgrootte. Maximale grootte van een pakket in de fysieke laag is 127 bytes. Dit resulteert in een maximale grootte van een frame in de datalinklaag van 102 bytes.
- Lage bandbreedte. Maximale overdrachtssnelheid van 250 kbps op 2.4 GHz en 20 kbps op 868 MHz. De 2.4 GHz frequentieband is dezelfde frequentie als wat ook gebruikt wordt door Wi-Fi. Op deze frequentie is er dus (grote) kans op interferentie.

De twee belangrijkste taken van 6LoWPAN zijn:

- Compressie van de header. Zoals hierboven al is uitgelegd blijven er maar 102 (van de 127) bytes over voor datapakketten uit de netwerklaag. Een IPv6 header is 40 bytes lang, dat wil zeggen dat er dan nog 62 bytes overblijven voor de transportlaag. Een UDP header is 8 bytes lang, een TCP header is 20 bytes lang. Zonder compressie is er 1 byte nodig voor de 6LoWPAN header en blijft er dus voor de data die daadwerkelijk verstuurd moet worden maar 53 bytes over bij gebruik van UDP, of 41 bytes bij gebruik van TCP. Met compressie is 6LoWPAN in staat om, bij gebruik van UDP, de headers dusdanig te comprimeren dat er maar liefst 108 bytes beschikbaar zijn voor de te versturen data. Meer dan een verdubbeling van het aantal beschikbare bytes!
- Opdelen en samenvoegen. Een IPv6 datapakket heeft een minimale grootte van 1280 bytes, vele malen groter dan een 6LoWPAN datapakket. Het is daarom een taak van de adaptation layer om een IPv6 datapakket dat verstuurd moet worden op te splitsen in meerdere 6LoWPAN datapakketten. En omgekeerd meerdere ontvangen 6LoWPAN datapakketten samen te voegen tot één IPv6 datapakket.

De adaptation layer zorgt ervoor dat er geen aanpassingen of beperkingen aan IPv6 nodig zijn. De functionaliteit van het opdelen en samenvoegen maakt het mogelijk dat er nog steeds grote hoeveelheden data verstuurd kunnen worden.

Voordelen	Nadelen
Bestaande IP gebaseerde technologieën zijn bekend en bewezen effectief.	Kleine pakketgrootte (127 bytes). (dit nadeel is een eigenschap van IEEE 802.15.4).
Netwerkbeheer kan worden uitgevoerd met bekende tools zoals ping en traceroute.	Lage bandbreedte (250 kbps). (dit nadeel is een eigenschap van IEEE 802.15.4).
6LoWPAN apparatuur kan direct worden aangesloten op IP gebaseerde netwerken (zoals het Internet) zonder dat een gateway of proxy nodig is.	Als in de transportlaag blijkt dat een TCP pakket niet correct ontvangen is dan moeten alle bijbehorende 6LoWPAN pakketten opnieuw verstuurd worden.
6LoWPAN is een IETF standaard. IETF standaarden zijn gratis toegankelijk en iedereen kan er gratis aan bijdragen (de IETF kent geen lidmaatschap).	
Het gebruik van 6LoWPAN is transparant; Een PC kan rechtstreeks communiceren met een sensor.	

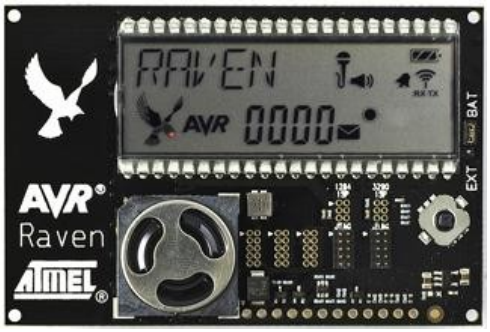
Tabel 1 Voor- en nadelen 6LoWPAN

Zie Bijlage C Onderzoeksrapport voor meer informatie over 6LoWPAN, IEEE 802.15.4 en IPv6.

5.2. Hardware

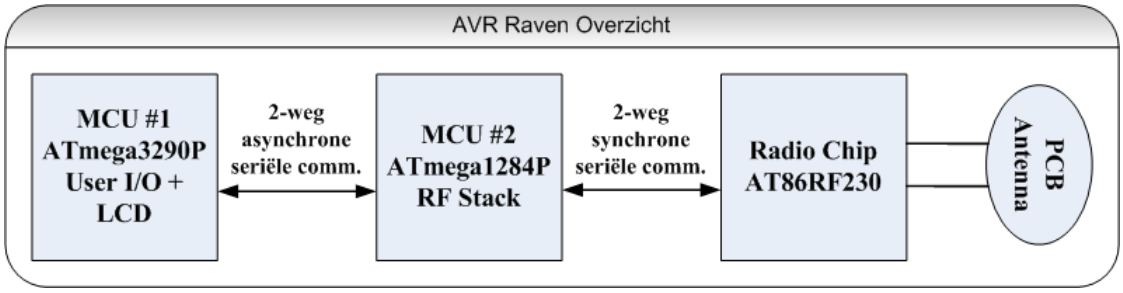
Het was erg lastig om hardware te vinden waarmee het mogelijk is om via 6LoWPAN te communiceren. Een van de redenen hiervoor is dat de 6LoWPAN standaard nog relatief nieuw is. 6LoWPAN zelf is weliswaar al in september 2007 een officiële standaard geworden, maar bepaalde door 6LoWPAN gebruikte technieken zijn pas recenter bekrachtigd. Zo is bijvoorbeeld in maart 2011 de specificatie met betrekking tot het routeren van IPv6 datapakketten op low-power netwerken voorgedragen om een Internet standaard te worden. En in september 2011 is de specificatie met betrekking tot header compressie een officiële Internet standaard geworden (RFC6282).

Ten tijde van het onderzoek waren er vier fabrikanten met een 6LoWPAN development kit, namelijk: Jennic, Arch Rock, Sensinode en Atmel. De hardware van Jennic had mijn voorkeur omdat deze beschikte over zowel een temperatuursensor als een lichtsensor. De Software Requirements Specificatie is gebaseerd op de beschikbaarheid van deze twee sensoren. Op het moment dat ik daadwerkelijk de hardware wilde aanschaffen, had Jennic de 6LoWPAN development kit van de markt gehaald. Navraag leerde dat Jennic een totaal ander mechanisme had gekozen voor het routeren van IPv6 datapakketten, dan hetgeen uiteindelijk de standaard is geworden. Ook de 6LoWPAN development kit van Arch Rock was niet meer beschikbaar. Arch Rock, een bedrijf gespecialiseerd in 6LoWPAN, was door Cisco overgenomen. Van de twee overgebleven fabrikanten is Sensinode afgevallen op basis van de prijs. Sensinode vroeg maar liefst € 2.499 voor één 6LoWPAN development kit. Ter vergelijking de 6LoWPAN development kit van Atmel kost +/- 100 euro.



Figuur 5 Foto AVR Raven

De Atmel 6LoWPAN development kit bestaat uit twee (identieke) sensoren, genaamd AVR Raven, en één USB stick, genaamd RZUSBSTICK. De sensoren hebben als nadeel dat deze alleen beschikken over een temperatuursensor. Wel beschikken de sensoren over User I/O pinnen. Hiermee is het in de toekomst alsnog mogelijk om een externe lichtsensor te verbinden met de AVR Raven. Een tweede nadeel is dat de AVR Raven beschikt over twee microcontrollers. Eén microcontroller, MCU #1, is verbonden met de temperatuursensor en de User I/O pinnen. De tweede microcontroller, MCU #2, is verbonden met de radiochip. De microcontrollers kunnen via een seriële verbinding met elkaar communiceren (zie onderstaande afbeelding). Dit brengt meer werk met zich mee, nu moet immers ook de seriële communicatie geïmplementeerd worden. Dit was niet nodig als de temperatuursensor en radiochip rechtstreeks met één microcontroller waren verbonden.



Figuur 6 AVR Raven overzicht

Met behulp van de RZUSBSTICK kan een PC communiceren met de sensoren. De USB stick bevat een microcontroller en een radiochip. De bekende besturingssystemen uit de PC wereld zoals Windows, OS X en Linux hebben geen ondersteuning voor 6LoWPAN. De RF stack verantwoordelijk voor de daadwerkelijke 6LoWPAN communicatie draait daarom op de microcontroller van de USB stick.

Figuur 7 Foto RZUSBSTICK

MCU	MCU #1	MCU #2
Type	ATmega3290P	ATmega1284P
Doel	User I/O + LCD	RF Stack
AVR Core	8-bit	8-bit
Frequentie	20 MHz	20 MHz
Flash	32 KiB	128 KiB
EEPROM	1 KiB	4 KiB
SRAM	2 KiB	16 KiB

Tabel 2 AVR Raven hardware specificatie

Figuur 8 Plug Computer

De RZUSBSTICK is verbonden met een zogenaamde Plug Computer. Een Plug Computer is een minicomputer in het formaat van een trafoblok die je rechtstreeks in een wandcontactdoos kunt steken. De Plug Computer is enerzijds, via de RZUSBSTICK, verbonden met 6LoWPAN en anderzijds verbonden met het bedraadde TASS netwerk. De taak van de Plug Computer is om het IPv6 verkeer te routeren tussen beide netwerken. De taak van de Plug Computer is hiermee vergelijkbaar met die van een Wi-Fi router.

5.3. Software

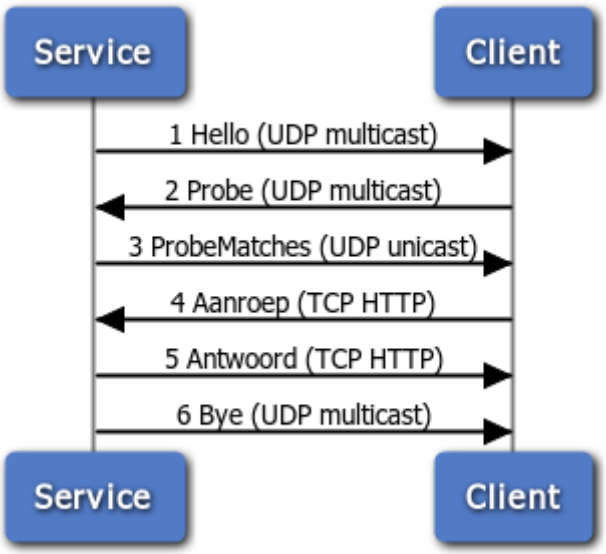
Tijdens mijn afstuderen moet ik een PC applicatie maken die gebruik maakt van 6LoWPAN, en niet 6LoWPAN zelf implementeren. Daarom ben ik op zoek gegaan naar een besturingssysteem met 6LoWPAN ondersteuning. Ik heb twee besturingssystemen gevonden, namelijk TinyOS en Contiki. TinyOS valt af omdat deze de door mij gebruikte Atmel hardware niet ondersteunt. Een ander voordeel van Contiki ten opzichte van TinyOS is dat Contiki gebaseerd is op de programmeertaal C. TinyOS is geschreven in nesC, een extensie op C. Contiki draait op MCU #2 van de AVR Raven en op de microcontroller van de RZUSBSTICK. De Plug Computer is voorzien van Debian GNU/Linux distributie.

5.4. Amigo communicatie

Eén van de belangrijkste requirements is compatibiliteit met het huidige Amigo platform.

Het Amigo platform maakt gebruik van webservices. Om te kunnen communiceren met een webservice moet er gebruik gemaakt worden van Simple Object Access Protocol (SOAP). SOAP is een protocol dat XML berichten stuurt over een HTTP verbinding (zie pijl 4 en 5 uit Figuur 9).

Elk Amigo netwerk is uniek. Het heeft zijn eigen apparaten, die elk een eigen service hebben met elk hun eigen mogelijkheden. Door deze samenstelling is het niet van tevoren mogelijk om te weten wat voor services beschikbaar zijn en op welk adres deze services te benaderen zijn. Om dit probleem op te lossen maakt Amigo gebruik van WS-Discovery. WS-Discovery is een protocol dat XML berichten stuurt over een UDP verbinding (zie pijl 1, 2, 3 en 6 uit Figuur 9).



Figuur 9 Bestaande Amigo communicatie

Het Amigo project is gestart in 2004. Toentertijd bevatte Java en C# nog geen ondersteuning voor webservices en WS-Discovery. Als onderdeel van het Amigo project is er door het European Microsoft Innovation Center (EMIC) een externe software bibliotheek geschreven voor C# met ondersteuning voor webservices en WS-Discovery. Het Fraunhofer instituut heeft een externe software bibliotheek geschreven voor Java met dezelfde functionaliteit.

6. Problemen en oplossingen

Tijdens mijn afstuderen stuitte ik op een aantal knelpunten. In dit hoofdstuk geef ik aan waar ik vastliep en wat ik heb bedacht om het probleem op te lossen.

6.1. IPv6 ondersteuning

Vanwege de keuze voor 6LoWPAN moet de communicatie (zie paragraaf 5.4) plaatsvinden via IPv6. Contiki, het gekozen besturingssysteem voor de sensor, heeft ondersteuning voor IPv6 en is zelfs IPv6 Ready fase 1 gecertificeerd. IPv6 is in december 1998 een officiële standaard geworden (RFC2460). Het Amigo project is gestart in 2004. De aanname was dat de tijdens het Amigo project geschreven externe software bibliotheken IPv6 zouden ondersteunen. Helaas bleek dat zowel de bestaande Amigo software bibliotheek voor C# als die voor Java alleen kunnen communiceren via IPv4.

Na een kort onderzoek kwam ik erachter dat de functionaliteit uit de bestaande externe software bibliotheken tegenwoordig geïntegreerd is in het .NET Framework versie 4.0. En dan specifiek in het Windows Communication Foundation (WCF) onderdeel van het .NET Framework. Als oplossing is ervoor gekozen om gebruik te maken van Visual Studio 2010 in combinatie met het .NET Framework versie 4.0. Vanwege de beperkte hoeveelheid beschikbare tijd is er geen onderzoek gedaan naar de ondersteuning voor webservices en WS-Discovery in Java.

De overstap van de bestaande externe software bibliotheken naar het .NET Framework versie 4.0 heeft als nadeel dat het implementeren van de Amigo communicatie voor de PC applicatie meer tijd heeft gekost dan initieel ingepland. Anderzijds is een voordeel van de overstap dat het .NET Framework wel onderhouden wordt, terwijl de bestaande externe software bibliotheken na afloop van het Amigo project niet meer zijn bijgewerkt.

Daarnaast is er op verzoek van Peter Vink een extra eindproduct gedefinieerd, namelijk een uitgebreide handleiding. Deze handleiding beschrijft hoe je met Visual Studio 2010 in combinatie met het .NET Framework versie 4.0 een Amigo service en een bijbehorende PC applicatie kunt maken. Deze documentatie kan zowel gebruikt worden voor nieuwe Amigo services, als voor het omzetten van bestaande Amigo services.

6.2. IPv6 multicast

Zoals beschreven in paragraaf 5.4 gebruikt Amigo verschillende vormen van communicatie, namelijk: UDP multicast, UDP unicast en TCP unicast. Multicast wordt gebruikt voor het versturen van de Hello & Bye berichten, en voor het ontvangen van Probe-berichten.

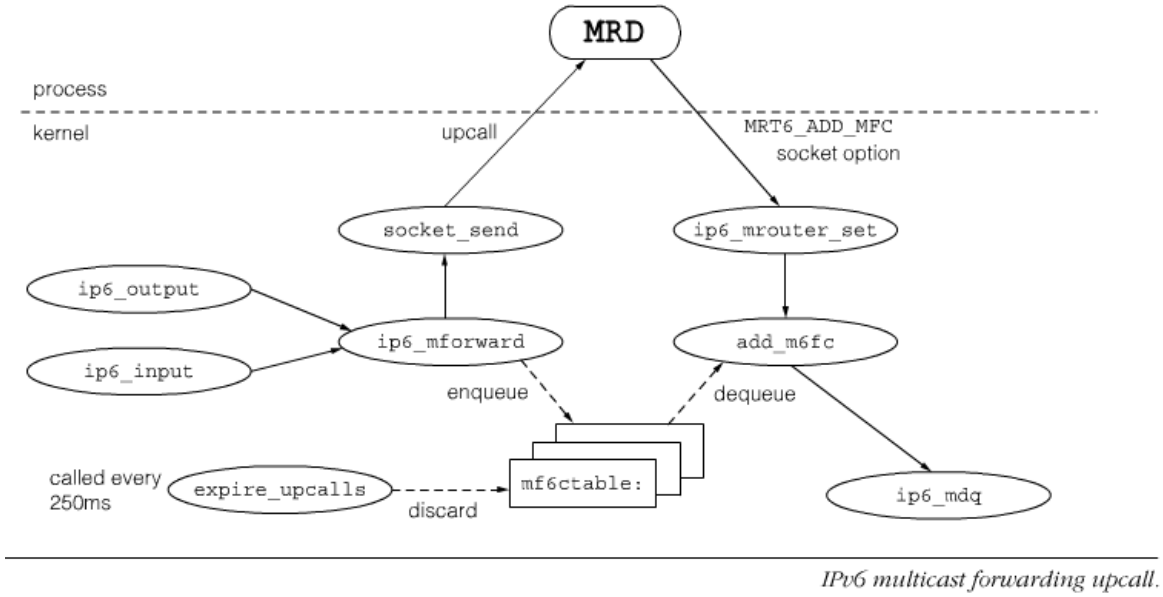
Multicast is een techniek waarbij je één pakket verstuurd en dat het netwerk er automatisch voor zorgt dat dit bij meerdere ontvangers aankomt. IPv6 Multicast staat beschreven in een aparte standaard genaamd “Multicast Listener Discovery Version 2 (MLDv2) for IPv6” (RFC3810). Een router stuurt een binnenkomend IPv6 multicast datapakket alleen door naar de aangesloten apparatuur als die zich van te voren geabonneerd heeft. Abonneren gaat op basis van een IPv6 multicast adres. De MLDv2 standaard beschrijft onder andere een Report Message dat je moet sturen om aan te geven dat je alle datapakketten wilt ontvangen die een specifiek IPv6 multicast adres als bestemming hebben. WS-Discovery maakt gebruik IPv6 multicast adres FF02::C.

Tijdens de implementatiefase ben ik tegen het probleem aangelopen dat er geen IPv6 multicast (Probe) berichten bij de sensor aankomen. Aan dit probleem liggen twee verschillende oorzaken ten grondslag.

Ten eerste blijkt dat Contiki geen ondersteuning bevat voor de MLDv2 standaard (zie [3]). Het is hierdoor voor mijn applicatie niet mogelijk om zich te abonneren op IPv6 multicast verkeer. Dit heeft tot gevolg dat de Probe-berichten niet naar de sensor gestuurd worden. Deze ontdekking verbaasde mij enorm, Contiki is immers IPv6 Ready fase 1 gecertificeerd. Het blijkt dat fase 1 opgevolgd is door fase 2 (zie [4]). Bij fase 2 is MLDv2 ondersteuning verplicht (zie [5]). Versturen van IPv6 multicast datapakketten is wel mogelijk, want bij het versturen van datapakketten is er geen verschil tussen unicast en multicast.

Ten tweede blijkt dat de Linux kernel niet zelfstandig in staat is om IPv6 multicast verkeer te routeren tussen 6LoWPAN en het TASS netwerk (zie [6]). De Linux kernel kan wel zelfstandig IPv6 unicast verkeer routeren. Dit is eenvoudig aan te zetten door in het /proc bestandssysteem de kernel parameters net.ipv6.conf.default.forwarding en net.ipv6.conf.all.forwarding op 1 te zetten.

Om IPv6 multicast verkeer te kunnen routeren heeft de Linux kernel hulp nodig van een externe applicatie genaamd Multicast Routing Daemon (MRD). Deze applicatie is geen onderdeel van de Linux kernel. Als er een IPv6 multicast datapakket binnenkomt dan geeft de Linux kernel dit door aan de MRD. Het is vervolgens de taak van de MRD om te beslissen wat er met het datapakket moet gebeuren op basis van het IPv6 adres van de verzender en het IPv6 multicast adres van de bestemming.

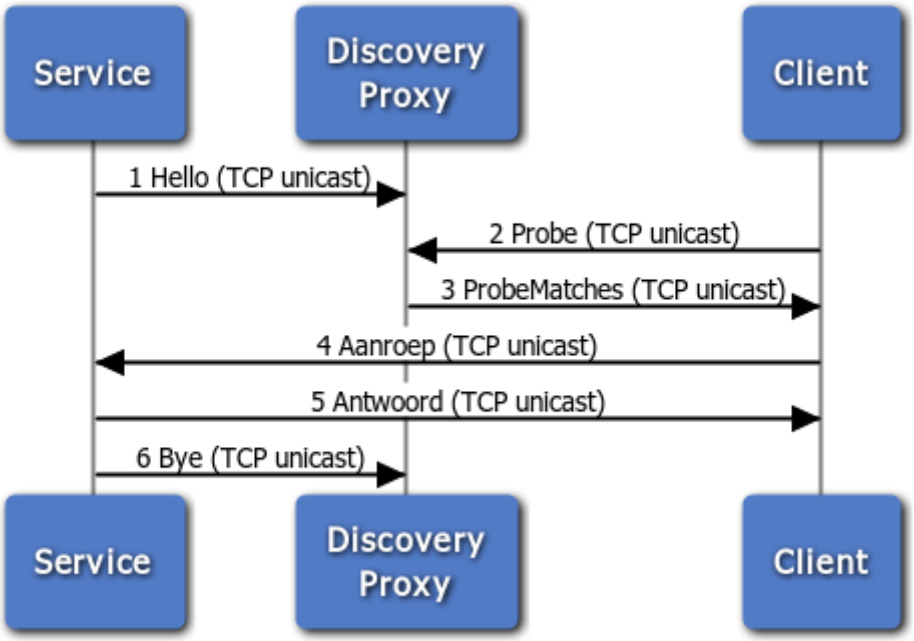


Figuur 10 Interactie tussen Linux kernel en MRD

De IPv6 multicast routing feature van de Linux kernel is zelfs in de meest recente Linux kernel nog gemarkeerd als experimenteel (Linux kernel 3.2.9; 1 maart 2012). Experimenteel betekent in deze situatie dat de interface tussen de Linux kernel en de MRD nog veel kan wijzigen. Het routeren van IPv6 multicast verkeer werkt alleen indien de MRD gebaseerd is op dezelfde interface als aangeboden wordt door de Linux kernel. Met andere woorden MRD versie *X* werkt alleen samen met de Linux kernel interface versie *Y* en niet met versie *Y+1*.

Ik heb verschillende Multicast Routing Daemons geprobeerd in combinatie met verschillende versies van de Linux kernel, maar het is mij niet gelukt om IPv6 multicast routing werkend te krijgen.

Het bestaande Amigo platform maakt alleen gebruik van multicast om te ontdekken wat voor services beschikbaar zijn in het lokale netwerk (zie paragraaf 5.4). Hiervoor maakt Amigo gebruik van de Ad Hoc Mode uit de WS-Discovery standaard. Naast Ad Hoc Mode beschrijft de WS-Discovery standaard ook een Managed Mode. In Managed Mode wordt multicast vervangen door unicast in combinatie met een Discovery Proxy. Als oplossing voor de in deze paragraaf beschreven IPv6 multicast problemen heb ik ervoor gekozen om, alleen voor IPv6, over te stappen op Managed Mode. De Amigo webservice stuurt de Hello & Bye berichten unicast naar de Discovery Proxy, en de PC applicatie stuurt de Probe unicast naar de Discovery Proxy. Zie onderstaande afbeelding ter verduidelijking.



Figuur 11 Managed Mode Amigo communicatie

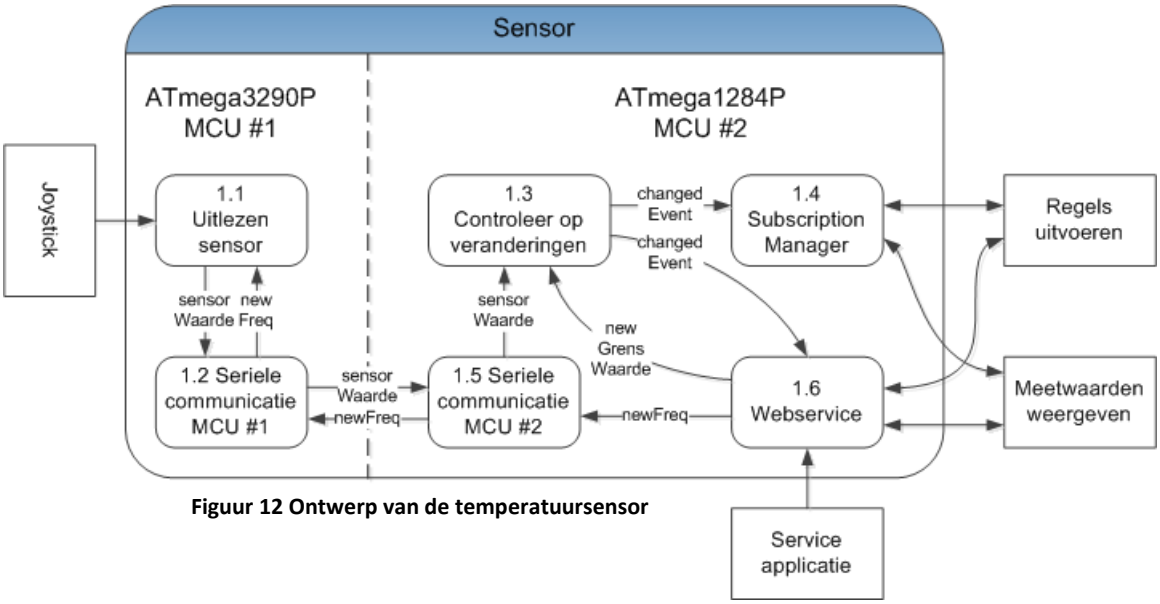
De Discovery Proxy is geen kant-en-klaar software pakket. Ik heb zelf, aan de hand van de WS-Discovery standaard, een Discovery Proxy geschreven in de programmeertaal C. De Discovery Proxy draait op de Plug Computer.

Eén van de belangrijkste requirements is dat de sensor meerdere jaren moet kunnen werken zonder vervanging van de batterijen. Een voordeel van de Discovery Proxy is dat de temperatuursensor nu niet meer hoeft te luisteren naar Probe-berichten en geen ProbeMatches-berichten meer hoeft te versturen. Deze taak is nu immers overgenomen door de Discovery Proxy (zie pijl 2 en 3 uit Figuur 11). De overstap op Managed Mode betekent dus minder netwerkverkeer voor de temperatuursensor met als gevolg dat deze energiezuiniger is dan in Ad Hoc Mode.

7. Eindproducten

7.1. Temperatuursensor

Het belangrijkste eindproduct is de temperatuursensor. Voor de temperatuursensor maak ik gebruik van de AVR Raven hardware. De AVR Raven bevat twee microcontrollers (zie Figuur 6 AVR Raven overzicht). De temperatuur wordt gemeten met behulp van een NTC-weerstand. Een NTC-weerstand, is een weerstand met een negatieve temperatuurcoëfficiënt. Dit betekent dat de elektrische weerstand afneemt als de temperatuur toeneemt. De NTC-weerstand is verbonden met MCU #1. De andere microcontroller, MCU #2, is verbonden met de radiochip. Alleen MCU #2 maakt gebruik van een besturingssysteem (zie paragraaf 5.3). Voor beide microcontrollers heb ik gebruik gemaakt van de programmeertaal C. Voor de temperatuursensor heb ik het onderstaande ontwerp gemaakt. Het ontwerp laat zien uit welke componenten de software bestaat. Ik heb geen gebruik gemaakt van UML omdat de programmeertaal C niet object georiënteerd is.

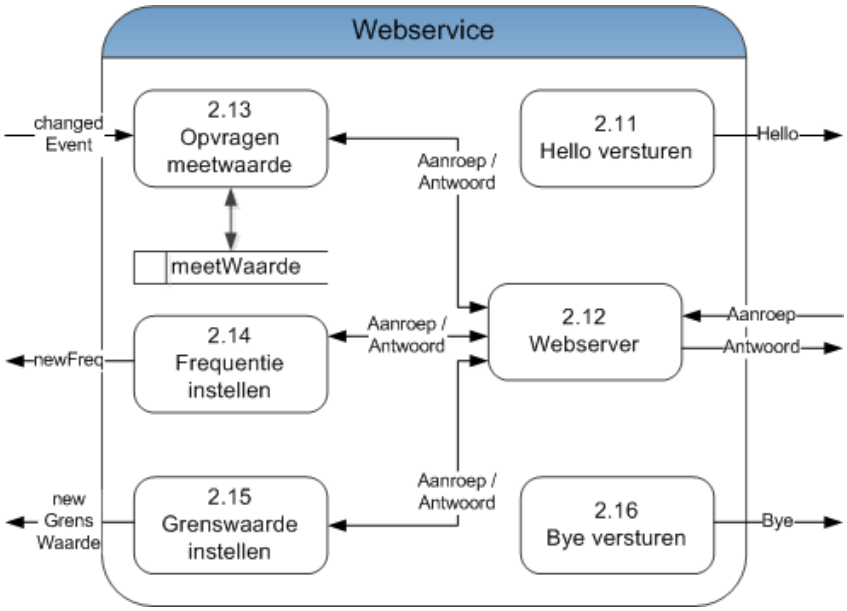


Figuur 12 Ontwerp van de temperatuursensor

De verdeling van de componenten is een bewuste keuze. Op deze manier hoeft alleen component 1.1 Uitlezen sensor en component 1.3 Controleer op veranderingen vervangen te worden bij gebruik van een ander type sensor (zoals lichtsensor, bewegingssensor, etc.).

Ik heb ervoor gekozen om zo veel mogelijk functionaliteit onder te brengen op MCU #2. Beide microcontrollers draaien op 20 MHz, maar MCU #2 heeft meer geheugen dan MCU #1 (16 KiB ten opzichte van 2 KiB). Daarnaast kan ik op MCU #2 gebruik maken van de standaard functies van het Contiki besturingssysteem. Ik hoef dus minder snel zelf een functie te implementeren. De enige taak voor MCU #1 is één keer per seconde de temperatuur uitlezen en naar MCU #2 sturen.

Component 1.6 Webservice is het belangrijkste component. Dit component is namelijk verantwoordelijk voor de draadloze communicatie, de kern van de afstudeeropdracht. Figuur 11 Managed Mode Amigo communicatie geeft de externe communicatie weer tussen component 1.6 Webservice en het Amigo platform. Component 1.6 Webservice is onderverdeeld in een aantal subcomponenten. Zie onderstaande afbeelding ter verduidelijking.



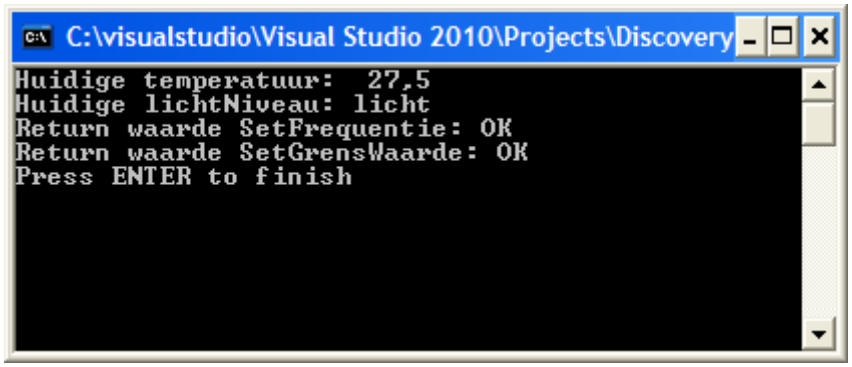
Figuur 13 Ontwerp webservice component

De software op MCU #2 moet tegelijkertijd luisteren naar binnenkomende meetwaarden via de seriële verbinding (component 1.5) en luisteren naar aanroepen komende van het Amigo platform (component 2.12). Daarom maak ik gebruik van processen. Ieder component, met uitzondering van component 1.6 Webservice, draait binnen een eigen proces. Component 1.6 bestaat uit drie processen: Hello (component 2.11), Webserver (component 2.12 tot en met 2.15) en Bye (component 2.16).

In Contiki is een proces een protothread (zie [11]). Protothreads zijn speciaal ontworpen voor systemen met een beperkte hoeveelheid geheugen (zie [12]). Een eigenschap van protothreads is dat alle protothreads gebruik maken van één gezamenlijke stack. Daarnaast zijn protothreads non-preemptive, dat wil zeggen dat protothreads nooit onderbroken worden. Bij non-preemptive multitasking blijft de microcontroller met één protothread bezig, totdat de protothread klaar is, of zelf besluit om de microcontroller vrij te geven. Dit heeft voor mij als voordeel dat ik de variabele waarin de meetwaarde wordt opgeslagen niet hoef te beveiligen. Omdat protothreads nooit onderbroken worden kan de situatie waarin de meetwaarde tegelijkertijd uitgelezen en bijgewerkt wordt immers niet voorkomen.

7.2. PC Applicatie

Om via het netwerk met de temperatuursensor te communiceren heb ik een eenvoudige PC applicatie geschreven in de programmeertaal C# (zie onderstaande screenshot).



Figuur 14 Screenshot PC Applicatie

De bestaande C# PC applicaties maken allemaal gebruik van een externe software bibliotheek geschreven door het European Microsoft Innovation Center (EMIC). Deze externe software bibliotheek heeft helaas geen IPv6 ondersteuning (zie paragraaf 6.1). Alle door Amigo gebruikte functionaliteit is tegenwoordig geïntegreerd in het .NET Framework versie 4.0. Als oplossing heb ik er daarom voor gekozen om voor de PC applicatie gebruik te maken van Visual Studio 2010 in combinatie met het .NET Framework versie 4.0.

7.3. Training document

De bestaande Amigo applicaties kunnen (nog) niet communiceren met de temperatuursensor. Dat komt door een tweetal oorzaken: Ten eerste heeft zowel de bestaande Amigo software bibliotheek voor C# als die voor Java geen IPv6 ondersteuning (zie paragraaf 6.1). Ten tweede ben ik noodgedwongen overgestapt van WS-Discovery Ad Hoc Mode naar Managed Mode (zie paragraaf 6.2). Daarom heb ik een trainingsdocument geschreven met uitleg hoe je met behulp van het .NET Framework versie 4.0 een applicatie kan schrijven, die wel kan communiceren via IPv6 en gebruik maakt van WS-Discovery Managed Mode. Dit document kan gebruikt worden als handleiding voor het schrijven van nieuwe applicaties en als leidraad voor het omzetten van bestaande applicaties. Het trainingsdocument is bijgevoegd als Bijlage E Amigo webservice met .NET Framework 4.0.

8. Conclusies

Uit het onderzoek bleek dat 6LoWPAN de meest geschikte draadloze techniek is om het Amigo netwerk uit te breiden met low-power draadloze sensoren. Tijdens de design & implementatiefase is het gelukt om een werkend eindproduct te maken op basis van deze technologie. De belangrijkste conclusie is dus dat 6LoWPAN inderdaad geschikt is voor het beoogde doel!

De temperatuursensor moet meerdere jaren kunnen werken zonder vervanging van de batterijen. Daarom maak ik gebruik van embedded hardware met beperkte hardware specificaties (zie Tabel 2 AVR Raven hardware specificatie). Vooral de aanwezige hoeveelheid flash en geheugen (SRAM) is een beperkende factor. Gelukkig past mijn implementatie van de door Amigo gebruikte standaarden in de aanwezige hoeveelheid flash en geheugen. Zie onderstaande tabel voor het geheugengebruik.

MCU	MCU #1	MCU #2
Type	ATmega3290P	ATmega1284P
Doel	User I/O + LCD	RF Stack
Flash aanwezig (bytes)	131072	32768
Flash gebruikt (bytes)	58612	11712
Flash gebruikt (percentage)	**44,7%**	**35,7%**
SRAM aanwezig (bytes)	16384	2048
SRAM gebruikt (bytes)	12845	452
SRAM gebruikt (percentage)	**78,4%**	**22,1%**

Tabel 3 Geheugengebruik temperatuursensor

6LoWPAN maakt het mogelijk om via IPv6 te communiceren over de IEEE 802.15.4 standaard. De belangrijkste nadelen van 6LoWPAN, kleine pakketgrootte en lage bandbreedte, zijn eigenschappen van de IEEE 802.15.4 standaard (zie Tabel 1 Voor- en nadelen 6LoWPAN). 6LoWPAN is ingezet om het Amigo netwerk uit te breiden met low-power draadloze sensoren. Voor dit doel spelen de nadelen van 6LoWPAN helemaal geen rol. Om een meetwaarde te versturen heb je immers helemaal geen snelle netwerkverbinding nodig.

De temperatuursensor en de PC applicatie verschillen heel erg van elkaar. De software van de temperatuursensor is geschreven in de programmeertaal C, maakt gebruik van het Contiki besturingssysteem en draait op een ARM processor. De PC applicatie daarentegen is geschreven in C# en draait onder Windows XP op een Intel processor. Door het implementeren van de door Amigo gebruikte standaarden (zoals SOAP en WS-Discovery), heb ik aangetoond dat de temperatuursensor kan samenwerken met de PC applicatie.

9. Aanbevelingen

Als eerste zou ik TASS willen aanbevelen om WS-Eventing te implementeren. Nu is het alleen mogelijk om de temperatuur op te vragen bij de sensor. WS-Eventing maakt het mogelijk om je te abonneren bij de sensor, waarna deze bij temperatuurveranderingen zelf een nieuwe meetwaarde kan opsturen. WS-Eventing was onderdeel van de oorspronkelijke opdracht en is meegenomen in de Software Requirements Specificatie en het systeemontwerp, maar er was helaas onvoldoende tijd om deze feature te implementeren.

Ik zou willen aanbevelen om onderzoek te doen hoe de bestaande Android en iOS Amigo applicaties omgezet kunnen worden naar IPv6 en naar WS-Discovery Managed Mode zodat deze ook gebruik kunnen maken van de temperatuursensor.

Het eindresultaat is niet alleen een temperatuursensor, maar ik heb ook een basis gelegd om het Amigo platform uit te breiden met andere low-power draadloze sensoren. Denk hierbij aan een lichtsensor, bewegingssensor, koolmonoxidemelder of een rookmelder. Deze voorbeelden zouden allemaal via 6LoWPAN aangesloten kunnen worden op het Amigo netwerk. Dat zou eventueel zelfs kunnen op basis van de door mij gebruikte hardware, de AVR Raven. Naast de temperatuursensor beschikt de AVR Raven namelijk over vier digitale I/O pinnen.

Aansluitende op het vorige punt zou het misschien een idee zijn om een elektronica student één printplaat te laten ontwerpen waar al deze sensoren op geïntegreerd zijn. Dan kan er ook meer nadruk gelegd worden op het lage stroomverbruik. De AVR Raven bevat bijvoorbeeld twee microcontrollers en een LCD scherm, terwijl het ook mogelijk moet zijn om een sensor te maken met maar een microcontroller en zonder LCD scherm.

Als laatste zou ik TASS willen aanbevelen om de ontwikkelingen op het gebied van Linux ondersteuning voor het routeren van IPv6 multicast verkeer te blijven volgen. Nu is deze feature van de Linux kernel nog gemarkeerd als experimenteel. De grotere Internet Providers van Nederland zoals UPC, Ziggo en KPN hebben bekend gemaakt voor het einde van het jaar IPv6 te gaan uitrollen (zie [13], [14] en [15]). Hierdoor komen steeds meer mensen in aanraking met IPv6. Persoonlijk zou het me daarom niet verbazen als dit ook aanleiding is voor de Linux gemeenschap om de IPv6 multicast ondersteuning te verbeteren. Wellicht dat er in de toekomst onderzoek gedaan kan worden om eventueel weer over te schakelen van WS-Discovery Managed Mode naar Ad Hoc Mode.

10. Evaluatie

Ik wilde heel graag een embedded afstudeeropdracht met de mogelijkheid om te programmeren. Na een gesprek met Peter Vink, heb ik gekozen voor een Amigo afstudeeropdracht, omdat hier beide aspecten in terugkomen. Peter is heel enthousiast over Amigo en zit vol met ideeën voor mogelijke uitbreidingen. Wat me aanspreekt in Amigo is dat het innovatief is, de toepasbaarheid en het feit dat het tastbaar is. Met behulp van de temperatuursensor wordt het bijvoorbeeld mogelijk om de verwarming, airco, zonneschermen, etc. automatisch aan te sturen. En het leuke aan het tastbare is dat ik iedereen, ook aan mensen zonder ICT achtergrond, de temperatuursensor kan laten zien en uitleggen waar ik aan gewerkt heb.

Via TASS ben ik de afgelopen jaren bij verschillende projecten gedetacheerd geweest, maar altijd in de functie van Software Test Engineer. Na jaren andermans code getest te hebben, wil ik in de toekomst heel graag zelf software ontwikkelen. Daarom had ik voor mezelf als doel gesteld om tijdens mijn afstuderen meer parate kennis op te bouwen van een programmeertaal. Dat doel is gelukt. Ik heb gebruik gemaakt van de programmeertalen C en C#. Ik heb vooral veel kennis en ervaring kunnen opdoen van de programmeertaal C.

Ik ben heel tevreden met de gekozen aanpak. Ik heb veel tijd besteed aan het onderzoek. Dit komt omdat 6LoWPAN gebaseerd is op IPv6. Ik heb dus onderzoek gedaan naar twee nieuwe technologieën. Daarnaast heb ik uiteraard onderzoek gedaan naar de werking van Amigo. Gedegen onderzoek is zeer belangrijk om de juiste basis te creëren om zo goed mogelijke beslissingen te kunnen nemen. Na het onderzoek heb ik eerst alle requirements uitgewerkt in een systeemontwerp en daarna per feature een detailontwerp gemaakt, de bijbehorende requirements geïmplementeerd en de code getest. Door cyclisch en iteratief per feature deze stappen te doorlopen heb ik nu een werkend eindproduct, ondanks het feit dat er onvoldoende tijd was om alle features uit te werken. Voor aanvang van het afstuderen had ik geen ervaring met de programmeertaal C. Daardoor vond ik het erg lastig om schattingen te maken voor de activiteiten uit de detail & implementatiefase.

Ik ben getrouwd, woon in een koopwoning en heb een zoontje van 15 maanden. Ik ben afhankelijk van mijn inkomen. Noodgedwongen ben ik daarom gestart met parttime afstuderen, 4 dagen in de week een drukke baan en 1 dag voor mijn studie. De combinatie van 4 dagen werken en 1 dag studeren is me erg zwaar gevallen. Deze periode verliep erg moeizaam. Uiteindelijk heb ik de laatste drie maanden (januari tot en met maart 2012) fulltime verlof opgenomen om mijn studie af te ronden. Parttime afstuderen raad ik iedereen af.

Ik ben heel erg blij met het eindresultaat. De temperatuursensor kan op halve graden nauwkeurig de temperatuur meten en is compatibel met het bestaande Amigo platform. Ook heb ik hiermee een basis gelegd om in de (nabije) toekomst Amigo uit te breiden met andere low-power draadloze sensoren. Daarnaast heb ik waardevolle kennis opgedaan van 6LoWPAN en IPv6.

Niet alleen TASS heeft baat bij deze opdracht. Tijdens het onderzoek naar de werking van IPv6 heb ik voor mij thuis een IPv6 tunnel aangevraagd bij SixXS (http://www.sixxs.net/). Inmiddels heb ik via SixXS de beschikking over een eigen IPv6 subnet. Ook de temperatuursensor wil ik thuis gaan gebruiken.

Binnen TASS wordt er gewerkt aan het speerpunt Embedded Services: “Everything becomes connected”. Dit speerpunt bestaat uit subgroepen die zich onder andere bezig houden met: embedded devices, Android en Amigo. Ik vind het heel leuk dat ik hiervoor gevraagd ben en dat ik op deze manier betrokken kan blijven bij de toekomstige ontwikkeling van het Amigo platform binnen TASS.

Op 27 april vindt de overdracht plaats van mijn afstudeeropdracht aan TASS. Ik vind het erg jammer dat ik onvoldoende tijd heb gehad om alle features uit te werken. Na de overdracht wil ik graag hier verder mee aan de slag gaan.

Literatuurlijst

[1] **KnowHow 02 / juni 2007**
Artikel Impact bladzijde 10 en 11
Telematica Instituut
http://www.telin.nl/index.cfm?type=knowhow

[2] **Contiki is IPv6 Ready**
http://www.osnews.com/story/20423/Contiki_Is_IPv6_Ready
https://www.ipv6ready.org/db/index.php/public/logo/01-000473/

[3] **Contiki heeft geen IPv6 multicast ondersteuning**
Zoek op: "Multicast Support"
http://dak664.github.com/contiki-doxygen/a01685.html#_details

[4] **IPv6 Ready fase 1 is opgevolgd door fase 2**
http://www.v6pc.jp/jp/pdf/101201_PressRelease_en.pdf

[5] **MLDv2 ondersteuning verplicht bij IPv6 Ready fase 2**
http://www.ipv6ready.org/?page=phase-2-about

[6] **IPv6 Advanced Protocols Implementation**
Hoofdstuk 2.10 IPv6 Multicast Forwarding
Auteur: Keiichi Shima en Qing Li
ISBN13: 9780123704795

[8] **6LoWPAN The Wireless Embedded Internet**
Auteur: Zach Shelby en Carsten Bormann
ISBN13: 9780470747995

[9] **Interconnecting Smart Objects With IP**
Auteur: Jean-Philippe Vasseur en Adam Dunkels
ISBN13: 9780123751652

[10] **Het W-model**
http://www.smartest.nl/verdieping/wmodel

[11] **Contiki processes**
http://dak664.github.com/contiki-doxygen/a01627.html

[12] **Protothreads – Lightweight, Stackless Threads in C**
http://www.sics.se/~adam/pt/

[13] **Gereedheid UPC-netwerk voor IPv6**
http://vragen.upc.nl/app/answers/detail/a_id/415/~/gereedheid-upc-netwerk-voor-ipv6

[14] **Ziggo en het nieuwe internetprotocol IPv6**
https://www.ziggo.nl/#entertainment/nieuws/ziggo/ziggo/0/ziggo-en-het-nieuwe-internetprotocol-ipv6

[15] **IPv6 wordt de nieuwe standaard voor internetadressen van KPN**
http://www.kpn.com/zakelijk/service/ipv6-wordt-de-nieuwe-standaard-voor-internetadressen-van-kpn.htm

Bijlagen

Deze pagina is opzettelijk leeg gelaten.

Draadloze communicatie voor Amigo

Bijlage A: Project Management Plan

Dit document is gepubliceerd door:
TASS BV
Eindhoven, Nederland

Commentaar en suggesties kunnen worden gestuurd naar:
TASS B.V.
Postbus 80060
5600 KA Eindhoven
Nederland
Tel: +31 40 2503200
Fax: +31 40 2503201

Document Geschiedenis

Versie	Status	Datum	Auteur	Wijzigingen
0.1	Proposed	17-10-2010	R. Zuidinga	Initiële versie.
0.2	Proposed	29-10-2010	R. Zuidinga	Review commentaar verwerkt.
0.3	Proposed	2-11-2010	R. Zuidinga	Geen inhoudelijke veranderingen. Goedgekeurd door Jos Boonen.
0.4	Proposed	23-11-2010	R. Zuidinga	Inhoud onderzoek in meer detail beschreven (zie paragraaf 2.4.2). Inhoud Increment 1 aangepast (zie §2.5.1). Inhoud Increment 2 aangepast (zie §2.5.2). Project Rapportage (detail planning) toegevoegd aan hoofdstuk 9.
0.5	Proposed	26-11-2010	R. Zuidinga	Open punt 1 & 2 opgelost. Hoofdstuk 9 herschreven. Planning aangepast (Fase 0 + 28 uur, Fase 2 INC1 + 11 uur en Fase 2 INC2 – 3 uur). Open punt 3 toegevoegd.
1.0	Approved	17-12-2010	R. Zuidinga	Geen inhoudelijke veranderingen. Goedgekeurd door Jos Boonen, Peter Vink.

Overzicht wijzigingen in versie 0.4:

- Systeem Test increment 1 vervallen.
- Temperatuursensor verplaatst van increment 1 naar increment 2.
- Grafische meetwaarden applicatie verplaatst van increment 1 naar increment 2.
- Simpele (niet grafische) meetwaarden applicatie toegevoegd aan increment 1.
- Omgeving (hardware / software stack) leren kennen toegevoegd aan increment 1.
- Frequentie metingen requirements (groep 6) verplaatst naar increment 3.
- Integratie locatiebepaling verplaatst van increment 2 naar increment 3.

Open Punten

Open punt 1: Opgelost, technische begeleider aangesteld.

Open punt 2: Opgelost, VPN toegang tot TASS intranet gekregen.

Open punt 3: Er was 128 uur gepland voor de drie increments (zie eerdere versie van dit document). Na het maken van de detail planning komt increment 1 uit op 139 uur en increment 2 op 125 uur. Increment 1 en 2 kosten samen dus 8 uur meer tijd dan initieel ingepland. Deze afwijking is erg klein en geaccepteerd. Echter increment 3 komt uit op 184 uur, 56 uur meer dan beschikbaar. Samen met Peter Vink zal er naar een oplossing gezocht worden om de benodigde tijd te beperken. Zeer waarschijnlijk zal dit betekenen dat er bepaalde requirements niet geïmplementeerd kunnen worden.

Inhoudsopgave

1. Introductie

1.1 Doel

Dit plan beschrijft:

- Hoe het project is georganiseerd.
- Wat het globale schema is en welke werkproducten wanneer worden geleverd.
- Op welke aannames en randvoorwaarden het plan is gebaseerd.
- Hoe de kwaliteit wordt gewaarborgd.
- Wat de afspraken met de klant zijn m.b.t. acceptatie en onderhoud.

Dit document is bedoeld als een overeenkomst tussen Ralph Zuidinga en TASS.
Dit document moet worden goedgekeurd door de klant, en de afstudeerbegeleider.

1.2 Distributielijst

Naam	Organisatie	Functie
Jos Boonen	Fontys	Afstudeerbegeleider
Peter Vink	TASS	Klant / Bedrijfsbegeleider
Tiny Henst	TASS	People manager
Technische begeleider	TASS	Technische begeleider

1.3 Contactgegevens

Naam	E-mail	Telefoon
Ralph Zuidinga	rzuidinga@eltimo.nl	06 – 5568 3393
Jos Boonen	j.boonen@fontys.nl	0877 – 875 239
Peter Vink	Peter.Vink@tass.nl	040 – 2503209
Tiny Henst	Tiny.Henst@tass.nl	040 – 2503208

1.4 Referenties

Referentie	Document ID	Titel
[HBPROJ]	TF09-QM-015	TASS Handboek Project Guide
[WBS]	WBS	Project Rapportage

1.5 Termen en Afkortingen

Term	Omschrijving
TASS	Technical Application Software Solutions
PMP	Project Management Plan; *dit document*
SRS	Software Requirements Specificatie
BAPAO	Big Amigo Pc that's Always On (de centrale server in het Amigo netwerk)
MoSCoW	Methode van prioriteiten stellen aan de requirements.
INC	Increment
Smoketest	Een smoketest is een sanity check; een test om vast te stellen of een software build een werkbare software build is.
TTM	Test Traceability Matrix
CE-HTML	Een taal om gebruikersinterface pagina's te maken voor consumentenelektronica apparaten zoals televisies.
X10	X 10 is een industriestandaard voor communicatie tussen modules via het lichtnet. De standaard wordt vooral gebruikt om Home Automation te kunnen realiseren.

1.6 Bedrijf

TASS software professionals is een dienstverlener op het gebied van technische en embedded software. Een mensgerichte organisatie die al meer dan 30 jaar voorop loopt in de ontwikkeling van software voor technisch hoogwaardige producten. Eerst als onderdeel van Philips en sinds 2007, met succes als zelfstandige organisatie.

De kernactiviteit van TASS is het aanbieden van services en oplossingen op het gebied van technische en embedded software. Embedded software speelt een steeds grotere rol in onze samenleving. Hoewel zelden zichtbaar voor de consument, vormt software vaak het hart van het product. TASS stelt deze expertise beschikbaar als een totaaloplossing dan wel op project of consultancy basis. Wat TASS zo speciaal maakt, is de enorme kennis en ervaring die TASS heeft opgedaan tijdens strategische state-of-the-art projecten.

2. Project Beschrijving

2.1 Aanleiding

Binnen TASS wordt er gewerkt aan het speerpunt Embedded Services: “alles wordt connected”. Technieken die hier gebruikt worden zijn veelal geleend uit de wereld van het Internet, zoals web services, maar met specifieke toevoegingen, zoals discovery. Uit eerdere onderzoeken is het zogenaamde ‘Amigo platform’ ontwikkeld waar diverse aspecten van een toekomstig in-home network hun plaats hebben gekregen (discovery, semantic discovery, interoperabiliteit, domotica, tracking etc.).

Het Amigo platform is service georiënteerd. Zo bestaat de huidige demonstatie opstelling van het Amigo platform bijvoorbeeld uit een lamp die als service de mogelijkheid aanbiedt om het licht aan of uit te zetten. Andere apparaten binnen het Amigo netwerk kunnen van deze service gebruik maken en op die manier daadwerkelijk de lamp bedienen. Het Amigo platform maakt gebruik van Web Services, UPnP, OSGI, WS-Discovery, TCP/IP, etc. De huidige demonstratie omgeving maakt momenteel zowel gebruik van een bedraad netwerk als van een draadloos Wi-Fi netwerk.

2.2 Probleem

Het Amigo platform ondersteunt nu alleen draadloze apparaten op basis van Wi-Fi en kan dus niet overweg met apparaten die gebaseerd zijn op andere draadloze standaarden zoals Zigbee of Bluetooth. Daarnaast wordt de ‘Plug and Play’ functionaliteit van het Amigo platform momenteel niet gebruikt. Het bestaande systeem moet uitgebreid worden met dynamisch gedrag (bijvoorbeeld dynamisch reageren op het aan en uitzetten van apparaten). En als laatste moet de gebruiker het systeem eenvoudiger kunnen bedienen. Het bestaande systeem moet daarvoor uitgebreid worden met de mogelijkheid om binnen een bepaalde context acties te kunnen koppelen aan gebeurtenissen (bijvoorbeeld als het donker is en je komt de kamer binnen dan moet het licht aan).

2.3 Opdracht

Het bestaande Amigo platform moet uitgebreid worden met low-power draadloze sensoren. De sensoren moeten meerdere jaren kunnen werken zonder dat de batterij moet worden vervangen. Er moet onderzocht worden welke draadloze technologie zoals Zigbee of Bluetooth hiervoor het beste geschikt is. Tijdens het onderzoek moeten er minimaal twee draadloze technologieën vergeleken worden. Daarnaast moet er een applicatie gemaakt worden (met nieuwe of aangepaste Amigo middleware services) die gebruik maakt van de sensoren en die dynamisch kan reageren op bepaalde gebeurtenissen (bijvoorbeeld kan reageren op berichten van de sensoren). De naam van het project, Dynamische Domotica, geeft al aan dat het dynamische gedrag heel belangrijk is. Als laatste moet de gebruiker het systeem eenvoudig kunnen bedienen (bijvoorbeeld door definitie van regels). Om bovenstaande te kunnen demonstreren moet een Amigo systeem gemaakt worden (hardware en software). Bepaalde hardware uit het bestaande Amigo systeem, zoals X10, moet worden hergebruikt in deze (nieuwe) demonstratie omgeving.

De volgende subdoelen zijn te onderkennen:

- Onderzoek naar draadloze technologieën voor uitbreiding van het Amigo systeem.
- Het bestaande Amigo systeem uitbreiden met low-power draadloze sensoren op basis van één van de onderzochte technologieën.
- Het maken van een applicatie, die voldoet aan de SRS, en gebruik maakt van de low-power draadloze sensoren en die het dynamische gedrag van het systeem laat zien.
- Onderzoek naar syntax / semantiek voor het vastleggen van regels.
- Applicatie uitbreiden zodat deze door de gebruiker eenvoudig te bedienen is.

2.4 Fasering

Het project zal worden onderverdeeld in vijf fases.

Op alle fases, met uitzondering van de eerste fase project definitie, is timeboxing toegepast. De beschikbare tijd voor dit afstudeerproject is verdeeld over de verschillende fases (zie kolom aantal uur). Na het afronden van de Software Requirements Specificatie moet de klant per requirement aangeven wat de prioriteit van dit requirement is. Tijdens de design en implementatie fases wordt een ontwerp gemaakt wat geschikt is voor de complete gewenste functionaliteit. De daadwerkelijke implementatie wordt volgens de prioriteit van de requirements gedaan. Het kan dus zijn dat sommige requirements met een lage prioriteit doorschuiven naar een latere fase of komen te vervallen.

Zowel de opdeling in fases als de toepassing van timeboxing is gedaan om te waarborgen dat er daadwerkelijk resultaat is na het afronden van de afstudeeropdracht.

2.4.1 Fase 0: Project definitie

Het vastleggen van het verwachte verloop van het project, afspraken tussen betrokken mensen en een tijdsplanning. Uitzoeken wat de wensen van de klant zijn. Deze wensen vervolgens vastleggen in concrete en testbare requirements. Verdiepen in de opzet en werking van het huidige Amigo project.

Mijlpaal / Oplevering	Verwachte datum	Goedkeuring door
Project Management Plan	26-11-2010	Peter Vink Jos Boonen
Software Requirements Specificatie	26-11-2010	Peter Vink

2.4.2 Fase 1: Onderzoek

Het huidige systeem moet uitgebreid worden met low-power draadloze sensoren die meerdere jaren kunnen werken zonder dat de batterij moet worden vervangen. Er moet onderzocht worden welke draadloze technologie hiervoor het beste geschikt is. Daarnaast moet tijdens het onderzoek ook bepaald worden welke hardware aangeschaft moet worden. De hardware moet uiteraard kunnen communiceren op basis van de gekozen draadloze technologie, moet beschikken over een licht- en temperatuursensor en moet gevoed kunnen worden door een of meerdere batterijen. Tot slot moet gekeken worden welke software stack, in combinatie met de hardware, gebruikt kan worden om te communiceren op basis van de gekozen draadloze technologie. Toelichting: Door voor de draadloze communicatie gebruik te maken van een bestaande software stack, blijft tijdens het project de focus behouden op de requirements uit de Software Requirements Specificatie.

Mijlpaal / Oplevering	Aantal uur	Verwachte datum	Goedkeuring door
Onderzoeksrapport	80	9-1-2011	Peter Vink

2.4.3 Fase 2: Design en implementatie 1

Tijdens deze fase moeten alle requirements uit de Software Requirements Specificatie geïmplementeerd worden met uitzondering van de requirements met betrekking tot het feit dat de gebruiker het systeem eenvoudig moet kunnen bedienen.

Deze fase is opgesplitst in twee delen; increment 1 en increment 2. Ieder increment wordt afgesloten met een demonstratie. De demonstratie aan het einde van increment 2 moet onder andere de volgende functionaliteit bevatten:

- Als het te koud wordt moet de verwarming aan,
- Als het warm genoeg is moet de verwarming uit,
- Als het (buiten) donker wordt en er is iemand in de kamer dan moet het licht aan,
- Als het (buiten) licht wordt dan moet het licht in de kamer uit.

De increments staan in meer detail beschreven in paragraaf 2.5.1 en 2.5.2.

De demonstratie moet dus dynamisch gedrag bevatten (bijvoorbeeld reageren op het feit dat de temperatuur onder een bepaalde waarde daalt), maar in deze fase zijn de regels hard-coded in de software vastgelegd. Het is voor de gebruiker dus nog niet mogelijk het systeem te bedienen.

INC	Mijlpaal / Oplevering	Aantal uur	Verwachte datum	Goedkeuring door
Increment 1	Product Risico Analyse	139	21-1-2011	Peter Vink
	Software Design Document		11-2-2011	Peter Vink
	Software		27-2-2011	Peter Vink
	Demonstratie		25-2-2011	Peter Vink
Increment 2	Software Design Document	125	11-3-2011	Peter Vink
	Test Specificatie / Rapportage		17-4-2011	Peter Vink
	Software		17-4-2011	Peter Vink
	Demonstratie		15-4-2011	Peter Vink

2.4.4 Fase 3: (Mini) onderzoek

De gebruiker moet het systeem eenvoudig kunnen bedienen. De gebruiker moet in staat zijn om het dynamische gedrag van het systeem vast te leggen (bijvoorbeeld door definitie van regels). De gebruiker moet dus de mogelijkheid krijgen om binnen een bepaalde context acties te kunnen koppelen aan gebeurtenissen (bijvoorbeeld als het donker is en je komt de kamer binnen dan moet het licht aan). Tijdens deze fase moet onderzocht worden welke syntax / semantiek het beste geschikt is voor het vastleggen van regels.

Mijlpaal / Oplevering	Aantal uur	Verwachte datum	Goedkeuring door
Onderzoeksrapport	40	8-5-2011	Peter Vink

2.4.5 Fase 4: Design en implementatie 2

Tijdens deze laatste fase moet de uitkomst van het onderzoek uit de vorige fase geïmplementeerd worden. Er moet een demonstratie gemaakt worden die laat zien dat een gebruiker het systeem eenvoudig kan bedienen. Deze fase is niet opgesplitst, en bestaat maar uit één deel; increment 3. Tot slot moet de bestaande Amigo locatiebepaling functionaliteit geïntegreerd worden met de nieuwe, tijdens dit project geschreven, Amigo software.

INC	Mijlpaal / Oplevering	Aantal uur	Verwachte datum	Goedkeuring door
Increment 3	Software Design Document	128	22-7-2011	Peter Vink
	Test Specificatie / Rapportage		28-8-2011	Peter Vink
	Software		28-8-2011	Peter Vink
	Demonstratie		26-8-2011	Peter Vink

2.5 Increments

2.5.1 Fase 2: Increment 1

Tijdens de fase voorafgaand aan dit increment is onderzocht welke draadloze technologie het beste geschikt is voor low-power draadloze sensoren, de benodigde hardware is aangeschaft en er is bepaald welke software stack gebruikt gaat worden (zie paragraaf 2.4.2).

Dit increment richt zich op het leren kennen van de omgeving en de software voor de lichtsensor. De omgeving bestaat uit de hardware en de software stack om te kunnen communiceren op basis van de gekozen draadloze technologie. Voor de embedded software die op de lichtsensor zelf draait moet eerst een software design ontworpen worden en vervolgens moeten de requirements met betrekking tot de lichtsensor geïmplementeerd worden. De temperatuursensor valt buiten de scope van dit increment.

Het leren kennen van de omgeving bestaat onder andere uit de volgende punten:

- Opzetten van een omgeving om te kunnen bouwen (compiler, linker).
- Uitzoeken hoe de waarde van de lichtsensor uit te lezen is.
- Uitproberen van de debug mogelijkheden.
- Inlezen in de documentatie van de software stack.

Tijdens dit increment moet een demonstratie gemaakt worden met de volgende functionaliteit:

- De lichtsensor moet kunnen rapporteren of het licht of donker is.
- Er moet een simpele (niet grafische) applicatie geschreven worden. Op het moment dat een lichtsensor een nieuwe waarde rapporteert dan moet de applicatie in real-time:
    - aangeven dat er een nieuwe waarde ontvangen is;
    - de nieuwe waarde tonen.

2.5.2 Fase 2: Increment 2

Dit increment richt zich op de software voor de temperatuursensor, het uitbreiden van de functionaliteit van de simpele (niet grafische) applicatie en het maken van een nieuwe grafische applicatie om de meetwaarden van de licht- en temperatuursensoren weer te geven.

Het eerste punt, de temperatuursensor, bestaat uit het bijwerken van het software design van de lichtsensor dusdanig dat het ook geschikt is voor de temperatuursensor en vervolgens het implementeren van de requirements met betrekking tot de temperatuursensor. Het tweede punt, het uitbreiden van de simpele (niet grafische) applicatie, bestaat uit het toevoegen van dynamisch gedrag (bijvoorbeeld reageren op het feit dat de temperatuur onder een bepaalde waarde daalt) en de integratie van X10 aansturing uit het bestaande Amigo systeem. In dit increment is het dynamische gedrag hard-coded in de software vastgelegd. Het derde en laatste punt bestaat uit het ontwerpen van een software design en vervolgens het implementeren van de requirements met betrekking tot de applicatie om de meetwaarden van de licht- en temperatuursensoren weer te geven.

Tijdens dit increment moet een demonstratie gemaakt worden met de volgende functionaliteit:

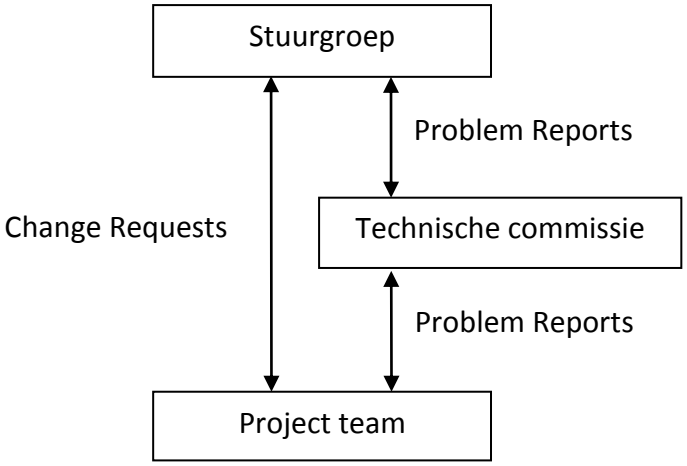
- Als het kouder wordt dan een bepaalde temperatuur dan moet de kachel aan.
- Als het warmer wordt dan een bepaalde temperatuur dan moet de kachel uit.
- Als het warmer wordt dan een bepaalde temperatuur dan moet de ventilator aan.
- Als het kouder wordt dan een bepaalde temperatuur dan moet de ventilator uit.
- Als het donker wordt dan moeten de gordijnen dicht.
- Als het donker wordt dan moet de verlichting aan.
- Als het buiten licht wordt dan moeten de gordijnen open.
- Als het buiten licht wordt dan moet de verlichting uit.
- Amigo middleware moet weten in welke ruimte een sensor zich bevindt.
- Aanwezige sensoren moeten worden getoond op een plattegrond.
- De plattegrond moet een CE-HTML pagina zijn.
- Op het moment dat een sensor een nieuwe waarde rapporteert dan moet de plattegrond in real-time:
    - aangeven dat er een nieuwe waarde ontvangen is;
    - de nieuwe waarde tonen.

2.5.3 Fase 4: Increment 3

Fase 4 is niet opgesplitst, en bestaat maar uit één deel; increment 3. De functionaliteit van increment 3 staat daarom beschreven in paragraaf 2.4.5 Fase 4: Design en implementatie 2.

3. Organisatie

3.1 Organisatieschema



3.2 Stuurgroep

De stuurgroep is verantwoordelijk voor:

- Periodieke voortgangscontrole in termen van tijd, functionaliteit en kwaliteit.
- Beslissingen over wijzigingen (Change Requests) die het Project Management Plan of de Software Requirements Specificatie beïnvloeden.
- Organisatorische zaken met betrekking tot resources, communicatie, mijlpalen, opleveringen, etc.
- Prioriteitsstelling binnen het project.
- Evaluatie van de resultaten.

De stuurgroep bestaat uit de volgende personen:

Rol	Naam	Functie in het project
Voorzitter	Peter Vink	Klant / Opdrachtgever
Lid / Notulist	Ralph Zuidinga	Project lid

Vergaderfrequentie:

- Op verzoek van een van de leden van de stuurgroep.
- Na het afsluiten van iedere fase.
- Bij fase 1 en 4 na het afsluiten van ieder increment
  (kort na de demonstratie waarmee het increment wordt afgesloten).
- Minstens één keer per twee weken.

3.3 Technische commissie

De technische commissie is verantwoordelijk voor:

- Feedback geven op het software design.
- Bespreken van problemen die zich hebben voorgedaan sinds de vorige vergadering.
- Bespreken van technische aspecten.
- Advies geven over technische beslissingen.
- Het nemen van beslissingen over Problem Reports.

De technische commissie bestaat uit de volgende personen:

Rol	Naam	Functie in het project
Voorzitter	Technische begeleider	Technische begeleider
Lid	Peter Vink	Bedrijfsbegeleider
Lid	Ralph Zuidinga	Project lid

Vergaderfrequentie:

- Op verzoek van een van de leden van de technische commissie.
- Bij fase 1 en 4 na het ontwerpen van het software design.
- Maximaal één keer per week.

3.4 Project Team

Het project team is verantwoordelijk voor:

- Uitvoeren van het onderzoek, ontwerpen van het software design, implementatie en testen van de software om zodoende het Dynamische Domotica project succesvol af te ronden.
- Het schrijven van de documenten zoals beschreven in paragraaf 2.4 Fasering.
- Planning en dagenverantwoording bijhouden en bespreken met de stuurgroep.
- Afstudeerbegeleider op de hoogte houden van de voortgang.

Het project team bestaat uit één persoon.

Naam	Periode	Niveau	Benodigde kennis
Ralph Zuidinga	Volledig	HBO Informatica	C of C++ en C# of Java

3.5 Afstudeerbegeleider

De afstudeerbegeleider is verantwoordelijk voor:

- Bijhouden of het afstuderen aan alle eisen vanuit de opleiding blijft voldoen.
- Regelmatig contact met het project lid (liefst per e-mail).
- Af en toe project lid bezoeken op de werkplek bij TASS
  (bijvoorbeeld bij de demonstraties aan het einde van een increment).
- Review commentaar geven op het (proposed) Project Management Plan.
- Meehelpen met het 'bewaken' van de planning / voortgang.
- Review commentaar geven op het (proposed) afstudeerverslag.
- Beoordelen van het afstudeerverslag en presentatie.

Er is één afstudeerbegeleider.

Naam	Organisatie
Jos Boonen	Fontys

4. Randvoorwaarden

4.1 Aannames en Afhankelijkheden

Het eindproduct, het te bouwen Amigo systeem, is gebaseerd op het in eerdere onderzoeken ontwikkelde 'Amigo platform' en hergebruikt een deel van de hardware uit het bestaande Amigo systeem. Binnen de scope van het project valt alleen de integratie van de bestaande hardware die daadwerkelijk hergebruikt wordt. Niet alle bestaande Amigo hardware wordt opgenomen in het eindproduct (de demonstratie).

Dit Project Management Plan is gebaseerd op de aanname dat de bestaande Amigo software werkt volgens de specificatie en vrij is van fouten. Binnen de scope van het project valt alleen het testen van de nieuw te ontwikkelen of aan te passen software. Er worden geen testen gemaakt voor de bestaande software die zonder aanpassingen 1-op-1 hergebruikt kan worden.

Project team is vrij in de keuze van ontwikkelomgeving.

De planning in dit Project Management Plan is gebaseerd op de aanname dat de benodigde apparatuur, infrastructuur en tools, zoals beschreven in onderstaande paragraaf, tijdig beschikbaar zijn. Dat wil zeggen uiterst op de in de tabel vermelde datum.

4.2 Apparatuur, Infrastructuur en Tools

Naam	Aantal	Datum	Toelichting
Amigo broncode en voorbeelden	1	26-11-2010	
BAPAO software	1	26-11-2010	
Laptop	1	26-11-2010	Om te kunnen gebruiken als BAPAO server. Dit moet een laptop zijn om ook thuis aan het project te kunnen werken.
VPN verbinding met TASS netwerk	1	26-11-2010	
Ontwikkelomgeving	1	14-1-2011	
Software om de coverage van de unit verificatie mee te meten.	1	14-1-2011	
Software om de complexiteit van de broncode mee te meten.	1	14-1-2011	
Locatiesensor	2	14-1-2011	
Draadloze lichtsensor	2	14-1-2011	Op basis van de technologie gekozen tijdens het onderzoek.
Draadloze temperatuursensor	2	14-1-2011	

Naam	Aantal	Datum	Toelichting
Draadloze router of indien niet mogelijk draadloze USB stick	1	14-1-2011	Op basis van de technologie gekozen tijdens het onderzoek.
Probe om software te debuggen / downloaden naar de hardware	1	14-1-2011	
Prisma test tool	1	14-1-2011	Voor het bepalen van de product risico analyse.
X10 computer interface	1	4-3-2011	Compatibel met Amigo
X10 doorsteek schakelmodule	3	4-3-2011	
X10 gordijnrail	1	4-4-2011	Benodigd (tijdelijk) voor het geven van een demonstratie aan het einde van increment 2, 3 en tijdens de afstudeerpresentatie.
Bureaulamp	1	4-4-2011	
Bureau ventilator	1	4-4-2011	
Elektrische kachel	1	4-4-2011	

4.3 Risico's

In deze sectie worden de top-5 project risico's vermeld met de preventieve acties die worden genomen om te verkomen dat deze risico's werkelijkheid worden.

#	Risico	Aktie
1	Geen werkend product na afloop van het project.	Design en implementatie (fase 2 en 4) opsplitsen in delen. Ieder increment afsluiten met een werkende demonstratie.
2	Project te groot voor 1 persoon.	Klant per requirement prioriteit laten aangeven (volgens MoSCoW methode). Belangrijkste requirements eerst implementeren.
3	Hardware niet verkrijgbaar.	Hardware laten maken, temperatuur of lichtsensor simuleren door een schakelaar of potmeter. In geval van nood: Overstappen op een andere draadloze technologie die onderzocht is tijdens fase 1.
4	Hardware defect.	Hardware die niet ingebouwd is, dus waarbij het mogelijk is om de elektronica aan te raken, kan snel stuk gaan. Kijken voor alternatief dat wel ingebouwd is, of minimaal twee stuks bestellen.
5	Benodigde hardware duurder dan budget.	Van te voren in overleg met TASS het maximale budget bepalen. In geval van nood: Op zoek naar tweedehands/gebruikte hardware.

5. Software Test Plan

5.1 Scope

De testen die tijdens dit project worden uitgevoerd richten zich alleen op verificatie, dat wil zeggen er wordt alleen gecontroleerd of het systeem volgens de Software Requirements Specificatie is gemaakt. Software verificatie wordt zowel uitgevoerd op unit niveau als ook op systeem niveau. Validatie, controleren dat het juiste systeem is gemaakt, valt buiten de scope van dit project. Na acceptatie (zie hoofdstuk 8) is het aan de klant om te controleren of het systeem doet wat hij wil.

5.2 Aanpak

5.2.1 Unit Verificatie

Het doel van de unit verificatie is om de functionele units (een component of klasse) onafhankelijk van elkaar te testen op een correcte werking. Tijdens de unit verificatie ligt de focus op de interface van de unit, en worden de publieke methodes en functies getest op een correcte werking. Dit wordt gedaan door test cases op te zetten met behulp van tooling zoals cppunit of nunit. De werking van de methodes en functies wordt getest door deze zowel met correcte als met foutieve invoer aan te roepen. De unit verificatie zal volledig bestaan uit geautomatiseerde test cases en is daardoor ook in te zetten als regressie test of smoketest.
De test cases van de unit verificatie worden ook gebruikt om de code coverage vast te stellen.

Start criteria
De unit verificatie start aan het begin van ieder increment.

Stop criteria
Unit test rapport beschikbaar.
Alle test cases uitgevoerd.
Geen openstaande defecten die een normaal gebruik in de weg staan.
Minimaal 95% functie coverage.
Minimaal 60% regel of statement coverage.

5.2.2 Systeem Verificatie

Het doel van de systeem verificatie is om vast te stellen of het systeem gemaakt is volgens de Software Requirements Specificatie. De systeem verificatie wordt uitgevoerd aan het einde van ieder increment, na afloop van de unit verificatie.

Om er zeker van te zijn dat daadwerkelijk getest wordt of alle requirements correct zijn geïmplementeerd wordt een Test Traceability Matrix (TTM) bijgehouden. De TTM is overzicht waarin een relatie gelegd wordt tussen een requirement en een of meerdere test cases.

Het is onmogelijk om uitputtend te testen, dit kost simpelweg te veel tijd. Daarom wordt er voor aanvang van de systeem verificatie een product risico analyse uitgevoerd waarbij voor groepjes van requirements het risico wordt geschat. Voor ieder groepje van bij elkaar behorende requirements wordt het bedrijfsrisico (impact) en het technische risico (kans) bepaald. Dit levert een matrix op met onderstaande vier kwadranten.

Kwadrant	Locatie	Bedrijfsrisico (impact)	Technisch risico (kans)
I	Rechtsboven	Hoog	Hoog
II	Linksboven	Laag	Hoog
III	Rechtsonder	Hoog	Laag
IV	Linksonder	Laag	Laag

Per kwadrant wordt de testaanpak bepaald. Het idee is om de testaandacht te richten op de meest risicovolle gebieden en daarmee de kwaliteit van testen te verhogen. Het uitgangspunt is om 'goed genoeg' te gaan testen in plaats van strikt vast te houden aan perfectie en iedere uithoek van het systeem trachten te raken. Hieronder volgt de gekozen testaanpak.

Kwadrant	Testaanpak
I	Test cases worden van te voren vastgelegd in een test case specificatie. Bij het definiëren van de test cases wordt zoveel mogelijk gebruik gemaakt van test technieken.
II & III	Test cases worden van te voren vastgelegd in een test case specificatie. Bij het definiëren van de test cases wordt gebruik gemaakt van use cases en 'gezond verstand'.
IV	Dit kwadrant wordt alleen getest indien er tijd over is. De benodigde test cases worden ter plekke bedacht.

Start criteria
Unit verificatie afgerond en unit test rapport beschikbaar.
(Approved) Product risico analyse beschikbaar.
(Approved) Test case specificatie beschikbaar.
(Approved) Test Traceability Matrix beschikbaar.
Subversion branch beschikbaar.

Stop criteria
Systeem test rapport beschikbaar. Alle test cases uitgevoerd. Geen openstaande defecten die een normaal gebruik in de weg staan.

5.3 Testomgeving

De hardware, software omgeving en tooling die nodig zijn voor het testen zijn opgenomen in de tabel in paragraaf 4.2 Apparatuur, Infrastructuur en Tools.

6. Configuratie Management Plan

TASS gebruikt voor configuratie management het online programma GForge. GForge is een programma wat gebruikt kan worden voor zowel configuratie beheer als change management. Alle interne projecten van TASS, dus ook alle afstudeeropdrachten, maken gebruik van GForge. Het voordeel van GForge is dat alle data die bij het project hoort (broncode, change management, documentatie) op één makkelijke en overzichtelijke manier bij elkaar staat. GForge is bereikbaar op het volgende URL: http://psln014.tass.lan/.

6.1 Configuratie Beheer

Deze sectie beschrijft hoe het configuratie beheer is georganiseerd:

- GForge maakt voor configuratie beheer gebruik van subversion.
- Onder Windows moet de TortoiseSVN subversion client gebruikt worden.
- Zowel broncode als documentatie moet onder configuratie beheer worden opgeslagen.
- De software executable(s) moeten altijd het subversion revisie nummer bevatten.
- Voordat er gestart wordt met systeem verificatie moet er een branch gemaakt worden.
- Er mag alleen vanuit een branch, na systeem verificatie, software geleverd worden.
- Van iedere software levering moet een baseline gemaakt worden.
- Aan het einde van iedere fase / increment moet er een baseline gemaakt worden.

6.2 Change Management

Deze sectie beschrijft hoe het change management is georganiseerd:

- Voor change management wordt gebruik gemaakt van GForge.
- Change Requests worden afgehandeld door de stuurgroep.
- Problem Reports worden afgehandeld door de technische commissie.

7. Kwaliteit

De technische kwaliteit wordt bewaakt door goed te controleren dat het systeem volgens de Software Requirements Specificatie is gemaakt (zie hoofdstuk 5 Software Test Plan).

Aan het einde van ieder increment wordt een demonstratie gegeven. Deze demonstratie moet uiteraard betrouwbaar zijn. De betrouwbaarheid wordt bewaakt door te controleren dat het mogelijk is om de demonstratie drie keer achter elkaar te geven zonder tegen problemen aan te lopen.

Onderdeel van de opdracht is ook dat de gebruiker het systeem eenvoudig moet kunnen bedienen. Fase 3 en 4 richten zich op dit onderdeel van de opdracht. De kwaliteit van dit onderdeel is lastiger te bewaken. De term ‘eenvoudig’ is namelijk subjectief, iedereen kan een andere mening hebben over hoe het systeem eenvoudig te bedienen is. De kwaliteit van dit onderdeel wordt bewaakt door de klant er nauw bij te betrekken (bijvoorbeeld: schetsen / interviews). Pas na goedkeuring van de klant wordt deze functionaliteit geïmplementeerd.

De kwaliteit wordt ook bewaakt door het proces te volgen zoals omschreven in de TASS Handboek Project Guide [HBPROJ], door te werken met fases en increments en door op regelmatige basis overleg met de stuurgroep, technische commissie en afstudeerbegeleider (zie hoofdstuk 3 Organisatie).

8. Acceptatie en Onderhoud

Het eindresultaat van dit project is een demonstratie omgeving. Een Amigo opstelling bestaande uit hardware (low-power draadloze sensoren) en software die dynamisch gedrag laat zien en eenvoudig door een gebruiker te bedienen is.

Omdat het eindresultaat geen commercieel product is, maar een demonstratie omgeving, zijn de eisen met betrekking tot acceptatie en onderhoud lager. Zo wordt er bijvoorbeeld geen garantie gegeven na acceptatie door de klant.

8.1 Acceptatie

Fase 4 is de laatste fase van het project en bestaat uit één deel, increment 3. Ieder increment wordt afgesloten met een demonstratie. Na afloop van deze laatste demonstratie wordt het product overgedragen aan de klant. Binnen twee weken na overdracht van het product zal de klant het product onderwerpen aan een acceptatietest.

Hiervoor geldt:

- De klant zal een productgeneratie uitvoeren op een eigen lokaal software archief.
- De klant zal een acceptatietest uitvoeren op basis van een door hem vastgesteld testplan.
- De klant zal de resultaten van de acceptatietest rapporteren via een acceptatie test rapport, waarin onder andere alle gevonden defecten worden vermeld.
- Defecten die een normaal gebruik van het product in de weg staan, dit ter beoordeling van de stuurgroep, worden binnen een afgesproken termijn door het project team hersteld waarna de klant de acceptatie test geheel of gedeeltelijk zal herhalen.
- Zodra geen defecten meer worden gevonden zal de klant schriftelijk bevestigen dat hij het product accepteert. Indien de klant binnen drie weken na de laatste aflevering niet reageert wordt het product geacht door hem te zijn geaccepteerd.

8.2 Garantie

Als tijdens de acceptatie testen fouten optreden die een normaal gebruik van het systeem in de weg staan, dit ter beoordeling van de stuurgroep, dan is het project team verantwoordelijk voor het opsporen, analyseren en oplossen daarvan. Deze fouten worden kosteloos binnen een afgesproken termijn hersteld. Er wordt geen garantie gegeven na acceptatie door de klant.

8.3 Onderhoud

Na acceptatie moet de klant zelf in staat zijn om de broncode te onderhouden. Daarom bestaat de overdracht van het product onder andere uit het Software Design Document, de broncode, interface documentatie en de gebruikte testcode. Om ervoor te zorgen dat de klant in staat is om de broncode te onderhouden zal de broncode niet te complex zijn en voorzien van duidelijk commentaar.

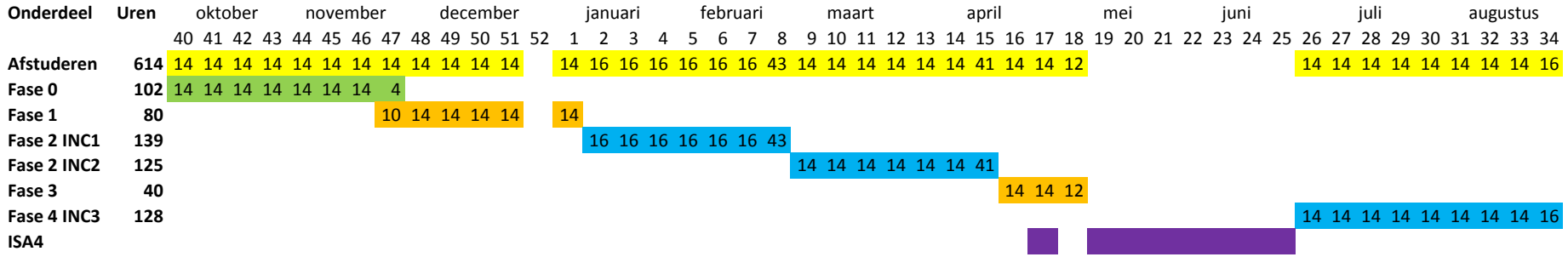
9. Planning

Op de volgende bladzijde volgt de detail planning van het project, ook wel Project Rapportage of Work Breakdown Structure genoemd. De Project Rapportage is een apart document (zie [WBS]). Ter verduidelijking van de planning is eenmalig een kopie van de detail planning in dit Project Management Plan opgenomen. Bij wijzigingen in de planning wordt alleen de Project Rapportage bijgewerkt. Met uitzondering van wijzigingen die impact hebben op de einddatum van de opdracht, dan wordt zowel de Project Rapportage als het Project Management Plan bijgewerkt. Let op: Het project bestaat uit vijf fases en drie increments. Echter de onderstaande detail planning begint bij: Fase 1: Onderzoek. De eerste fase, Fase 0: Project definitie, is niet in de detail planning opgenomen.

Open punt: Er was 128 uur gepland voor de drie increments (zie eerdere versie van dit document). Na het maken van de detail planning komt increment 1 uit op 139 uur en increment 2 op 125 uur. Increment 1 en 2 kosten samen dus 8 uur meer tijd dan initieel ingepland. Deze afwijking is erg klein en geaccepteerd. Echter increment 3 komt uit op 184 uur, 56 uur meer dan beschikbaar. Samen met Peter Vink zal er naar een oplossing gezocht worden om de benodigde tijd te beperken. Zeer waarschijnlijk zal dit betekenen dat er bepaalde requirements niet geïmplementeerd kunnen worden.

Fase	INC	Activiteiten	Uren		
			Baseline	Gedaan	Te doen
		Onderzoek	**80**	**0**	**0**
1	0	Onderzoeksrapport	80	0	0
		Design en implementatie 1 - Increment 1	**139**	**0**	**0**
		Design en implementatie 1 - Increment 2	**125**	**0**	**0**
2	1	Product Risico Analyse	4	0	0
2	1	Hardware: Opzetten compiler / toolchain	6	0	0
2	1	Hardware: Uitproberen debug mogelijkheden	2	0	0
2	1	Hardware: Uitproberen SW stack voor draadloze communicatie	6	0	0
2	1	Hardware: Uitlezen waarde licht sensor	4	0	0
2	1	Lichtsensor: Software Design Document (SDS)	40	0	0
2	1	Lichtsensor: Implementatie groep 1	40	0	0
2	1	Lichtsensor: Unit verificatie groep 1	13	0	0
2	1	Simpele applicatie om meetwaarden te tonen	8	0	0
2	1	INC1: Demonstratie	4	0	0
2	1	INC1: Afstudeerverslag	12	0	0
2	2	Hardware: Uitlezen waarde temperatuur sensor	4	0	0
2	2	Temp. sensor: Update SDS	8	0	0
2	2	Temp. sensor: Implementatie groep 2	16	0	0
2	2	Temp. sensor: Unit verificatie groep 2	5	0	0
2	2	Heartbeat (SA5): Update SDS	4	0	0
2	2	Heartbeat (SA5): Implementatie groep 3	4	0	0
2	2	Heartbeat (SA5): Unit verificatie groep 3	1	0	0
2	2	Meetwaarden GUI: Software Design Document	20	0	0
2	2	Meetwaarden GUI: Implementatie groep 4	20	0	0
2	2	Meetwaarden GUI: Unit verificatie groep 4	7	0	0
2	2	Simpele applicatie: X10 aansturing	4	0	0
2	2	Simpele applicatie: Hard-coded dynamisch gedrag	4	0	0
2	2	INC2: Systeem Test Specificatie	7	0	0
2	2	INC2: Uitvoering Systeem Test	4	0	0
2	2	INC2: Systeem Test Rapportage	1	0	0
2	2	INC2: Demonstratie	4	0	0
2	2	INC2: Afstudeerverslag	12	0	0
		Onderzoek	**40**	**0**	**0**
3	0	Onderzoeksrapport	40	0	0
		Design en implementatie 2	**184**	**0**	**0**
4	3	Regels: Software Design Document (SDS)	40	0	0
4	3	Regels: Grafisch vastleggen, wijzigen en verwijderen (AR1, AR2, AR3, AR9, AR10, AR13 en AR14)	40	0	0
4	3	Regels: Persistente opslag per gebruiker (AR3, AR4 en AR9)	4	0	0
4	3	Regels: Inlezen en uitvoeren van de regels (AR2, AR3 en AR13)	16	0	0
4	3	Regels: Integratie locatiesensor (AR5)	8	0	0
4	3	Regels: Aansturen X10 apparatuur (AR6)	8	0	0
4	3	Regels: Logging (AR11)	4	0	0
4	3	Regels: Unit verificatie	27	0	0
4	3	Frequentie metingen: Update SDS	4	0	0
4	3	Frequentie metingen: Implementatie groep 6	4	0	0
4	3	Frequentie metingen: Unit verificatie groep 6	1	0	0
4	3	INC3: Systeem Test Specificatie	7	0	0
4	3	INC3: Uitvoering Systeem Test	4	0	0
4	3	INC3: Systeem Test Rapportage	1	0	0
4	3	INC3: Demonstratie	4	0	0
4	3	INC3: Afstudeerverslag	12	0	0
		Ingeplande uren	568	0	0
		Beschikbare uren	504	0	0

Vanwege het feit dat dit een parttime afstudeeropdracht betreft heeft het project een langere doorlooptijd. Daarom volgt hieronder een globale planning van het project. Deze globale planning is gebaseerd op de detail planning uit de Project Rapportage. Het doel van de globale planning is om (grafisch) aan te geven hoelang een fase / increment duurt en om zodoende een indicatie te geven van de start- en einddatum van iedere fase / increment.



Fase	Startdatum	Einddatum
Fase 0	vrijdag 8-10-2010	vrijdag 26-11-2010
Fase 1	vrijdag 26-11-2010	zondag 9-1-2011
Fase 2 INC1	vrijdag 14-1-2011	zondag 27-2-2011
Fase 2 INC2	vrijdag 4-3-2011	zondag 17-4-2011
Fase 3	vrijdag 22-4-2011	zondag 8-5-2011
Fase 4 INC3	vrijdag 1-7-2011	zondag 28-8-2011

Op 26 augustus 2010 heeft Peter Vink vastgesteld dat ik in vergelijking met een voltijd student reeds 80 uur gewerkt heb aan mijn afstudeeropdracht. Daarna is het afstuderen tijdelijk stil komen te liggen. Op vrijdag 8 oktober 2010 (week 40) heb ik het afstuderen opnieuw opgepakt. Dat is ook de begindatum van bovenstaande planning. In totaal besteed ik dus 80 + 614 = 694 uur aan mijn afstuderen.

tass
TSS TECHNOLOGY SOLUTIONS

Draadloze communicatie voor Amigo

Bijlage B: Software Requirements Specificatie

Dit document is gepubliceerd door:
TASS BV
Eindhoven, Nederland

Commentaar en suggesties kunnen worden gestuurd naar:
TASS B.V.
Postbus 80060
5600 KA Eindhoven
Nederland
Tel: +31 40 2503200
Fax: +31 40 2503201

Document Geschiedenis

Versie	Status	Datum	Auteur	Wijzigingen
0.1	Draft	01-08-2010	R. Zuidinga	Eerste versie (alleen hoofdstuk 3).
0.2	Proposed	16-11-2010	R. Zuidinga	Review commentaar verwerkt. Document verder afgemaakt.
0.3	Proposed	22-11-2010	R. Zuidinga	Technische requirements gegroepeerd. Requirement TR4 toegevoegd. Requirement AR13 & AR14 toegevoegd.
0.4	Proposed	22-11-2010	R. Zuidinga	MoSCoW prioriteiten toegevoegd.
0.5	Proposed	28-11-2010	R. Zuidinga	Omschrijving verbeterd van requirements: SA7, AR7 en AR12. Vier nieuwe requirements toegevoegd: SL6, AM4, AM5 en TR5.
1.0	Approved	17-12-2010	R. Zuidinga	Geen inhoudelijke veranderingen. Goedgekeurd door Peter Vink.

Open Punten

Geen.

Inhoudsopgave

1. Introductie

1.1 Doel

Dit document beschrijft:

- De functionele en niet-functionele requirements voor de te ontwikkelen software
- De eventuele relaties en interfaces van de software met andere producten
- Hoe de software gebruikt kan worden.

Dit document moet worden goedgekeurd door de klant.

1.2 Distributielijst

Naam	Organisatie	Functie
Jos Boonen	Fontys	Afstudeerbegeleider
Peter Vink	TASS	Klant / Bedrijfsbegeleider
Tiny Henst	TASS	People manager

1.3 Contactgegevens

Naam	E-mail	Telefoon
Ralph Zuidinga	rzuidinga@eltimo.nl	06 – 5568 3393
Jos Boonen	j.boonen@fontys.nl	0877 – 875 239
Peter Vink	Peter.Vink@tass.nl	040 – 2503209
Tiny Henst	Tiny.Henst@tass.nl	040 – 2503208

1.4 Referenties

Referentie	Document ID	Titel
[PMP]	PMP	Project Management Plan

1.5 Termen en Afkortingen

Term	Omschrijving
TASS	Technical Application Software Solutions
PMP	Project Management Plan
SRS	Software Requirements Specificatie; *dit document*
CE-HTML	Een taal om gebruikersinterface pagina's te maken voor consumentenelektronica apparaten zoals televisies.
MoSCoW	Methode van prioriteiten stellen aan de requirements.

2. Product Beschrijving

2.1 Het Product

Dit nieuwe product betreft een uitbreiding op het bestaande Amigo platform en heeft als belangrijkste doel het gebruiksgemak te verbeteren. Dit wordt bereikt door een software applicatie waarmee het mogelijk is om acties te kunnen koppelen aan gebeurtenissen. En door de introductie van een licht- en temperatuursensor. Met deze twee nieuwe sensoren wordt het aantal 'triggers' voor een gebeurtenis uitgebreid met licht en temperatuur. Zo is het middels deze uitbreiding op het bestaande Amigo platform bijvoorbeeld mogelijk om vast te leggen dat het licht aan moet gaan als de gebruiker een donkere kamer binnenloopt.

Het bestaande Amigo platform kent al een locatiesensor, dus in geval van het bovengenoemde voorbeeld zou het licht altijd aangaan als de gebruiker de ruimte binnenkomt, ongeacht of het in de ruimte licht of donker is.

Het bestaande Amigo platform moet dus worden uitgebreid met een licht- en temperatuursensor. De sensoren moeten meerdere jaren kunnen werken zonder vervanging van de batterijen. Dit ook met het oog op het gebruiksgemak; een gebruiker wil niet iedere maand de batterijen vervangen. We gebruiken daarom een low-power draadloze technologie. Deze technologie is bepaald tijdens een onderzoek voorafgaande aan de eerste design en implementatie fase (zie: Project Management Plan). De hoofdtaak van de sensoren is het uitvoeren van metingen en het versturen van de meetwaarden. De embedded software, die op de sensoren moet draaien, is (uiteraard) niet voorzien van een grafische gebruikersinterface. Paragraaf 3.1 omschrijft alle eisen die aan deze embedded software worden gesteld.

Naast de sensoren, wordt het bestaande Amigo platform ook uitgebreid met een software applicatie, waarmee de gebruiker op een eenvoudige manier zelf acties kan koppelen aan gebeurtenissen. De gebruikersinterface van deze applicatie moet worden geschreven in CE-HTML, zodat deze kan worden getoond op een televisie. Hiermee is het gebruiksgemak ten opzichte van het bestaande Amigo platform verbeterd omdat het nu voor het eerst voor de gebruiker mogelijk is om zelf acties te koppelen aan gebeurtenissen. Voorheen was dit alleen mogelijk op een computer. De eisen, die aan deze software worden gesteld, staan beschreven in paragraaf 3.2.2.

Tot slot wordt er ook nog een tweede software applicatie gemaakt. Het doel van deze applicatie is om de meetwaarden van de licht- en temperatuursensoren weer te geven. Ook de gebruikersinterface van deze applicatie moet worden geschreven in CE-HTML. De gebruiker kan het systeem niet beïnvloeden met deze applicatie. De eisen, die aan deze software worden gesteld, staan beschreven in paragraaf 3.2.1.

3. Specifieke Requirements

Dit hoofdstuk beschrijft de functionaliteit van het software product in detail.

Vanwege het relatief lage aantal requirements is ervoor gekozen om de requirements te groeperen naar onderdeel van het product. Alle requirements met betrekking tot één onderdeel staan bij elkaar in een van de onderstaande paragrafen. In zo'n paragraaf staan zowel de functionele als de niet-functionele requirements.

3.1 Sensoren

3.1.1 Algemeen

Deze algemene sensor requirements zijn van toepassing op zowel de lichtsensor als op de temperatuursensor. Let op: Deze requirements zijn dus niet van toepassing op de bestaande Amigo sensoren (zoals de locatiesensor).

ID:	SA1	**Versie:**	0.1	**Status:**	Approved	**Prioriteit:**	Should
Titel:	Timing requirement aanzetten van een sensor						
Omschrijving: Binnen 30 seconden na het aanzetten van een sensor moet het eerste event verstuurd zijn.							

ID:	SA2	**Versie:**	0.2	**Status:**	Approved	**Prioriteit:**	Could
Titel:	Frequentie metingen instelbaar via web service						
Omschrijving: De sensoren moeten in staat zijn om met twee verschillende frequenties (langzaam / snel) metingen te doen. Het moet met behulp van een web service mogelijk zijn om de gewenste frequentie te kiezen. Toelichting: De mogelijkheid om een langzame frequentie te kiezen is opgenomen om de levensduur van de batterij te verlengen.							

ID:	SA3	**Versie:**	0.2	**Status:**	Approved	**Prioriteit:**	Could
Titel:	Standaard waarde van de frequentie metingen						
Omschrijving: Na het aanzetten van de sensor gaat deze standaard op frequentie snel metingen doen.							

ID:	SA4	**Versie:**	0.2	**Status:**	Approved	**Prioriteit:**	Could
Titel:	Tijdsintervallen behorende bij de frequenties van meten						
Omschrijving: Bij frequentie langzaam moet er één keer per dertig minuten een meting worden gedaan, en bij frequentie snel moet er één keer per vijf minuten een meting worden gedaan.							

ID:	SA5	Versie:	0.1	Status:	Approved	Prioriteit:	Should
Titel:	Heartbeat event						
Omschrijving: De sensor moet één keer per uur een zogenaamd heartbeat event versturen. Toelichting: Deze eis is toegevoegd om het mogelijk te maken dat de rest van het systeem kan opmerken wanneer de batterij van een sensor leeg is.							

ID:	SA6	Versie:	0.1	Status:	Approved	Prioriteit:	Must
Titel:	Demonstratie timers						
Omschrijving: Er moet een speciale versie van de software worden opgeleverd, waarin alle timers een kleinere waarde hebben, zodat de software beter geschikt is voor het geven van demonstraties.							

ID:	SA7	Versie:	0.5	Status:	Approved	Prioriteit:	Must
Titel:	Levensduur batterij						
Omschrijving: Een sensor moet minimaal 1 jaar onafgebroken kunnen werken zonder dat de batterij vervangen hoeft te worden. De software moet er dus voor zorgen dat het stroomverbruik van de sensoren continu zo laag mogelijk is. Toelichting: Dit requirement kan geïmplementeerd worden door in de tijd dat een sensor niets hoeft te doen de hardware in een speciale 'power saving' mode te zetten.							

3.1.2 Lichtsensor

ID:	SL1	Versie:	0.1	Status:	Approved	Prioriteit:	Must
Titel:	Definitie licht event						
Omschrijving: Een lichtsensor moet licht events kunnen versturen. In een licht event staat of het licht of donker is.							

ID:	SL2	Versie:	0.1	Status:	Approved	Prioriteit:	Should
Titel:	Timing requirement aanzetten van een lichtsensor						
Omschrijving: Bij het aanzetten van de lichtsensor moet zo snel mogelijk een lichtmeting worden gedaan. De gemeten waarde moet worden vergeleken met een vooraf ingestelde grenswaarde om te bepalen of het licht of donker is. Meteen daarna moet er een licht event worden verstuurd.							

ID:	SL3	Versie:	0.1	Status:	Approved	Prioriteit:	Must
Titel:	Grenswaarde om te bepalen of het licht of donker is						

Omschrijving:
De lichtsensor bevat een standaard grenswaarde. Deze standaard grenswaarde wordt tijdens de implementatie gekozen en tijdens een demonstratie ter goedkeuring aan de opdrachtgever voorgelegd.

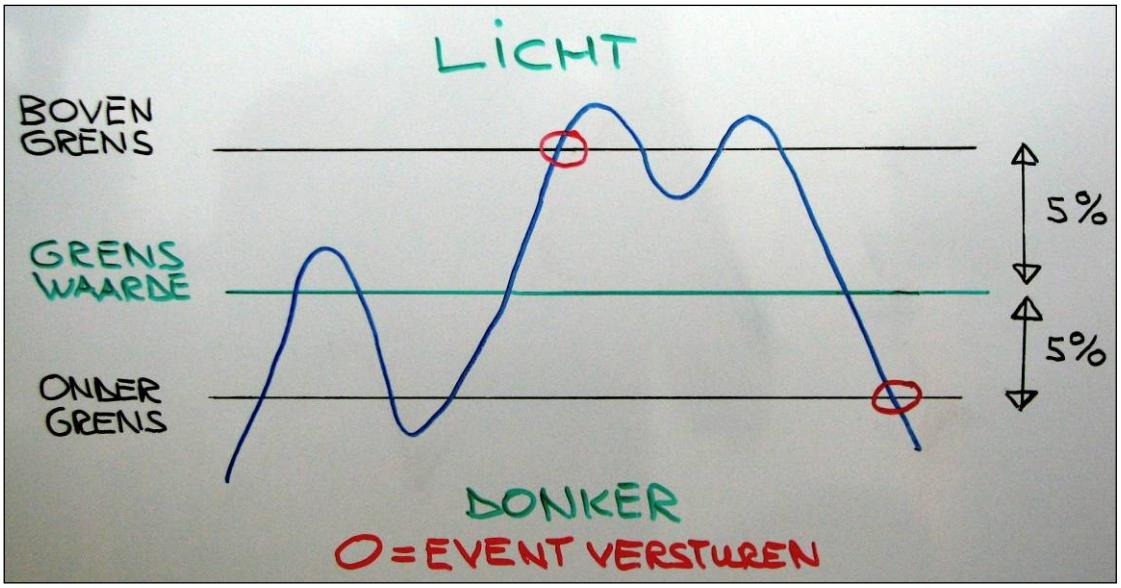
ID:	SL4	Versie:	0.1	Status:	Approved	Prioriteit:	Could
Titel:	Grenswaarde instelbaar via web service						

Omschrijving:
Het moet met behulp van een web service mogelijk zijn om een nieuwe waarde voor de grenswaarde op te geven. Deze nieuwe grenswaarde wordt vanaf dat moment gebruikt, maar wordt niet opgeslagen. Als de lichtsensor zijn spanning verliest, dan is de aangepaste grenswaarde verloren.

ID:	SL5	Versie:	0.2	Status:	Approved	Prioriteit:	Must
Titel:	Algoritme om te bepalen wanneer een licht event te versturen						

Omschrijving:
Een lichtsensor mag alleen een licht event versturen wanneer dat noodzakelijk is volgens onderstaand algoritme. Toelichting: Ten eerste moet het algoritme ervoor zorgen dat er zo min mogelijk licht events verstuurd worden, dit om de levensduur van de batterij te verlengen.
Ten tweede moet het algoritme voorkomen dat er onnodig veel events worden verstuurd in de situatie dat het gemeten lichtniveau schommelt rond de grenswaarde.

Algoritme:
Na de eerste lichtmeting wordt de waarde van alle volgende lichtmetingen vergeleken met een bovengrens en een ondergrens. De bovengrens is de grenswaarde + vijf procent en de ondergrens is de grenswaarde – vijf procent. Zodra de gemeten waarde boven de bovengrens uitkomt (dus het wordt licht) terwijl in het laatst verstuurde licht event staat dat het donker is, dan moet er een licht event worden verstuurd om aan te geven dat het licht is geworden. Het omgekeerde is ook vereist. Zodra de gemeten waarde onder de ondergrens uitkomt (dus het wordt donker) terwijl in het laatst verstuurde licht event staat dat het licht is, dan moet er een licht event worden verstuurd om aan te geven dat het donker is geworden. Zie onderstaande afbeelding ter verduidelijking. Alleen op de rood omcirkelde momenten moet er een licht event worden verstuurd.



ID:	SL6	Versie:	0.5	Status:	Approved	Prioriteit:	Could
Titel:	Huidige waarde lichtsensor opvraagbaar via web service						
Omschrijving: Het moet met behulp van een web service mogelijk zijn om de huidige waarde van de lichtsensor op te vragen (licht of donker). Met andere woorden, als de web service wordt aangeroepen, moet er een (extra) licht event worden verstuurd.							

3.1.3 Temperatuursensor

ID:	ST1	Versie:	0.1	Status:	Approved	Prioriteit:	Must
Titel:	Definitie temperatuur event						
Omschrijving: Een temperatuursensor moet temperatuur events kunnen versturen. In een temperatuur event staat de huidige temperatuur in graden Celsius met een nauwkeurigheid van 1/10de graad.							

ID:	ST2	Versie:	0.1	Status:	Approved	Prioriteit:	Should
Titel:	Timing requirement aanzetten van een temperatuursensor						
Omschrijving: Bij het aanzetten van de temperatuursensor moet zo snel mogelijk een temperatuurmeting worden gedaan. Meteen daarna moet er een temperatuur event worden verstuurd.							

ID:	ST3	Versie:	0.1	Status:	Approved	Prioriteit:	Should
Titel:	Huidige temperatuur opvraagbaar via web service						
Omschrijving: Het moet met behulp van een web service mogelijk zijn om de huidige temperatuur op te vragen. Met andere woorden, als de web service wordt aangeroepen, moet er een (extra) temperatuur event worden verstuurd.							

ID:	ST4	Versie:	0.2	Status:	Approved	Prioriteit:	Must
Titel:	Algoritme om te bepalen wanneer een temperatuur event te versturen						

Omschrijving:

Een temperatuursensor mag alleen een temperatuur event versturen wanneer dat noodzakelijk is volgens onderstaand algoritme. Toelichting: Ten eerste moet het algoritme ervoor zorgen dat er zo min mogelijk temperatuur events verstuurd worden, dit om de levensduur van de batterij te verlengen. Ten tweede moet het algoritme voorkomen dat er onnodig veel events worden verstuurd omdat de gemeten temperatuur (meestal) altijd anders is.

Algoritme:

De temperatuursensor moet van de laatste tien metingen de gemeten waarde onthouden. Daarnaast moet bij iedere temperatuurmeting het gemiddelde van de laatste tien metingen berekend en onthouden worden. Indien er minder dan tien metingen beschikbaar zijn dan moet het gemiddelde berekend worden op basis van alle reeds gemaakte metingen.

Zodra het verschil tussen de laatst berekende gemiddelde temperatuur (hierna aangeduid als: [nieuw gemiddelde]) en de gemiddelde temperatuur behorende bij de meting die gebruikt is voor het vorige verstuurde event (hierna aangeduid als: [verstuurd gemiddelde]) groter of gelijk is aan 1 graden Celsius dan moet er een nieuw temperatuur event worden verstuurd. Let op: Een temperatuur event bevat de huidige temperatuur, maar om te bepalen of er een nieuw event moet worden verstuurd, worden twee gemiddelde temperaturen met elkaar vergeleken.

Meting nummer	Gemeten waarde	Gemiddelde waarde	Temperatuur Event
1	22,0	22,00	Ja
2	22,5	22,25	Nee
3	22,7	22,40	Nee
4	22,6	22,45	Nee
5	22,9	22,54	Nee
6	23,3	22,67	Nee
7	23,1	22,73	Nee
8	23,3	22,80	Nee
9	23,5	22,88	Nee
10	24,1	23,00	Ja
11	24,5	23,25	Nee
12	24,7	23,47	Nee

Meting nummer	Gemeten waarde	Gemiddelde waarde	Temperatuur Event
13	24,9	23,56	Nee
14	25,1	23,68	Nee
15	24,8	23,79	Ja
16	25,0	23,87	Nee
17	25,3	24,02	Nee
18	25,1	24,20	Nee
19	25,4	24,43	Nee
20	25,2	24,60	Nee
21	25,5	24,70	Nee
22	25,9	25,22	Ja
23	25,6	25,29	Nee
24	26,1	25,39	Nee

Voorbeeld: Na meting nummer 10 is er een event verstuurd met daarin de huidige temperatuur. Het event bevat dus de waarde 24,1. Vanaf nu is [verstuurd gemiddelde] gelijk aan 23 graden Celsius. Na meting nummer 11 wordt gecontroleerd of het verschil tussen [nieuw gemiddelde] en [verstuurd gemiddelde] groter of gelijk is aan 1 graden Celsius. Er wordt geen temperatuur event verstuurd, want het verschil is kleiner dan 1 graden Celsius (23,25 – 23 = 0,25). Na meting nummer 15 wordt er wel een nieuw temperatuur event verstuurd, want het verschil is nu wel groter dan 1 graden Celsius (24,13 – 23 = 1,13).

3.2 Applicatie

3.2.1 Applicatie 1: Meetwaarden

ID:	AM1	Versie:	0.2	Status:	Approved	Prioriteit:	Must
Titel:	Plattegrond met locatie en waarde van de sensoren						
Omschrijving: De gebruikersinterface van de applicatie moet op een plattegrond de volgende items tonen: • De locatie van de aanwezige licht- en temperatuursensoren, • De laatste gerapporteerde waarde van de sensoren.							

ID:	AM4	Versie:	0.5	Status:	Approved	Prioriteit:	Could
Titel:	Locatie personen tonen op plattegrond						
Omschrijving: De gebruikersinterface van de applicatie moet op een plattegrond de volgende items tonen: • De locatie van de aanwezige personen.							

ID:	AM2	Versie:	0.2	Status:	Approved	Prioriteit:	Should
Titel:	Sensor batterij leeg indicatie op plattegrond						
Omschrijving: Op het moment dat de applicatie van een werkende licht- en/of temperatuursensor voor de tweede keer geen heartbeat event heeft ontvangen, dan moet de applicatie aan de gebruiker kenbaar maken van welke sensor de batterij leeg is. Hierbij moet de gebruikersinterface aangeven om wat voor een soort sensor het gaat en waar deze zich bevindt. De laatste gerapporteerde waarde (zie: requirement AM1) mag nu niet meer getoond worden.							

ID:	AM3	Versie:	0.2	Status:	Approved	Prioriteit:	Must
Titel:	Input events waar meetwaarden applicatie naar moet luisteren (1 van 2)						
Omschrijving: De applicatie moet overweg kunnen met de volgende drie soorten events: • Licht (nieuw), • Temperatuur (nieuw), • Heartbeat (nieuw).							

ID:	AM5	Versie:	0.5	Status:	Approved	Prioriteit:	Could
Titel:	Input events waar meetwaarden applicatie naar moet luisteren (2 van 2)						
Omschrijving: De applicatie moet overweg kunnen met de volgende soorten events: • Locatie (bestaand).							

3.2.2 Applicatie 2: Regels

ID:	AR1	Versie:	0.2	Status:	Approved	Prioriteit:	Must
Titel:	Regels vastleggen, wijzigen en verwijderen						
Omschrijving: De applicatie moet het voor de gebruiker mogelijk maken om de volgende acties uit te voeren: • Regels vastleggen, • Vastgelegde regels inzien, • Vastgelegde regels wijzigen, • Vastgelegde regels verwijderen. Toelichting: Door middel van een onderzoek wordt bepaald welke syntax / semantiek het beste geschikt is voor het vastleggen van regels.							

ID:	AR2	Versie:	0.1	Status:	Approved	Prioriteit:	Should
Titel:	Tegelijkertijd regels uitvoeren en nieuwe accepteren						
Omschrijving: De applicatie moet automatisch de vastgelegde regels van alle gebruikers uitvoeren. Tegelijkertijd moet de gebruiker de in requirement AR1 genoemde acties kunnen blijven uitvoeren.							

ID:	AR3	Versie:	0.1	Status:	Approved	Prioriteit:	Should
Titel:	Regels moeten persoonsgebonden zijn						
Omschrijving: De regels moeten persoonsgebonden zijn. Het moet voor persoon *X* niet mogelijk zijn om de regels van persoon *Y* te zien.							

ID:	AR4	Versie:	0.1	Status:	Approved	Prioriteit:	Must
Titel:	Regels moeten persistent worden opgeslagen						
Omschrijving: De vastgelegde regels moeten worden opgeslagen, zodat deze niet verloren gaan als de applicatie wordt afgesloten.							

ID:	AR5	Versie:	0.2	Status:	Approved	Prioriteit:	Must
Titel:	Input events waar regels applicatie naar moet luisteren						
Omschrijving: De applicatie moet overweg kunnen met de volgende vier soorten events: • Licht (nieuw), • Temperatuur (nieuw), • Heartbeat (nieuw), • Locatie (bestaand).							

ID:	AR6	Versie:	0.1	Status:	Approved	Prioriteit:	Must
Titel:	Apparatuur die de regels applicatie moet kunnen aansturen						
Omschrijving: De applicatie moet de volgende apparatuur kunnen aansturen / bedienen: • Lamp aan / uit, • Gordijn open / dicht, • Ventilator aan / uit, • Kachel aan / uit.							

ID:	AR7	Versie:	0.5	Status:	Approved	Prioriteit:	Could
Titel:	Applicatie moet frequentie metingen sensoren instellen						
Omschrijving: Zodra de applicatie een locatie event ontvangt waarin staat dat er iemand een ruimte binnenkomt waar een sensor aanwezig is, dan moet de applicatie ervoor zorgen dat alle sensoren in de betreffende ruimte hun metingen gaan doen op de hoogst mogelijke frequentie. Het omgekeerde is ook vereist. Zodra er vijf minuten lang niemand meer in een ruimte met een sensor aanwezig is, dan moet de applicatie ervoor zorgen dat alle sensoren in de betreffende ruimte hun metingen gaan doen op de laagst mogelijke frequentie. Let op: Dit requirement is alleen van toepassing op licht- en temperatuursensoren.							

Requirement AR8 is vervangen door requirement AM2.

ID:	AR9	Versie:	0.1	Status:	Approved	Prioriteit:	Must
Titel:	Aantal regels per gebruiker						
Omschrijving: De applicatie moet per gebruiker maximaal 25 regels kunnen vastleggen.							

ID:	AR10	Versie:	0.1	Status:	Approved	Prioriteit:	Should
Titel:	Timing requirement aanzetten regels applicatie						
Omschrijving: Het opstarten van de applicatie mag maximaal 30 seconden duren. Na deze opstarttijd: • Moet de gebruiker de in requirement AR1 genoemde acties kunnen uitvoeren, • Moet de applicatie gereed zijn om de reeds vastgelegde regels uit te voeren.							

ID:	AR11	Versie:	0.2	Status:	Approved	Prioriteit:	Should
Titel:	Logging regels applicatie						
Omschrijving: De applicatie moet in een tekst bestand de volgende logging bijhouden: • Welke regels vastgelegd worden (regel + gebruiker), • Welke events ontvangen worden (event + sensor), • Welke actie gedaan wordt als gevolg van welke regel. Iedere regel moet beginnen met de huidige datum en tijd.							

ID:	AR12	Versie:	0.5	Status:	Approved	Prioriteit:	Wont
Titel:	Conflicterende regels						
Omschrijving: De applicatie moet bestand zijn tegen conflicterende regels.							

ID:	AR13	Versie:	0.3	Status:	Approved	Prioriteit:	Should
Titel:	Regel altijd of alleen indien aanwezig uitvoeren						
Omschrijving: Bij het vastleggen van een nieuwe regel en bij het wijzigen van een bestaande regel moet je kunnen opgeven of de regel alleen uitgevoerd moet worden als je zelf in de betreffende ruimte aanwezig bent of dat de regel altijd uitgevoerd moet worden ongeacht of er iemand in de ruimte aanwezig is.							

ID:	AR14	Versie:	0.3	Status:	Approved	Prioriteit:	Should
Titel:	Standaard waarde van de wanneer regel uitvoeren optie						
Omschrijving: Standaard moeten regels alleen uitgevoerd worden als de persoon die de regel heeft vastgelegd in de betreffende ruimte aanwezig is.							

3.3 Technische Requirements

ID:	TR1	Versie:	0.2	Status:	Approved	Prioriteit:	Must
Titel:	Software sensoren in C of C++						
Omschrijving: De embedded software voor de sensoren moet ontwikkeld worden in C of C++.							

ID:	TR2	Versie:	0.2	Status:	Approved	Prioriteit:	Must
Titel:	Software applicaties in C# of Java						
Omschrijving: De software voor de applicaties moet ontwikkeld worden in C# of Java.							

ID:	TR3	Versie:	0.2	Status:	Approved	Prioriteit:	Should
Titel:	Gebruikersinterface applicaties in CE-HTML						
Omschrijving:							
De gebruikersinterface van de applicaties moet worden geschreven in CE-HTML.							

ID:	TR4	Versie:	0.3	Status:	Approved	Prioriteit:	Must
Titel:	Compatibiliteit met bestaande Amigo platform						

Omschrijving:

Alle nieuw te ontwikkelen software moet gemaakt worden volgens onderstaande standaarden om compatibiliteit met het huidige Amigo platform te garanderen.

Standaarden:

- Simple Object Access Protocol (SOAP)
  http://www.w3.org/TR/soap/
- Web Services Description Language (WSDL)
  http://www.w3.org/TR/wsdl of: http://www.w3.org/TR/wsdl20/
- Web Services Dynamic Discovery (WS-Discovery)
  http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.html
- Web Services Eventing (WS-Eventing)
  http://www.w3.org/TR/ws-eventing/

Toelichting: Waar het kan moet de software werken volgens de relevante delen van bovenstaande standaarden. Maar bovenstaande standaarden hoeven niet volledig geïmplementeerd te worden in de te ontwikkelen software.

ID:	TR5	Versie:	0.5	Status:	Approved	Prioriteit:	Must
Titel:	Betrouwbaarheid versturen events						

Omschrijving:

De software voor de licht- en temperatuursensor moet ervoor zorgen dat gemiddeld in 90% van de situaties dat er een event verstuurd moet worden er ook daadwerkelijk een event ontvangen wordt door de meetwaarden of regels applicatie. Toelichting: Het UDP protocol biedt bijvoorbeeld geen garantie dat de verstuurde gegevens daadwerkelijk aankomen.
Let op: Dit requirement is alleen van toepassing op licht- en temperatuursensoren.

4. Overzicht Requirements

Hieronder volgt een overzicht van het ID, titel en prioriteit van alle requirements.

ID	Titel	Prioriteit
SA1	Timing requirement aanzetten van een sensor	SHOULD
SA2	Frequentie metingen instelbaar via web service	COULD
SA3	Standaard waarde van de frequentie metingen	COULD
SA4	Tijdsintervallen behorende bij de frequenties van meten	COULD
SA5	Heartbeat event	SHOULD
SA6	Demonstratie timers	MUST
SA7	Levensduur batterij	MUST
SL1	Definitie licht event	MUST
SL2	Timing requirement aanzetten van een lichtsensor	SHOULD
SL3	Grenswaarde om te bepalen of het licht of donker is	MUST
SL4	Grenswaarde instelbaar via web service	COULD
SL5	Algoritme om te bepalen wanneer een licht event te versturen	MUST
SL6	Huidige waarde lichtsensor opvraagbaar via web service	COULD
ST1	Definitie temperatuur event	MUST
ST2	Timing requirement aanzetten van een temperatuursensor	SHOULD
ST3	Huidige temperatuur opvraagbaar via web service	SHOULD
ST4	Algoritme om te bepalen wanneer een temperatuur event te versturen	MUST
AM1	Plattegrond met locatie en waarde van de sensoren	MUST
AM2	Sensor batterij leeg indicatie op plattegrond	SHOULD
AM3	Input events waar meetwaarden applicatie naar moet luisteren (1 van 2)	MUST
AM4	Locatie personen tonen op plattegrond	COULD
AM5	Input events waar meetwaarden applicatie naar moet luisteren (2 van 2)	COULD
AR1	Regels vastleggen, wijzigen en verwijderen	MUST
AR2	Tegelijkertijd regels uitvoeren en nieuwe accepteren	SHOULD
AR3	Regels moeten persoonsgebonden zijn	SHOULD
AR4	Regels moeten persistent worden opgeslagen	MUST
AR5	Input events waar regels applicatie naar moet luisteren	MUST
AR6	Apparatuur die de regels applicatie moet kunnen aansturen	MUST
AR7	Applicatie moet frequentie metingen sensoren instellen	COULD
AR9	Aantal regels per gebruiker	MUST
AR10	Timing requirement aanzetten regels applicatie	SHOULD
AR11	Logging regels applicatie	SHOULD
AR12	Conflicterende regels	WONT
AR13	Regel altijd of alleen indien aanwezig uitvoeren	SHOULD
AR14	Standaard waarde van de wanneer regel uitvoeren optie	SHOULD
TR1	Software sensoren in C of C++	MUST
TR2	Software applicaties in C# of Java	MUST
TR3	Gebruikersinterface applicaties in CE-HTML	SHOULD
TR4	Compatibiliteit met bestaande Amigo platform	MUST
TR5	Betrouwbaarheid versturen events	MUST

Draadloze communicatie voor Amigo

Bijlage C: Onderzoeksrapport

Dit document is gepubliceerd door:
TASS BV
Eindhoven, Nederland

Commentaar en suggesties kunnen worden gestuurd naar:
TASS B.V.
Postbus 80060
5600 KA Eindhoven
Nederland
Tel: +31 40 2503200
Fax: +31 40 2503201

Document Geschiedenis

Versie	Status	Datum	Auteur	Wijzigingen
0.1	Draft	10-05-2011	R. Zuidinga	Eerste versie.
0.2	Proposed	13-05-2011	R. Zuidinga	Tweede versie.

Open Punten

Geen.

Inhoudsopgave

1. Introductie

1.1 Doel

De onderzoeksvraag luidt: “Welke draadloze technologie is het beste geschikt voor uitbreiding van het Amigo platform?”. De criteria waar de technologie aan moet voldoen om in aanmerking te komen zijn:

1. De sensoren, op basis van de geselecteerde technologie, moeten meerdere jaren kunnen werken zonder vervanging van de batterijen.
2. De technologie moet compatibel zijn met het bestaande Amigo platform.

Er bestaan verschillende draadloze technologieën zoals: Wi-Fi, Bluetooth, Z-Wave, ZigBee, 6LoWPAN, Wavenis, etc. Tijdens het onderzoek heb ik twee draadloze technologieën met elkaar vergeleken, namelijk 6LoWPAN en ZigBee. Beide technologieën voldoen aan het eerste criteria. Al snel was duidelijk dat 6LoWPAN het beste geschikt was voor deze opdracht omdat 6LoWPAN net als Amigo op Internet protocollen gebaseerd is. 6LoWPAN voldoet hiermee ook aan het tweede criteria, terwijl bij ZigBee nog een extra vertaalslag nodig is.

Het onderzoek bestaat uit de volgende deelonderzoeken:

1. Verdiepen in de werking van 6LoWPAN.
2. Kennis en ervaring opdoen met de werking van IPv6.
3. Met welke hardware is het mogelijk om via 6LoWPAN te communiceren?
4. Welk besturingssysteem heeft 6LoWPAN ondersteuning?
5. Proof-of-concept.

1.2 Distributielijst

Naam	Organisatie	Functie
Jos Boonen	Fontys	Afstudeerbegeleider
Peter Vink	TASS	Klant / Bedrijfsbegeleider
Tiny Henst	TASS	People manager

1.3 Contactgegevens

Naam	E-mail	Telefoon
Ralph Zuidinga	rzuidinga@eltimo.nl	06 – 5568 3393
Jos Boonen	j.boonen@fontys.nl	0877 – 875 239
Peter Vink	Peter.Vink@tass.nl	040 – 2503209
Tiny Henst	Tiny.Henst@tass.nl	040 – 2503208

1.4 Referenties

Geen.

1.5 Termen en Afkortingen

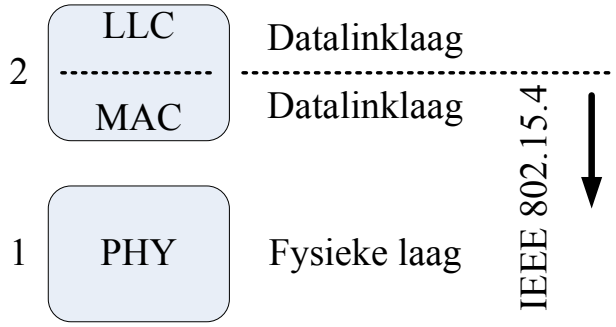
Term	Omschrijving
TASS	Technical Application Software Solutions
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
DHCP	Dynamic Host Configuration Protocol
EUI	Extended Unique Identifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IPv4	Internet Protocol versie 4
IPv6	Internet Protocol versie 6
ISP	Internet Service Provider
LAN	Local Area Network
LLC	Logical Link Control layer
MAC	Media Access Control layer
MAC adres	Een uniek identificatienummer dat aan een apparaat in een netwerk is toegekend (ook wel hardware adres genoemd).
MCU	Microcontroller
OSI	ISO reference model for Open Systems Interconnection
PCB	Printed Circuit Board
PHY	Physical layer
ROLL	Routing Over Low power and Lossy networks
RPL	IPv6 Routing Protocol for Low power and Lossy Networks
WPAN	Wireless Personal Area Network

2. IEEE 802.15.4

Zowel 6LoWPAN als ZigBee communiceren over de IEEE 802.15.4 standaard. In dit hoofdstuk wordt eerst het ontstaan van de IEEE 802.15.4 standaard uitgelegd en vervolgens worden de plaats van de standaard in het OSI-model en de belangrijkste eigenschappen uitgelegd.

Het Institute of Electrical and Electronics Engineers (IEEE) is een organisatie die zich onder andere bezighoudt met de formulering van LAN normen. Binnen de IEEE organisatie is in het verleden de 802.15 werkgroep opgericht. Deze werkgroep heeft als taak om onderzoek te doen naar een oplossing voor draadloze communicatie geschikt voor eenvoudige hardware (lage complexiteit) en waarmee een lange batterij levensduur haalbaar is. Dit impliceert een lage transmissiesnelheid. Hieruit is de IEEE 802.15.4 standaard ontstaan. De eerste versie van deze standaard is uitgekomen in 2003, met later in juni 2006 een revisie.

De naam van de standaard is voluit: “Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)”. De naam zegt het al, de IEEE 802.15.4 standaard beschrijft de fysieke laag en (een gedeelte van) de datalinklaag van het OSI model. De datalinklaag bestaat uit twee sublagen, namelijk: de “Media Access Control layer (MAC)” en de “Logical Link Control layer (LLC)”. Zie onderstaande afbeelding ter verduidelijking.



Kenmerken van IEEE 802.15.4 zijn:

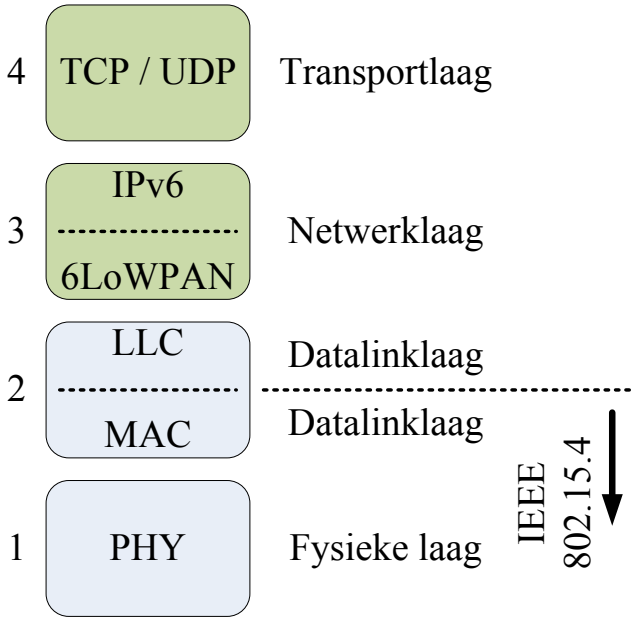
- Kleine pakketgrootte. Maximale grootte van een pakket in de fysieke laag is 127 bytes. Dit resulteert in een maximale grootte van een frame in de MAC laag van 102 bytes. Als er in de LLC laag encryptie wordt aangezet, dan 'kost' dit nog eens minimaal 9 tot maximaal 21 bytes afhankelijk van de gekozen encryptie methode. In de slechtste situatie blijven er dus maar 81 bytes over voor datapaketten uit de netwerklaag.
- Lage bandbreedte. Maximale overdrachtsnelheid van 250 kbps op 2.4 GHz, 40 kbps op 915 MHz en 20 kbps op 868 MHz. De 2.4 GHz frequentieband is dezelfde frequentie als wat ook gebruikt wordt door Wi-Fi. Op deze frequentie is er dus (grote) kans op interferentie. De frequentiebanden mogen niet overal ter wereld gebruikt worden. Onderstaande tabel geeft aan waar welke frequentie gebruikt mag worden.

Frequentie	Locatie
2.4 GHz	Amerika en Europa
915 MHz	Alleen in Amerika
868 MHz	Alleen in Europa

- Ondersteuning voor zowel 16-bit 'short' als voor 64-bit 'extended' MAC adressen. Vanuit de PC wereld zijn we 48-bit MAC adressen gewend ook wel MAC-48 of EUI-48 genoemd. De IEEE 802.15.4 standaard maakt echter gebruik van 16-bit MAC adressen en van 64-bit MAC adressen ook wel EUI-64 genoemd.

3. 6LoWPAN

6LoWPAN staat voor: “IPv6 over Low-Power Wireless Personal Area Networks”. De 6LoWPAN standaard is opgesteld door de gelijknamige werkgroep van de Internet Engineering Task Force (IETF). De werkgroep is in 2004 opgericht met als doel om het mogelijk te maken via IPv6 te communiceren over de IEEE 802.15.4 standaard. Dit wordt gerealiseerd door een zogenaamde “adaptation layer”, een laag die IPv6 compatibel maakt met IEEE 802.15.4. Zie onderstaande afbeelding ter verduidelijking.
De 6LoWPAN standaard is in september 2007 bekrachtigd (RFC4944).



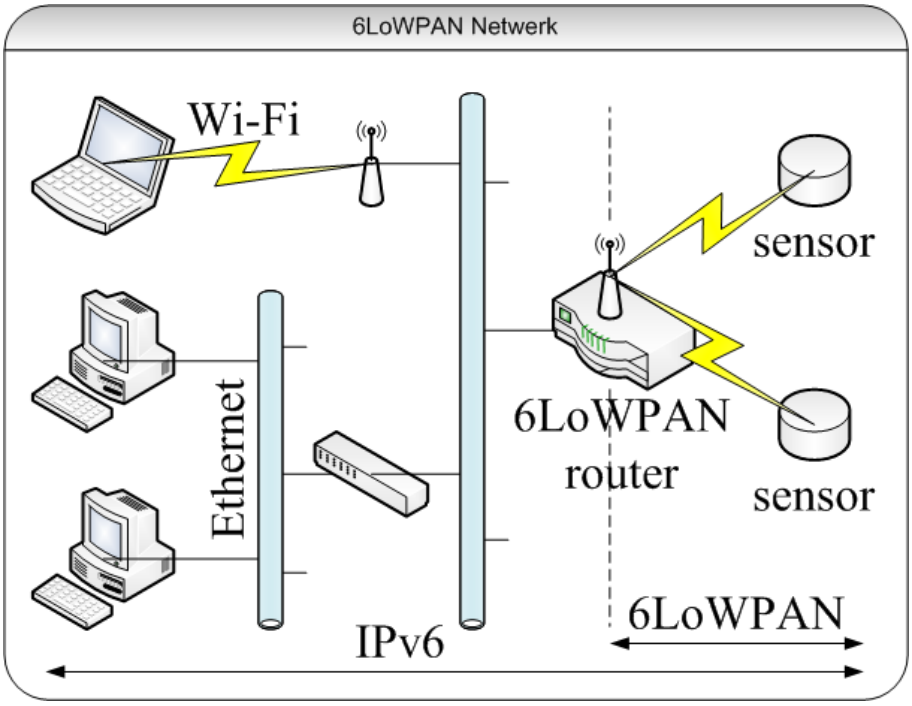
De twee belangrijkste taken van de 6LoWPAN adaptation layer zijn:

- Compressie van de header. Zoals in hoofdstuk 2 besproken blijft er, zonder encryptie, maar 102 (van de 127) bytes over voor datapakketten uit de netwerklaag. Een IPv6 header is 40 bytes lang, dat wil zeggen dat er dan nog 62 bytes overblijven voor de transportlaag. Een UDP header is 8 bytes lang, een TCP header is 20 bytes lang. Zonder compressie is er 1 byte nodig voor de 6LoWPAN header en blijft er dus voor de data die daadwerkelijk verstuurd moet worden maar 53 bytes over bij gebruik van UDP, of 41 bytes bij gebruik van TCP. Met compressie is 6LoWPAN in staat om, bij gebruik van UDP, de headers dusdanig te comprimeren dat er maar liefst 108 bytes beschikbaar zijn voor de te versturen data. Meer dan een verdubbeling van het aantal beschikbare bytes!
- Opdelen en samenvoegen. Een IPv6 datapakket heeft een minimale grootte van 1280 bytes, vele malen groter dan een 6LoWPAN datapakket. Het is daarom een taak van de adaptation layer om een IPv6 datapakket dat verstuurd moet worden op te splitsen in meerdere 6LoWPAN datapaketten. En omgekeerd meerdere ontvangen 6LoWPAN datapaketten samen te voegen tot één IPv6 datapakket.

De adaptation layer zorgt ervoor dat er geen aanpassingen of beperkingen aan IPv6 nodig zijn. De functionaliteit van het opdelen en samenvoegen maakt het mogelijk dat er nog steeds grote hoeveelheden data verstuurd kunnen worden. Maar het is de verwachting van de 6LoWPAN werkgroep dat er bij de meeste toepassingsgebieden van 6LoWPAN maar weinig data verstuurd hoeft te worden. In combinatie met de compressie van de header is de verwachting dat dit dan in één 6LoWPAN datapakket past.

Oorspronkelijk werden datapaketten gerouteerd in de datalinklaag (laag 2) op basis van de MAC adressen. Deze techniek wordt: “Mesh-Under” genoemd. Deze techniek bleek echter nadelen te hebben. Binnen de Internet Engineering Task Force (IETF) is daarom de ROLL werkgroep opgericht. ROLL staat voor: “Routing Over Low power and Lossy networks”. Deze werkgroep is bezig met een nieuwe standaard genaamd: “IPv6 Routing Protocol for Low power and Lossy Networks” afgekort tot RPL. Bij RPL worden datapaketten gerouteerd in de netwerklaag (laag 3). Deze techniek wordt ook wel: “Route-Over” genoemd. De RPL standaard is nog een draft. De laatste revisie, 18, is gedateerd 4 februari 2011.

Bij het gebruik van 6LoWPAN gaat het netwerk er als volgt uitzien:



Voor- en nadelen van 6LoWPAN:

Voordelen	Nadelen
Bestaande IP gebaseerde technologieën zijn bekend en bewezen effectief.	Kleine pakketgrootte (127 bytes). (dit nadeel is een eigenschap van IEEE 802.15.4).
Netwerkbeheer kan worden uitgevoerd met bekende tools zoals ping en traceroute.	Lage bandbreedte (250 kbps). (dit nadeel is een eigenschap van IEEE 802.15.4).
6LoWPAN apparatuur kan direct worden aangesloten op IP gebaseerde netwerken (zoals het Internet) zonder dat een gateway of proxy nodig is.	Als in de transportlaag blijkt dat een TCP pakket niet correct ontvangen is dan moeten alle bijbehorende 6LoWPAN pakketten opnieuw verstuurd worden.
6LoWPAN is een IETF standaard. IETF standaarden zijn gratis toegankelijk en iedereen kan er gratis aan bijdragen (de IETF kent geen lidmaatschap).	
Het gebruik van 6LoWPAN is transparant; Een PC kan rechtstreeks communiceren met een sensor.	

4. Internet Protocol

De 6LoWPAN standaard maakt het alleen mogelijk om via IPv6 te communiceren, communicatie via IPv4 is niet mogelijk. IPv6 staat voor: Internet Protocol versie 6. Iedereen is bekend met IPv4, maar IPv6 wordt nog weinig gebruikt. Ongeacht van welke Internet Protocol versie je gebruik maakt; om te kunnen communiceren heb je een IP-adres nodig. Dit hoofdstuk bevat eerst een korte uitleg over IPv4, daarna over IPv6.

4.1 IPv4

Een IPv4-adres is een 32 bits getal, bestaande uit 4 groepjes van 8 bits. Ieder groepje wordt decimaal weergegeven met een punt als scheidingsteken, bijvoorbeeld: 10.10.4.55.

Bij IPv4 heeft een netwerk interface altijd precies één IP-adres. Dit IP-adres kan op drie verschillende manieren worden verkregen / toegewezen:

- Een dynamisch IP-adres via DHCP,
- Een vast IP-adres via DHCP (op basis van het MAC adres),
- Een op het apparaat handmatig ingesteld vast IP-adres.

Het IP-adres zelf, het getal, is een publiek of een privé IP-adres. Een publiek IP-adres is een uniek adres en wordt uitgedeeld door de Internet Service Provider (ISP). Door dit publieke IP-adres is het apparaat voor de rest van de wereld zichtbaar en vanuit het Internet benaderbaar.

De binnen IPv4 voor privégebruik vrijgehouden IP-adressen zijn:

RFC1918 naam	Eerste IP-adres	Laatste IP-adres	Klasse omschrijving	Aantal adressen
24-bit blok	10.0.0.0	10.255.255.255	1 klasse A netwerk	16.777.216
20-bit blok	172.16.0.0	172.31.255.255	16 opeenvolgende klasse B netwerken	1.048.576
16-bit blok	192.168.0.0	192.168.255.255	256 opeenvolgende klasse C netwerken	65.536

4.2 IPv6

Een IPv6-adres is een 128 bits getal, bestaande uit 8 groepjes van 16 bits. Ieder groepje wordt hexadecimaal weergegeven met een dubbelepunt als scheidingsteken. De dubbelepunt wordt echter bij IPv4 al gebruikt in de adres balk van een webbrowser om het poort nummer op te geven. Daarom is afgesproken dat als je een website wil bezoeken op basis van een IPv6 IP-adres je rechte haken om het IP-adres moet zetten.
Voorbeeld: `http://[FE80::0A:00:27:FF:FE:F9:8F:A6].`

Bovenstaand IP-adres is volledig uitgeschreven. IPv6 kent de volgende twee afspraken waarmee IP-adressen korter kunnen worden opgeschreven:

1. Precies één aaneengesloten reeks met groepjes bestaande uit vier nullen mogen worden weggelaten. Op de plaats van de reeks moet dan een dubbele dubbelepunt.
2. Als een reeks met nullen begint dan mogen deze worden weggelaten.

`FE80:0000:0000:0000:0A00:27FF:FEF9:8FA6`	Aaneengesloten reeks nullen verwijderen.
`FE80::0A00:27FF:FEF9:8FA6`	Voorloopnullen verwijderen.
`FE80::A00:27FF:FEF9:8FA6`	Zelfde IP-adres; korte notatie.

Net als bij IPv4 kent IPv6 ook publieke en privé IP-adressen. Bij IPv6 worden echter andere termen gebruikt: een publiek IP-adres wordt een globaal adres genoemd, en een privé IP-adres wordt een 'link local' adres genoemd. De termen globaal en link local geven het bereik van het IP-adres aan. Link local adressen zijn adressen die uitsluitend op één link (= netwerk interface) worden verstuurd en nooit voorbij een router komen.

Start bits	IPv6 notatie	Omschrijving
001	2000::/3	Globale adressen
1111 1110 10	FE80::/10	Link local adressen

Bij IPv6 heeft een netwerk interface altijd een link local adres en daarnaast vaak ook een globaal adres (voor Internet toegang). Dit in tegenstelling tot IPv4 waar een netwerk interface altijd maar precies één IP-adres heeft.

Net als bij IPv4 zijn IPv6-adressen opgedeeld in een netwerkgedeelte en een hostgedeelte. Bij IPv4 wordt dit aangegeven door het netwerkadres en een netmasker. Bij IPv6 vormen de eerste (linker) 64 bits het netwerkdeel, terwijl de laatste (rechter) 64 bits het hostgedeelte vormen. Het netwerkdeel wordt veelal prefix genoemd. IPv6 kent geen netmasker, de lengte van de prefix (het aantal bits) wordt weergegeven door een slash gevolgd door de lengte, bijvoorbeeld: /64. De /3 in bovenstaande tabel geeft aan dat voor globale adressen de eerste 3 van de 64 bits vastliggen. Er zijn dus 2^{61} netwerken mogelijk met ieder 2^{64} IP-adressen.

Bij IPv4 wordt vaak DHCP gebruikt om een IP-adres te verkrijgen. Een nadeel van DHCP is het feit dat het IP-adres iedere keer anders is. Voor servers is dit ongewenst. Bij servers wordt daarom vaak het IP-adres handmatig ingesteld op de server zelf, of er wordt handmatig een reservering voor een vast IP-adres toegevoegd aan de configuratie van de DHCP server. DHCP bestaat nog steeds bij IPv6, maar wordt weinig gebruikt.

Nieuw in IPv6 is: “stateless autoconfiguration”. Bij stateless autoconfiguration kan een PC zelf zijn globale IP-adres bepalen. Routers versturen hiervoor periodiek de prefix, de bijbehorende lengte en het IP-adres van de gateway (meestal is de router zelf de gateway). De prefix bepaalt de eerste 64 bits van het adres, de laatste 64 bits worden afgeleid van het MAC adres. Hardware op basis van de IEEE 802.15.4 standaard heeft een 64-bit MAC adres, maar in de PC wereld worden 48-bit MAC adressen gebruikt. Als de netwerkinterface een 48-bit MAC adres heeft dan wordt in het midden FFFE ingevoegd. Als laatste wordt bit 6 omgezet om aan te geven dat het hostgedeelte gebaseerd is op een uniek MAC adres. Bij link local IP-adressen is de prefix altijd gelijk aan FE80::/10. Link local IP-adressen werken dus altijd (ook zonder router). Het hostgedeelte wordt op dezelfde manier bepaald als hierboven beschreven.

`08:00:27:F9:8F:A6`	48-bit MAC adres
`08:00:27:FF:FE:F9:8F:A6`	FFFE ingevoegd
`0A:00:27:FF:FE:F9:8F:A6`	Bit 6 omgezet
`FE80::0A:00:27:FF:FE:F9:8F:A6`	Link local IP-adres

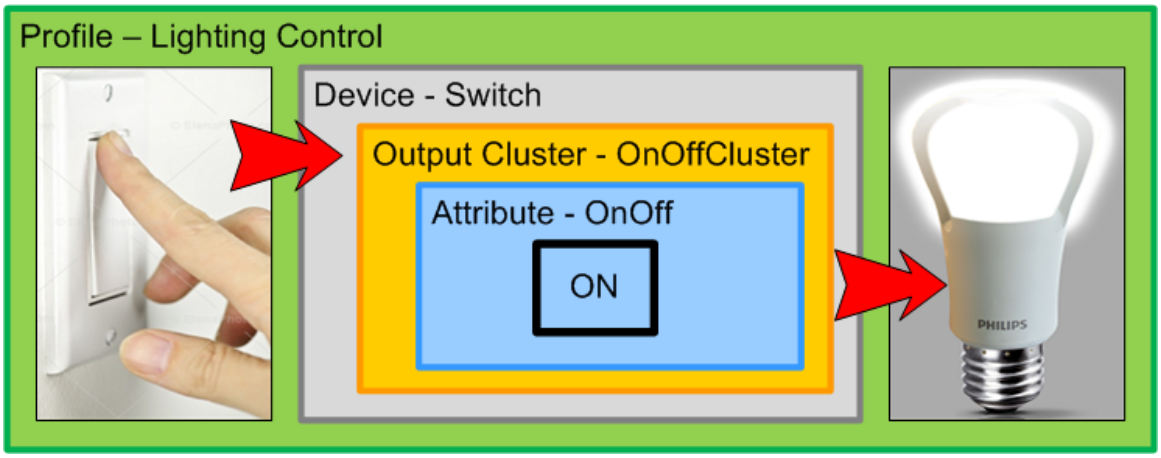
5. ZigBee

ZigBee is een open standaard voor draadloze netwerken, gebaseerd (of opgebouwd) op de IEEE 802.15.4 standaard. De ZigBee standaard is opgesteld door de ZigBee Alliance. De ZigBee Alliance is een non–profit consortium opgericht in 2002 dat samenwerkt met verschillende industriële partners om betrouwbare, draadloze controle en monitoring producten te ontwikkelen gebaseerd op een algemene, open standaard. Inmiddels zijn er honderden bedrijven aangesloten bij de ZigBee Alliance.

De eerste versie van de ZigBee standaard is in december 2004 bekrachtigd. De ZigBee Alliance heeft ervoor gekozen om het jaartal waarin de standaard officieel bekrachtigd is als versienummer te gebruiken. In 2006 is een verbeterde versie van de standaard bekrachtigd en de meest recente versie is in oktober 2007 bekrachtigd. Er bestaan echter twee versies van de meest recente specificatie, namelijk 2007 (Residential) en ZigBee PRO. In totaal bestaan er dus vier versies, namelijk: 2004, 2006, 2007 (Residential) en ZigBee PRO. Maar de allereerste versie, versie 2004, is inmiddels obsolete verklaard.

De grote kracht van de ZigBee standaard zijn de applicatieprofielen. Regelmatig publiceert de ZigBee Alliance een publiek applicatieprofiel. In een dergelijk profiel worden een verzameling apparaten en het communicatie protocol ertussen gestandaardiseerd voor een specifieke toepassing. Het eerste applicatieprofiel, Home Automation, is aangekondigd in november 2007. Andere voorbeelden van applicatieprofielen zijn: Remote Control, Health Care, Building Automation en Smart Energy. In deze applicatieprofielen wordt ieder netwerkbericht tot in detail gespecificeerd. Dit heeft als grote voordeel dat hierdoor producten van verschillende fabrikanten met elkaar kunnen communiceren.

Voorbeeld netwerkbericht om lamp aan te zetten:



Voor- en nadelen van ZigBee:

Voordelen	Nadelen
Interoperabiliteit door applicatieprofielen waarin de netwerkberichten tot in detail zijn gespecificeerd.	Leveranciersafhankelijkheid. Veel bedrijven maken gebruik van encryptie waardoor alleen apparatuur van dezelfde fabrikant met elkaar kan communiceren.
Meer keuze aan ZigBee apparatuur doordat de ZigBee standaard langer bestaat (dan de 6LoWPAN standaard).	ZigBee is een open standaard, maar je moet lid zijn van de ZigBee Alliance ($9.500 per jaar) om deel te mogen nemen aan werkgroepen en mee te mogen beslissen over de toekomst van de ZigBee standaard.
	Verplichte certificering (kosten).
	Kleine pakketgrootte (127 bytes). (dit nadeel is een eigenschap van IEEE 802.15.4).
	Lage bandbreedte (250 kbps). (dit nadeel is een eigenschap van IEEE 802.15.4).

6. Hardware

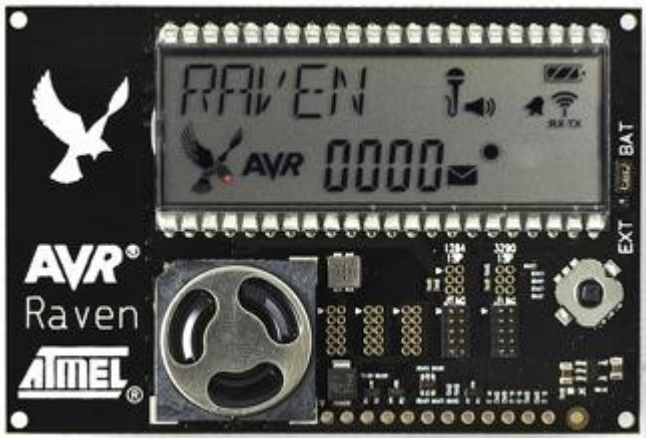
6.1 RZUSBSTICK

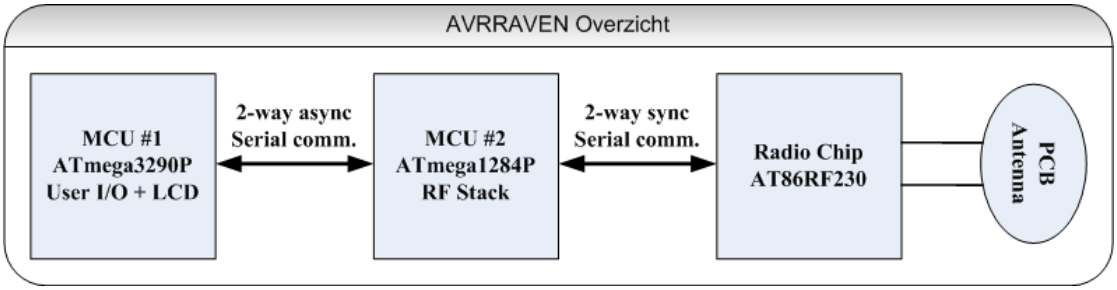
De RZUSBSTICK, de naam zegt het al, is een USB stick die een microcontroller en een radio chip bevat. Met behulp van deze USB stick kan een PC communiceren met de AVR Raven. De bekende besturingssystemen uit de PC wereld zoals Windows, OSX en Linux hebben nog geen ondersteuning voor 6LoWPAN. De RF Stack verantwoordelijk voor de daadwerkelijke 6LoWPAN communicatie draait daarom op de microcontroller van de USB stick.

Type	AT90USB1287
AVR Core	8-bit
Frequentie	16 MHz
Flash	128 KiB
EEPROM	4 KiB
SRAM	8 KiB

6.2 AVR Raven

AVR Raven is de naam van de hardware die ik, in overleg met TASS, gekozen heb voor de sensoren. De afmetingen van de sensoren zijn: 8 cm × 5,5 cm en wordt gevoed door twee 1.5V LR44 knoopcelbatterijen. De AVR Raven bevat alleen een temperatuursensor. Een externe lichtsensor is verbonden met de User I/O pinnen van de AVR Raven.



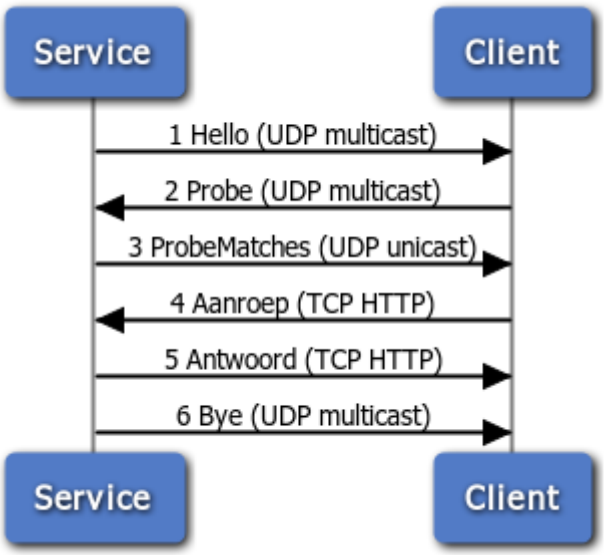


MCU#	MCU #1	MCU #2
Type	ATmega3290P	ATmega1284P
Doel	User I/O + LCD	RF Stack
AVR Core	8-bit	8-bit
Frequentie	20 MHz	20 MHz
Flash	32 KiB	128 KiB
EEPROM	1 KiB	4 KiB
SRAM	2 KiB	16 KiB

Naast de interne flash / EEPROM bevat de AVR Raven 2 MiB externe flash en 256 bytes externe EEPROM. De externe flash is aangesloten op MCU #1 en de EEPROM is aangesloten op MCU #2. Let op: De verbinding tussen MCU #1 en #2 is asynchroon, de verbinding tussen MCU #2 en de radio chip is synchroon. De AT86RF230 radio chip werkt alleen op de 2.4 GHz frequentieband. De PCB antenne heeft een maximale versterking van 5 dB.

7. Amigo communicatie

Eén van de belangrijkste requirements is compatibiliteit met het huidige Amigo platform.



Het Amigo platform maakt gebruik van webservices. Om te kunnen communiceren met een webservice moet er gebruik gemaakt worden van Simple Object Access Protocol (SOAP). SOAP is een protocol dat XML berichten stuurt over een HTTP verbinding (zie pijl 4 en 5).

Elk Amigo netwerk is uniek. Het heeft zijn eigen apparaten, die elk een eigen service hebben met elk hun eigen mogelijkheden. Door deze samenstelling is het niet van tevoren mogelijk om te weten wat voor services beschikbaar zijn en op welk adres deze services te benaderen zijn. Om dit probleem op te lossen maakt Amigo gebruik van WS-Discovery. WS-Discovery is een protocol dat XML berichten stuurt over een UDP verbinding (zie pijl 1, 2, 3 en 6).

Het Amigo project is gestart in 2004. Toentertijd bevatte Java en C# nog geen ondersteuning voor webservices en WS-Discovery. Als onderdeel van het Amigo project is er door het European Microsoft Innovation Center (EMIC) een externe software bibliotheek geschreven voor C# met ondersteuning voor webservices en WS-Discovery. Het Fraunhofer instituut heeft een externe software bibliotheek geschreven voor Java met dezelfde functionaliteit.

Literatuurlijst

IPv6 Advanced Protocols Implementation
Hoofdstuk 2.10 IPv6 Multicast Forwarding
Auteur: Keiichi Shima en Qing Li
ISBN13: 9780123704795

6LoWPAN The Wireless Embedded Internet
Auteur: Zach Shelby en Carsten Bormann
ISBN13: 9780470747995

Interconnecting Smart Objects With IP
Auteur: Jean-Philippe Vasseur en Adam Dunkels
ISBN13: 9780123751652

Running IPv6
Auteur: Iljitsch Van Beijnum
ISBN13: 9781590595275

ZigBee geschiedenis
http://en.wikipedia.org/wiki/ZigBee#History

Draadloze communicatie voor Amigo

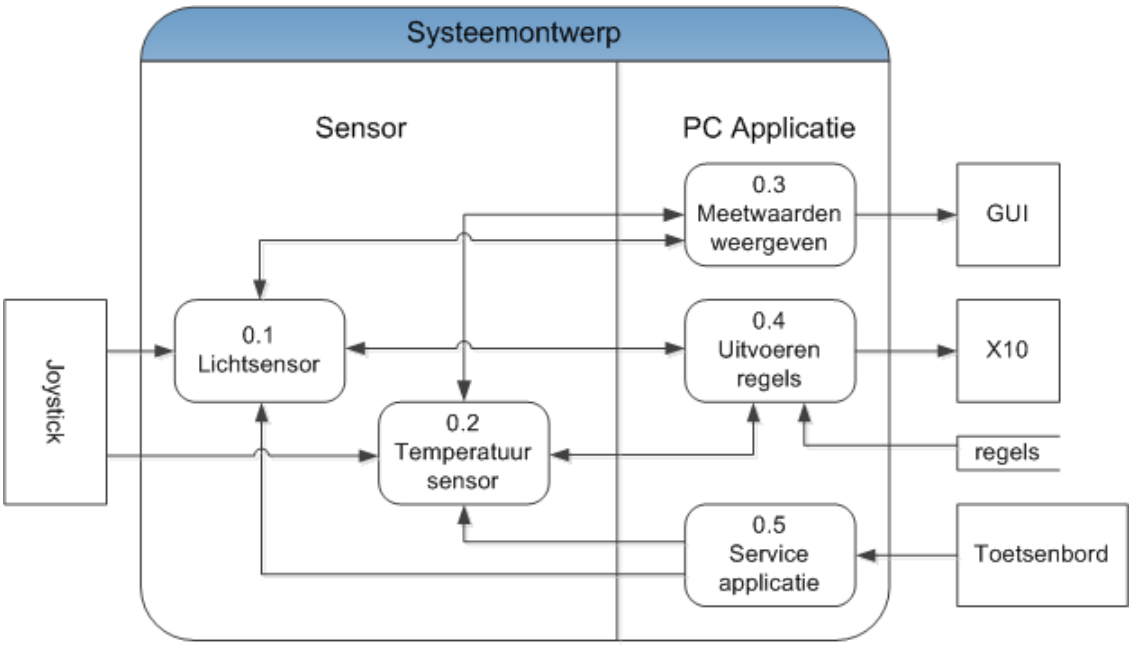
Bijlage D: Software Design Specificatie

Dit document is gepubliceerd door:
TASS BV
Eindhoven, Nederland

Commentaar en suggesties kunnen worden gestuurd naar:
TASS B.V.
Postbus 80060
5600 KA Eindhoven
Nederland
Tel: +31 40 2503200
Fax: +31 40 2503201

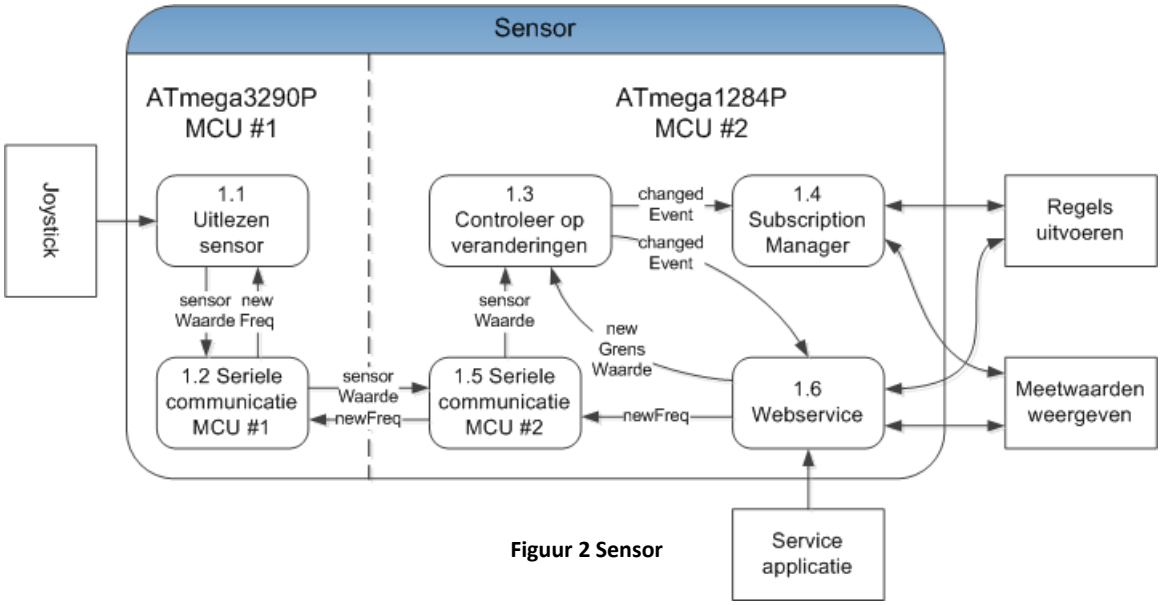
Inhoudsopgave

1. Systeemontwerp

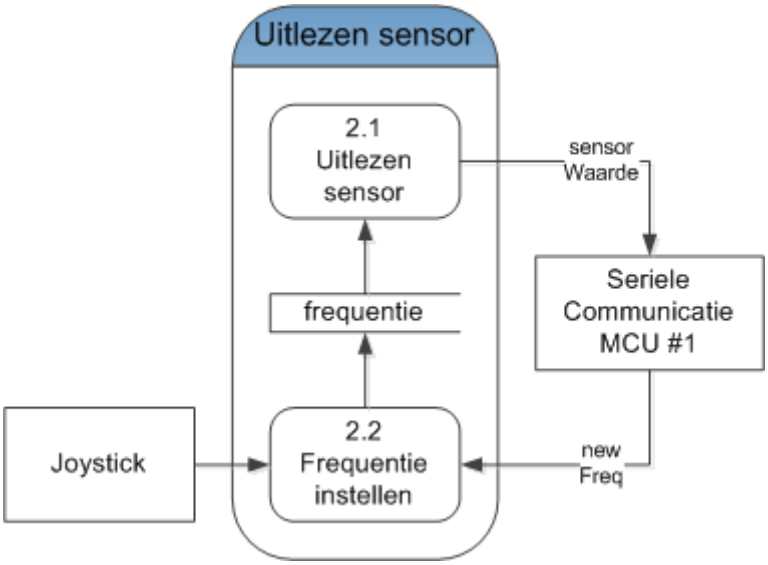


Figuur 1 Systeemontwerp

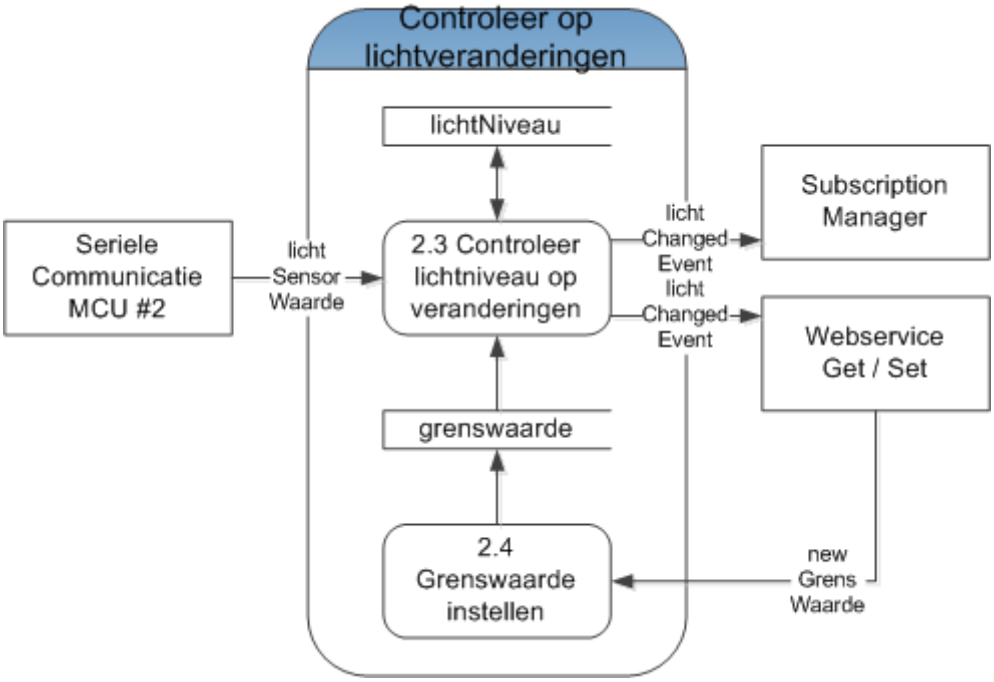
1.1. Sensor



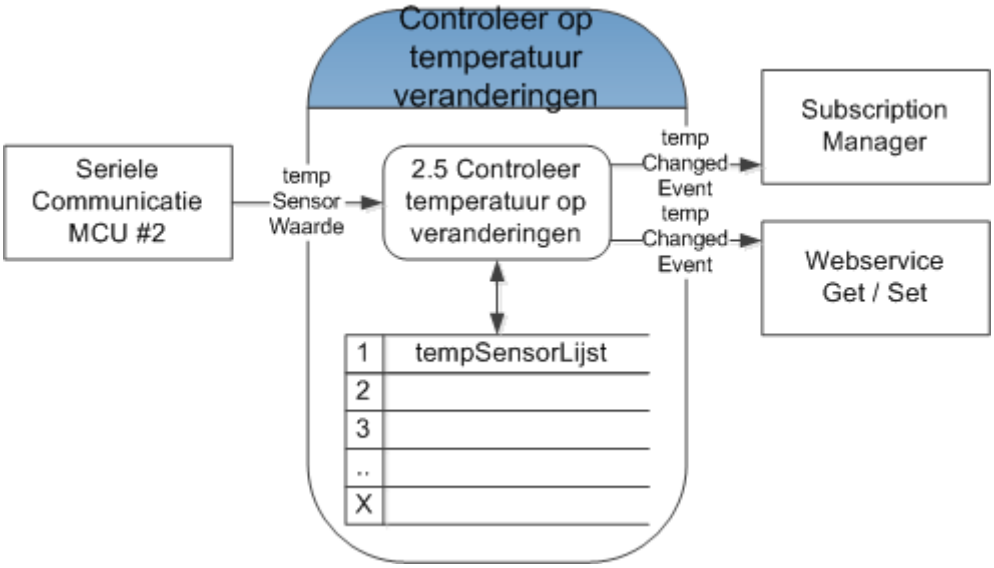
Figuur 2 Sensor



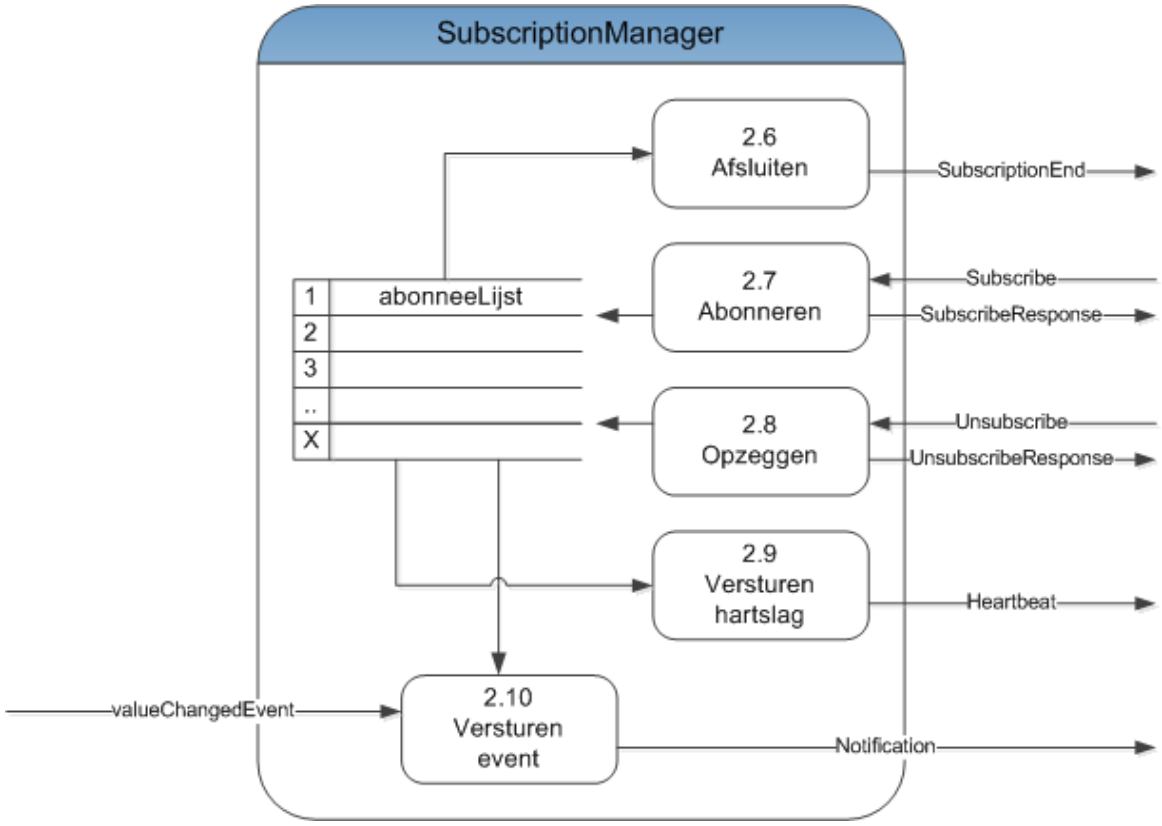
Figuur 3 Uitlezen sensor



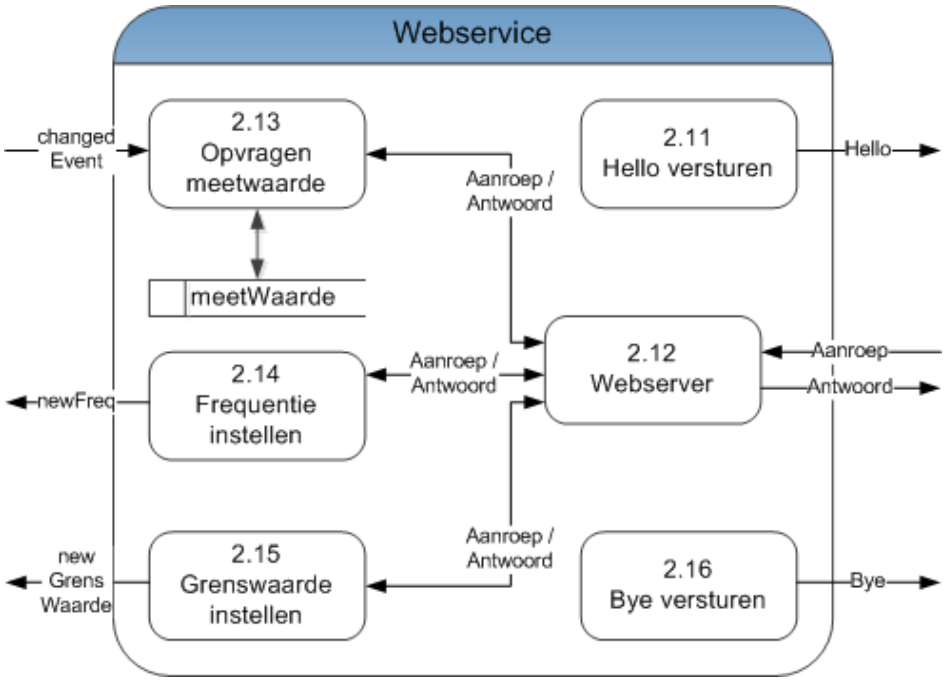
Figuur 4 Controleer op lichtveranderingen



Figuur 5 Controleer op temperatuurveranderingen

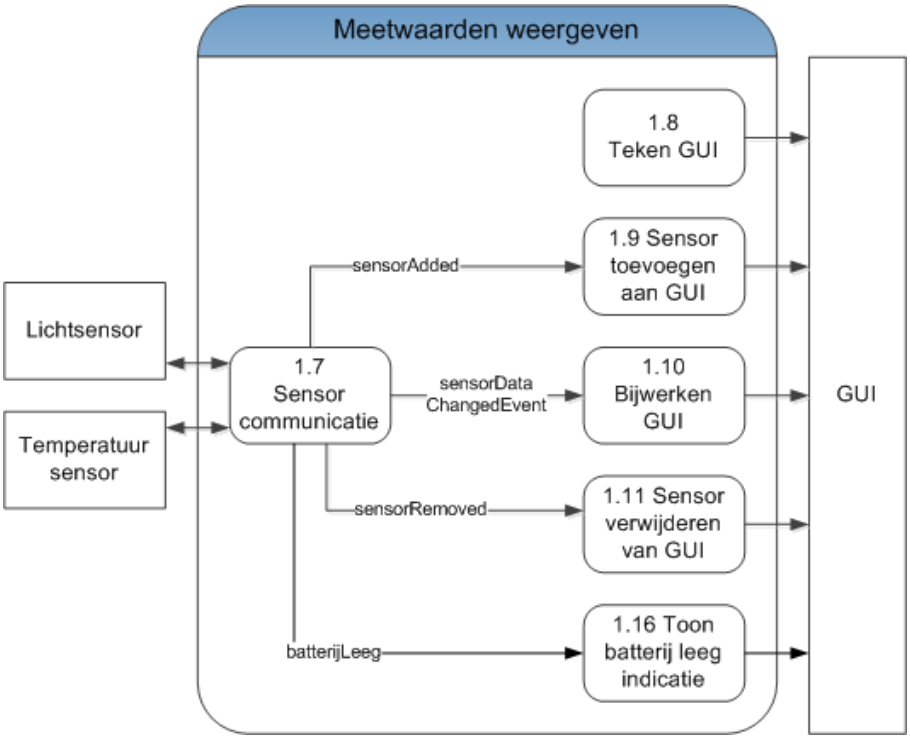


Figuur 6 SubscriptionManager

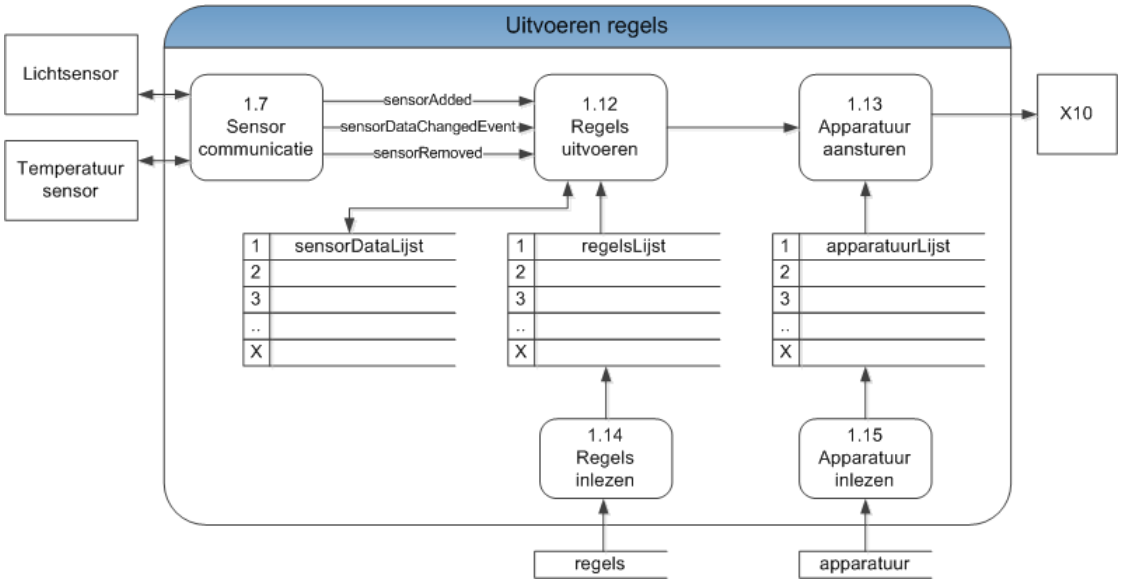


Figuur 7 Webservice

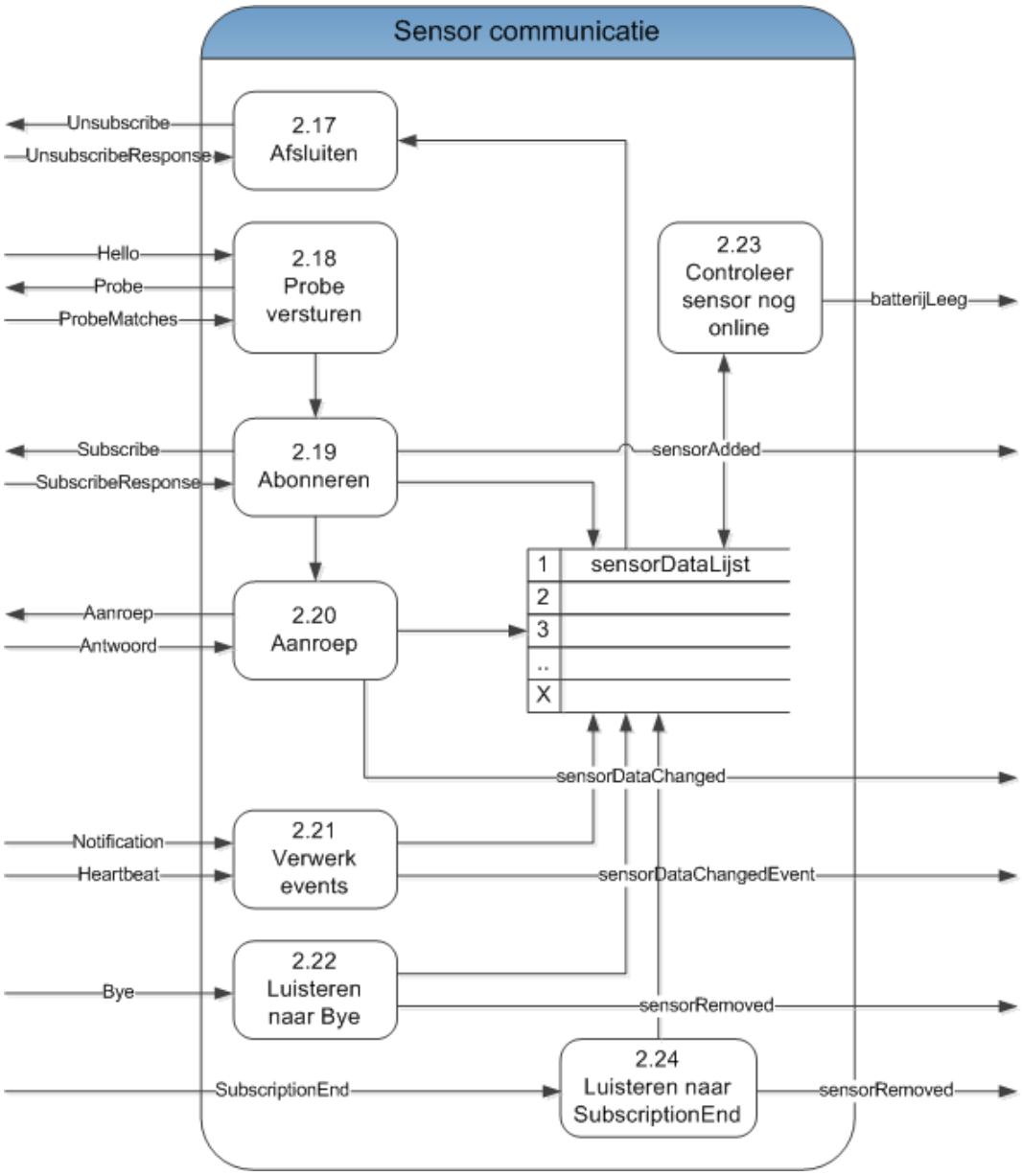
1.2. PC Applicatie



Figuur 8 Meetwaarden weergeven

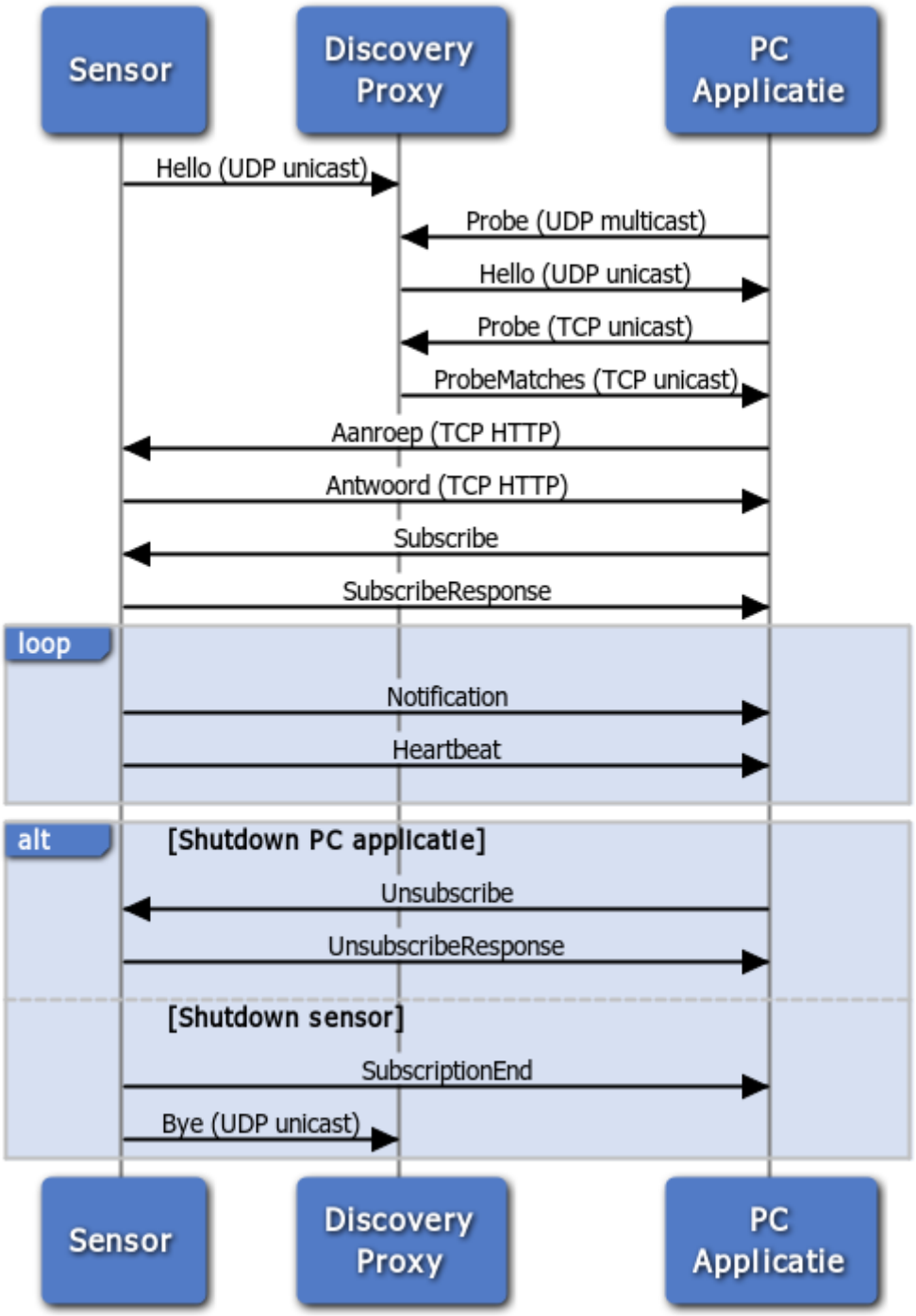


Figuur 9 Uitvoeren regels



Figuur 10 Sensor communicatie

2. Externe interface



Figuur 11 Externe communicatie

3. Detailontwerp

Globale variabele	Waarde
Proxy	soap.udp://[AAAA::1]:3702/DiscoveryProxy
GlobalUUID	urn:uuid:4fa77ef8-9e8b-4e5d-b662-3c5c34b58b11
Service	6LoWPANService
XAddrs	http://[AAAA::11::22FF:FE33:4455]/service.cgi
APP_MAX_DELAY	500

3.1. Temperatuursensor MCU #2

3.1.1. Hello versturen

Taak: Hello bericht versturen naar de Discovery Proxy.
Wanneer: Bij het opstarten van de sensor.
Type bericht: UDP unicast naar poort nummer 3702.
Standaard: WS-Discovery.
Locatie: http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.html

Inhoud bericht:

```
 1 <?xml version="1.0" encoding="utf-8"?>
 2 <s:Envelope
 3   xmlns:a="http://www.w3.org/2005/08/addressing"
 4   xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01"
 5   xmlns:s="http://www.w3.org/2003/05/soap-envelope" >
 6   <s:Header>
 7     <a:Action>
 8       http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/Hello
 9     </a:Action>
10     <a:MessageID>urn:uuid:$__UUID__$</a:MessageID>
11     <a:To>$__Proxy__$</a:To>
12     <d:AppSequence InstanceId="$__InstanceId__$"
   MessageNumber="$__SeqNr__$" />
13   </s:Header>
14   <s:Body>
15     <d:Hello>
16       <a:EndpointReference>
17         <a:Address>$__GlobalUUID__$</a:Address>
18       </a:EndpointReference>
19       <d:Scopes>urn:amigo</d:Scopes>
20       <d:Types>$__Service__$ d:TargetService</d:Types>
21       <d:XAddrs>$__XAddrs__$</d:XAddrs>
22       <d:MetadataVersion>1</d:MetadataVersion>
23     </d:Hello>
24   </s:Body>
25 </s:Envelope>
```

Implementatie stappen:

- $__UUID__$ (regel 10) moet vervangen worden door een unieke UUID.
- $__Proxy__$ (regel 11) moet vervangen worden door: soap.udp://[AAAA::1]:3702/DiscoveryProxy.
- $__InstanceId__$ (regel 12) moet vervangen worden door de globale variabele instanceId. De globale variabele InstanceId moet tijdens het opstarten de waarde krijgen van de huidige tijd uitgedrukt in seconden sinds 1 januari 1970. InstanceId mag daarna niet meer veranderen. Alleen bij het herstarten van de software moet InstanceId opnieuw geïnitialiseerd worden.
- $__SeqNr__$ (regel 12) moet vervangen worden door de globale variabele seqNr.
- $__GlobalUUID__$ (regel 17) moet vervangen worden door: urn:uuid:4fa77ef8-9e8b-4e5d-b662-3c5c34b58b11.
- $__Service__$ (regel 20) moet vervangen worden door: 6LoWPANService.
- $__XAddrs__$ (regel 21) moet vervangen worden door: http://[AAAA::11::22FF:FE33:4455]/service.cgi.

FOR X = 1 TO 3

- Random tijd wachten tussen 0 en 500 milliseconden wachten.
- Bericht versturen.

ENDFOR

- De globale variabele seqNr moet met 1 opgehoogd worden.

Dit bericht moet via UDP verstuurd worden. Een eigenschap van UDP is dat er geen garantie is dat het bericht daadwerkelijk aankomt. Om de kans dat het bericht verloren gaat te verkleinen moet het bericht drie keer verstuurd worden. Deze aanpak wordt ook gehanteerd door de huidige Amigo implementatie. Let op: De drie berichten moeten identiek zijn! De XML tags die een unieke waarde moeten hebben, zoals: MessageNumber, mogen niet opgehoogd worden.

3.1.2. Webserver

Taak: Binnenkomende berichten beantwoorden.
Wanneer: Continue.
Type bericht: Binnenkomende TCP unicast berichten op poort nummer 80.
Standaard: SOAP 1.1 (XML over HTTP).
Locatie: http://www.w3.org/Protocols/rfc2616/rfc2616.html
http://www.w3.org/TR/2000/NOTE-SOAP-20000508

Een aanroep bestaat uit HTTP headers en XML data. In de HTTP header SOAPAction staat welke functie aangeroepen moet worden en in de XML data staan de input parameters.

Voorbeeld aanroep:

```
 1  POST /service.cgi HTTP/1.1\r\n
 2  User-Agent: curl/7.21.0 (arm-unknown-linux-gnueabi) libcurl/7.21.0
    OpenSSL/0.9.8o zlib/1.2.3.4 libidn/1.15 libssh2/1.2.6\r\n
 3  Host: [AAAA::11:22FF:FE33:4455]\r\n
 4  Accept: */*\r\n
 5  Content-Type: text/xml; charset=utf-8\r\n
 6  SOAPAction: "http://www.tass.nl/setFrequentie"\r\n
 7  Content-Length: 289\r\n
 8  \r\n
 9  <?xml version="1.0" encoding="utf-8"?>
10  <soap:Envelope
11   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
12   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
13  <soap:Body>
14  <setFrequentie xmlns="http://www.tass.nl/">
15  <newFreq>2</newFreq>
16  </setFrequentie>
17  </soap:Body>
18  </soap:Envelope>
```

Implementatie stappen:

WHILE (waar)
- Wacht op binnenkomend bericht.
- Controleer of het een HTTP POST bericht is (regel 1).
- Controleer of er een HTTP SOAPAction header aanwezig is (regel 6).
- Controleer of de functie door ons wordt aangeboden (regel 6).
- Controleer of de XML data de juiste input parameters bevat (regel 9 tot en met 18).
- IF één van bovenstaande controles is gefaald THEN
  - Foutmelding versturen.
- ENDIF
- CALL functie(input parameters) RETURNING antwoord.
- Antwoord versturen.

ENDWHILE

3.1.3. Opvragen meetwaarde

Dit component biedt de volgende twee functies aan:

- `char *getLichtNiveau(char *reply);`
- `char *getTemperatuur(char *reply);`

Deze functies worden aangeroepen door component Web Server (zie §3.1.2). De returnwaarde is het complete antwoord, dat wil zeggen: HTTP + XML. Component Web Server verzendt vervolgens het antwoord naar de PC applicatie.

Returnwaarde getTemperatuur:

```
 1 HTTP/1.1 200 Ok\r\n
 2 Server: Contiki/2.5\r\n
 3 Content-Length: 332\r\n
 4 Content-Type: text/xml; charset=utf-8\r\n
 5 \r\n
 6 <?xml version="1.0" encoding="utf-8"?>
 7 <soap:Envelope
 8  xmlns:t="http://www.tass.nl/"
 9  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
10  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
11 <soap:Body>
12 <t:getTemperatuur>
13 <t:temperatuur>18.0</t:temperatuur>
14 </t:getTemperatuur>
15 </soap:Body>
16 </soap:Envelope>
```

Returnwaarde getLichtNiveau:

```
 1 HTTP/1.1 200 Ok\r\n
 2 Server: Contiki/2.5\r\n
 3 Content-Length: 333\r\n
 4 Content-Type: text/xml; charset=utf-8\r\n
 5 \r\n
 6 <?xml version="1.0" encoding="utf-8"?>
 7 <soap:Envelope
 8  xmlns:t="http://www.tass.nl/"
 9  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
10  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
11 <soap:Body>
12 <t:getLichtNiveau>
13 <t:lichtNiveau>licht</t:lichtNiveau>
14 </t:getLichtNiveau>
15 </soap:Body>
16 </soap:Envelope>
```

Let op: Aan het einde van de laatste regel hoeft geen enter.

Let op: Content-Length moet de grootte van de XML data (regel 6 tot en met 16) bevatten. Dit moet gelijk zijn aan de bestandsgrootte van het XML bestand.

3.1.4. Frequentie instellen

Dit component biedt de volgende twee functies aan:

- `char *setFrequentie(char *reply, int newFreq);`

Deze functie wordt aangeroepen door component Web Server (zie §3.1.2). De returnwaarde is het complete antwoord, dat wil zeggen: HTTP + XML. Component Web Server verzendt vervolgens het antwoord naar de PC applicatie.

Returnwaarde setFrequentie:

```
 1 HTTP/1.1 200 Ok\r\n
 2 Server: Contiki/2.5\r\n
 3 Content-Length: 318\r\n
 4 Content-Type: text/xml; charset=utf-8\r\n
 5 \r\n
 6 <?xml version="1.0" encoding="utf-8"?>
 7 <soap:Envelope
 8  xmlns:t="http://www.tass.nl/"
 9  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
10  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
11 <soap:Body>
12 <t:setFrequentie>
13 <t:result>OK</t:result>
14 </t:setFrequentie>
15 </soap:Body>
16 </soap:Envelope>
```

Let op: Aan het einde van de laatste regel hoeft geen enter.

Let op: Content-Length moet de grootte van de XML data (regel 6 tot en met 16) bevatten. Dit moet gelijk zijn aan de bestandsgrootte van het XML bestand.

3.1.5. Grenswaarde instellen

Dit component biedt de volgende twee functies aan:

- `char *setGrensWaarde(char *reply, int newGrensWaarde);`

Deze functie wordt aangeroepen door component Web Server (zie §3.1.2). De returnwaarde is het complete antwoord, dat wil zeggen: HTTP + XML. Component Web Server verzendt vervolgens het antwoord naar de PC applicatie.

Returnwaarde setGrensWaarde:

```
 1 HTTP/1.1 200 Ok\r\n
 2 Server: Contiki/2.5\r\n
 3 Content-Length: 320\r\n
 4 Content-Type: text/xml; charset=utf-8\r\n
 5 \r\n
 6 <?xml version="1.0" encoding="utf-8"?>
 7 <soap:Envelope
 8  xmlns:t="http://www.tass.nl/"
 9  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
10  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
11 <soap:Body>
12 <t:setGrensWaarde>
13 <t:result>OK</t:result>
14 </t:setGrensWaarde>
15 </soap:Body>
16 </soap:Envelope>
```

Let op: Aan het einde van de laatste regel hoeft geen enter.

Let op: Content-Length moet de grootte van de XML data (regel 6 tot en met 16) bevatten. Dit moet gelijk zijn aan de bestandsgrootte van het XML bestand.

3.1.6. Bye versturen

Taak: Bye bericht versturen naar de Discovery Proxy.
Wanneer: Bij het afsluiten van de sensor.
Type bericht: UDP unicast naar poort nummer 3702.
Standaard: WS-Discovery.
Locatie: http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.html

Inhoud bericht:

```
 1  <?xml version="1.0" encoding="utf-8"?>
 2  <s:Envelope
 3    xmlns:a="http://www.w3.org/2005/08/addressing"
 4    xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01"
 5    xmlns:s="http://www.w3.org/2003/05/soap-envelope" >
 6    <s:Header>
 7      <a:Action>
 8        http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/Bye
 9      </a:Action>
10      <a:MessageID>urn:uuid:$__UUID__$</a:MessageID>
11      <a:To>$__Proxy__$</a:To>
12      <d:AppSequence InstanceId="$__InstanceId__$"
    MessageNumber="$__SeqNr__$" />
13    </s:Header>
14    <s:Body>
15      <d:Bye>
16        <a:EndpointReference>
17          <a:Address>$__GlobalUUID__$</a:Address>
18        </a:EndpointReference>
19      </d:Bye>
20    </s:Body>
21  </s:Envelope>
```

Implementatie stappen:

Zie §3.1.1 Hello versturen.

3.1.7. Seriële communicatie MCU #2

Contiki bevat een beperkt aantal applicaties (afgekort apps). Dit zijn kant en klare bouwblokken die je in je project kunt hergebruiken. Zo bevat Contiki ook een applicatie genaamd raven-lcd-interface. Deze applicatie is verantwoordelijk voor de seriële communicatie aan de kant van MCU #2. Door in de Makefile van je project APPS=raven-lcd-interface op te nemen wordt deze applicatie onderdeel van je eigen project. De broncode van deze applicatie staat in de directory: contiki/platform/avr-raven/apps/raven-lcd-interface. De broncode zit in één bestand, ravenlcd.c, en creëert een proces genaamd raven_lcd_process. De bijbehorende Makefile zet de compiler vlag RAVEN_LCD_INTERFACE.

Bij het opstarten start de executie altijd in de functie met de naam main. Deze functie zit in het bestand: contiki/platform/avr-raven/contiki-raven-main.c. Deze functie initialiseert de hardware en start het autostart proces. Het autostart proces is de 'main' van je eigen project. Alleen als de compiler vlag RAVEN_LCD_INTERFACE gezet is dan worden de volgende stappen uitgevoerd:

1. Initialiseren van de RS232 poort naar MCU #1,
2. Het zetten van een RS232 input handler,
3. Het raven_lcd_process starten.

Daarna wordt pas het autostart proces gestart.

Het raven_lcd_process registreert een ICMP6 callback handler. Alleen als dit lukt dan wordt in een oneindige loop de functie raven_gui_loop aangeroepen. De functie raven_gui_loop verwerkt de binnenkomende events. Dit zijn of ICMP6 events of het event SERIAL_CMD wat betekent dat er via de seriële verbinding een commando is ontvangen van MCU #1.

Het bestand platform/avr-ravenlcd/main.h bevat onder andere een specificatie van de commando's op de seriële interface tussen MCU #1 en MCU #2.

Commando's van MCU #1 → MCU #2:

Commando	Waarde
NULL_CMD	0
SEND_TEMP	0x80
SEND_PING	0x81
SEND_ADC2	0x82
SEND_SLEEP	0x83
SEND_WAKE	0x84

Commando's van MCU #2 → MCU #1:

Commando	Waarde
REPORT_PING	0xC0
REPORT_PING_BEEP	0xC1
REPORT_TEXT_MSG	0xC2
REPORT_WAKE	0xC3

Met behulp van het commando SEND_TEMP wordt de temperatuur overgestuurd. De temperatuur komt binnen in een string. De string bevat alleen de meetwaarde (zonder spaties of eenheid) en kan dus eventueel eenvoudig geconverteerd worden naar een gebroken getal.

Als MCU #2 het commando SEND_SLEEP ontvangt dan wordt een energiebesparingmodes geactiveerd. Let op: Als je wilt gaan debuggen op MCU #2 dan kan het handig zijn om het activeren van de energiebesparingmodes tijdelijk uit te zetten. Uit de bestaande code valt op te maken dat als MCU #2 het commando SEND_WAKE ontvangt dat deze dan meteen moet antwoorden met REPORT_WAKE. Enkel bij het commando SEND_WAKE moet je een bevestiging terugsturen.

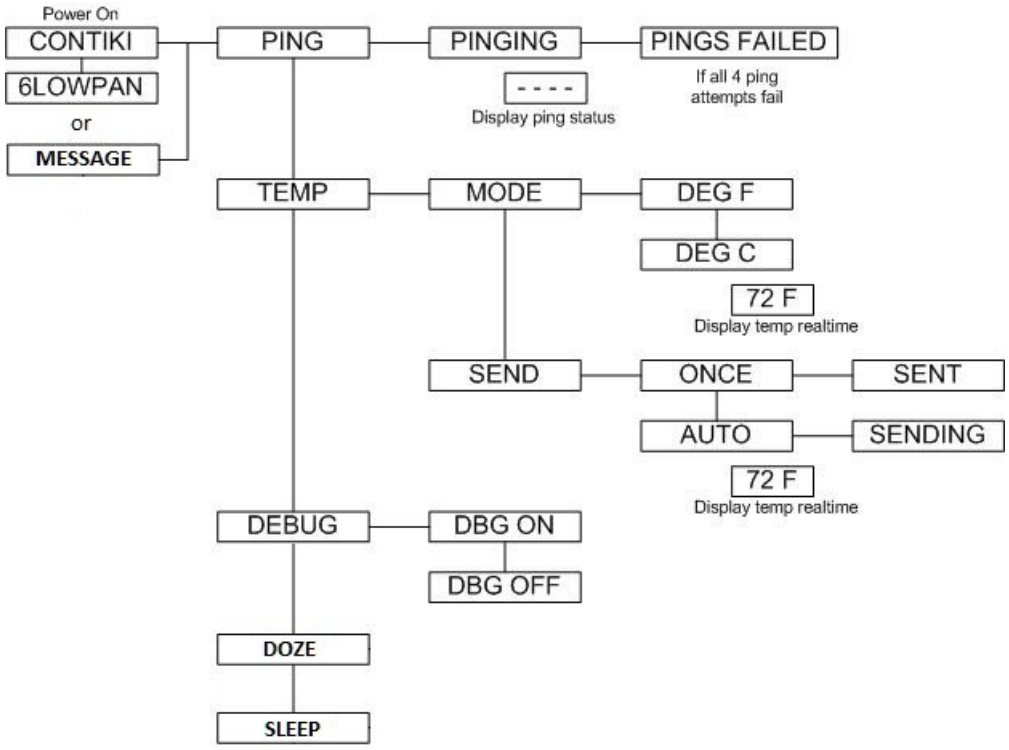
Een leuk extraatje is de ICMP6 callback handler. Bij een binnenkomende ping reply, dus een antwoord op een ping verstuurd vanaf de sensor, wordt het commando REPORT_PING naar MCU #1 verstuurd. Bij een binnenkomende ping request, dus als er een ping gestuurd wordt naar de sensor, wordt het commando REPORT_PING_BEEP naar MCU #1 verstuurd.

3.2. Temperatuursensor MCU #1

3.2.1. Raven LCD

MCU #1, type ATmega3290P, is verantwoordelijk voor de user I/O en de aansturing van de LCD. MCU #2, type ATmega1284P, is verantwoordelijk voor het aansturen van de radiochip.

Contiki bevat een kant en klare applicatie voor op MCU #1, genaamd Raven LCD. Deze applicatie staat in de directory: contiki/platform/avr-ravenlcd. Let op: Deze applicatie wordt wel meegeleverd met Contiki, maar maakt geen gebruik van het Contiki OS. Het Contiki besturingssysteem draait alleen op MCU #2. De Raven LCD applicatie is in staat de temperatuursensor uit te lezen en de meetwaarde serieel te versturen naar MCU #2. Daarnaast bevat de applicatie een simpel menu. De menu opties worden getoond op het display. Met behulp van een joystick is het mogelijk om door het menu te bladeren en opties te selecteren.



Figuur 12 Raven LCD menustructuur

Om in het menu te komen moet je de joystick naar rechts duwen. Het eerste submenu PING wordt dan getoond op het display. Door de joystick naar boven of beneden te duwen kun je door de submenu's bladeren. Door de joystick naar rechts of links te duwen, ga je naar een menu op een lager of hoger niveau. Een optie kiezen, bijvoorbeeld DBG OFF, kan wederom door de joystick naar rechts te duwen.

De temperatuursensor en de JTAG interface delen dezelfde GPIO. Daarom doet de Raven LCD applicatie bij het opstarten de JTAG interface uitzetten. Als er een waarschuwingssymbool in de vorm van een driehoek met daarin een uitroepteken getoond wordt op het display dan staat de JTAG interface nog aan terwijl er temperatuurmetingen gedaan worden. Doordat de Raven LCD applicatie de JTAG interface uitzet is het niet meer mogelijk om MCU #1 te flashen. Indien MCU #1 van nieuwe software voorzien moet worden dan moet je eerst, met behulp van de optie DBG ON, de JTAG interface aanzetten. Let op: Foutmeldingen bij het flashen van MCU #1 kunnen erop duiden dat de JTAG interface nog uitstaat.

Zoals je in Figuur 12 kunt zien kan de Raven LCD applicatie de temperatuur meten zowel in graden Celsius als in graden Fahrenheit. Daarnaast kan de applicatie de meetwaarde eenmalig of 'automatisch' opsturen. Automatisch betekent dat de sensor iedere *X* seconden wordt uitgelezen waarna de meetwaarde wordt opgestuurd. Bovenin het bestand menu.c heb ik de temperatuur ingesteld op graden Celsius en op automatisch opsturen. Het aantal seconden, de *X*, staat standaard ingesteld op 1 seconde (zie timer.c).

De temperatuur wordt gemeten met behulp van een NTC-weerstand. Een NTC-weerstand is een weerstand met een negatieve temperatuurcoëfficiënt. Dit betekent dat de elektrische weerstand afneemt als de temperatuur toeneemt. De NTC is verbonden met een analoog digitaal converter (ADC). De ADC is geïntegreerd in de microcontroller (MCU #1). De gebruikte NTC-weerstand is van fabrikant Murata en heeft typenummer NCP18WF104J03RB. De temperatuursensor heeft als nadeel dat de NTC-weerstand zo dicht naast de aansluiting voor de externe voeding zit dat deze de NTC verhit. Om nauwkeurige temperatuurmetingen te krijgen moet je dus batterijen gebruiken.

De Raven LCD applicatie bevat een tabel waarin aan de hand van de ADC-waarde de bijbehorende temperatuur opgezocht kan worden. Met behulp van deze tabel kan de temperatuur in hele graden nauwkeurig worden opgezocht (van -15 tot en met 60 graden Celsius). De bestaande tabel heeft een aantal nadelen, namelijk:

1. De temperatuur is niet nauwkeurig genoeg. Ik wil graag de temperatuur weten op een halve graad nauwkeurig. Interpoleren gaat niet, want de grafiek is niet lineair.
2. De ondergrens van -15°C is te hoog. We hebben afgelopen winter strenge nachtvorst gehad, met temperaturen onder de -15 graden Celsius. Ik wil graag een ondergrens van -20°C.
3. De applicatie bevat geen documentatie hoe men aan de ADC waarden gekomen is.

Daarom heb ik ervoor gekozen om zelf, met behulp van Excel, de ADC waarde te berekenen van -20 tot en met 60 graden Celsius in stappen van een halve graad. Ik heb hiervoor de formule gebruikt uit het gegevensblad van de fabrikant van de NTC-weerstand (zie pagina 21).

Formule voor het berekenen van de weerstand als functie van de temperatuur:

$$R = R0 * e^{B * \left(\frac{1}{T} - \frac{1}{T0}\right)}$$

Symbool	Omschrijving	Waarde
R	De weerstand van de NTC bij een temperatuur T	
T	De temperatuur in graden Kelvin	graden Celsius + 273,15
R0	De weerstand van de NTC bij de referentietemperatuur	100.000 kΩ
T0	De referentietemperatuur uitgedrukt in graden Kelvin	25°C = 298,15°K
B	B-constante voor de gebruikte NTC-weerstand	4250
e	Getal van Euler	2,71828182845905

Met behulp van bovenstaande formule is het mogelijk om voor een gegeven temperatuur de weerstand te berekenen. Daarna kan met onderstaande formule de uitgangswaarde van de ADC berekend worden. De gebruikte ADC is 10 bits, de uitgangswaarden lopen dus van 0 tot en met 1024. Als op de ingang van de ADC de volledige voedingsspanning wordt aangeboden (geen weerstand) dan is de uitgangswaarde van de ADC gelijk aan 1024. Als op de ingang van de ADC nul volt wordt aangeboden (volledige weerstand) dan is de uitgangswaarde van de ADC gelijk aan nul. De voedingsspanning van de ADC wordt ook gebruikt als voeding van de serieschakeling van vaste weerstand van 100kΩ en NTC. De NTC heeft een weerstandswaarde van 100kΩ bij 25 °C. Dit betekent dat bij 25°C de halve voedingspanning over de NTC staat en naar de ingang van de ADC gaat. Dus is de uitgangswaarde van de ADC gelijk aan 512.

Formule voor het berekenen van de uitgangswaarde van de ADC:

$$Uitgangswaarde\ ADC = \frac{R}{(R + R0)} * 1024$$

Als gevolg van deze wijziging heb ik ook overal het type van de variabele temperatuur moeten veranderen van een integer in een double. Voordat de temperatuur verstuurd wordt naar MCU #2 wordt deze omgezet naar een string. In de originele code werd het symbool C of F aan de string toegevoegd afhankelijk van de gekozen eenheid van temperatuur. Ik heb dit aangepast; Nu wordt alleen de temperatuur verstuurd, zodat de string eenvoudig geconverteerd kan worden naar een gebroken getal.

Deze pagina is opzettelijk leeg gelaten.

Draadloze communicatie voor Amigo

Training Document
Bijlage E: Amigo webservice met NET Framework 4.0

Dit document is gepubliceerd door:
TASS BV
Eindhoven, Nederland

Commentaar en suggesties kunnen worden gestuurd naar:
TASS B.V.
Postbus 80060
5600 KA Eindhoven
Nederland
Tel: +31 40 2503200
Fax: +31 40 2503201

Document Geschiedenis

Versie	Status	Datum	Auteur	Wijzigingen
0.1	Proposed	27-02-2012	R. Zuidinga	Initiële versie.

Inhoudsopgave

1. Introductie

1.1 Doel

Het doel van dit document is om uit te leggen hoe je met behulp van het .NET Framework versie 4.0 een eenvoudige Amigo webservice met WS-Discovery kunt maken.

Het Amigo project maakt onder andere gebruik van op SOAP gebaseerde webservices en WS-Discovery. Om een applicatie te maken die compatibel is met Amigo moet men deze standaarden implementeren. Het Amigo project is gestart in 2004. Toentertijd bevatte Visual Studio nog geen ondersteuning voor webservices en WS-Discovery. Microsoft was een van de deelnemers aan het Amigo project. Speciaal voor het Amigo project heeft Microsoft een (externe) software bibliotheek, genaamd EMIC, geschreven met ondersteuning voor webservices en WS-Discovery. Als onderdeel van het Amigo project is in het verleden een training document gemaakt met uitleg hoe je een eenvoudige webservice met WS-Discovery kunt maken op basis van deze EMIC bibliotheek (zie: [ATD]). Later is deze functionaliteit geïntegreerd in Visual Studio. Dit is een nieuw training document met dezelfde uitleg echter gebaseerd op Visual Studio 2010 en zonder gebruik te maken van een externe software bibliotheek.

Om dit document goed te kunnen volgen is bekendheid met de WS-Discovery standaard vereist (zie: [WSD]).

1.2 Termen en Afkortingen

Term	Omschrijving

1.3 Referenties

Referentie	Document ID	Titel
[ATD]	IST-2004-004182	Amigo Training Document: Using Web-Services https://gforge.inria.fr/frs/download.php/3255/Web_Services-Tutorial.pdf
[WSD]	N.v.t.	Web Services Dynamic Discovery (WS-Discovery) 1.1 http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.html

1.4 Distributielijst

Naam	Organisatie	Functie
Peter Vink	TASS	Klant / Bedrijfsbegeleider

2. Simpele Webservice

2.1 Service implementatie

File → New → Project.
Selecteer: Visual C# → WCF → WCF Service Library.
Naam: TemperatuurServiceLibrary.
Solution Name: TemperatuurServiceLibrary.

Sluit alle openstaande bestanden in Visual Studio.
Hernoem: IService1.cs naar: ITemperatuurService.cs
Hernoem: Service1.cs naar: TemperatuurService.cs

In ITemperatuurService.cs komt de input voor de WSDL. Boven iedere methode die je wilt aanbieden moet [OperationContract] staan (dit vervangt [WebMethod] uit de oude EMIC bibliotheek). Boven de interface moet [ServiceContract] staan. In TemperatuurService.cs komt de implementatie van de aangeboden methodes.

Beide bestanden gebruiken dezelfde externe bibliotheken:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Runtime.Serialization;
5  using System.ServiceModel;
6  using System.Text;
```

Code van ITemperatuurService.cs:

```
1  namespace TemperatuurServiceLibrary
2  {
3      [ServiceContract]
4      public interface ITemperatuurService
5      {
6          [OperationContract]
7          string GetLichtNiveau();
8
9          [OperationContract]
10         float GetTemperatuur();
11
12         [OperationContract]
13         string SetFrequentie(int NewFreq);
14
15         [OperationContract]
16         string SetGrensWaarde(int NewGrensWaarde);
17     }
18 }
```

Code van TemperatuurService.cs:

```
1  namespace TemperatuurServiceLibrary
2  {
3      public class TemperatuurService : ITemperatuurService
4      {
5          public string GetLichtNiveau()
6          {
7              return "licht";
8          }
9
10         public float GetTemperatuur()
11         {
12             return 18.0F;
13         }
14
15         public string SetFrequentie(int NewFreq)
16         {
17             return "OK";
18         }
19
20         public string SetGrensWaarde(int NewGrensWaarde)
21         {
22             return "OK";
23         }
24     }
25 }
```

Dit type project bevat het bestand App.config. Dit is een XML configuratie bestand wat de webservice beschrijft met behulp van endpoints. Een endpoint bestaat altijd uit de volgende onderdelen: **A**dres, **B**inding en **C**ontract. Adres is het adres waarop de webservice bereikbaar is, binding is het type van de webservice en contract is de interface (de WSDL). Als je het bestand App.config opent zie je dat er standaard al twee endpoints gedefinieerd zijn, namelijk: één met binding wsHttpBinding en één met binding mexHttpBinding. De eerste binding, wsHttpBinding, is één van de drie types die beschikbaar zijn voor het maken van een webservice. De laatste binding, mexHttpBinding, is een binding die door Visual Studio gebruikt wordt.

De volgende drie types zijn beschikbaar voor het maken van een webservice:

Binding	Omschrijving
basicHttpBinding	Deze binding voldoet aan de WS-I Basic Profile 1.1 specificatie. Gebruik deze binding voor maximale compatibiliteit met oudere (ASMX) webservices en client applicaties.
wsHttpBinding	Deze binding voldoet aan de pre-2007 draft van de WS-* specificaties. Gebruik deze binding om compatibel te zijn met oudere versies van WCF en het .NET framework.

Binding	Omschrijving
ws2007HttpBinding	Dit is de nieuwste binding en voldoet aan de OASIS standaarden. Deze binding biedt ondersteuning voor Security, ReliableSession, en TransactionFlow.

Let op: De binding namen zijn hoofdletter gevoelig.

Zoals je in de tabel kunt zien is ws2007HttpBinding de nieuwste binding en daarom gaan we deze gebruiken. Er moeten daarvoor twee wijzigingen gedaan worden in de App.config. Ten eerste moet wsHttpBinding vervangen worden door ws2007HttpBinding en ten tweede moet security expliciet uitgezet worden (security is buiten de scope van deze tutorial). Klik op Save All om alle veranderingen op te slaan.

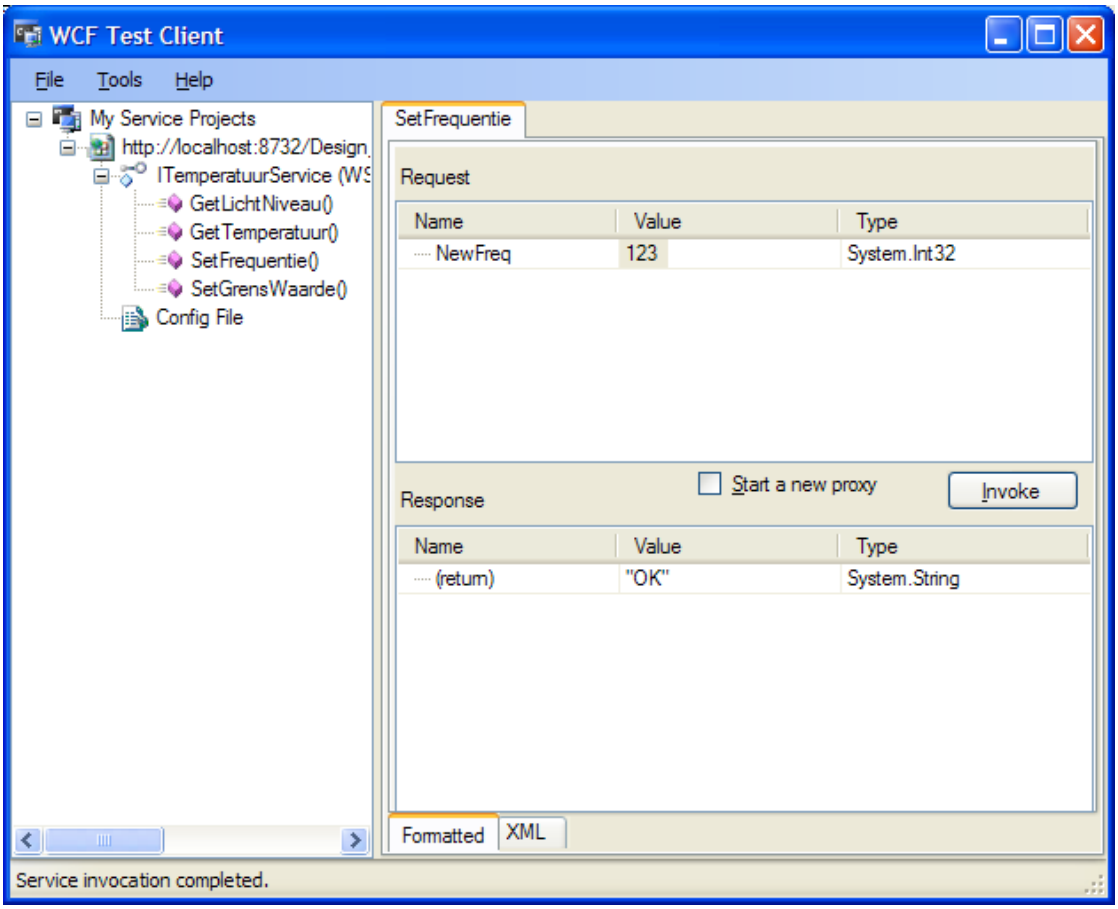
Wijzigingen in App.config:

```
1   <system.serviceModel>
2     <bindings>
3       <ws2007HttpBinding>
4         <binding name="NoSecurity">
5           <security mode="None">
6             <transport clientCredentialType="None" />
7             <message clientCredentialType="None" />
8           </security>
9         </binding>
10      </ws2007HttpBinding>
11    </bindings>
12    <services>
13      <service name="TemperatuurServiceLibrary.TemperatuurService">
14        <endpoint address="" binding="ws2007HttpBinding"
    bindingConfiguration="NoSecurity"
15          contract="TemperatuurServiceLibrary.ITemperatuurService">
16          <identity>
17            <dns value="localhost" />
18          </identity>
19        </endpoint>
```

Het moet nu mogelijk zijn om de webservice te bouwen zonder errors of warnings.
Je kunt dit testen door: Build → Clean Solution, gevolgd door: Build → Build Solution.

2.2 Service testen vanuit Visual Studio

Tot nu toe hebben we alleen de webservice geïmplementeerd. Voordat de webservice daadwerkelijk gebruikt kan worden moet deze eerst gehost worden in bijvoorbeeld IIS of een Windows applicatie. Door middel van de mexHttpBinding is het echter toch mogelijk om op een eenvoudige manier de webservice te testen. Als je op het groene Play icoon klikt dan wordt het programma WcfSvcHost gestart om de service te hosten. WcfSvcHost wordt geminimaliseerd gestart (zie icoon in de notification area van de Windows taakbalk). Meteen daarna wordt de WcfTestClient gestart. Met behulp van WcfTestClient is het mogelijk om de door de service aangeboden methodes aan te roepen en de return waarde te controleren (zie onderstaande screenshot).



2.3 Host implementatie

File → New → Project.
Selecteer: Visual C# → Windows → WPF Application.
Naam: TemperatuurServiceHost.
Solution Name: TemperatuurServiceHost.

Rechts klik op References → Add Reference, selecteer tabblad .NET, kies System.ServiceModel en klik op OK.

Rechts klik op References → Add Reference, selecteer tabblad Browse, navigeer naar TemperatuurServiceLibrary.dll en klik op OK. Deze DLL staat in de bin/Debug directory van het TemperatuurServiceLibrary project.

Aan MainWindow.xaml.cs moeten de volgende twee using statements worden toegevoegd:
using System.ServiceModel;
using TemperatuurServiceLibrary;

Code van MainWindow.xaml:

```
1  <Window x:Class="TemperatuurServiceHost.MainWindow"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          Title="Temperatuur Service Host" Height="300" Width="350">
5      <Grid>
6          <Button Height="23" HorizontalAlignment="Left" Margin="51,60,0,0"
7             Name="start" VerticalAlignment="Top" Width="75"
8             Click=" start_Click">Start</Button>
9          <Button Height="23" HorizontalAlignment="Right" Margin="0,60,56,0"
10            Name="stop" VerticalAlignment="Top" Width="75" IsEnabled="False"
11            Click=" stop_Click">Stop</Button>
12         <Label Height="23" HorizontalAlignment="Left" Margin="43,0,0,111"
13            Name="label1" VerticalAlignment="Bottom" Width="88">Service
   Status:</Label>
14         <TextBox IsReadOnly="True" Margin="133,0,59,107" Name="status"
15            Text="Service Stopped" Height="26"
   VerticalAlignment="Bottom"></TextBox>
16     </Grid>
17 </Window>
```

Code van MainWindow.xaml.cs

```
1   public partial class MainWindow : Window
2   {
3       private ServiceHost temperatuurSeviceHost;
4
5       public MainWindow()
6       {
7           InitializeComponent();
8       }
9
10      private void handleException(Exception ex)
11      {
12          MessageBox.Show(ex.Message, "Exception",
13              MessageBoxButton.OK, MessageBoxImage.Error);
14      }
15
16      private void start_Click(object sender, RoutedEventArgs e)
17      {
18          try
19          {
20              temperatuurSeviceHost = new
    ServiceHost(typeof(TemperatuurService));
21              temperatuurSeviceHost.Open();
22              stop.IsEnabled = true;
23              start.IsEnabled = false;
24              status.Text = "Service Running";
25          }
26          catch (Exception ex)
27          {
28              handleException(ex);
29          }
30      }
31
32      private void stop_Click(object sender, EventArgs e)
33      {
34          try
35          {
36              temperatuurSeviceHost.Close();
37              stop.IsEnabled = false;
38              start.IsEnabled = true;
39              status.Text = "Service Stopped";
40          }
41          catch (Exception ex)
42          {
43              handleException(ex);
44          }
45      }
46  }
```

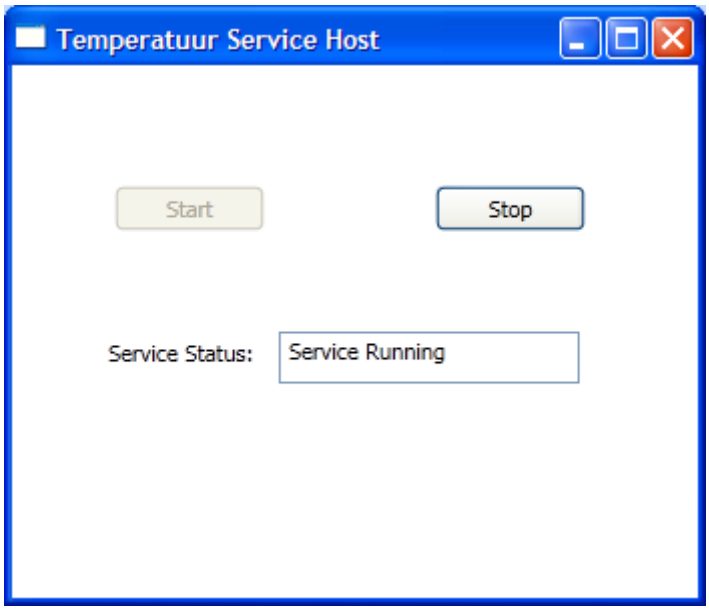
Als laatste moet een kopie van het bestand App.config uit het TemperatuurServiceLibrary project toegevoegd worden aan het huidige project. Je moet het bestand eerst kopiëren en daarna toevoegen aan het project. Toevoegen aan het project kan op de volgende manier: Rechts klik op het Project → Add → Existing Item. Selecteer All files (*.*) bij Objects of type en klik op OK.

Het moet nu mogelijk zijn om de service te bouwen zonder errors of warnings.
Je kunt dit testen door: Build → Clean Solution, gevolgd door: Build → Build Solution.

In de App.config staat het URL waarop de webservice gehost wordt, namelijk:
http://localhost:8732/Design_Time_Addresses/TemperatuurServiceLibrary/Service1/
Zoals je aan het URL kunt zien wordt er gebruik gemaakt van TCP poortnummer 8732. Bij TASS staat standaard de Windows Firewall aan. Gebruikers hebben geen rechten om de firewall uit te zetten, maar het is wel mogelijk om een poort te openen.

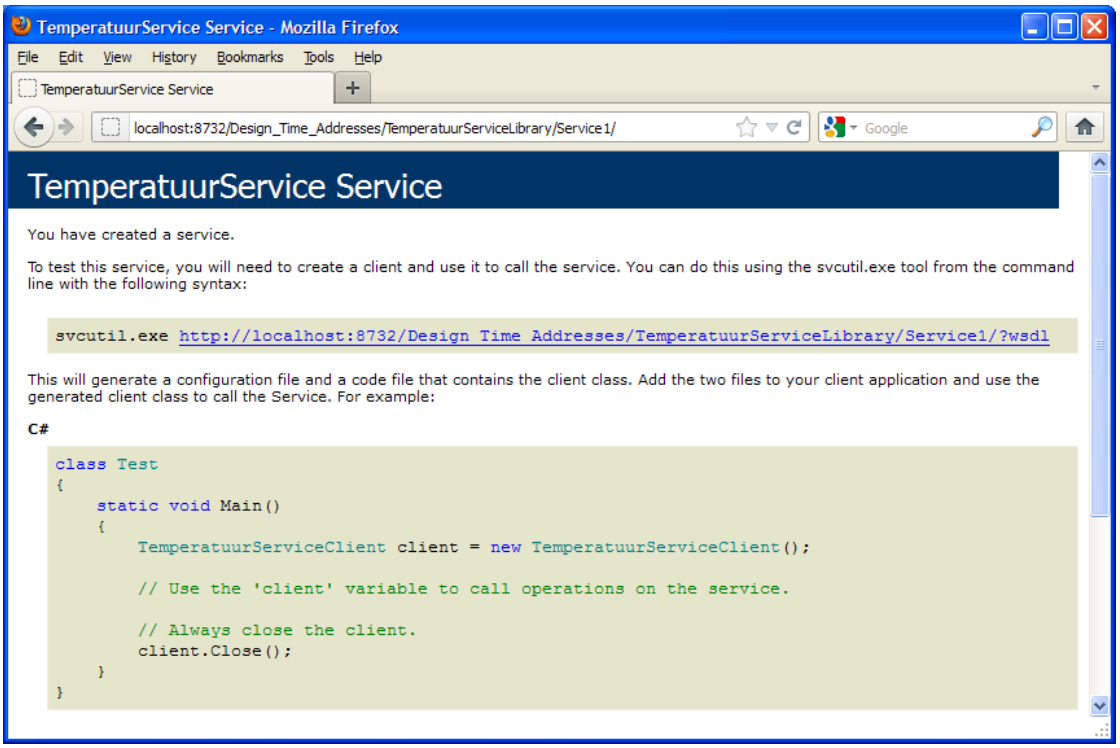
Poort openen onder Windows XP:
1. Klik op Start → Settings → Control Panel.
2. Dubbelklik op Security Center.
3. Klik op Manage security settings for Windows Firewall.
4. Selecteer het Exceptions tabblad.
5. Klik op Add Port, Name: WCF Service, Port: 8732, Type: UDP en klik op OK.
6. Klik op OK om het Windows Firewall venster af te sluiten.

Je kunt nu het programma starten door op het groene Play icoon te klikken. Klik daarna op Start om de service te starten. Het volgende venster wordt dan getoond:



2.4 Service testen met browser

Je kunt nu met behulp van een web browser testen dat de service gestart en bereikbaar is. Open een web browser en ga naar het URL uit het bestand App.config.
http://localhost:8732/Design_Time_Addresses/TemperatuurServiceLibrary/Service1/
Als alles goed gaat krijg je nu het onderstaande venster te zien.



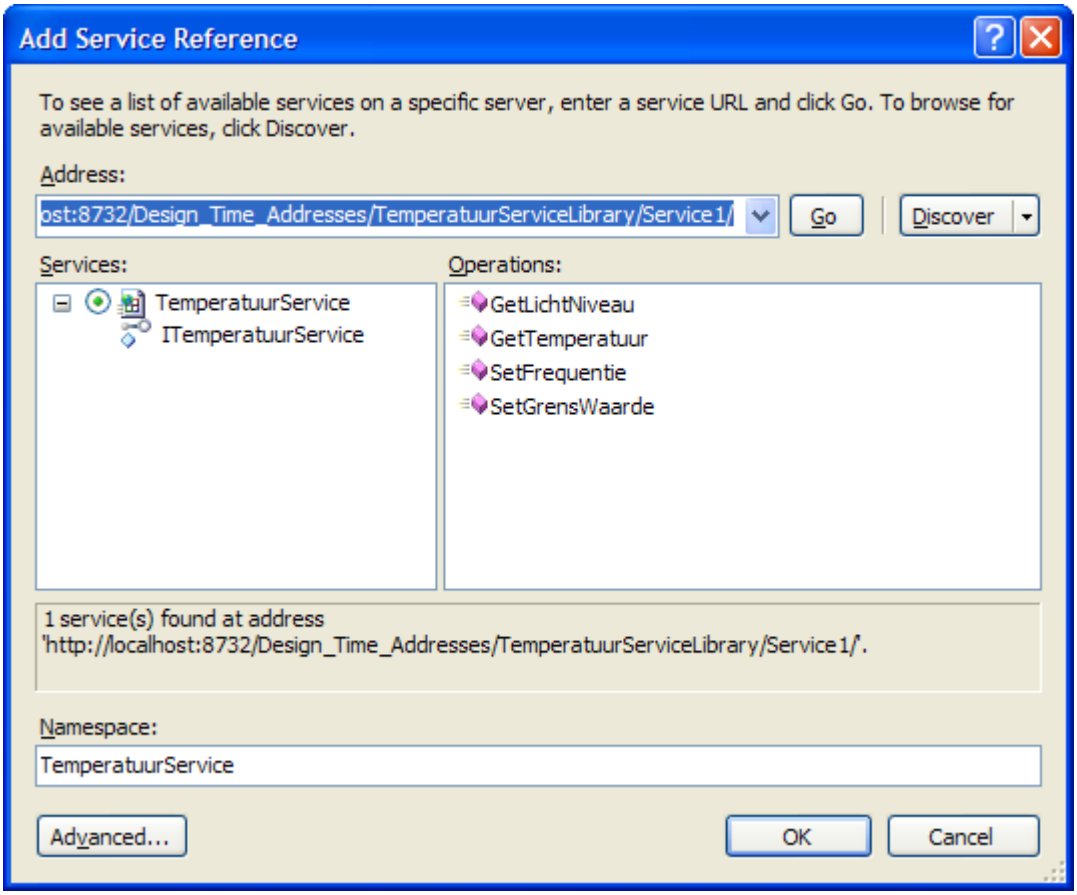
De getoonde pagina bevat een hyperlink naar de WSDL van de webservice. Het beschikbaar stellen van de WSDL is ook een van de taken van de mexHttpBinding (zie: Bindings).

2.5 Client implementatie

File → New → Project.
Selecteer: Visual C# → Windows →Console Application.
Naam: TemperatuurClient.
Solution Name: TemperatuurClient.

Rechts klik op References → Add Reference, selecteer tabblad .NET, kies System.ServiceModel en klik op OK.

Voor de volgende stap moet de Temperatuur Service Host draaien en de service gestart zijn.
Rechts klik op References → Add Service Reference, vul bij Address het URL van de service in: http://localhost:8732/Design_Time_Addresses/TemperatuurServiceLibrary/Service1/ en klik op GO. Dubbelklik op TemperatuurService, selecteer ITemperatuurService, vul bij Namespace TemperatuurService in en klik op OK.



Aan Program.cs moeten de volgende twee using statements worden toegevoegd:
using System.ServiceModel;
using TemperatuurClient.TemperatuurService;

Code van Program.cs:

```
1         static void Main(string[] args)
2         {
3             // Create a proxy object and connect to the service
4             TemperatuurServiceClient proxy = new
  TemperatuurServiceClient();
5
6             float temperatuur = proxy.GetTemperatuur();
7             Console.WriteLine("Huidige temperatuur: {0}", temperatuur);
8
9             string lichtNiveau = proxy.GetLichtNiveau();
10            Console.WriteLine("Huidige lichtNiveau: {0}", lichtNiveau);
11
12            string rv;
13            rv = proxy.SetFrequentie(123);
14            Console.WriteLine("Return waarde SetFrequentie: {0}", rv);
15
16            rv = proxy.SetGrensWaarde(1000);
17            Console.WriteLine("Return waarde SetGrensWaarde: {0}", rv);
18
19            proxy.Close();
20            Console.WriteLine("Press ENTER to finish");
21            Console.ReadLine();
22        }
```

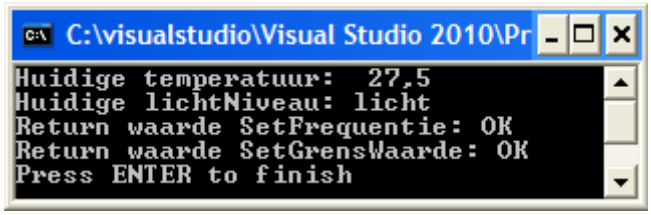
Opmerking: TemperatuurServiceClient is de naam van de lokale proxy klasse.

De App.config van de host en client moeten altijd met elkaar overeenkomen. In de App.config van de client moeten daarom dezelfde twee veranderingen doorgevoerd worden als aan de host zijde, dat wil zeggen wsHttpBinding moet (op drie plaatsen) vervangen worden door ws2007HttpBinding (geen screenshot) en de security moet expliciet uitgezet worden (zie onderstaande screenshot).

Security wijziging in App.config:

```
1 <security mode="None">
2       <transport clientCredentialType="None" />
3       <message clientCredentialType="None" />
4 </security>
```

Voor de volgende stap moet de Temperatuur Service Host draaien en de service gestart zijn.
Je kunt nu het programma starten door op het groene Play icoon te klikken. Je krijgt dan de volgende uitvoer te zien:



3. WS-Discovery

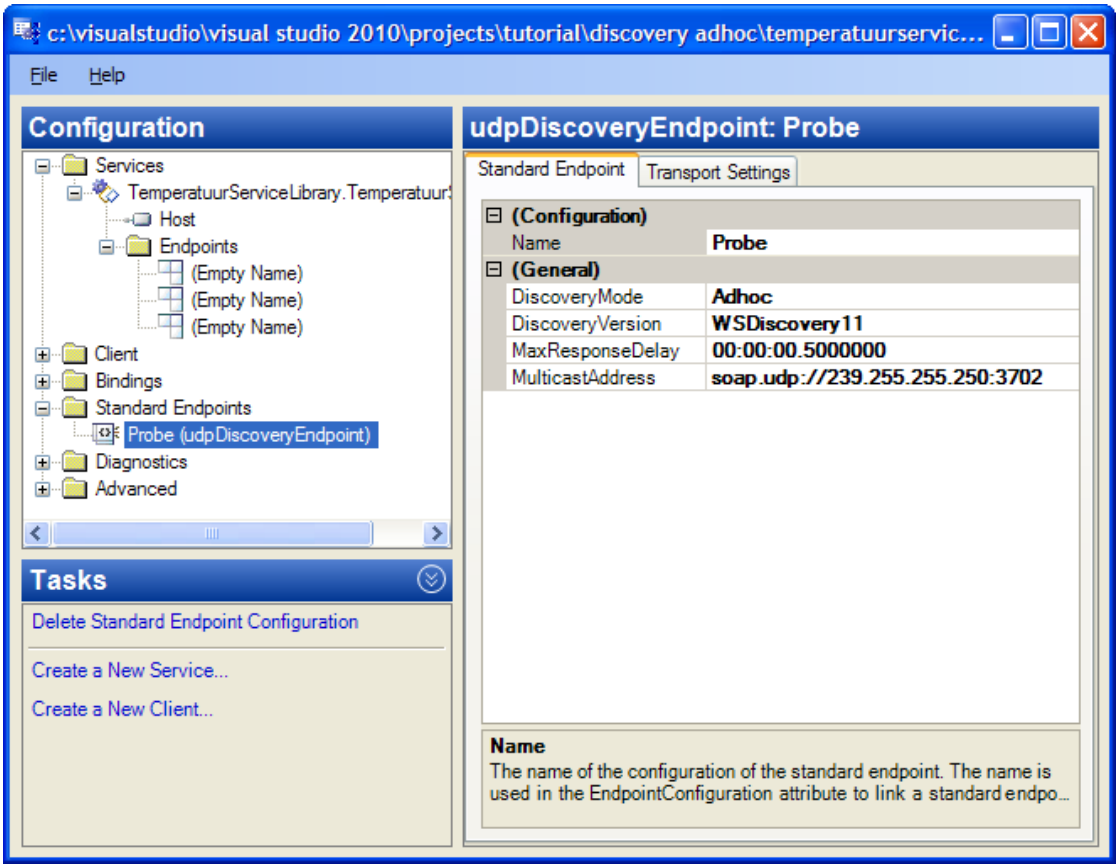
3.1 Host implementatie

Om WS-Discovery toe te voegen aan de TemperatuurServiceHost hoeft alleen het bestand App.config aangepast te worden. Tot nu toe hebben we dit bestand 'met de hand' aangepast, maar nu gaan we het bestand aanpassen met behulp van de WCF Service Configuration Editor. Rechts klik op App.config en kies Edit WCF Configuration. Het kan zijn dat deze optie niet in het menu staat. Open dan eenmalig de editor via: Tools → WCF Service Configuration Editor. Sluit de editor meteen weer af en vanaf nu staat de optie wel in het menu dat getoond wordt als je rechts klikt op App.config.

Endpoint toevoegen voor het ontvangen van Probe berichten:

1. Open Advanced → Service Behaviors → Empty Name in het configuratie venster. Klik op Add in het behavior venster, selecteer serviceDiscovery en klik op Add.
2. Open Services → TemperatuurServiceLibrary.TemperatuurService → Endpoints in het configuratie venster. Rechts klik op Endpoints en kies New Service Endpoint. Klik op Kind, klik op het pijltje naar beneden en selecteer udpDiscoveryEndpoint.
3. Selecteer Standard Endpoints in het configuratie venster. Rechts klik op Standard Endpoints, kies New Standard Endpoint Configuration, selecteer udpDiscoveryEndpoint en klik op OK. Klik op Name in het behavior venster en verander de naam in Probe.
4. Selecteer de in stap 2 gemaakte endpoint. Klik op EndpointConfiguration, klik op het pijltje naar beneden en selecteer Probe.

Vooral de in stap 3 aangemaakte endpoint configuratie (zie screenshot) is interessant.



Endpoint toevoegen voor het versturen van Hello en Bye berichten:

1. Open Advanced → Service Behaviors → Empty Name → serviceDiscovery → Announcement Endpoints in het configuratie venster. Rechts klik op Announcement Endpoints en kies New Client Endpoint. Klik op Kind, klik op het pijltje naar beneden en selecteer udpAnnouncementEndpoint.
2. Open Standard Endpoints in het configuratie venster. Rechts klik op Standard Endpoints, kies New Standard Endpoint Configuration, selecteer udpAnnouncementEndpoint en klik op OK. Klik op Name in het behavior venster en verander de naam in Hello.
3. Selecteer de in stap 1 gemaakte endpoint. Klik op EndpointConfiguration, klik op het pijltje naar beneden en selecteer Hello.

Let op: WS-Discovery maakt gebruik van UDP poortnummer 3702. Deze poort moet nog geopend worden in de Windows Firewall anders kan de TemperatuurServiceHost geen Probe berichten ontvangen (zie: Poort openen onder Windows XP).

3.2 Client implementatie

We gaan nu de client aanpassen zodat deze met behulp van WS-Discovery op zoek gaat naar het adres van de webservice. Maar het bestand App.config bevat nog steeds het adres van de webservice. Om er zeker van te zijn dat de client het via WS-Discovery verkregen adres gebruikt moet je eerst het adres in de App.config veranderen in het volgende ongeldige adres: http://192.0.2.1:8732/Design_Time_Addresses/TemperatuurServiceLibrary/Service1/

Rechts klik op References → Add Reference, selecteer tabblad .NET, kies System.ServiceModel.Discovery en klik op OK.

De rest van de aanpassingen moeten gedaan worden in het bestand Program.cs.

Het volgende using statement moet worden toegevoegd:
using System.ServiceModel.Discovery:

Wijziging in Program.cs:

```
 1  static void Main(string[] args)
 2  {
 3   // Use Discovery to find the TemperatuurService service
 4   DiscoveryClient client = new DiscoveryClient(new UdpDiscoveryEndpoint());
 5   FindCriteria findCriteria = new FindCriteria(typeof(ITemperatuurService));
 6   FindResponse findResponse = client.Find(findCriteria);
 7   client.Close();
 8
 9   EndpointAddress address = findResponse.Endpoints[0].Address;
10   TemperatuurServiceClient proxy = new TemperatuurServiceClient();
11   proxy.Endpoint.Address = address;
```

4. IPv6

Een IPv6 adres is een 128 bits getal, bestaande uit 8 groepjes van 16 bits. Ieder groepje wordt hexadecimaal weergegeven met een dubbelepunt als scheidingsteken. De dubbelepunt wordt echter al gebruikt in een URL om het poort nummer op te geven. Daarom is afgesproken dat je in een URL rechte haken om een IPv6 adres moet zetten.

4.1 Host implementatie

In theorie zou je om de TemperatuurServiceHost via IPv6 te laten communiceren alleen de IP adressen hoeven aan te passen. In praktijk blijkt dit echter tegen te vallen.

De TemperatuurServiceHost bevat op de volgende drie plaatsen een IP adres:

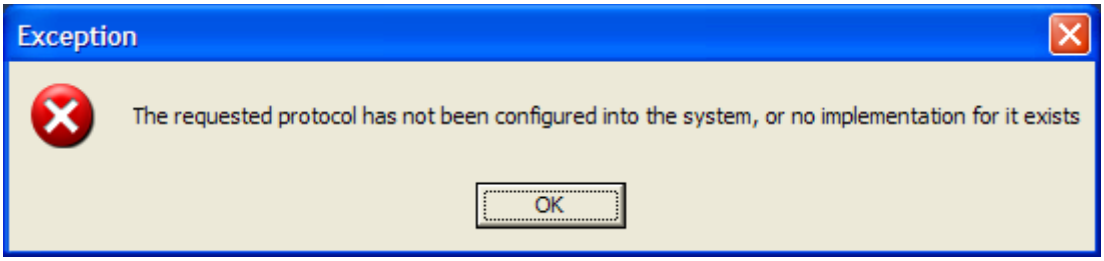
1. Het adres van de webservice,
2. Het multicast IP adres van het udpDiscoveryEndpoint,
3. Het multicast IP adres van het udpAnnouncementEndpoint.

Het eerste IP adres is opgeslagen in het bestand App.config, de laatste twee IP adressen zijn alleen zichtbaar als je het bestand App.config opent in de WCF Service Configuration Editor. De laatste twee IP adressen zijn (tot nu toe) niet opgeslagen omdat beide gelijk zijn aan het door de WS-Discovery standaard voorgeschreven multicast IPv4 adres.

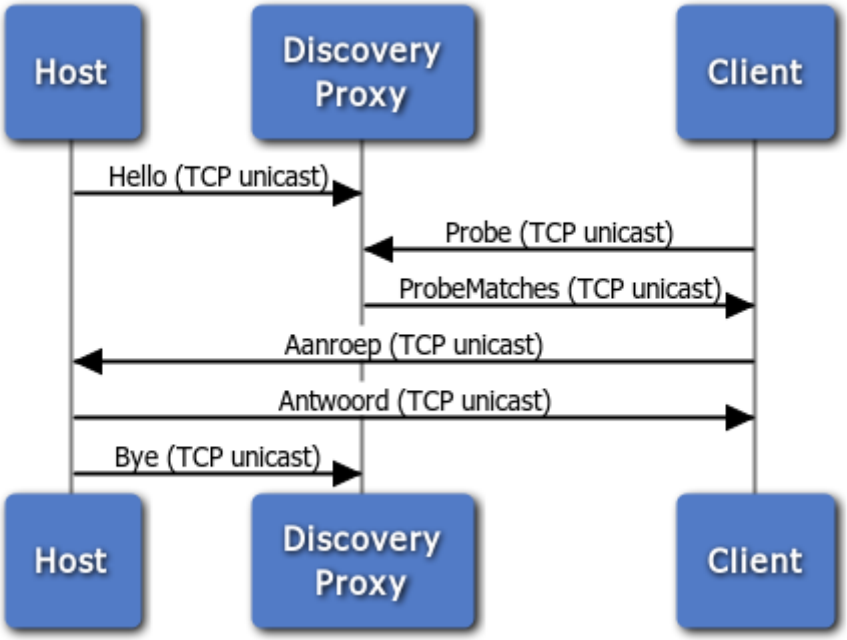
Het adres van de webservice moet veranderd worden in:
http://[IPv6_adres]:8732/Design_Time_Addresses/TemperatuurServiceLibrary/Service1/,
waarbij je *IPv6 adres* moet vervangen door het IPv6 adres van de PC.

Een nog betere oplossing is om een hostname te gebruiken in plaats van een IPv6 adres. De webservice is dan zowel via IPv4 als via IPv6 bereikbaar. Dit werkt echter alleen indien de gebruikte DNS server zowel een IPv4 als een IPv6 adres bevat voor de gebruikte hostname. Op het moment van dit schrijven (februari 2012) bevat de TASS DNS server nog geen IPv6 adressen.

Volgens de WS-Discovery standaard moet indien IPv6 gebruikt wordt het multicast adres FF02::C gebruikt worden. Als ik echter de laatste twee IP adressen vervang door: “soap.udp://[FF02::C]:3702” dan krijg ik runtime, als ik op de Start knop druk, de volgende foutmelding:



Een (tijdelijke) oplossing voor dit probleem is om over te schakelen van Adhoc Discovery naar Managed Mode Discovery. Bij Managed Mode wordt er geen gebruik gemaakt van UDP multicast, maar wordt de Hello en Bye TCP unicast verstuurd naar een Discovery Proxy. Daarnaast hoeft de host niet meer te luisteren naar Probe berichten. Het luisteren naar en beantwoorden van Probe berichten wordt overgenomen door de Discovery Proxy.



De Discovery Proxy is geschreven in de programmeertaal C en is geïnstalleerd op een plugtop. De plugtop maakt gebruik van het Debian GNU/Linux besturingssysteem en is bereikbaar via IP-adres AAAA::1. Let op: De Hello, Probe, ProbeMatches en Bye berichten maken vanaf nu gebruik van TCP. Een eigenschap van TCP is dat een bericht alleen succesvol verstuurd kan worden als er aan de ontvangende zijde iemand luistert. Dat wil zeggen dat eerst de Discovery Proxy gestart moet worden, daarna de Host en als laatste de Client.

Verwijderen van Probe endpoint:

1. Open het bestand App.config in de WCF Service Configuration Editor.
2. Open Services → TemperatuurServiceLibrary.TemperatuurService → Endpoints in het configuratie venster. Selecteer het endpoint waarbij Kind gelijk is aan udpDiscoveryEndpoint. Rechts klik op dit endpoint en kies Delete Endpoint. Klik op OK om te bevestigen.
3. Open Standard Endpoints in het configuratie venster. Rechts klik op Probe (udpDiscoveryEndpoint) en kies Delete Standard Endpoint Configuration. Klik op OK om te bevestigen.

Aanpassen Hello endpoint:

1. Open het bestand App.config in de WCF Service Configuration Editor.

2. Open Standard Endpoints in het configuratie venster. Rechts klik op Hello (udpAnnouncementEndpoint) en kies Delete Standard Endpoint Configuration. Klik op OK om te bevestigen.
3. Open Advanced → Service Behaviors → Empty Name → serviceDiscovery → Announcement Endpoints → Empty Name in het configuratie venster.
   Klik op Kind, klik op het pijltje naar beneden en selecteer announcementEndpoint.
   Klik op Binding, klik op het pijltje naar beneden en selecteer ws2007HttpBinding.
   Klik op BindingConfiguration, klik op het pijltje naar beneden en selecteer NoSecurity.
   Klik op Address en vul het volgende adres in: “http://[AAAA::1]:3702/”.

4.2 Client implementatie

Uiteraard moet ook de client aangepast worden zodat deze de Probe via TCP unicast verstuurd naar de Discovery Provy. Hiervoor moet de code in Program.cs gewijzigd worden.

Wijziging in Program.cs:

```
1         private const string probeAddress = "http://[AAAA::1]:3702";
2
3         static void Main(string[] args)
4         {
5             // Use Discovery to find the TemperatuurService service
6             Uri discoveryProxyUri = new Uri(probeAddress);
7             EndpointAddress discoveryProxyAddress =
8               new EndpointAddress(discoveryProxyUri);
9             WS2007HttpBinding binding = new WS2007HttpBinding();
10            binding.Security.Mode = SecurityMode.None;
11            DiscoveryEndpoint discoveryProxyEndpoint =
12              new DiscoveryEndpoint(binding, discoveryProxyAddress);
13            DiscoveryClient discoveryClient =
14              new DiscoveryClient(discoveryProxyEndpoint);
15
16            FindCriteria findCriteria =
17              new FindCriteria(typeof(ITemperatuurService));
18            FindResponse findResponse = discoveryClient.Find(findCriteria);
19            discoveryClient.Close();
20
21            TemperatuurServiceClient proxy =
22              new TemperatuurServiceClient();
23            proxy.Endpoint.Address = findResponse.Endpoints[0].Address;
```

Je kunt nu de IPv6 aanpassingen testen door eerst de Discovery Proxy te starten, daarna de TemperatuurServiceHost en als laatste op het groene Play icoon te klikken.

A. Wireshark logging

Deze appendix bevat de Wireshark logging van de netwerk communicatie tussen de host en client uit hoofdstuk 3, dat wil zeggen van een op IPv4 gebaseerde webservice met WS-Discovery.

Hello (Service → UDP Multicast):

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://www.w3.org/2005/08/addressing">
- <s:Header>
    <a:Action s:mustUnderstand="1">http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/Hello</a:Action>
    <h:AppSequence InstanceId="1330180494" MessageNumber="1" xmlns:h="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01" />
    <a:MessageID>urn:uuid:d93d09ef-5d15-4323-8d8d-60ac9b7da66e</a:MessageID>
    <a:To s:mustUnderstand="1">urn:docs-oasis-open-org:ws-dd:ns:discovery:2009:01</a:To>
  </s:Header>
- <s:Body>
  - <Hello xmlns="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
    - <a:EndpointReference>
        <a:Address>http://192.168.1.15:8732/Design_Time_Addresses/TemperatuurServiceLibrary/Service1/</a:Address>
      - <Identity xmlns="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity">
          <Dns>localhost</Dns>
        </Identity>
      </a:EndpointReference>
      <d:Types xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01"
        xmlns:dp0="http://tempuri.org/">dp0:ITemperatuurService</d:Types>
      <XAddrs>http://192.168.1.15:8732/Design_Time_Addresses/TemperatuurServiceLibrary/Service1/</XAddrs>
      <MetadataVersion>0</MetadataVersion>
    </Hello>
  </s:Body>
</s:Envelope>
```

Probe (Client → UDP Multicast):

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://www.w3.org/2005/08/addressing">
- <s:Header>
    <a:Action s:mustUnderstand="1">http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/Probe</a:Action>
    <a:MessageID>urn:uuid:70dc4a84-0173-4cbd-9453-b7990ca460bb</a:MessageID>
  - <a:ReplyTo>
      <a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>
    </a:ReplyTo>
    <a:To s:mustUnderstand="1">urn:docs-oasis-open-org:ws-dd:ns:discovery:2009:01</a:To>
  </s:Header>
- <s:Body>
  - <Probe xmlns="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
      <d:Types xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01"
        xmlns:dp0="http://tempuri.org/">dp0:ITemperatuurService</d:Types>
      <Duration xmlns="http://schemas.microsoft.com/ws/2008/06/discovery">PT20S</Duration>
    </Probe>
  </s:Body>
</s:Envelope>
```

ProbeMatch (Service → Client UDP Unicast):

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://www.w3.org/2005/08/addressing">
- <s:Header>
    <a:Action s:mustUnderstand="1">http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/ProbeMatches</a:Action>
    <h:AppSequence InstanceId="1330180494" MessageNumber="3" xmlns:h="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01" />
    <a:RelatesTo>urn:uuid:70dc4a84-0173-4cbd-9453-b7990ca460bb</a:RelatesTo>
    <a:MessageID>urn:uuid:4e4e0cb5-157a-4823-8618-f3a4cee577f7</a:MessageID>
  </s:Header>
- <s:Body>
  - <ProbeMatches xmlns="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01" xmlns:i="http://www.w3.org/2001/XMLSchema-
      instance">
    - <ProbeMatch>
      - <a:EndpointReference>
          <a:Address>http://192.168.1.15:8732/Design_Time_Addresses/TemperatuurServiceLibrary/Service1/</a:Address>
        - <Identity xmlns="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity">
            <Dns>localhost</Dns>
          </Identity>
        </a:EndpointReference>
        <d:Types xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01"
          xmlns:dp0="http://tempuri.org/">dp0:ITemperatuurService</d:Types>
        <XAddrs>http://192.168.1.15:8732/Design_Time_Addresses/TemperatuurServiceLibrary/Service1/</XAddrs>
        <MetadataVersion>0</MetadataVersion>
      </ProbeMatch>
    </ProbeMatches>
  </s:Body>
</s:Envelope>
```

Aanroep HTTP (Client → Service TCP Unicast):

```
POST /Design_Time_Addresses/TemperatuurServiceLibrary/Service1/ HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
Host: 192.168.1.15:8732
Content-Length: 580
Expect: 100-continue
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

HTTP Continue (Service → Client TCP Unicast):

```
HTTP/1.1 100 Continue
```

Aanroep SOAP (Client → Service TCP Unicast):

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://www.w3.org/2005/08/addressing">
- <s:Header>
    <a:Action s:mustUnderstand="1">http://tempuri.org/ITemperatuurService/GetTemperatuur</a:Action>
    <a:MessageID>urn:uuid:180fda1e-43cf-419f-befe-25656e437322</a:MessageID>
  - <a:ReplyTo>
      <a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>
    </a:ReplyTo>
    <a:To s:mustUnderstand="1">http://192.168.1.15:8732/Design_Time_Addresses/TemperatuurServiceLibrary/Service1/</a:To>
  </s:Header>
- <s:Body>
    <GetTemperatuur xmlns="http://tempuri.org/" />
  </s:Body>
</s:Envelope>
```

Antwoord (Service → Client TCP Unicast):

```
HTTP/1.1 200 OK
Content-Length: 459
Content-Type: application/soap+xml; charset=utf-8
Server: Microsoft-HTTPAPI/1.0
Date: Sat, 25 Feb 2012 13:35:30 GMT
```

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://www.w3.org/2005/08/addressing">
- <s:Header>
    <a:Action s:mustUnderstand="1">http://tempuri.org/ITemperatuurService/GetTemperatuurResponse</a:Action>
    <a:RelatesTo>urn:uuid:180fda1e-43cf-419f-befe-25656e437322</a:RelatesTo>
  </s:Header>
- <s:Body>
  - <GetTemperatuurResponse xmlns="http://tempuri.org/">
      <GetTemperatuurResult>18</GetTemperatuurResult>
    </GetTemperatuurResponse>
  </s:Body>
</s:Envelope>
```

Opmerking: Antwoord is één bericht.

Bye (Client → UDP Multicast):

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://www.w3.org/2005/08/addressing">
- <s:Header>
    <a:Action s:mustUnderstand="1">http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/Bye</a:Action>
    <h:AppSequence InstanceId="1330180494" MessageNumber="4" xmlns:h="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01" />
    <a:MessageID>urn:uuid:0bbadd89-739b-4895-ba56-822b032aa76e</a:MessageID>
    <a:To s:mustUnderstand="1">urn:docs-oasis-open-org:ws-dd:ns:discovery:2009:01</a:To>
  </s:Header>
- <s:Body>
  - <Bye xmlns="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
    - <a:EndpointReference>
        <a:Address>http://192.168.1.15:8732/Design_Time_Addresses/TemperatuurServiceLibrary/Service1/</a:Address>
      - <Identity xmlns="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity">
          <Dns>localhost</Dns>
        </Identity>
      </a:EndpointReference>
      <d:Types xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01"
        xmlns:dp0="http://tempuri.org/">dp0:ITemperatuurService</d:Types>
      <XAddrs>http://192.168.1.15:8732/Design_Time_Addresses/TemperatuurServiceLibrary/Service1/</XAddrs>
      <MetadataVersion>0</MetadataVersion>
    </Bye>
  </s:Body>
</s:Envelope>
```