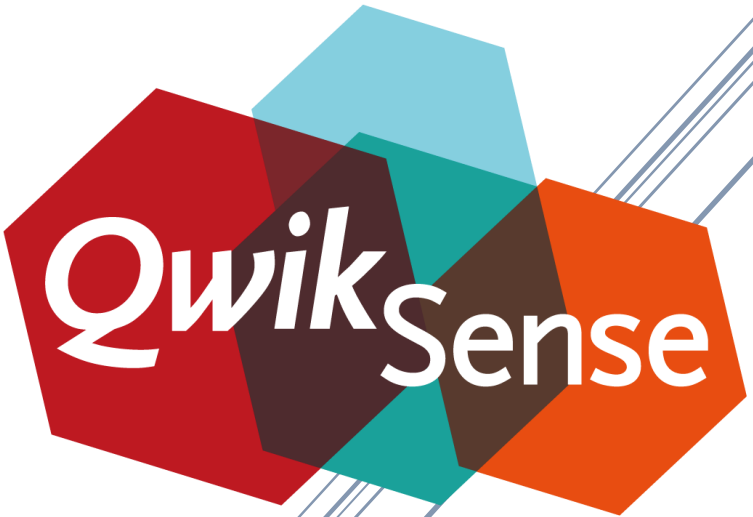
AFSTUDEERSCRIPTIE

Onderzoek naar en ontwikkeling van een betaalbaar klimaatsensornetwerk voor grootschalige toepassing.



Auteur: Felix Mann (1549433)

Eerste examinator: Jan Zuurbier

Samenvatting

In deze scriptie beschrijf ik het onderzoek en project dat ik bij QwikSense heb uitgevoerd, met als doel het ontwikkelen van een goedkoper alternatief voor het huidige meetsysteem dat QwikSense gebruikt om het werkklimaat van bedrijven te meten. Naast de hoge prijs van het bestaande systeem speelde ook de gebruikersvriendelijkheid een rol in de beslissing om op zoek te gaan naar een alternatief. De hoofdvraag is hierbij als volgt:

Op welke manier is het huidige product dat QwikSense gebruikt voor haar werkzaamheden te innoveren en goedkoper te produceren terwijl de flexibiliteit van de oplossing vergroot wordt?

Er moest een betaalbaar alternatief komen voor de huidige oplossing en het liefst een oplossing waarbij QwikSense niet meer afhankelijk is van een derde partij voor het leveren en onderhoud van de apparatuur en infrastructuur. Ik ben begonnen met een onderzoek naar de huidige oplossing en alternatieven die op de markt beschikbaar zijn. Uit dit onderzoek bleek dat veel commerciële oplossingen te duur zijn om op grote schaal te implementeren. Bovendien zijn veel commerciële oplossingen vaak gebonden aan een service die de sensordata bij de eindgebruiker brengt, een dienst die QwikSense ook wil verlenen.

Met de resultaten van het onderzoek ben ik gaan prototypen en testen om tot een werkend sensornetwerk te komen. Op basis van een ATmega328-P microcontroller en een XBee-communicatiemodule in combinatie met diverse sensoren kwam een prototype sensornode tot stand. De gateway bestaat uit een Raspberry Pi en een XBee-module voor communicatie met de sensornodes.

QwikSense heeft het ontwikkelen van het dashboard, de web-interface waarin de eindgebruiker de verzamelde en verwerkte data kan inzien, uitbesteed aan het bedrijf StarterSquad. Samen hebben we de communicatie tussen de gateway en de webservice tot stand gebracht.

Het resultaat is een modulaire sensornode voor minder dan de helft van de prijs van de bestaande oplossing waar bovendien later nog sensoren bij te zetten en uit te verwijderen zijn. In alle stappen van de ontwikkeling is er gelet op efficiëntie en energiezuinigheid van het product zodat het ook op bijvoorbeeld een zonnepaneel en een accu kan functioneren.

Keywords: Internet of Things, Sensor, Energiebesparing, Werkcomfort, Microcontroller

Begrippenlijst

Communicatie-module – Een chip of module die data draadloos naar een soortgelijke chip of module kan versturen. Door deze module toe te voegen aan een (bestaand) product of microprocessor kan men (in sommige gevallen zelfs zonder aanpassing in code) draadloos communiceren.

Dashboard – Een grafische interface die een overzicht toont van een verzameling gegevens.

Gateway – Een embedded computer die metingen van sensornodes ontvangt en verstuurt naar een externe webservice.

GPIO – General Purpose Input and Output, de benaming voor een of meerdere pinnen waarvan de functionaliteit tijdens het draaien van een programma op gebruikersniveau kan worden bepaald. Deze pinnen kunnen vaak uitgelezen worden en er kan naar geschreven worden om te communiceren met een extern systeem.

IEEE 802.15.4 – Een standaard voor draadloze PAN-netwerken met lage bandbreedte. Deze standaard omschrijft hoe signalen draadloos verstuurd en ontvangen kunnen worden en hoe meerdere transcievers kunnen communiceren zonder elkaar in de weg te zitten.

(the) Internet of Things – Het concept dat alledaagse voorwerpen direct verbonden zijn met het internet en informatie uit kunnen wisselen. Er wordt vaak geprobeerd om producten toegevoegde waarde te geven door ze te koppelen aan het internet. Denk aan bijvoorbeeld thermostaten, digitale fotolijstjes met WiFi-functionaliteit, een weegschaal die online gewichtsdata opslaat.

Library – Een verzameling functies die door programmeurs te hergebruiken zijn. Hierdoor hoeven deze functies niet opnieuw geschreven te worden.

Mesh Network – Een netwerk waarbij modules samenwerken om het bereik te vergroten. In tegenstelling tot een netwerk met een geplande infrastructuur zijn mesh-netwerken in staat om veranderingen in de omgeving of samenstelling van het netwerk op te vangen zonder herconfiguratie van het netwerk.

Microchip – Een elektrisch circuit met een specifieke functie, maar dan geïntegreerd in de vorm van een chip. Een microchip is vaak gebaseerd op analoge circuits.

Microcontroller – Een kleine computer op een chip, bestaande uit een processorkern, geheugen en programmeerbare inputs en outputs. Deze apparaten worden vooral gebruikt in producten die op batterijen werken. Door een microcontroller in slaapstand zetten daalt het stroomverbruik bij sommige microcontrollers tot slechts enkele picowatts.

.NET Object Remoting – Een techniek van Microsoft die verwerkt is in het .NET Framework en een applicatie in staat stelt een object aan een externe assembly bloot te stellen. De client-assembly kan zich op dezelfde machine bevinden of gebruik

maken van een TCP/IP-verbinding. Hoewel de techniek als 'legacy' beschouwd wordt is het wel een techniek die volledig geïmplementeerd is in Mono. Dit in tegenstelling tot *Windows Communication Foundation*.

PAN – Personal Area Network, een netwerk dat gebruikt wordt voor communicatie tussen apparaten waarbij de apparaten elkaar diensten kunnen verlenen die normaal via een kabel zouden verlopen. Toepassingen variëren van het delen van een internetverbinding tot een mesh-netwerk van sensoren.

Polling rate – Interval waarmee er naar data gevraagd wordt aan een extern systeem.

Sensor – Een microchip die in staat is een omgevingsvariabele (zoals bijvoorbeeld temperatuur of luchtdruk) om te zetten in een analoog of digitaal signaal.

Sensornode – Een apparaat dat een of meerdere sensoren uitleest en deze data verzamelt en doorstuurt naar een centrale locatie waar de data verwerkt kan worden.

Transciever – Een elektronische module die zowel data kan zenden als ontvangen.

WebCore – Een project in ontwikkeling zodat een programmeur met relatief weinig moeite een webservice/website dynamisch kan laten genereren met zo min mogelijk code. WebCore is een product dat ik in mijn vrije tijd ontwikkel. Voor meer info zie: http://www.thisiswhytheinternetexists.com/products/webcore

Inhoudsopgave

1 Inleiding

Jaarlijks gooien bedrijven miljoenen euro's weg. Veel bedrijven kijken niet of nauwelijks om naar het binnenklimaat van hun gebouw. Zonder er erg in te hebben ontstaat vaak een ongezond werkklimaat. Dit resulteert in een verhoogd ziekteverzuim. Zo komt het vaak voor dat werkplaatsen met een constante temperatuur worden verwarmd of gekoeld zonder dat dit wordt aangepast aan de actuele bezettingsgraad in dat deel van het gebouw. Als gevolg hiervan wordt er veel energie verspild.

Het is niet nieuw om na te denken over energiebesparing en het verlagen van de stookkosten, maar bedrijven zijn zich vaak nog niet bewust van het besparingspotentieel dat er te behalen is op het gebied van comfort. Door de informatie over de temperatuur te verrijken met informatie over CO_2, geluid, licht en luchtvochtigheid ontstaat er een overzicht om het comfort van de werkplaats te meten. Als deze informatie omgezet kan worden in een implementeerbaar advies ligt hier een kans voor bedrijven om kosten te besparen op ziekteverzuim én de stookkosten waarbij er niet wordt bespaard op comfort, maar juist dóór comfort.

1.1 Over QwikSense

QwikSense is een start-up die op het moment van schrijven in Amersfoort gevestigd is. Zij houdt zich bezig met het onderzoeken van de correlaties tussen een aantal omgevingsfactoren op de werkvloer van bedrijven. Wanneer een bedrijf door QwikSense gemonitord wordt kunnen er uit de verzamelde data correlaties gelegd worden en kunnen er conclusies getrokken worden. Deze conclusies worden omgezet in een advies voor het creëren van energie-efficiënte en gezonde werkomgeving.

Het bestaande concept dat QwikSense aanbiedt, bestaat uit een aantal sensoren die verspreid over de werkvloer en op vijf parameters metingen verrichten, het betreft: CO_2-niveau, geluidsniveau, licht, temperatuur en luchtvochtigheid.

Figuur 1 De parameters waar QwikSense met de huidige oplossing op meet

QwikSense plaatst sensornodes bij een bedrijf en onderzoekt de klimaatbeheersingsapparatuur waarover een bedrijf beschikt. De sensoren houden over langere periode bij hoe het klimaat op de werkvloer is. Momenteel kan QwikSense alleen op basis van deze gegevens een bedrijf adviseren om de apparatuur efficiënter in te stellen. Dit kan resulteren in een aanzienlijke stroombesparing maar ook in een vermindering van ziekteverzuim. De kracht van het advies van QwikSense ligt vooral in het kijken naar een langere periode en het

afleiden van verbanden tussen omstandigheden op de werkvloer en de afstelling van de klimaatbeheersingsapparatuur.

In een verder stadium moet een gebouwbeheerder zelf inzicht krijgen in de huidige situatie en op basis hiervan aanpassingen laten uitvoeren door de persoon/het bedrijf dat de controle over het klimaatbeheersingssysteem heeft. Voor het gebruik van de sensoren en het advies en inzicht dat QwikSense een bedrijf verschaft betaalt een bedrijf maandelijks een bedrag. Hiervoor krijgt het bedrijf geld terug door energiebesparing en door een optimaler werkklimaat daalt het ziekteverzuim. Bedrijven zijn nog niet overtuigd van de besparing die ze op lange termijn kunnen halen met advies van QwikSense over het (juiste) gebruik van de faciliteiten binnen een gebouw. Door middel van een duidelijke en inzichtelijke gebruikersinterface wil QwikSense hier verandering in brengen. Ook moet de installatie en het beheer laagdrempelig zijn. Een leek moet (weliswaar met de juiste instructies) al het onderhoud moeten kunnen uitvoeren.

1.2 Mijn rol in de stage

De bedoeling is dat ik voor QwikSense een eigen oplossing ontwikkel die sensormetingen doet, verwerkt en opstuurt naar een webservice. Deze gegevens kunnen vanaf de webservice weer in een mooie gebruikersinterface gegoten worden en op een computer of mobiel apparaat getoond worden. Ik vergelijk hardware (microcontrollers, sensoren, batterijen, zonnepanelen, etc.) en zoek naar een flexibele en betaalbare oplossing die goed te onderhouden is. Verder ontwikkel ik de firmware die de sensormetingen verwerkt en opstuurt naar een gateway. Ook de gateway en de bijbehorende software schrijf ik. In overleg met Rino, Paul en Iwein realiseer ik een eigen sensornetwerk voor QwikSense dat ingezet kan worden als vervanging van het bestaande systeem.

1.3 Reden tot dit onderzoek

QwikSense heeft nog geen klanten die vaste metingen verrichten. Om als bedrijf te groeien wil QwikSense binnen een bedrijf vaste sensoren plaatsen. Zo kan de situatie in de gaten gehouden worden en kunnen er over langere periode adviezen worden gegeven aan een bedrijf. Tijdens mijn stage is QwikSense bezig met het ontwikkelen van een dashboard dat zowel voor een beheerder van een gebouw als voor een eindgebruiker geschikt is om snel de situatie op de werkvloer in kaart te kunnen brengen en direct de vinger op de zere plek te leggen. Mijn taak is het ontwikkelen van nieuwe apparatuur die goedkoper is dan de apparatuur van MakeMoreSense en bovendien modulair is in opzet waardoor kosten bespaard kunnen worden of (naar wens van de klant) de sensoren preciezer.

1.4 Doelstelling, Onderzoeksvraag en deelvragen

Deze scriptie dient als advies voor QwikSense dat gebruikt kan worden om een alternatief voor het huidige meetsysteem van MakeMoreSense te produceren. Hierbij wordt er gekeken naar de potentiele verbeterpunten t.o.v. de bestaande oplossing. Dit doe ik door de onderstaande onderzoeksvraag te beantwoorden:

Op welke manier is het huidige product dat QwikSense gebruikt voor haar werkzaamheden te innoveren en goedkoper te produceren terwijl de flexibiliteit van de oplossing vergroot wordt?

Deze vraag behandel ik in verschillende subvragen:

1. Met welk probleem houdt QwikSense zich bezig en waarom is het een probleem?
2. Hoe werkt de bestaande oplossing?
   a. Waaruit bestaat de sensornode?
   b. Waaruit bestaat de gateway?
   c. Hoe werken de sensornode, gateway en webservice samen om meetdata te verzamelen en weer te geven in een dashboard?
   d. Welke functionaliteit ontbreekt er in de huidige oplossing?
   e. Welke functionaliteit werkt goed en kan ik meenemen in het samenstellen van een eigen sensoren?
3. Welke (kant-en-klare) alternatieven zijn er verkrijgbaar op de markt en in hoeverre zijn deze bruikbaar?
4. Hoe kan ik zelf een sensornode bouwen?
   a. Welke hardware heb ik nodig om een eigen sensor samen te stellen?
   b. Welke technieken kan ik gebruiken om er voor te zorgen dat er weinig tot geen onderhoud aan de sensornodes plaats hoeft te vinden?
   c. Op welke manieren is een sensornode modulair te maken? (Is er een oplossing die er voor zorgt dat een klant zelf kan kiezen wat er gemeten wordt en met welke precisie?)
   d. Op welke manieren kan het aansluiten van een sensornode vergemakkelijkt worden?
5. Welke hardware heb ik nodig om een eigen gateway samen te stellen?
6. Hoe goed presteert de nieuwe sensor?
   a. Is er daadwerkelijk verbetering t.o.v. de sensornode uit de bestaande oplossing?

1.5 Structuur van dit document

Dit hoofdstuk gaf informatie over de huidige situatie en beantwoordde onderzoeksvraag *1*. Hoofdstuk 2 behandelt de bestaande oplossing die QwikSense gebruikt voor pilotprojecten. Hiervan wordt bekeken waarom de oplossing niet voldoet aan de wensen van QwikSense. Ook wordt er gekeken naar zaken die wel goed geregeld zijn binnen de bestaande oplossing. Hoofdstuk 3 behandelt de alternatieven voor de bestaande oplossing die momenteel op de markt zijn. In hoofdstuk 4 wordt het proces van het bouwen van de sensornode behandeld en een deel van de functionaliteit in de software van de sensornode wordt toegelicht. Ook wordt de sensornode vergeleken met de bestaande sensornode. In hoofdstuk 5 wordt de bouw van de gateway en de software van de gateway behandeld. Hoofdstuk 6 bevat de conclusies en aanbevelingen om verder te gaan met de ontwikkeling van dit project.

2 Huidige oplossing

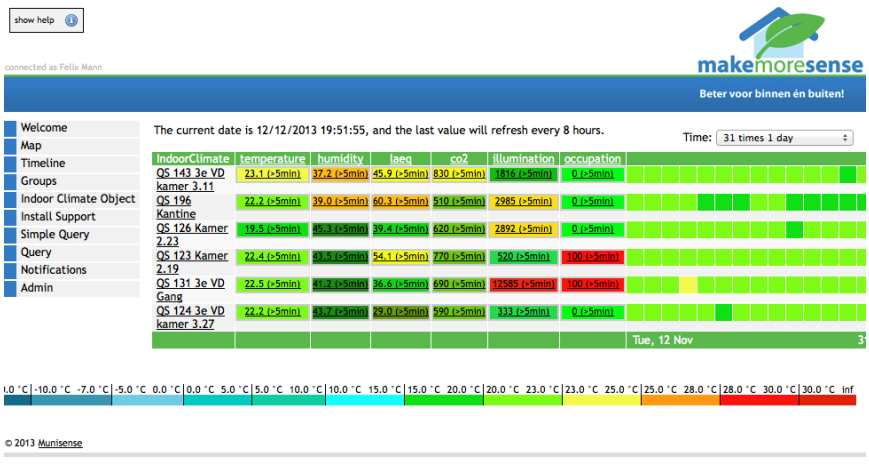
In het vorige hoofdstuk werden de activiteiten van QwikSense, het onderwerp van dit onderzoek en de bijbehorende deelvragen behandeld. In dit hoofdstuk komen de samenstelling van de oplossing die QwikSense momenteel gebruikt (2.1) en wat de problemen zijn met de deze oplossing (2.2) aan bod.

2.1 Waaruit bestaat de bestaande oplossing?

De oplossing die momenteel gebruikt wordt bestaat uit twee componenten: de gateway en de sensornodes. De uitgebreide specificaties en een klein onderzoek naar de werking van de sensornodes is bijgesloten als bijlage 3. Zowel de sensornodes als de gateway worden geproduceerd door het bedrijf MuniSense. MakeMoreSense verkoopt de oplossing door.

De gateway draait op een AMD Geode LX800 processor die gebruik maakt van de x86-architectuur en het operating system is Voyage Linux. Omdat de gateway verbonden is met het internet is het noodzakelijk dat er een LAN-verbinding is of er moet een SIM-kaart geplaatst worden zodat de gateway te allen tijde verbonden is met het internet. Een gateway met 3G-functionaliteit kost €695,- inclusief BTW en een gateway met alleen een LAN-verbinding kost €495,- inclusief BTW.

De sensornode van MakeMoreSense is een kastje dat vijf typen metingen doet: hoeveelheid licht, geluidsniveau, CO_2-niveau, luchtvochtigheidsgraad en temperatuur. Net als in de Gateway is er in de sensornode een Zigbee-module aanwezig. Deze module verzorgt de communicatie tussen verschillende sensornodes en de Gateway. Verder bevat de sensornode een meerkleurige LED waarmee waarschuwingen kan worden gegeven als meetwaarden boven een (door de eindgebruiker) ingestelde norm komen. Ook is een kleine speaker aanwezig die een akoestisch alarm kan geven wanneer een bepaalde meetwaarde overschreden wordt. Een sensor met alleen sensoren voor temperatuur en luchtvochtigheid kost €195,- inclusief BTW. Een sensornode met extra sensoren (CO_2, geluid en licht) kost €395,- inclusief BTW. Deze sensor kan door het hoge stroomverbruik van de CO_2-sensor niet (lang) op batterijstroom werken. Deze CO_2-sensor heeft een opwarmtijd van drie minuten (Futurlec, 2013) en verbruikt gemiddeld 50mA. Om deze sensor gedurende 24 uur te kunnen gebruiken is er een accu van 1200mAh nodig.

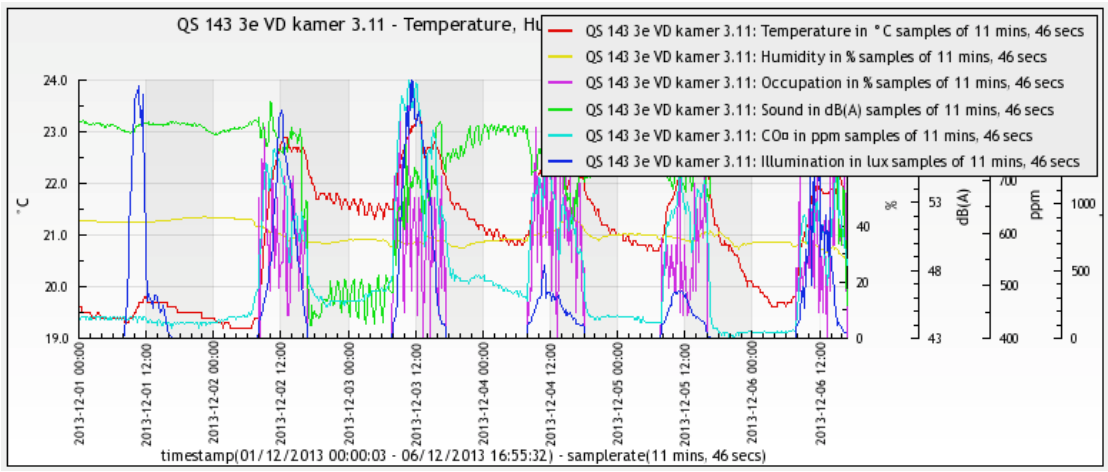


Figuur 2: Het dashboard van MakeMoreSense

De sensornodes sturen metingen draadloos door naar de gateway. De gateway stuurt de metingen van alle sensoren door naar de webservice van MakeMoreSense. Vanaf de webservice kan de data in een web-interface bekeken worden.

In de presentatie van de data die MakeMoreSense levert (zie **Error! Reference source not found.**) lijkt het alleen te gaan om het visualiseren van cijfers. Er wordt gebruik gemaakt van kleuren om aan te geven of een waarde te hoog/laag of goed is. De interface voelt in het gebruik aan als een verouderde website. Functies zitten op een contra-intuïtieve locatie en lijken op sommige plekken helemaal geen functie te hebben. Dit zijn zaken waar de gebruiker gaandeweg wel mee zou kunnen leren omgaan maar dit bevordert op geen enkele wijze het gebruik van het product (namelijk een dashboard dat inzicht geeft in de actuele situatie).

De grafiek in Figuur 3 toont de data van een recente meting. Dit wordt gerenderd als een afbeelding en vervolgens naar de browser van de gebruiker gestuurd. Dit heeft als gevolg dat bijvoorbeeld de informatie in de grafiek rechtsboven niet zichtbaar is. Met de technieken die tegenwoordig bestaan is het mogelijk om deze informatie een stuk duidelijker te presenteren.



Figuur 3: De grafiek die MakeMoreSense gebruikt om meetdata grafisch weer te geven

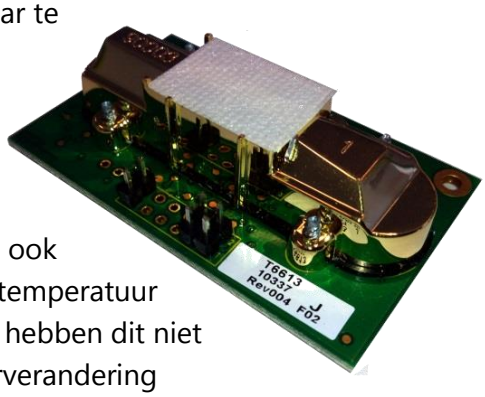
2.2 Waarom voldoet de bestaande oplossing niet?

QwikSense wil duidelijkheid brengen in de brei van gegevens en er voor zorgen dat bedrijven actie kunnen ondernemen als de werknemers of het gebouw niet optimaal functioneren. Bij grote bedrijven (waar een dito hoeveelheid sensoren geplaatst wordt) is het dus nodig dat er een grote investering wordt gedaan. QwikSense wil deze investering wel doen maar beschikt niet over het benodigde kapitaal om dit te realiseren. Een tussenoplossing zou zijn om een deel van de kosten van een sensor in rekening te brengen bij de klant en/of een deel van de kosten te verwerken in abonnementskosten die QwikSense aan haar klanten berekent.

Een bijkomend probleem bij de bestaande oplossingen is dat ze niet voldoende flexibiliteit bieden. Wanneer een klant van QwikSense een sensor voor fijnstof zou willen hebben of een sensor op zonne-energie zou willen laten werken is dit met veel oplossingen niet standaard mogelijk. In een aantal gevallen is het noodzakelijk om

een andere sensornode aan te schaffen. Door een eigen sensor te ontwikkelen met flexibiliteit in het achterhoofd kan QwikSense sensoren leveren die zowel modulair als betaalbaar zijn. Veel sensoren zouden achteraf in een sensornode bijgeplaatst kunnen worden. Kortom: *we willen een sensornode kunnen up- en downgraden*.

QwikSense is met de huidige oplossing afhankelijk van de webservice van MakeMoreSense. De geleverde apparatuur maakt gebruik van de netwerkinfrastructuur van MakeMoreSense om de data beschikbaar te stellen d.m.v. die webservice.



2.3 Wat is er goed aan de bestaande oplossing?

De sensornode zelf heeft sensoren die het benodigde bereik zonder problemen kunnen meten en bijvoorbeeld de CO_2-sensor heeft een zeer groot bereik en kalibreert zichzelf om de metingen ook over langere tijd consistent te houden. Ook meet deze sensor de temperatuur en compenseert de gemeten waarde hiermee. Veel CO_2-sensoren hebben dit niet en kunnen dus na verloop van tijd of bij een te grote temperatuurverandering foutieve meetresultaten geven. Dit vormt voor piekmetingen geen probleem maar voor metingen met een kleiner verschil (zoals in een kantooromgeving) is dit niet praktisch.

Wat betreft gebruikersgemak is er door de oorspronkelijke ontwikkelaar van de hard- en software wel nagedacht over wát er in te stellen moet zijn in het webportaal, maar niet hóe. Het voordeel van de manier die ze hanteren is dat er niet ter plekke met behulp van een laptop zaken geprogrammeerd hoeven te worden. Zaken als de alarmdrempel kunnen zo aangepast worden zonder dat een techneut ter plekke aanwezig hoeft te zijn. Helaas is het bijgeleverde webportaal niet overzichtelijk en daardoor niet bruikbaar voor de eindgebruiker. Bovendien heeft MakeMoreSense er voor gekozen een aantal zaken niet aan de eindgebruiker (in dit geval QwikSense) uit handen te geven, waaronder het toevoegen en verwijderen van sensoren, het instellen van de samplerate van de sensoren, en de mogelijkheid om de instellingen van de gateway in te zien en te wijzigen.

Een oplossing die met weinig moeite te installeren is maar aanzienlijk minder kost terwijl er meer vrijheid wordt geboden m.b.t. configuratie van de gateway en de sensornodes is het streven van dit onderzoek.

Dit hoofdstuk behandelde de bestaande oplossing die QwikSense gebruikt voor pilotprojecten. Er is gekeken naar de sterke en zwakke plekken in de oplossing van MakeMoreSense. De sensornodes en de gateway zijn duur en QwikSense heeft te weinig beheer over de gateways om optimaal te kunnen meten. Bovendien kosten de sensoren en de gateway veel bij het aanschaffen maar zijn er ook nog maandelijkse kosten verbonden aan het gebruik van de sensoren. De sensoren zijn niet volledig zelf samen te stellen en bieden in veel gevallen te veel of te weinig informatie. Het komende hoofdstuk behandelt de alternatieven voor de huidige oplossing.

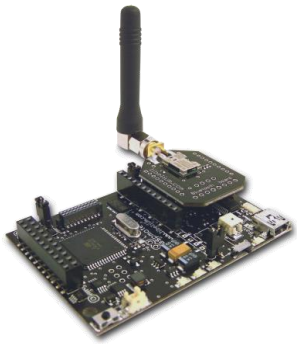
3 (Commerciële) alternatieven voor de bestaande oplossing

In het vorige hoofdstuk is de huidige oplossing die QwikSense gebruikt behandeld. Hierbij is aan bod gekomen waar de huidige oplossing uit bestaat en wat er goed is en wat er mis is met deze oplossing. In het komende hoofdstuk behandel ik de belangrijkste alternatieven voor het bestaande meetsysteem, te weten de WaspMote van Libelium en het zelf bouwen van een meetsysteem.

Om temperatuur te meten zijn er een grote hoeveelheid producten op de markt. Tegenwoordig komt de kwaliteit van de buiten- en binnenlucht steeds vaker in het nieuws. Zowel op commercieel als 'open source'-vlak wordt er gekeken naar oplossingen om mensen meetgegevens te verschaffen tegen een zo laag mogelijke prijs. Het doel is daarbij vaak om een centrale database te maken met meetgegevens. Iedereen kan meedoen om zo een kaart van de wereld te maken met meetgegevens die voor iedereen in te zien zijn.

3.1 WaspMote

De WaspMote (Figuur 4) is een soort *prototyping*- sensornode dat ook ingezet kan worden als permanente oplossing. Libelium (de fabrikant) levert zowel de gateways en sensornodes als de sensoren zelf.



Figuur 4: Een foto van de WaspMote

Zoals veel producten in deze markt is de WaspMote gebaseerd op microcontrollers van Atmel en wordt de Arduino-software gebruikt om code voor deze microcontrollers te ontwikkelen. Voor ontwikkelaars die al eens met een 'normale' Arduino gewerkt hebben kan dit een groot voordeel zijn, de bekende tools kunnen gebruikt worden en in veel gevallen kan eigen code zonder problemen gebruikt worden op deze alternatieve Arduino's. Bovendien biedt Libelium een aantal verschillende modules aan voor communicatie waaronder 868MHz/900MHz, GPRS, 3G, Ethernet, WiFi, Bluetooth, NFC en RFID. Al deze radios worden standaard ondersteund en met veel van deze radios is het mogelijk om *Over the Air*-updates uit te voeren. Bij een OTA-update wordt de microcontroller in een sensornode voorzien van nieuwe code zonder dat deze uit de sensornode gehaald hoeft te worden. Bovendien kunnen er met deze techniek meerdere sensornodes tegelijk geüpdatet worden. (Libelium, 2013)

Libelium Sensor voor Temperatuur	**€**	**2,50**
Libelium Sensor voor Luchtvochtigheid	€	20,00
Libelium Sensor voor Lichtsterkte	€	2,50
Libelium Sensor voor Geluid	€	20,00
Libelium Sensor voor CO2	€	50,00
Libelium Smart Cities board	€	120,00
Libelium WaspMote ZB	€	120,00
Totaal	**€**	**335,00**

Tabel 1: Prijs van de onderdelen die benodigd zijn om een WaspMote sensor te bouwen.

Nog een voordeel aan de WaspMote is dat het aantal sensoren dat beschikbaar is groot is. Libelium levert voor de sensoren die beschikbaar zijn kant en klare libraries. Daarnaast zijn bestaande sensoren ook eenvoudig aan te sluiten omdat het bordje beschikt over een SPI-poort, een I2C-poort, twee hardware UART-poorten, acht digitale In/Outputs en zeven analoge Inputs.

3.1.1 Kosten

De WaspMote heeft helaas ook een keerzijde, de aanschafprijs is nog hoger dan die van de bestaande oplossing die QwikSense gebruikt. Een WaspMote met een Zigbee ZB-radio kost €130 volgens de productcatalogus van (Libelium, 2013). Voor de PRO-variant wordt €150 gevraagd. Deze kale WaspMote bevat behalve een temperatuursensor en accelerometer geen sensoren. Voor nog eens €120 is er dan een *Smart Cities* sensorbord beschikbaar zonder sensoren. Als we een sensor willen bouwen die op de zelfde parameters meet als de bestaande sensor van QwikSense dan komen we op een totaalprijs van €335 per sensor (zie Tabel 1). Deze samengestelde sensor beschikt niet over de sensor voor de bezettingsgraad en de sensor voor fijnstof. Deze worden namelijk niet geleverd door Libelium.

Naast de sensornode dient er nog een gateway aangeschaft te worden voor gebruik met de WaspMote. Deze verkoopt Libelium onder de naam Meshlium. De belangrijkste varianten van deze gateway zijn de Meshlium ZigBee-PRO-AP en de Meshlium ZigBee-PRO-3G-AP. Het eerste model van de twee beschikt over een WiFi-module om verbinding te maken met het internet en een Zigbee Pro module om met de sensornodes te communiceren. Het tweede model beschikt naast de modules van de eerste variant tevens over een 3G-module zodat ook zonder WiFi metingen doorgestuurd kunnen worden. Voor de module zonder 3G wordt €690 gevraagd terwijl voor de module met 3G €930 betaald moet worden. (Libelium, 2013)

3.1.2 Eindoordeel

De WaspMote is erg interessant om in de prototyping-fase van de ontwikkeling van een sensornode te gebruiken. Bovendien neemt Libelium met dit product een groot deel van het werk uit handen, voor de sensoren die Libelium levert zijn libraries beschikbaar. Zodra de sensor geproduceerd wordt is het wellicht interessanter om te kijken naar een goedkoper alternatief, een totaalprijs van €335 is te hoog om een interessant alternatief voor de sensornode van MakeMoreSense te zijn. Bovendien is de gateway duurder dan die van MakeMoreSense. Daar tegenover staat dan weer dat er aan het gebruik van de WaspMote en Meshlium (in tegenstelling tot bij MakeMoreSense) geen maandelijkse kosten verbonden zijn (Libelium, 2013).

3.2 Zelf samenstellen uit componenten

Omdat een kant-en-klare sensornode-oplossing te duur is heb ik onderzoek gedaan naar de prestaties van losse sensoren op de markt. Bij een aantal fabrikanten heb ik evaluatiechips aangevraagd en in veel gevallen werden deze kosteloos toegestuurd. Ook heb ik een aantal chips gekocht omdat er geen gratis evaluatiechips werden aangeboden.

Voor bijna elke omgevingsfactor zijn wel sensoren te vinden, sommige sensoren hebben echter een zeer hoog energieverbruik. Omdat de sensornode overal neer te zetten moet zijn en weinig onderhoud mag vereisen is het belangrijk dat er goed gekeken moet worden naar het energieverbruik van de sensoren en losse componenten. Een sensor die gemiddeld meer verbruikt dan er in een periode van 24

uur aan stroom gewonnen kan worden door bijvoorbeeld een zonnepaneel is per definitie al niet geschikt.

Bijlage 6 is een overzicht van de geselecteerde sensoren. Dit is geëxporteerd vanuit de Excel-sheet.

3.3 Conclusie

Er zijn veel producten op de markt die niet voldoen aan de eisen die QwikSense stelt. Veel van deze producten zijn gericht op consumenten en hebben daarom een kant-en-klaar ecosysteem met apps en een webservice die zij beheren. Het andere uiterste is de WaspMote die volledig naar wens aan te passen is maar prijstechnisch niet interessant is voor productie op grotere schaal. De meest interessante optie is dus het zelf bouwen van een sensornode.

Dit hoofdstuk behandelde de alternatieven voor de huidige hardware-oplossing die QwikSense in gebruik heeft. In het volgende hoofdstuk wordt het bouwen van een zelf samengestelde sensornode behandeld. Ook wordt de voor de sensornode geschreven software behandeld.

4 Een sensornode bouwen

In het vorige hoofdstuk zijn de alternatieven voor de huidige oplossing die QwikSense hanteert aan bod gekomen. De conclusie hieruit was dat zelf een sensor bouwen een interessante optie is. Het komende hoofdstuk gaat over het proces van het ontwikkelen en prototypen van een sensornode. Hierbij wordt er gekeken naar welke onderdelen deel uitmaken van een sensornode. Daarna wordt de keuze voor de door mij gebruikte draadloze technologie en de gekozen microcontroller toegelicht. Verder wordt de structuur van de sensornode-software toegelicht en tot slot komen certificering van het product en de Bill of Materials aan bod.

Binnen 'the Internet of Things' spelen microcontrollers een grote rol. Met een relatief goedkope chip en een vorm van (draadloze) communicatie is sensordata te koppelen aan het internet en kunnen motoren en circuits aangestuurd worden. Communicatie-modules krijgen steeds meer functionaliteit en programmeerbare microcontrollers worden steeds energiezuiniger en efficiënter.

Binnen de ontwikkeling van software en hardware zijn ontwikkelaars en fabrikanten continu bezig met het verbeteren van hun producten en diensten. Bij sensoren is dit niet anders. Het is daarom belangrijk om een product te maken waarbij het mogelijk is om in te spelen op de actuele ontwikkelingen. Fabrikanten van sensoren proberen steeds vaker zogenaamde *drop in replacements* te maken voor sensoren. Dit zijn sensoren waarvan de software en de aansluiting op dezelfde manier werkt als een oudere sensor. De kosten en het stroomverbruik van de nieuwe sensor zijn echter vaak wel lager. Door een modulair ontwerp aan te houden zorg je er voor dat je product langer mee kan gaan. Je maakt een ontwerp voor het 'moederbord' en plaatst er met de tijd betere en goedkopere sensoren in.

4.1 Fase 1 - prototyping

Het prototypen van een sensor houdt in dat er met behulp van losse componenten een elektronische oplossing wordt samengesteld waarbij een minimum aan functionaliteit van het uiteindelijke product aanwezig is. De uiteindelijke sensornode moet in ieder geval deze taken voor zijn rekening nemen:

- Een omgevingsvariabele meting met behulp van een sensor. De sensor draagt in veel gevallen de verantwoordelijkheid voor het omzetten van een analoge meting in één of meerdere bytes
- (Optioneel) een correctie uitvoeren op een meting, bijvoorbeeld als de sensor een waarde teruggeeft die buiten de specificatie van de sensor valt
- Metingen van meerdere sensoren verzamelen en versturen naar een gateway die de zorg draagt voor de verbinding met het internet en de verdere verwerking van de sensorwaarden.

4.1.1 Opties voor het samenstellen van een sensornode

Omdat tegenwoordig communicatiemodules over steeds meer rekencapaciteit en functionaliteit beschikken is het tegenwoordig een optie om uitsluitend een communicatiemodule te gebruiken voor sensormetingen. Dit bespaart energie (er is

geen losse microcontroller nodig) maar beperkt de mogelijkheden: zo is het bijvoorbeeld bij de Xbee niet mogelijk om voorwaardelijk een meting te versturen. Dit is een probleem wanneer een sensornode aangepast wordt, als er geen CO_2-sensor aanwezig is dan dient de sensornode natuurlijk geen CO_2-waarde te versturen.

Dit probleem wordt door het bedrijf Silicon Labes bijvoorbeeld opgelost door een sterkere processor en meer werk- en flashgeheugen in de communicatiemodules te integreren in de EM35x-serie van ZigBee-communicatiemodules. Met een eigen ontwikkelomgeving zijn sensoren eenvoudig te programmeren en kunnen er regels worden toegevoegd voor bijvoorbeeld het voorwaardelijk versturen van metingen.

Een nadeel aan deze oplossing is dat voor het samenstellen van een sensor op basis van Zigbee er een development-kit van €4477 (Digi-Key, 2013) nodig is. De software (IAR Workbench for ARM) maakt ongeveer de helft uit van deze kosten. De rest van de kosten bestaat uit de hardware en de software die Silicon Labs heeft ontwikkeld om eenvoudig een programma voor de EM35x samen te kunnen stellen (Ember AppBuilder). Voor QwikSense is het op het moment niet mogelijk om deze investering te doen. Bovendien beschikt dit soort oplossingen vaak niet over een extra UART-poort om met bepaalde sensoren te kunnen communiceren.

De laatste optie is het samenstellen van een eigen sensornode die bestaat uit een microcontroller die sensoren detecteert, uitleest, waar nodig calibreert en vervolgens de metingen doorstuurt naar de gateway d.m.v. een communicatiemodule. De gateway zorgt er voor dat de metingen via het internet bij een opslag (webservice) komen. De in 3.1 besproken WaspMote van het bedrijf Libelium is op deze wijze opgebouwd. Voor veel van de microcontrollers die op de markt zijn, stellen fabrikanten een ontwikkelomgeving beschikbaar. In het geval van Libelium wordt er gebruik gemaakt van de ontwikkelomgeving voor Arduino die onder de CC-licentie beschikbaar is. Daarom lijkt deze oplossing voor QwikSense het meest interessant. Bovendien kan er op een bekend platform (bv. Arduino) code ontwikkeld worden die hergebruikt kan worden in andere oplossingen. Ook kan er besloten worden om een andere communicatiemodule in te zetten zonder de code voor het uitlezen van een sensor te hoeven veranderen.

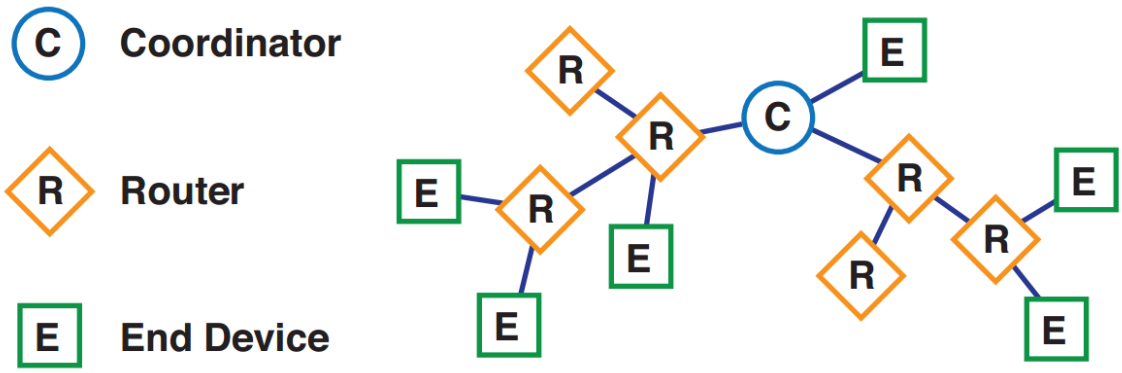
4.1.2 Keuze van een communicatie-module

Om de data van een microcontroller naar het internet te krijgen hebben we een losse communicatie-module nodig. Deze chip draagt de verantwoordelijkheid voor het verplaatsen van data naar een ontvangende module. Omdat we in dit geval gebruik willen maken van meerdere sensoren en we moeten letten op het stroomverbruik is het gebruik van veelgebruikte technieken zoals WiFi en Bluetooth uitgesloten. Bovendien zijn beide technieken redelijk beperkt in mesh-netwerk-functionaliteit. In een netwerk van sensoren is mesh-networking vaak wel gewenst en vaak zelfs noodzakelijk.

Het voordeel van mesh-networking is dat het mogelijk is om het netwerk aan te passen (doordat een apparaat verplaatst wordt of een storing heeft) en het netwerk

deze wijziging opvangt en er voor zorgt dat data toch bij het eindpunt komt (Faludi, Building Wireless Sensor Networks, 2010, p. 26). Tegelijkertijd wordt er altijd naar de kortste route gezocht waarmee berichten bij het eindpunt kunnen komen. In Figuur 5 is een voorbeeld van een mesh-netwerk te zien. Alle routers fungeren als doorgeefluik van data. Elk *End Device* hoeft alleen maar een meting te versturen naar een *router* in de buurt en deze zorgt ervoor dat het bericht bij de *coördinator* komt. Indien een *End Device* direct met de *coördinator* kan communiceren dan is dit de snelste route.

Het voordeel van deze structuur is dat de End Nodes bijna geen verantwoordelijkheden hebben binnen het netwerk en het grootste deel van de tijd kunnen 'slapen' (Faludi, Building Wireless Sensor Networks, 2010, p. 161). In de slaapmodus ontvangen ze geen berichten en alleen een vooraf geprogrammeerde ontwakingsperiode of een interrupt kan de microcontroller dan wakker maken. Door het grootste deel van de tijd te slapen is het mogelijk om een accuduur van meerdere jaren op een enkele Lithium-knoopcelbatterij te realiseren (Faludi, Arduino and XBee battery test results, 2010).



Figuur 5: Een visualisatie van de Zigbee-netwerkstructuur

De 802.15.4 standaard voor draadloze communicatie voorziet in mesh-mogelijkheden en wordt toegepast in Zigbee-modules. De modules die op de markt zijn worden door de fabrikant veelal geoptimaliseerd voor gebruik op een accu. Het bedrijf Digi maakt Zigbee-gecertificeerde modules voor mesh-networking onder de naam *Xbee*. Zigbee wordt in veel oplossingen gebruikt omdat het erg goed schaalt en relatief goedkoop is.

In de duurdere Zigbee-modules is het mogelijk om code te programmeren zodat sensoren direct op de Zigbee-module aangesloten kunnen worden. Dit maakt een losse microcontroller overbodig. In verband met de uitbreidbaarheid en schaalbaarheid van de oplossing kiezen we er voor om een losse microcontroller te gebruiken zodat we de data slim kunnen vormgeven in zo min mogelijk bytes. Ook kunnen we in de Arduino in het EEPROM data opslaan voor bijvoorbeeld kalibratie. Tot slot is het op deze wijze ook mogelijk om over te stappen naar een andere soort

communicatie-module zonder daarbij alle code voor het uitlezen van sensoren te hoeven herschrijven.

4.1.3 Keuze van een microcontroller

Er zijn op de markt veel oplossingen voor het bouwen van een embedded device. Bij het kiezen van een microcontroller kies je vaak voor de oplossing die 'exact past'. Aan 'te veel' geheugen heb je niets en de kans dat het stroomverbruik van een chip met meer functies lager is dan het stroomverbruik van een minder geavanceerde chip is klein. Verder spelen ook de kosten van een chip ook een rol.

De keuze voor een microcontroller is gevallen op de Atmel ATmega328P-PU, een microcontroller die gebruikt wordt in veel varianten van de populaire hobbyisten-bordjes zoals de Arduino Uno. Voor deze (en vele andere chips van Atmel) is een breed scala aan libraries beschikbaar en doordat de chip op 1.8V, 3.3V én op 5V functioneert, is het mogelijk de chip ook voor *low-power* toepassingen in te zetten. Verder biedt deze microcontroller 32KB aan opslaggeheugen voor code, 1KB EEPROM-opslag en 2KB aan SRAM-geheugen (Atmel, 2009, p. 1). Het apparaat beschikt over een intern kristal die gebruikt kan worden om het apparaat op 8MHz te laten functioneren i.p.v. de gebruikelijke 16MHz (zelfs 32Khz is mogelijk!). Dit heeft als voordeel dat het voltage waarop de microcontroller werkt kan worden verlaagd wat leidt tot een lager stroomverbruik (Nate, 2013). De verdere motivatie achter deze keuze voor de ATmega328P-PU als microcontroller is te vinden in bijlage 4. te zetten is het stroomverbruik flink terug te dringen. In volledige slaapstand is het verbruik terug te dringen naar slechts 0.1 micro-ampère (Atmel, 2009, p. 1) en (Nate, 2013). De chip is dan wel alleen nog wakker te maken met een fysieke interrupt. De Xbee is in staat deze interrupt te genereren. Dit betekent dat de Xbee bepaalt wanneer de Arduino wakker is (Faludi, Building Wireless Sensor Networks, 2010, p. 163).

4.1.4 Keuze van sensoren

Op de markt zijn allerlei sensoren beschikbaar. Sommige sensoren meten een parameter (bijvoorbeeld temperatuur) en andere sensoren meten meerdere zaken (luchtdruk, temperatuur). Bij het kiezen van sensoren is het belangrijk dat het stroomverbruik laag is en de opwarmtijd kort is. Daarnaast moet het meetbereik goed zijn (niet te groot, niet te klein), de afwijking zo klein mogelijk en het liefst is de sensor van een grote fabrikant die de sensor gedurende langere tijd in grote hoeveelheden kan aanleveren. Deze gegevens heb ik voor een aantal sensoren verzameld en in een Excel-werkblad (bijlage 6) gezet. Op basis van deze gegevens heb ik een aantal sensoren besteld en getest.

Voor elke sensor die we getest hebben heb ik een driver geschreven voor de sensornode-software. Deze driver bevat het protocol dat een sensor spreekt en hoe de gegevens van de sensor omgerekend kunnen worden tot een getal. In veel gevallen beschrijft de driver ook hoe er getest kan worden of een sensor in de sensornode aanwezig is. De sensor detecteert dus welke metingen er verricht kunnen worden. Een voorbeeld van hoe deze detectie plaats vindt:

```
bool FliX_AD7410::begin(byte address)
{
  // Enable I2C
  Wire.begin();                                 // connect I2C
  adt7410Address = address;
  if(!SENSORUTILS.checkDevicePresent(address)) return false;
  _theConfig = SENSORUTILS.getReg(adt7410Address, ADT7410IdReg);
  SENSORUTILS.setConfig(adt7410Address, ADT7410ConfigReg, B10000001);
  return true;
}
```

```
boolean SensorUtilities::checkDevicePresent(byte targetAddress){
  char error;
  Wire.beginTransmission(targetAddress);
  error = Wire.endTransmission();
  return (error == 0);
}
```

In bovenstaande codefragmenten wordt er een transmission gestart naar een adres op de I2C-bus. Wanneer er zich een apparaat met dat adres op de bus bevindt zal de error-code 0 zijn. Wanneer de code anders is dan 0 is er iets fout gegaan en wordt de sensor als afwezig beschouwd. Totdat de sensornode opnieuw start worden er van deze sensor geen metingen doorgegeven. Zowel de SensorUtilities-klasse als alle klassen waarvan de naam begint met FliX_ zijn door mij geschreven. De data die hier voor nodig was, was de documentatie van de Arduino Wire library (Arduino, 2008) en de informatie uit de datasheet van de desbetreffende chips.

4.1.5 Sensordata en protocollen

Als fabrikant van een product leg je vast welke functies een product biedt als het communiceert met externe systemen. Ook kun je eigen functionaliteit toevoegen die bijvoorbeeld meerdere taken achter elkaar doet maar die met een enkel commando aan te roepen is. Het gevolg is dat heel veel sensoren dus globaal op de zelfde manier werken maar toch net een andere taal spreken. Om dit probleem op te vangen heb ik gebruik gemaakt van de *Adafruit Unified Sensor Driver* van (Townsend, Adafruit_Sensor GitHub Repository, 2013). Deze library maakt het toevoegen van sensoren en uitlezen van sensordata eenvoudig en zorgt bovendien voor minder unieke code per sensor.

Omdat de library meer ruimte innam dan in mijn ogen noodzakelijk heb k het op een aantal vlakken geoptimaliseerd en waar nodig herschreven. Het eindresultaat is dat het toevoegen van een nieuwe sensor (met bijbehorend protocol) aan de firmware slechts enkele honderden bytes kost. Bij normale Arduino-libraries komt het vaak voor dat een library tussen de 900 bytes en 6000 bytes aan ruimte inneemt. De bespaarde ruimte kan gebruikt worden om een breder scala aan sensoren te ondersteunen.

Het gevolg van het gebruik van de sensor/driver-constructie is dat er op zeer eenvoudige wijze gewisseld kan worden van sensortype. Er worden een aantal typen sensoren ondersteund, maar mocht een type niet meer leverbaar zijn of de klant specifiekere eisen stellen dan is het geen probleem om een ander type sensor te kiezen. Als een sensor nog niet in de firmware ondersteund wordt is het aan de hand

van een eenvoudig template mogelijk een sensor-driver te schrijven. De basiscode voor het schrijven voor een sensor-driver is bijgevoegd als bijlage 5. Voor de meeste sensoren die gebruik maken van het I2C-protocol is het alleen nodig het adres van de sensor en de locatie van het data-register toe te voegen aan het template. Voor andere sensortypen is het iets meer werk, code voor SPI-communicatie is nog niet aanwezig in de SENSORUTILITIES-klasse. Voor een sensor die met een UART uitgelezen kan worden moet het protocol dat de sensor spreekt in een driver vastgelegd worden.

Voor de volgende sensoren heb ik een driver geschreven:

- Analog Devices AD7410 (temperatuur)
- CoZir Ambient 2K CO_2Sensor GC-0010 (CO_2)
- Texas Instruments DS1621 (temperatuur)
- Texas Instruments DS1631 (temperatuur)

Voor de volgende sensoren heb ik bestaande libraries geïmplementeerd en/of bewerkt:

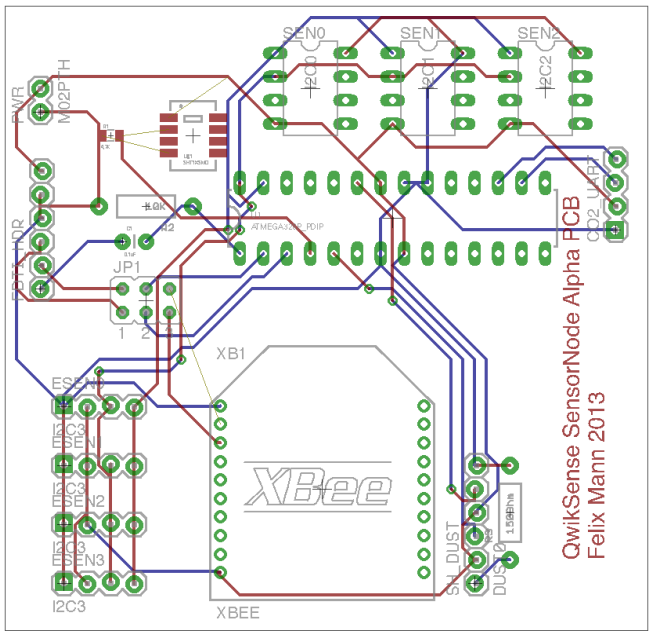
- Sensirion DHT-22 (luchtvochtigheid, temperatuur)
- Bosch BMP085/180 (temperatuur, hoogte, luchtdruk) op basis van (Townsend, Bosch BMP085 Breakout Board/Using the BMP085 (API v2), 2013).
- Sharp GP2Y1010AU0F ((fijn)stofsensor)

4.2 Fase 2 - werkend concept

Na het onderzoek naar de sensoren en de implementatie van de drivers voor de belangrijkste sensoren heb ik een werkend concept gerealiseerd. Het apparaat meet op de volgende parameters en kan deze doorsturen naar de webservice met behulp van de gateway (die ik bespreek in hoofdstuk 5):

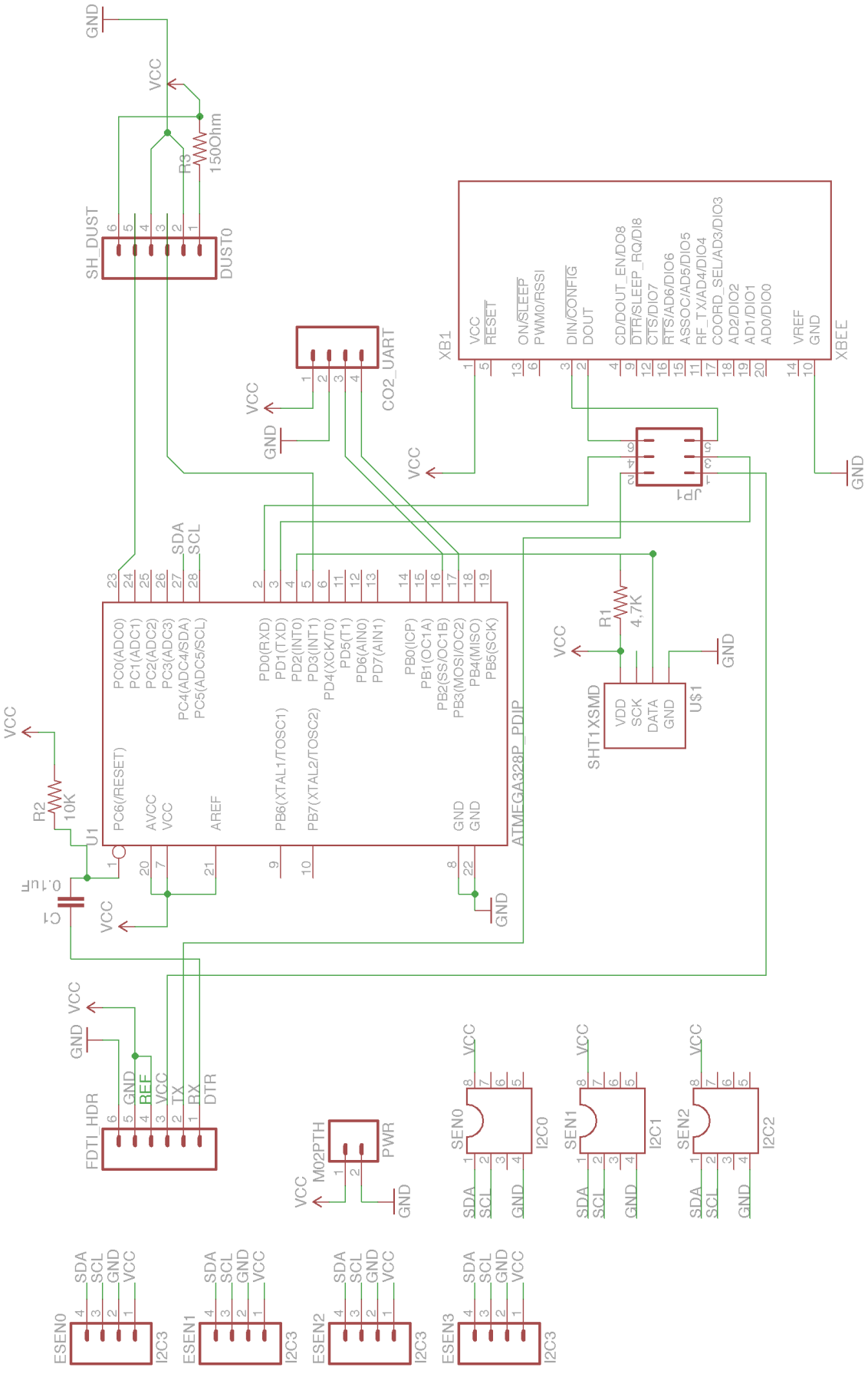
- Temperatuur
- Luchtvochtigheid
- CO_2-concentratie
- Luchtdrukniveau relatief aan zeeniveau
- Geschatte hoogte op basis van luchtdrukniveau
- Concentratie stof/fijnstof
- Periodiek gemeten geluidsniveau [N.B. Is momenteel niet geïmplementeerd!]

Figuur 6 is een ontwerp voor een PCB dat geproduceerd kan worden. Voor de CO_2-sensor is er een aparte aansluiting (helemaal rechts). Theoretisch kan elke sensor die gebruik maakt van een UART hier op



Figuur 6: PCB-ontwerp voor een SensorNode

aangesloten worden. Er dient dan wel rekening mee gehouden te worden dat er een driver voor de gekozen sensor geimplementeerd wordt. Ook voor de (fijn)stof-sensor is er een eigen aansluiting op het bord (rechts van de Xbee-module). Voor het aansluiten van extra I2C-sensoren zijn er vier extra aansluitingen. Door deze aansluitingen wordt een sensor voorzien van 3,3v, GND, een SCK en een SDA-signaal. Voor zowel de vier extra I2C-aansluitingen als de drie in te prikken sensoren (SEN0, SEN1, SEN2) geldt dat er op de sensor zelf een adres gekozen moet worden. Als er twee sensoren op het zelfde adres willen communiceren, zal er maar soms één, maar vaak geen van de twee sensoren functioneren. In Figuur 7 is een elektrisch schema te zien van de sensornode. Hier is beter op te zien welke chips welke signalen op welke pin binnen krijgen.



Figuur 7: Elektrisch schema opbouw nieuwe sensornode

De software die de sensornode gebruikt om metingen van sensoren te verzamelen en d.m.v. de Xbee te versturen is zelf geschreven. Ik heb de software geschreven in de Arduino IDE omdat dit een simpele IDE is waar het gebruik van libraries eenvoudig is. Bovendien is de IDE naar wens onder de motorkap aan te passen.

Figuur 8: Overzicht van de functies die aanwezig zijn in de hoofdklasse van de sensor-software.

De Sensornode-software kan op verschillende manieren gecompileerd worden. In eerste instantie communiceerde ik d.m.v. een seriële verbinding met de sensornode en las ik zo de meetwaarden uit. Naar mate het project vorderde begon de Xbee een belangrijker deel van het project uit te maken. Zowel de Xbee-library als alle seriële communicatiecode namen een significante hoeveelheid geheugen in. Voor testdoeleinden is het mogelijk om met een wijziging alle seriële of XBee-code uit te zetten.

4.2.1.1 Gebruikte Libraries

Ik heb voor het schrijven van de sensornode-software gebruik gemaakt van de volgende libraries:

- EEPROMex van (Elenbaas, 2012)
- EEPROMAnything van (Arduino Playground, sd)
- idDHTLib van (Niesteszeck, 2013)
- Adafruit_Sensor van (Townsend, Adafruit_Sensor GitHub Repository, 2013)
- Narcoleptic van (Knight, 2010)
- Interface to Sharp GP2Y1010AU0F Particle Sensor van (Nafis, 2010)

Veel van deze libraries zijn niet in de vorm toegepast waarin ze zijn verkregen. Waar mogelijk zijn er delen van de code verwijderd die niet nodig zijn en zijn de libraries verpakt in drivers.

4.2.1.2 void setup()

Binnen de setup-functie initialiseer ik de gebruikte libraries en lees ik de EEPROM-variabelen uit die gebruikt kunnen worden voor bijvoorbeeld kalibratie van sensoren. Voor het inlezen en uitlezen van EEPROM maak ik gebruik van de library van (Elenbaas, 2012) om zo efficiënt mogelijk met het beperkt aantal lees/schrijfcycli van het EEPROM-geheugen om te gaan en de Arduino Playground library EepromAnything (Arduino Playground, n.d.) voor het opslaan van verschillende datatypen in het EEPROM-geheugen.

Vervolgens wordt er gecontroleerd welke sensoren er aanwezig zijn op het bord. Hiervoor wordt op elk sensor-object de functie begin() aangeroepen. Alle drivers die dit ondersteunen hebben een returnwaarde van true als de sensor aanwezig is en een returnwaarde van false als dit niet het geval is. Deze waarden worden opgeslagen

voor gebruik in de loop(). Indien USE_XBEE gedefined is wordt in deze functie ook de Xbee-module geïnitialiseerd.

4.2.1.3 void loop()

In de functie loop() kunnen er twee dingen gebeuren:

Wanneer USE_XBEE niet gedefinieerd is draait het programma in seriële modus. Dit houdt in dat er commando's kunnen worden gestuurd naar de sensornode en met behulp van deze commando's kunnen er instellingen veranderd worden en meetwaarden uitgelezen worden. In Figuur 9 worden de beschikbare commando's weergegeven.

Figuur 9: Beschikbare commando's in Seriële modus

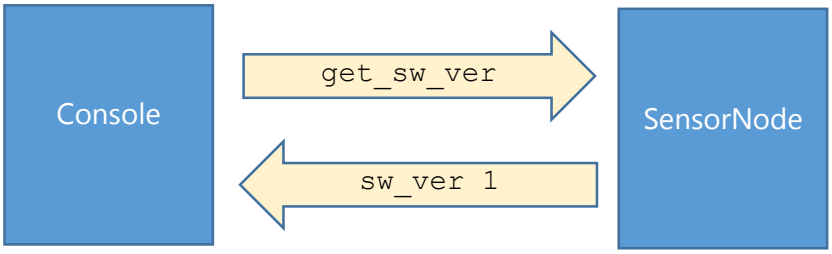
Veel van deze commando's hebben sinds het gebruik van de XBee hun bruikbaarheid verloren (b.v.b. `get_node_id` en `set_node_id` omdat de ID van de node bekend is als de XBee een bericht verzendt, deze berichten hebben een afzender en een geadresseerde) maar zijn nog wel aanwezig voor debugging-werkzaamheden. Bovendien wordt de XBee in API-modus gebruikt. Dit houdt in dat er conform een bepaald protocol gecommuniceerd moet worden met de XBee. Het voordeel hiervan is dat een programmeur verder geen omkijken heeft naar de daadwerkelijke inhoud van de berichten. De tegenhanger van de API-modus is de *transparant* modus. In deze modus zijn twee XBees met elkaar verbonden en de data die aan de ene kant verstuurd wordt komt aan de andere kant aan. In deze modus vindt geen routering plaats en beide nodes zijn hier gelijk. Sommige commando's zijn void en returnen geen data. Commando's die wel data returnen geven vooraf met het response_type aan wat voor response het is. Figuur 11 geeft weer hoe de SensorNode reageert op een request. In dit geval wordt er door de console een request gedaan naar de softwareversie. Omdat het mogelijk is dat een commando pas later een response krijgt (of er tussendoor een periodieke meting naar de console gestuurd wordt) wordt de response voorzien van een response_type. Na het response_type wordt de daadwerkelijke data meegestuurd. Dit werkt ook voor het versturen van commando's. Het eenmalig laten pulseren van de LED's in de kleur Deep Pink zou met het volgende commando gerealiseerd kunnen worden:

Figuur 10: Beschikbare responscodes in de seriële modus

```
pulse_led_single 0 255 20 147
```

Omdat dit commando verder geen resultaat oplevert (anders dan de pulserende LED 0 in de kleur R255, G20, B147) wordt er geen response verzonden. Als er geen serieel commando klaar staat wordt de functie refreshValuesIfTimeElapsed() aangeroepen.

Wanneer USE_XBEE wél gedefinieerd is vindt al het bovenstaande niet plaats en wordt de refreshValuesIfTimeElapsed() altijd aangeroepen. Daarnaast wordt de slaap-functie van de Arduino ingeschakeld om stroom te besparen. Dit gebeurt in de seriële modus niet.



Figuur 11: Een request en response-cyclus tussen een console en de SensorNode

4.2.1.4 Uitdagingen

Een microcontroller heeft de beschikking over een kleine hoeveelheid opslag- en RAM-geheugen. Het is daarom altijd extra belangrijk dat er zorgvuldig met RAM- en opslaggeheugen omgegaan wordt. Zoals aangegeven in 4.1.5 (Sensordata en protocollen) heb ik de losse libraries die ik gebruikte voor het aansturen van de sensoren geanalyseerd en ben ik met deze kennis begonnen aan het schrijven van drivers in de stijl van (Townsend, Adafruit_Sensor GitHub Repository, 2013). Dit heeft een aantal kilobytes opgeleverd.

Omdat er op een gegeven moment geen grote geheugenverbruikers meer zijn is het vaak interessant om even helemaal andersom te beginnen: gebruik je alle datatypen wel zo efficiënt mogelijk? Bij het bekijken en gebruiken van libraries en voorbeeldcode viel het me op dat er bijna altijd int's gebruikt worden om een getal op te slaan. In het geval van deze Arduino hebben we het over een 16-bits signed integer die ons de ruimte geeft om een getal van -32768 tot 32767 op te slaan. Op veel plekken is echter de opslag van een getal kleiner dan 255 voldoende. Hiervoor kan gebruik worden gemaakt van een (unsigned) char of een byte. Om ook negatieve getallen op te kunnen slaan is de signed char een optie, deze kan waarden van -128 tot 127 opslaan. Door op deze manier te besparen halveer je het geheugengebruik. Omdat we in totaal 2048 bytes aan RAM-geheugen tot onze beschikking hebben is het zaak om hier heel zuinig mee om te gaan, zeker in arrays. Het gebruik van de juiste datatypen scheelt in dit geval 50% aan geheugen.

De grootste winst was te behalen met het verwijderen van strings, characters en alle code die te maken heeft met schrijven naar een UART. Een character array zoals "Actuele temperatuur:" kost 21 bytes (20 karakters en een '\o' als termination character) die gedurende de looptijd van het programma in het RAM blijven staan. Het is mogelijk om deze array alleen tijdens het gebruik in te laden d.m.v. het PROGMEM-keyword. Door dit voor een character array te plaatsen wordt de compiler geïnstrueerd de variabele in het flashgeheugen te laten staan. Om de variabele te gebruiken moet deze telkens eerst worden gekopieerd (Arduino.cc, n.d.).

Verdere winst was te behalen in het verwijderen van functies die bij bijna elke sensor voorkomen (zoals bijvoorbeeld functies voor het uitlezen van een bepaald register, of het schrijven naar een register). Deze functies heb ik verzameld in de klasse

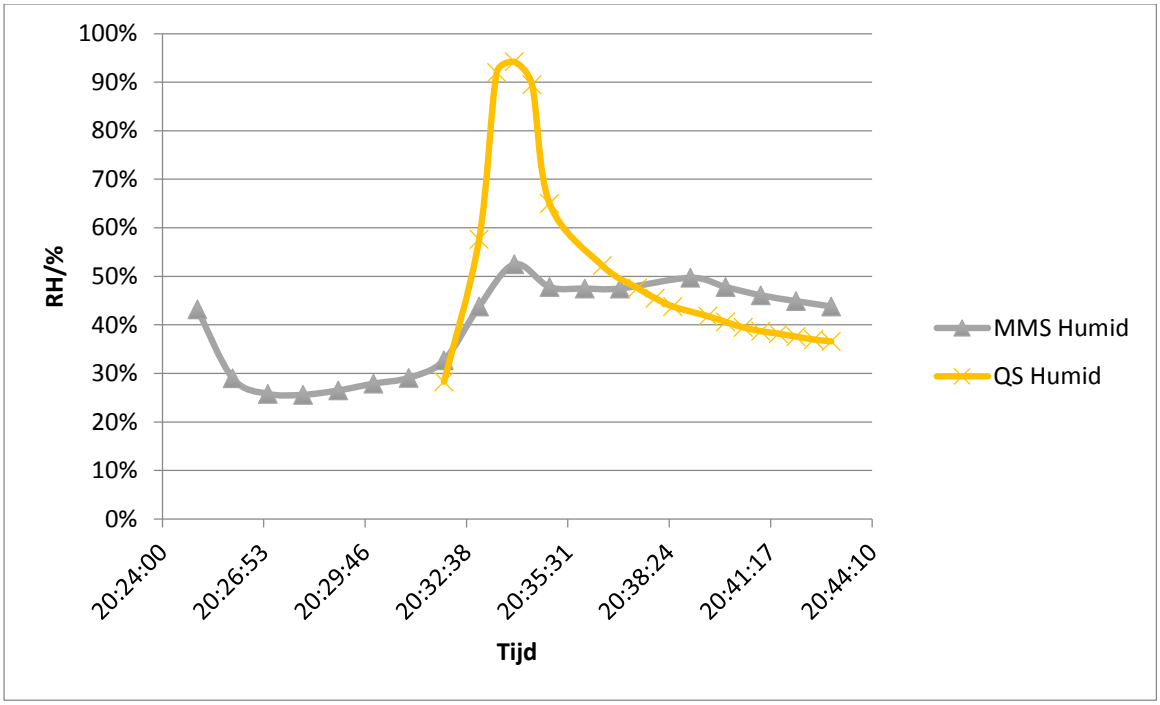
SENSORUTILITIES. Voorbeelden van functies die door veel I2C-sensoren gebruikt worden zijn het uitlezen/schrijven van een byte naar een register, het zetten/lezen van een bit in een register en het uitlezen/schrijven van meerdere bytes in een register. Er is geen standaard voor in welk register welke bit opgeslagen moet worden, de functionaliteit van elk type chip is zodanig anders dat een standaard nooit alles zou kunnen dekken. Door de adressen van een specifieke chip op te slaan heb je genoeg informatie om een chip uit te lezen. Op deze manier zijn de drivers uit 4.1.5 tot stand gekomen. De register-adressen voor het uitlezen van sensordata zijn te vinden in de datasheets die fabrikanten beschikbaar stellen.

4.3 Fase 3 - configuratie en testen

Na fase 2 was de sensor klaar voor een praktijktest. Voor deze test werden de sensoren in een testopstelling geplaatst en gedurende een periode van 48 uur gemonitord. Binnen deze testomgevingen wordt gekeken hoe de sensoren reageren op veranderingen in de omgeving en hoe accuraat de meetwaarden zijn in vergelijking met de bestaande oplossing van MakeMoreSense. Doordat de sensoren uit de bestaande oplossing tijdens een externe pilot niet beschikbaar waren was er slechts één kans om de sensoren daadwerkelijk naast elkaar te testen. Door een technische storing zijn de metingen na 20:44:10 helaas niet opgeslagen. Er was geen tijd om de test te herhalen voor de afrondingsdatum van deze scriptie. Alleen de meetresultaten van 20:24:00 tot 20:44:10 waren intact.

Figuur 12 geeft het temperatuurverloop van de eigen sensor en een sensor uit de huidige oplossing van QwikSense over een periode van twintig minuten. Binnen deze periode worden de sensoren gedurende tien minuten afgekoeld in de vriezer. Hiermee testen we de snelheid waarmee de sensor reageert op neergaande veranderingen in temperatuur en luchtvochtigheid. Daarnaast wordt het zendvermogen van de sensornodes getest: als er geen metingen binnen komen betekent het dat de Zigbee-chip niet krachtig genoeg is of niet goed geconfigureerd is. Na de afkoelperiode worden de sensoren vlak naast elkaar geplaatst. Hiermee wordt voorkomen dat de sensoren daadwerkelijk verschillende temperaturen aan het meten zijn. De sensoren zijn bij een verwarming geplaatst zodat het effect van een afslaande verwarming in de meetresultaten kan worden gezien. Ook hier kijken we weer hoe de veranderingen in de omgeving invloed hebben en hoe snel de sensornodes hier op reageren. Van deze periode zijn de meetresultaten verloren gegaan.

In Figuur 12 is te zien hoe de QS-sensor zich in eerste instantie sneller aanpast en bij het verwijderen uit de koeling (rond 20:32:00) omhoog springt. Dit heeft te maken met het feit dat condens op de luchtvochtigheids-sensor bevriest. Bij ontdooiing schiet de luchtvochtigheid ineens omhoog. Bij de MMS-sensor gebeurt dit ook, maar de meetwaarde lijkt er een stuk minder door beïnvloed. Gelukkig herstelt de eigen sensor zich snel. De MMS-sensor heeft meer tijd nodig om zich te herstellen.



Figuur 12: Het verloop van de temperatuur binnen een periode van twintig minuten, gemeten met verschillende sensoren

[Conclusie wel of geen verbetering t.o.v. MMS met motivatie]

4.4 Productie en certificering

Voor het produceren van de sensor heb ik advies ingewonnen bij het bedrijf Upp Energy. Dit bedrijf houdt zich bezig met het meten van energieverbruik binnen bedrijven en heeft hier zelf hardware voor ontwikkeld. De ervaringen van Upp Energy kunnen we gebruiken bij het produceren en laten certificeren van een eigen sensor. Wanneer de sensornodes en gateway in productie genomen worden en aan bedrijven geleverd gaan worden is onder andere een CE-certificering verplicht. De CE-certificering is een verklaring dat een product is geproduceerd in overeenstemming met de minimale veiligheids- en kwaliteitswetten die gelden in Europa (van Aken, Hoefnagels, van Randen, & Sonneveld, 1996). Ten tijde van het schrijven van dit document zijn we nog niet met een producerende partij in gesprek geweest.

4.5 Bill of materials

Om de sensor te produceren is er een lijst met onderdelen nodig. Voor elk onderdeel is de afnameprijs per stuk, per tien, per honderd en per duizend aangegeven. Ook is er voor elke afnamehoeveelheid een totaalprijs berekend. Deze sheet bevestigt dat de productiekosten van de sensornode significant gedrukt kunnen worden door ze in grotere aantallen te produceren.

Parameter	Type	Price / 1	Price / 10	Price / 100	Price / 1000
Pressure/temp	Bosch BMP085	€ 6,24	€ 4,92	€ 3,50	€ 3,06
Sound	Empty sensor				€ -
Humidity	Sensirion DHT22	€ 3,26	€ 3,55	€ 3,00	€ 3,00
Motion	Empty sensor				€ -
PM10	Empty sensor				€ -
PM25	Empty sensor				€ -
CO2	Cozir GC-0010	€ 130,00	€ 92,96	€ 85,00	€ 70,00
Temperature	Maxim DS1631A	€ 3,74	€ 2,63	€ 1,89	€ 1,59
Solar	MPT4.8-75	€ 8,05	€ 7,47	€ 6,75	€ 5,58
Battery	Adafruit 1100mAh	€ 7,46	€ 7,46	€ 7,46	€ 7,46
Zigbee	XB24-BWIT-004	€ 14,03	€ 14,03	€ 14,03	€ 14,03
	Totaal / stuk	**€ 172,78**	**€ 133,02**	**€ 121,63**	**€ 104,72**
	Totaal * Aantal	€ 172,78	€ 1.330,20	€ 12.163,00	€ 104.720,00

Figuur 13: De kosten voor de onderdelen waaruit een standaard sensornode bestaat

4.6 Resultaat

Ik heb een prototype-sensor kunnen maken tegen een aanzienlijk lager bedrag dan de aanschafprijs van een sensornode van MakeMoreSense. De sensornode is in een aantal opzichten verbeterd ten opzichte van de oude sensornode. De grootste verbetering is het feit dat de sensornode modulair opgebouwd is. Hierdoor is een sensor volledig aan te passen aan de wensen en eisen van de klant. Bovendien kunnen er sensoren gerealiseerd worden die bijvoorbeeld voorzien zijn van een accu en een zonnecel. Deze accu's zijn dus ook te plaatsen op locaties waar geen stroomvoorziening is. Tot slot leidt de modulaire opbouw ook tot een sensor die eenvoudig te onderhouden is. In veel gevallen is een onderdeel van de sensornode in een handomdraai te vervangen.

Daar staat tegenover dat het automatisch kalibreren van de sensoren nog niet geïmplementeerd is. Ook moet bidirectionele communicatie met de SensorNodes mogelijk zijn zodat bijvoorbeeld de polling rate van de sensornodes veranderd kan worden en de slaapmodus gefinetuned kan worden.

Om de sensor daadwerkelijk op grote schaal te produceren moet QwikSense op zoek naar een partij die mijn ontwerpen wil produceren tegen een schappelijke prijs. Zeker in kleine aantallen is dit vaak moeilijk.

Dit hoofdstuk behandelde het proces van het ontwikkelen en prototypen van een sensornode. Hierbij werd er gekeken naar welke onderdelen deel uitmaken van een sensornode. Daarna wordt de keuze voor de door mij gebruikte draadloze technologie en de gekozen microcontroller toegelicht. Verder werd de structuur van de sensornode-software toegelicht en tot slot kwamen certificering van het product en de Bill of Materials aan bod.

5 Een gateway bouwen

De bestaande oplossing maakt gebruik van een centrale node binnen het netwerk van sensoren die beschikt over een verbinding met een webservice. De hoofdreden om een centrale node te gebruiken is dat een verbinding met het internet een relatief kostbare bezigheid is voor een computer met beperkte rekenkracht. Bovendien is het op sommige plekken lastig om een vaste verbinding te gebruiken en wordt er gebruik gemaakt van mobiel internet waarbij al het dataverkeer per Kilobyte afgerekend wordt. Sensormetingen hebben doorgaans zeer weinig rekenkracht nodig. Bij het ontwerpen van een sensornetwerk is het daarom vaak verstandig om een microcontroller met zeer beperkte rekenkracht in de sensornodes te plaatsen en een krachtigere processor in de gateway die de verantwoordelijkheid draagt voor het versturen van data. Dit heeft als bijkomend voordeel dat de data gecomprimeerd verstuurd kan worden. Hoe meer metingen er tegelijk verstuurd worden, hoe hoger het percentage data dat per meetmoment bespaard wordt.

5.1 Gebruikte hardware

Aan de gateway worden een aantal eisen gesteld. Het belangrijkste eis was een vermindering van de kosten die een gateway van MakeMoreSense met zich mee brengt. In de tijd dat MakeMoreSense dit systeem ontwierp heeft het bedrijf gekozen voor het gebruik van de x86-architectuur. Tegenwoordig zijn er verscheidene ARM-architectuur-gebaseerde oplossingen. Een ARM-SoC (System on Chip) kost tegenwoordig niet veel meer, een Raspberry Pi (met een Broadcom SoC) kost voor het basismodel ongeveer 34 euro en het model met extra werkgeheugen en een netwerkkaart kost ongeveer 38 euro.

	Raspberry Pi Model B	MakeMoreSense Gateway
CPU (MHz, Cache)	Broadcom BCM2835 (700MHz, 16KB L1, 128Kb L2), 697 BogoMIPS	AMD Geode LX800(500Mhz, 64KB L1, 128Kb L2), 997 BogoMIPS
RAM (MB, MHz)	512MB, 400MHz	256MB, 400Mhz
Ethernet-chip	SMSC LAN9512 (10/100mbps)	VIA VT6105M (10/100mbps)
Opslag	SD-kaart	Compact-Flash kaart (4GB)
Aantal USB-poorten	2x USB1.1/2.0	2xUSB1.1/2.0
Prijs met LAN	Ongeveer €80 incl. BTW en accessoires.	€595 incl. BTW.

Tabel 2: Vergelijking in aanschafprijs tussen de Raspberry Pi Model B en de gateway uit de bestaande oplossing die QwikSense gebruikt.

Tabel 2 laat zien dat de apparaten wat betreft hardware redelijk aan elkaar gewaagd zijn. Beiden beschikken ze over een processor die weinig stroom gebruikt en hebben ze de beschikking over een ethernet-poort en twee USB 2.0-poorten. De Raspberry Pi heeft een zwakkere processor maar tweemaal zo veel RAM-geheugen als de MMS-gateway (MakeMoreSense). Het prijsverschil van 495 Euro maakt dat beetje extra rekencapaciteit van de MMS-gateway niet goed. Bovendien is de Raspberry Pi zowel softwarematig als hardwarematig een stuk eenvoudiger uit te breiden. Op de USB-poort kunnen alle 3G, Wi-Fi, Bluetooth en soortgelijke modules aangesloten worden zolang ze ondersteund worden door Linux. Er zijn vanaf ongeveer 6 Euro USB-modules voor WiFi-communicatie beschikbaar die werken met de Raspberry Pi. Een USB-module voor 3G kost ongeveer 30 Euro los en wordt vaak gratis bij een

jaarabonnement voor data geleverd. Naast via USB kunnen er op de GPIO-pinnen microcontrollers of communicatiemodules aangesloten worden. Er zijn ook kabels beschikbaar waarmee de GPIO-header naar een breadboard gebracht kan worden. Zeker in de prototyping-fase van een project kan dit erg interessant zijn.

De MMS-gateway ondersteunt een beperkt aantal externe modules voor WiFi- en 3G-functionaliteit. Voor deze geschikte WiFi- en 3G-upgrades worden aanzienlijke kosten in rekening gebracht: 3G-functionaliteit heeft een meerprijs van 100 Euro. De gebruiker heeft geen mogelijkheid (in elk geval niet zonder de medewerking van MakeMoreSense) om de Linux-versie te updaten of een eigen Wi-Fi-module te gebruiken. Overigens beschikken zowel de MMS-gateway als de Raspberry Pi over een seriële console om diagnostieke informatie uit te lezen of de configuratie van de gateway te wijzigen. Deze console is niet bedoeld voor gebruik door de eindgebruiker maar voor onderhoud door de leverancier.

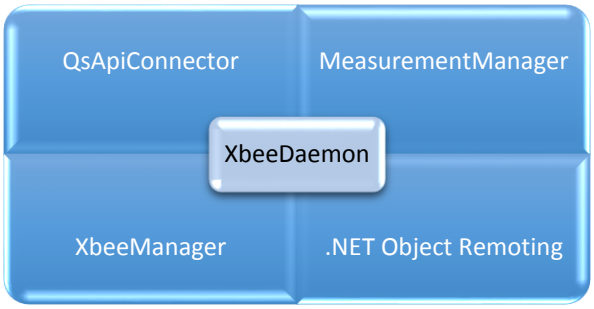
5.2 Gebruikte software

Hoewel er oplossingen bestaan voor het rapporteren van Zigbee-sensordata naar een webservice zijn veel oplossingen te duur. Met de basis van een Raspberry Pi ben ik aan de slag gegaan met het schrijven van een programma dat de sensordata van de sensornodes verzamelt en doorgeeft aan een webservice. De eisen aan de gateway zijn onder andere dat er diagnostieke informatie lokaal beschikbaar moet zijn zodat er op locatie gekeken kan worden of alle sensoren goed functioneren en wat de status van de sensoren is.

5.2.1 Verzamelen van sensordata

Voor het verzamelen van data was software nodig die berichten van de sensornodes kan verzamelen en (waar nodig) aanpassen. Het is hierbij belangrijk dat de software stabiel is. Om een inzicht te krijgen in een werkklimaat wil je een zo compleet mogelijk beeld. Als er sensormetingen missen geeft dit geen goed beeld van de betrouwbaarheid van de sensor.

XbeeDaemon is een door mij geschreven C#-applicatie die gebruik maakt van de GBee Library (http://xbee.codeplex.com). Omdat de library in haar originele vorm niet bruikbaar was voor mijn toepassing heb ik enkele aanpassingen gedaan aan de klassen en de library als geheel opgenomen in het XbeeDaemon project. De XbeeDaemon is een programma dat op de achtergrond zal draaien op de Raspberry Pi.



Figuur 14: Schematische weergave van de componenten waar de XbeeDaemon uit bestaat.

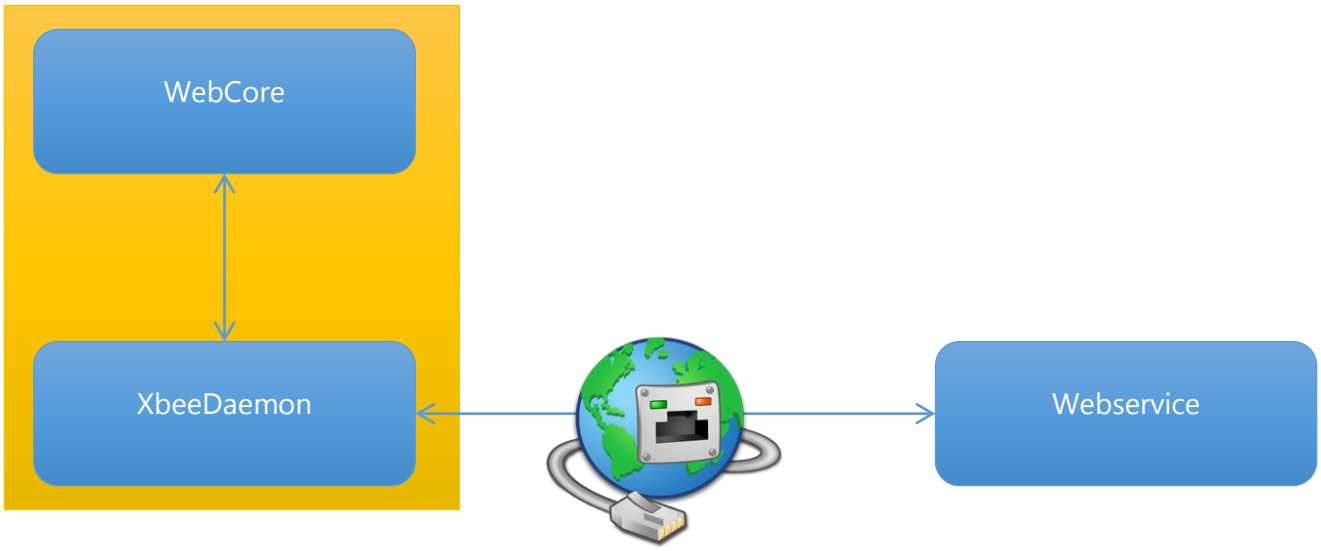
De **XbeeManager** opent een verbinding met de Xbee-module die op de Raspberry Pi is aangesloten en stuurt binnenkomende sensornode-berichten door naar de **MeasurementManager**. Er wordt door de MeasurementManager periodiek

gecontroleerd of er metingen gedaan zijn die nog niet verstuurd zijn naar de externe webservice. Deze metingen worden naar de **QsApiConnector** gestuurd. Deze klasse comprimeert de metingen en stuurt ze periodiek naar de webservice. Door metingen van verschillende sensoren te versturen als één bericht wordt de overhead van het versturen van een meting aanzienlijk verlaagd. Tot slot zorgt de techniek **.NET Object Remoting** er voor dat de XbeeManager door een externe executable te manipuleren is: alle variabelen die gebruikt worden om metingen en Xbees te beheren zijn in te zien en te manipuleren.

5.2.2 Diagnostiek en web-interface

Bij het plaatsen van sensoren is het heel handig om direct te kunnen zien of het bereik goed is en de sensor naar behoren functioneert. Ook als er geen internetverbinding naar buiten mogelijk is kan deze data nuttig zijn. In de web-interface van MakeMoreSense zijn het ID, en de signaalsterkte van een sensornode te zien. Om te voorkomen dat ik 'het wiel opnieuw uit zou gaan vinden' heb ik hiervoor gebruik gemaakt van een eigen project dat ik buiten deze scriptie om ontwikkel.

Het originele project heet WebCore en is een C# library die een programmeur in staat stelt om in enkele regels code een web-interface te laten genereren die zowel op mobiele devices als op desktops werkt zonder verlies van functionaliteit. Met behulp van de in 5.2.1 genoemde techniek .NET Object Remoting heb ik een daemon gemaakt die WebCore en XbeeDaemon aan elkaar koppelt onder de naam PiCore



(omdat de applicatie origineel voor de Raspberry Pi ontwikkeld is).

5.2.3 WebCore Handlers

WebCore maakt gebruik van 'Handlers'. Deze Handlers hebben een of meer verantwoordelijkheden. Wanneer een browser een request stuurt naar WebCore wordt er gekeken of er een handler is die de request af kan handelen. Dit gebeurt door te kijken naar de Request-URL. Bij de url http://picore/login is alles na de servernaam belangrijk voor het zoeken naar een handler. Is er een handler die login afhandelt? Dan krijgt deze handler de request. Verder is er nog een onderscheid

tussen een SimpleRequest en een ComplexRequest. Wanneer een request zoals http://picore/login gedaan wordt is dit een simple request. Wanneer de request wordt gedaan met POST-data of er nog zaken achter het handler-keyword staan is het een ComplexRequest. Voorbeeld: http://picore/nodes/0013A20040AA14E5/config/channel/set.

Voor PiCore heb ik een aantal handlers geschreven. De overige functionaliteit (b.v. inloggen en sessies) worden door WebCore afgehandeld.

- AboutHandler (geeft informatie over de sensor en QwikSense) [/about]
- HomePageHandler (landing page voor PiCore) [/]
- MeasurementHandler (handelt requests voor het weergeven van metingen af) [/measurements]
- NodeHandler (Geeft een status/overzicht van de Nodes in het sensornetwerk) [/nodes]
- SettingsHandler (Bevat intellingen voor configuratie en debugging) [/admin]

5.2.3.1 AboutHandler

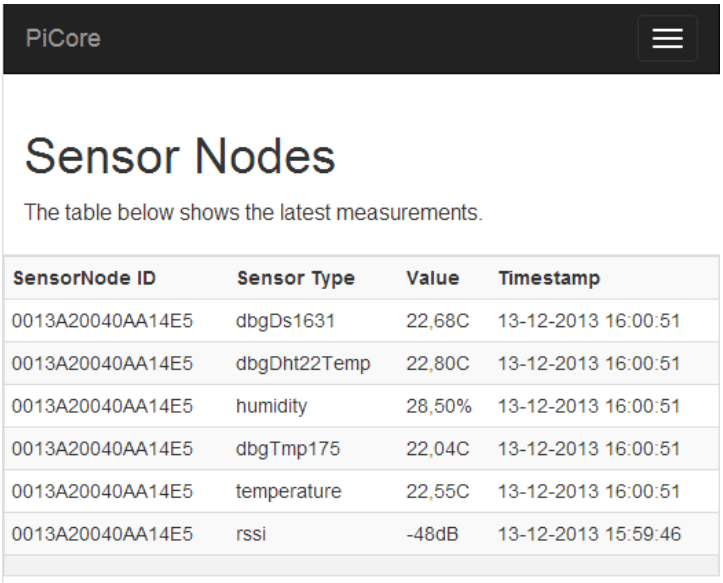
Deze handler kan alleen simple requests afhandelen en geeft een kort verhaal over wat het apparaat op dit IP-adres is en waar het vandaan komt.

5.2.3.2 HomePageHandler

Deze handler kan alleen simpele requests afhandelen en geeft de status van de gateway weer. Indien gewenst kunnen deze gegevens alleen zichtbaar worden gemaakt als de gebruiker ingelogd is en beheerdersrechten heeft.

5.2.3.3 MeasurementHandler

Deze handler handelt zowel simple als complex requests af. Door naar /measurements te navigeren in een browser wordt het beeld van figuur X weergegeven. In dit beeld zijn van alle sensoren in het netwerk de laatste metingen te zien en het laatste tijdstip dat de meting gedaan is. Hier is dus ook te zien of een bepaalde sensor metingen doorgeeft. Ook zien we hier dat de meting met het Sensor Type rssi in dit geval al ongeveer een minuut lang geen nieuwe waarde heeft gekregen. De complex requests die de MeasurementHandler af kan handelen zijn onder andere het weergeven van de metingen van een enkele sensornode, het weergeven van metingen van een sensor in de sensornode, en het weergeven van een selectie van de gegevens van een sensor in een sensornode (in de vorm van JSON). Afhankelijk van het soort request wordt er

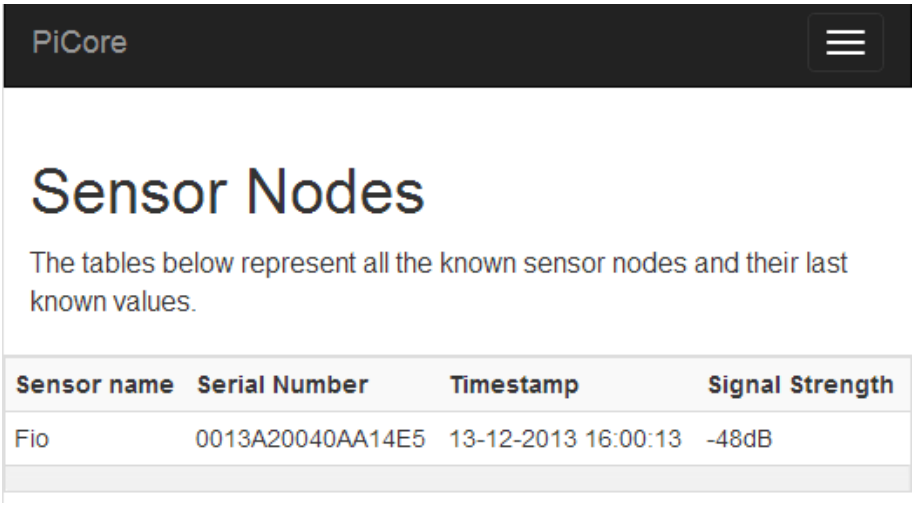


Figuur 15: Een screenshot van de MeasurementHandler in PiCore

een pagina teruggestuurd of alleen de data. Door het gebruik van de techniek AJAX wordt zo de belasting van de server (die in sommige gevallen alleen JSON hoeft te versturen i.p.v. gegenereerde HTML-code) verlaagd. Grafieken van een sensor worden op de client gerenderd door middel van de charts.js library van (Downie, n.d.).

5.2.3.4 NodeHandler

Deze handler handelt zowel simple als complex requests af. Door naar /nodes te navigeren in een browser wordt het beeld van figuur X weergegeven. In dit beeld zijn alle nodes in het sensornetwerk weergegeven met de laatst gemeten signaalsterkte en het laatste tijdstip dat er metingen zijn ontvangen van de sensornode. Verder worden het adres van de sensornode en de *Friendly name* van de sensor weergegeven.



Figuur 16: Een screenshot van de NodeHandler in PiCore

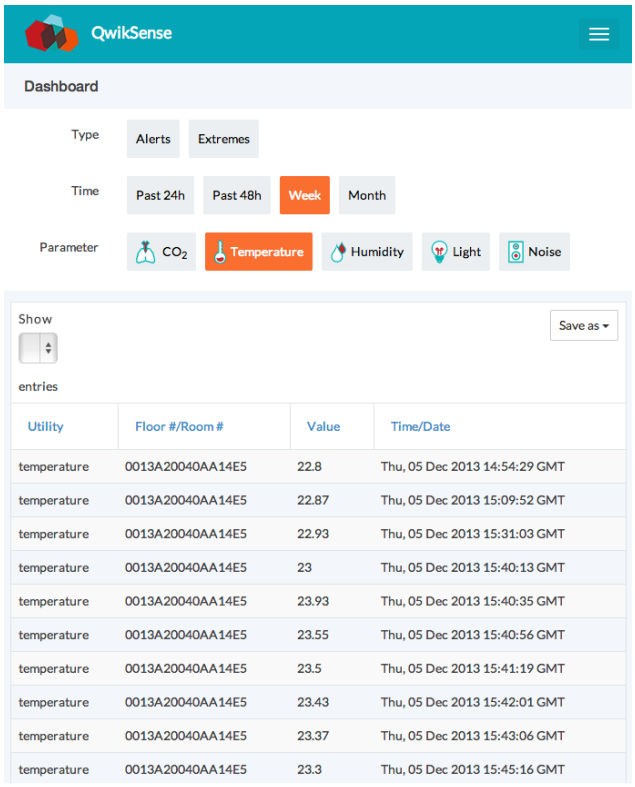
5.3 Resultaat

Evenals de sensornode is de gateway een stuk goedkope geworden dan de gateway uit de bestaande oplossing. Bovendien is een koppeling met de eigen webservice gerealiseerd en kan sensornode-data nu bekeken worden in het online dashboard.

Verder kan er in de web-interface van de gateway gekeken worden naar de diagnostieke informatie, waaronder actuele metingen en signaalsterkte van de nodes. De sensornode kan tegen een geringe meerprijs voorzien worden van 3G of Wi-Fi en daarbij is er geen beperking aan welke mobiele provider er gebruikt wordt.

6 Conclusies en aanbevelingen

Binnen mijn stageperiode bij QwikSense heb ik een hoop geleerd over sensornetwerken en Zigbee. Deze kennis heb ik gebruikt om een prototype sensornode te bouwen. De sensornode is goedkoper om te produceren dan de sensornode van MakeMoreSense. Bovendien zijn er geen maandelijkse kosten verbonden aan het gebruik van de eigen sensoren. De nieuwe sensoren zijn volledig in eigen beheer en zaken als de polling rate zijn door QwikSense zelf beheerbaar. Tot slot is de nieuwe sensor modulair: klanten van QwikSense kunnen bijvoorbeeld een sensornode met uitsluitend een temperatuursensor bestellen om kosten te drukken. Wanneer er nieuwere (goedkopere) sensoren op de markt komen kunnen deze zonder of met kleine aanpassingen in de sensornode-software worden gebruikt in de nieuwe sensornodes.



Hoewel niet alle functionaliteit die ik wilde implementeren in de opgeleverde versie zit is het belangrijkste doel behaald, er is een sensor geproduceerd die aanzienlijk goedkoper is dan de sensor uit de bestaande oplossing en QwikSense kan sensornodes samenstellen naar wens van de klant. Bovendien kunnen deze sensoren communiceren met de eigen webservice van QwikSense en is door middel van de web-interface in de gateway op locatie te controleren of de sensornodes correct functioneren. De bespaarde kosten geven ruimte voor verdere softwareontwikkeling en groei van QwikSense. Wellicht kan deze sensor een opstapje vormen naar een sensornode versie 2.0, de sensornodes bieden immers veel flexibiliteit. De duurste sensoren (zoals bijvoorbeeld de CO_2-sensor) zouden overgezet kunnen worden naar een sensornode versie 2.0. In de tussentijd heeft QwikSense een sensor die getest en gebruikt kan worden voor pilots en kleinschalige productie.

Aanbevelingen voor de verdere ontwikkeling van de opgeleverde producten:

6.1 Sensornode

- Ontwerpen van een behuizing voor de sensornode
- (laten) Produceren van de PCB's
- PCB's en behuizing laten certificeren
- Uitwisseling custom commando's via XBee implementeren

6.2 Gateway

- Testen van een module die binnen de Raspberry Pi geplaatst kan worden i.p.v. extern via USB
- 3G-software installeren voor de meestgebruikte USB-modems en de configuratie bewerkbaar maken in XbeeDaemon
- Toevoegen van meer diagnostieke informatie zoals netwerktopologie
- Toevoegen van meer configuratie-instellingen op de configuratiepagina
- Toevoegen van node-configuratie-functies voor bijvoorbeeld de polling rate/sleep mode
- Toevoegen van een modus waarin alle nodes wakker blijven (voor installatiemodus)
- Toevoegen van pairing-modus (nodes mogen dan alleen verbinden als deze aan staat)

7 Bronnen

Arduino. (2008, 08 12). *Arduino - Wire*. Retrieved 09 05, 2013, from Arduino: http://arduino.cc/en/reference/wire

Arduino Playground. (n.d.). *Reading and Writing Data Structures to EEPROM*. Retrieved 10 12, 2013, from Arduino Playground: http://playground.arduino.cc/Code/EEPROMWriteAnything

Arduino.cc. (n.d.). *Arduino - PROGMEM*. Retrieved 10 02, 2013, from Arduino Reference: http://www.arduino.cc/en/Reference/PROGMEM

Atmel. (2009, Oktober). *ATmega48PA/ATmega88PA/ATmega168PA/ATmega328P Datasheet Summary*. Retrieved from Atmel Corporation: http://www.atmel.com/Images/8161s.pdf

CubeSensors - Features. (2013, 11 26). Retrieved from CubeSensors - Improving indoor living: http://cubesensors.com/#features

Digi-Key. (2013, 12 9). *EM35X-DEV-IAR Product Page*. Retrieved from Digi-Key Corporation: http://www.digikey.nl/short/t18vr

Downie, N. (n.d.). *Chart.js | HTML5 Charts for your website*. Retrieved 12 04, 2013, from Chart.js | HTML5 Charts for your website: http://www.chartjs.org

Elenbaas, T. (2012, 07 22). *Extended EEPROM library for Arduino*. Retrieved 10 12, 2013, from Thijs Elenbaas Weblog: http://thijs.elenbaas.net/2012/07/extended-eeprom-library-for-arduino/

Faludi, R. (2010, 8 30). *Arduino and XBee battery test results*. Retrieved 10 12, 2013, from Rob Faludi Blog: http://www.faludi.com/projects/arduino-and-xbee-battery-test-results/

Faludi, R. (2010). *Building Wireless Sensor Networks*. (B. Jepson, Ed.) Sebastopol, United States of America: O'Reilly.

Futurlec. (2013, 09 01). *Technical Information - CO2 Gas Concentration Module Datasheet*. Retrieved from Futurlec: http://www.futurlec.com/Datasheet/Sensor/MH-Z14.pdf

Knight, P. (2010, 07). *narcoleptic - Sleep library for Arduino*. Retrieved from Google Code / narcoleptic: https://code.google.com/p/narcoleptic/

Libelium. (2013, 12 15). *Libelium Over the Air Programming (OTAP)*. Retrieved from Libelium web site: http://www.libelium.com/products/waspmote/OTA/

Libelium. (2013, 11 26). *Libelium Waspmote Product Catalogue*. Retrieved from Libelium Waspmote Product Catalogue: http://www.libelium.com/xhjs76gd/libelium_products_catalogue.pdf

Libelium. (2013, 11 26). *Products*. Retrieved from Waspmote - Wireless Sensor Networks 802.15.4 ZigBee Mote - Open Source Sensor Device | Libelium: http://www.libelium.com/products/waspmote/

Nafis, C. (2010, 11 13). *Interface to Sharp GP2Y1010AU0F Particle Sensor*. Retrieved from Sharp Dust Sensor and Arduino: http://sensorapp.net/?p=479

Nate. (2013, 10 23). *Adventures in Low Power Land*. Retrieved from Sparkfun Tutorials: https://www.sparkfun.com/tutorials/309

Niesteszeck, N. (2013, 10 05). *DHT11 & DHT22 interrupt driven library for arduino*. Retrieved from Github / niesteszeck / idDHTLib: https://github.com/niesteszeck/idDHTLib

Sensemakers. (2013, 11 26). *Air Quality Egg | sensemakers*. Retrieved from Sensemakers Portfolio: http://sensemake.rs/portfolio-item/air-quality-egg/

Townsend, K. (2013, 11 01). *Adafruit_Sensor GitHub Repository*. Retrieved from adafruit on GitHub: https://github.com/adafruit/Adafruit_Sensor

Townsend, K. (2013, 12 2). *Bosch BMP085 Breakout Board/Using the BMP085 (API v2)*. Retrieved from AdaFruit Learning System: http://learn.adafruit.com/bmp085/using-the-bmp085-api-v2

van Aken, D., Hoefnagels, W., van Randen, A., & Sonneveld, M. (1996). *Handboek Ontwerpen van veilige producten.* Utrecht: Lemma.