# Programmeerproject 2013-2014

## Tweede jaar Bachelor in de Computerwetenschappen

Titularis: Prof. Theo D'Hondt
Begeleidende Assistenten:
Kevin Van Vaerenbergh (kevvaere@vub.ac.be)
Lode Hoste (lhoste@vub.ac.be)
Yves Vandriessche (yvdriess@vub.ac.be)
project2ba@dinf.vub.ac.be

## Context

In het programmeerproject van 1ste bachelor hebben jullie een eerste groot programma in Scheme gebouwd. In dit project maken jullie een complete softwaretoepassing. Dit wil zeggen dat je uiteindelijke "product" niet enkel bestaat uit een werkend programma. Enerzijds dien je je toepassing te beschrijven, zowel gericht naar de eindgebruiker als naar andere ontwikkelaars die eventuele aanpassingen aan jouw project moet kunnen uitvoeren. Anderzijds dient je code aan bepaalde kwaliteitseisen te voldoen, waaronder een uitvoerige documentatie, het hanteren van coding conventies en testing.

In dit project is er veel ruimte gelaten om je creativiteit op bot te vieren. Dit document bevat een algemene omschrijving van het onderwerp, de opdracht, de 'minimale' functionele vereisten en de niet-functionele vereisten zoals code stijl en rapportering. Je wordt aangemoedigd om de originele opdracht te verrijken met eigen inbreng, zolang dit in de thema van de opdracht past.

# Opdracht

## Opzet

Het onderwerp van je toepassing is het ontwerpen van een domotica energie monitoring systeem. Verschillende input sensoren zullen ter beschikking zijn en je toepassing moet kunnen communiceren met deze sensoren. De toepassing moet het gebruik van de verschillende sensoren kunnen voorstellen, gebruik makende van een grafische user interface.

In het kader van deze toepassing komen dus een aantal “nieuwe” elementen aan bod:

- Communiceren met **domotica hardware (**sensoren en actuatoren) om informatie uit de omgeving te verzamelen.
- Werken met **communicatie protocols** (TCP/IP, ZigBee) om een netwerk uit te bouwen tussen verschillende apparaten en computer platformen.
- Een project uitbouwen dat werkt over **verschillende platformen** (hardware en software).
- Een **grafische user interface** (GUI) om de verzamelde informatie te bekijken en te navigeren.
- Een lokale **databank** gebruiken om niet alleen sensor geschiedenis in op te slaan, maar ook om je programma **persistent** te maken: bv. verder lopen zoals ervoor na een stroompanne.

Deze opdracht werd origineel in twee delen gesplitst. Dit document beschrijft de gecombineerde en dus finale opdracht voor. De vereiste van het eerste deel worden hier dus nogmaals herhaald. In het eerste deel lag de nadruk op het schrijven van je domotica energie monitoring systeem, met bijhorende grafische interface en databank. Het tweede deel werkt de netwerk communicatie uit, een accurate simulatieomgeving voor de gegeven domotica apparaten en het verzorgen van de communicatie met de werkelijke domotica hardware.

## Examinatie

Mits er al een partiële examinatie plaatsvond aan het einde van Deel 1, kan de eind examinatie van dit programmeerproject op twee verschillende manieren verlopen:

- *Je bent tevreden met de quotering van Deel 1.* Je blijft je punten van Deel 1 behouden. Je maakt een eindverslag zoals beschreven in de rapportering sectie van dit document. Tijdens de examenperiode kom je individueel je project en eindverslag mondeling verdedigen. Bij deze examinatie ligt de nadruk op de vereisten van Deel 2. Daarnaast maak je ook duidelijk welke delen van Deel 1 je aangepast, veranderd of verwijderd hebt sinds je tussentijdse projectverdediging.
- *Ik ben niet tevreden met de quotering van Deel 1* en wens hiervoor opnieuw geëxamineerd te worden. Je maakt een rapportering die beide de vereisten van Deel 1 en Deel 2 omvat. Bij de examinatie tijdens de examenperiode zal je een presentatie en demo geven van heel je project. De score combinatie van deze rapportering en mondelinge examinatie wordt je uiteindelijke eindcijfer voor dit vak.

Het correct invullen van alle functionele en niet-functionele vereisten zal je maar een voldoende slaagcijfer geven. *Het is de bedoeling dat je voor dit programmeerproject een grote eigen inbreng kan demonstreren, in de vorm van aantal extra functionaliteiten.*

# Functionele Vereisten

Hieronder beschrijven we de functionaliteit die je toepassing zeker moet bevatten. Zoals eerder vermeld word je aangemoedigd en zelfs aangeraden om met extra functionaliteit in je project te verwerken. Let wel op dat de nadruk in dit project nog steeds ligt op de kwaliteit van je code, en niet zozeer op het aantal features. Bespreek dus zeker je plannen grondig met de begeleidende assistenten, alvorens er al te veel tijd aan te besteden.

---

## Opstelling

De virtuele domotica opstelling waarmee we werken bevat drie soorten hardware:

- Een normale computer of server die dienst doet als aanspreekpunt van het domotica-systeem voor de gebruiker: **de majordomo** of huisopzichter.
- Per kamer een gelimiteerde en geïntegreerde computer die de fysieke connecties beheert van omringende sensoren en actuatoren: **de steward**.
- Verschillende domotica **devices** of apparaten: sensoren en actuatoren die rondom het huis geplaatst worden. Er zijn sensoren die beweging kunnen meten, lichtintensiteit, temperatuur, stroomverbruik, etc.  Actuatoren kunnen ook acties ondernemen, zoals een schakelaar die de stroom afsluit of een automatisch rolluik. Vele apparaten combineren verschillende actuatoren en sensoren, zoals bijvoorbeeld een thermostaat die een thermometer, bewegingssensor, lichtsensor en schakelaar bevat.

We verdelen de functionele vereisten in twee delen. Deel 1 bevat de vereisten voor de majordomo. Je broncode voor het eerste deel van de opgave draait op de majordomo: een desktop of server computer platform. Deel 2 bevat de vereisten voor de steward. Je broncode voor het tweede deel van de opgave draait op het Raspberry Pi computer platform. Let op dat de simulatie van de steward een functionele vereiste is van van Deel 1. *Het moet mogelijk zijn om de majordomo te gebruiken zonder de steward hardware of desbetreffende functionaliteiten van Deel 2.*

# Deel 1

## Simulatie en communicatie

De majordomo communiceert met de stewards, op een onrechtstreekse manier verzameld hij zo verschillende gegevens van sensoren en stuurt hij actuatoren aan. Dit communicatiekanaal moet gesimuleerd kunnen worden, zodat de majordomo kan getest en gedemonstreerd worden zonder de steward hardware. We leggen niet vast hoe je deze simulatie moet opstellen, het ontwikkelen van een passend ontwerp is deel van de opdracht.

## Grafische interface

Voor de grafische interface maak je gebruik van de faciliteiten in Racket, een dialect van Scheme. Je dient dus deze “Racket Graphical Interface Toolkit” onder de loep te nemen. Functionaliteit is heel belangrijk. De minimaal te ondersteunen functionaliteit van je GUI wordt toegelicht aan de hand van gebruikers scenarios die je kan vinden in **Appendix A**.

We willen ook benadrukken dat je moet trachten de echte functionaliteit van je applicatie te scheiden van de grafische interface. Een goede vuistregel gaat als volgt: stel dat je een andere interface moet toevoegen (bv. één die meer geschikt is om te gebruiken op een smartphone), hoeveel van je code kan je hergebruiken, hoeveel moet je aanpassen?

Met toestemming van de begeleidende assistenten mag je, in de plaats van de Racket GUI, een website of web applicatie bouwen. Let op dat je hiervoor nog steeds binnen Racket moet werken, zonder externe programma’s of frameworks. Bekijk dus eerst de web application bibliotheek van Racket alvorens toestemming te vragen.

## Persistentie

Maak gebruik van een database om de grote hoeveelheid aan informatie op te slaan. Als DBMS gebruiken jullie SQLite. Een bibliotheek voor het aanroepen van SQLite databases is beschikbaar voor Racket.

Naast het bijhouden en oproepen van verzamelde sensor-informatie, moet je de database ook gebruiken om **persistentie** te verzekeren. Dit wil zeggen dat na een stroomonderbreking (je sluit Racket af) je programma terug moet kunnen opstarten en automatisch verder lopen zoals ervoor. Je mag ervan uitgaan dat bij het opstarten een `(start)` functie zonder argumenten wordt aanroepen. De database moet dus transparant voor de gebruiker opslaan wat de configuratie-opties zijn, welke apparaten aanwezig zijn, de doorlopende opdrachten, etc. Tip: begin pas aan je databank en persistentie module nadat de rest van de functionaliteit in orde is.

# Deel 2

De functionele vereisten van Deel 1 gelden ook voor Deel 2. Dit wil zeggen dat je tijdens de examinatie nog steeds moet kunnen aantonen dat de User Interface en Persistentie delen van je project voldoen aan de voorwaarden zoals uitgelijnd in de opgave van Deel 1. Het simulatie onderdeel blijft, maar wordt hieronder gedeeltelijk vervangen door een nieuwe functionele vereiste.

---

## Steward op de Raspberry Pi

De bedoeling van het Steward programma is dat deze draait op een kleine, goedkope en minder stroom verbruikende computer, zogenaamde embedded hardware. De basis vereiste hier is dat alle functionaliteit van de Steward werkt op de uitgedeelde Raspberry Pi hardware. De Scheme omgeving die je hiervoor gebruikt is *Racket R5RS* ofwel *Slip*. Het gebruiken van Slip wordt aanzien als het invullen van deze basis vereiste plus een extra functionaliteit, zie de Extra Functionaliteiten sectie later in dit document.

De communicatie tussen de Majordomo computer en een Steward verloopt via een TCP/IP netwerk. Het spreekt vanzelf dat je de TCP/IP functies (die niet R5RS zijn) uit Racket mag gebruiken hiervoor. Het is ten sterkste aangeraden om deze code migratie in verschillende stappen te ondernemen. Een richtlijn hiervoor kan je vinden in een apart document genaamd ‘Stewart Raspberry Pi migratie richtlijn’ op Pointcarré. Hoe je concreet de communicatie verzorgd en data uitwisselt tussen de Steward en Majordomo (viz. de communicatie interface) is deel van de opdracht. Het design hiervan zal worden beoordeeld op een gelijkaardige manier als het design van een ADT.

---

## Zigbee Communicatie en Devices

Het draadloos communicatie protocol Zigbee wordt gebruikt om de communicatie te verzorgen tussen de Steward en de rondomliggende apparaten. Een Xbee[1] radio is aangesloten aan de Steward, wat hem toelaat Zigbee berichten te ontvangen en uit te sturen. Deze communicatie interface verloopt door middel van ‘API Frames’, een vector van bytes die gestructureerde data voorstelt. Als minimum functionaliteit moet je de volgende frames ondersteunen:

| Frame Type (Hex) | Frame Name |
|---|---|
| `0x10` | Zigbee Transmit Request |
| `0x8B` | Zigbee Trasmit Status |
| `0x90` | Zigbee Receive Packet |

Via deze API frames kan je berichten sturen naar en ontvangen van de Zigbee sensoren en actuatoren.
Een domotica opstelling met coordinator-Xbee, sensoren en actuatoren zal worden aangeboden om je project uit te testen.
Gedetailleerde instructies en specificaties over de Zigbee API, apparaten en de Scheme Xbee interface kan je terugvinden in een appart document genaamd ‘Xbee interface’.

---

[1] Zigbee is de naam van het communicatie protocol. Xbee is de merknaam van de radio hardware.

## Simulatie van de Opstelling

Het moet mogelijk zijn op een Steward om in een gesimuleerde opstelling te werken. Dit wil zeggen dat de Zigbee interface en de communicatie met de apparaten moet kunnen gesimuleerd werken. Deze simulatie moet volledig overeenkomen met de Zigbee communicatie en apparaat gedrag zoals in de ‘Zigbee Communicatie’ functionele vereiste hierboven. Met andere woorden, het moet triviaal zijn om de gesimuleerde opstelling simulatie te vervangen met een werkelijke opstelling.

## Scenario

Voor de samenwerking van alle componenten in je programmeerproject te demonstreren, moet je minstens één demonstratiescenario ondersteunen. Een voorbeeld scenario is bijvoorbeeld de simulatie van een dag en nacht cyclus, waar de Majordomo ervoor gaat zorgen dat bepaalde lampen uit en aan worden gezet. Minimaal moet dit scenario alle minimale vereisten van Deel 1 & 2 demonstreren. Het is de bedoeling dat al je scenario’s werken op beide een volledige hardware en volledige simulatie opstelling. Gebruik zulke scenario’s ook om je extra functionaliteiten te demonstreren. Uw scenario’s presenteer je als demo tijdens de examinatie.

# Extra functionaliteit

Hier geven we een lijst van extra functionaliteiten. Deze zijn niet verplicht, maar dienen als steun of inspiratiebron voor het uitwerken van je eigen extra functionaliteiten. Zoals eerder vermeld is het de bedoeling dat je een eigen inbreng kan demonstreren.

---

## Live setup management

Met behulp van de Xbee discovery pakketten en het bijhouden van recent bevestigde (ACK) berichten is het mogelijk om een ‘live’ beeld te krijgen van de actieve sensoren in de kamers. Het is immers logisch dat bepaalde end-devices op termijn kapot gaan en vervangen worden. De detectie van langdurig inactieve apparaten kan dus ook deel uitmaken van deze uitbreiding.
Let op: je kan de scheme functie “xbee-list-nodes” hiervoor niet gebruiken. Deze functie houdt een lijst bij van alle apparaten die ontdekt zijn sinds de start van het programma. Deze lijst zal dus nooit kleiner worden als een apparaat niet meer antwoordt.

---

## Power budgettering

Elk actief apparaat verbruikt kostbare energie. Om kosten te besparen kunnen we met behulp van domotica en slimme software een power budgettering opstellen. Zo een power budgettering laat toe om defecte apparaten op te sporen die uitzonderlijk veel energie verbruiken (bv. een frigo waarbij de deur niet goed meer sluit zal bijzonder veel energie verbruiken). Dit kan gedetecteerd worden door een aantal gemiddeldes van het apparaat te bereken en die te weerspiegelen op de geschiedenis (weken, maanden, jaren geleden). Een oudere frigo zal immers altijd wat efficiëntie verliezen. Een ander luik van power budgettering is het opstellen van limieten per apparaat of kamer. Zo kan men bijvoorbeeld een game console (Xbox, PS4) automatisch laten uitschakelen wanneer deze ‘teveel’ energie verbruikt heeft op een dag, week of maand. Ook kan je een water boiler enkel actief maken op het nachttarief waardoor de gebruiker energiekosten kan besparen. Zoals voor vele extra’s: hoe intelligenter en geavanceerder de oplossing, hoe interessanter!
Tip: je kan bijvoorbeeld ook de Xbee API commando’s gebruiken om een energie threshold op een end-device te installeren. Zo kan je de server ontlasten van een teveel aan taken.

---

## Smart causality

Je zal ontdekken dat draadloze communicatie over het Xbee protocol geen garantie biedt op het aankomen van berichten. In de ‘smart causality’ functionaliteit denken we aan het garanderen en optimaliseren van de communicatie. Een voorbeeld: de gebruiker vergist zich van schakelaar om een bepaalde lamp uit te schakelen en drukt tweemaal heel snel op de knop (aan - uit). Aangezien het (POW ON) bericht naar de lamp toevallig mislukt was tussen dit moment, kunnen we detecteren dat een korte sequentie van (POW ON) en (POW OFF) eigenlijk betekend dat deze lamp helemaal niet aan moet staan en we dus de communicatie kunnen laten vallen. Dit bespaart energie voor de lamp en ontlast het draadloze spectrum.

## Advanced sleep mode & prioritisation

De meeste Xbee apparaten ondersteunen geavanceerde slaap commando's om energie te sparen. Zo kan je bijvoorbeeld de 'wake-up' van een apparaat minimaliseren. Een boiler kan bijvoorbeeld slechts om het uur naar het netwerk luisteren. Een slimme implementatie kan bijvoorbeeld rekening houden met het feit dat sommige apparaten minder luisteren en hun 'retry' cyclus voor gefaalde berichten minder agressief instellen. Het is ook zeker interessant om dan bijvoorbeeld andere berichten voorrang te geven zodat je het bericht voor je lamp niet moet wachten tot boiler eens luistert.

## Smart behaviors

Een koppeling tussen een Xbee schakelaar en een Xbee lamp is noodzakelijk voor een traditionele interactie in een huis. Dit vereist een systeem dat bepaalde 'events' kan omzetten naar andere 'acties'. Een bewegingssensor, tijd, sferen (feestje/romantisch) kunnen allerlei acties impliceren.

## Extra hardware

Je kan bij ons terecht om extra hardware te verkrijgen voor je eigen scenario op te stellen. Zo hebben we nog drukknoppen, tilt en magneet sensoren, Kinect camera's en meer.

## SRF support

Het ondersteunen van het SRF (serial radio frequency) protocol kan je ook als extra functionaliteit aanbieden. We hebben ook een hele hoop sensoren en relays die op SRF en LLAP[2] werken. Dit protocol is eenvoudige dan de XBee API frames en werkt met behulp van een usb stick op je raspberry pi. En 'Hello world' bericht zoals "a—HELLO——" kan je bijvoorbeeld rechtsreeks over de seriële poort versturen. De adressen zijn integers tussen 0 en 99 en passen in de eerste '—' symbolen (bv: "a01HELLO——" wordt verstuurd naar apparaat 01).

## Web API

Het aanbieden van functionaliteit over het web kan interessant zijn om Android clients te schrijven. Met een Web API bedoelden we niet een heuse website maar bv. een REST API: "PUT http://majordomo.myhome.be/kamer1/lamp2 (POW ON)"
Een voorbeeld van een REST interface kan je vinden op twitter: https://dev.twitter.com/docs/api/1.1.

[2] http://openmicros.org/index.php/articles/85-llap-lightweight-local-automation-protocol/297-llap-reference

# Kwaliteitseisen code

Algemene informatie omtrent de kwaliteitseisen vind je in het document “Kwaliteitseisen Programmeerprojecten Softwareproject en Systeemprogrammatuur” dat beschikbaar is op PointCarré. Lees dit document grondig! Bovendien zal je in de cursus Algo&Data 2 (prof. De Meuter) ondervinden hoe goede Scheme code, gebruik makende van modules, eruit ziet.

Het is vanzelfsprekend dat het hanteren van een consistente en leesbare code stijl vereist is. We leggen geen code stijl op, maar vragen wel de bestaande Scheme stijl conventies te hanteren. Als je niet zeker bent van je code stijl, word je aangeraden om ‘Scheme style guides’ online op te zoeken[3].

Specifiek voor dit project willen we nog het gebruik van modules benadrukken. Splits je toepassing op in kleine onderdelen. Dit helpt bijvoorbeeld om de grafische interface strikt gescheiden te houden van de rest van je toepassing. Praktisch gezien zal een software ontwerp met goed gekozen modules je veel tijd besparen bij het tweede onderdeel van dit project.

# Testing

Unit testen is een methode om softwaremodules of stukjes broncode afzonderlijk te testen. Bij unit testen zal voor iedere ADT een of meerdere tests ontwikkeld worden. Hierbij worden dan verschillende testcases doorlopen. In het ideale geval zijn alle testcases onafhankelijk van andere tests. Eventueel worden hiertoe stukken code geschreven die stukken van je programma "simuleren" om andere stukken te testen (bv. het communiceren met een sensor, het zelf aanmaken van objecten om de database aan te spreken, ...). Het doel van unit testen is om functionele units onafhankelijk van elkaar te kunnen testen op correcte werking. Telkens je iets wijzigt aan de code kan je dan je unit testen doorlopen om te controleren of alles nog werkt. Een batterij aan tests zal van onschatbaar waarde zijn wanneer je je broncode aanpast voor het tweede deel van dit project.

We vragen niet om unit tests te maken voor elke kleine functionaliteit. Wel vragen we een aantal automatische tests die ons kan helpen te overtuigen dat je functionaliteit wel degelijk werkt.

Ook vragen we je niet om de grafische interface te testen, enkel de functionaliteit van het programma. Indien je de indruk krijgt dat bepaalde applicatielogica die zich in de interface bevindt de moeite waard is om te testen, dan is dit waarschijnlijk een teken dat die code zich op een verkeerde plaats bevindt.

# Deliverables

Hieronder vind je een lijst van deliverables. Deze lijst beschrijft alles wat je moet indienen voor de examenperiode in het eerste semester. Voorbladen voor de documenten zijn niet nodig, beperk jezelf tot een titel. Let op dat je al je documenten voorziet van een inleiding

---

[3] Voorbeeld Scheme style guides: http://community.schemewiki.org/?scheme-style ; http://web.archive.org/web/20020809131500/www.cs.dartmouth.edu/~cs18/F2002/handouts/scheme-tips.html ; http://mumble.net/~campbell/scheme/style.txt

en een conclusie. Natuurlijk mag je zeker niet vergeten je naam en rolnummer te vermelden.

- Listings van de broncode van je werkende toepassing in Racket (+ pdf)
- Een handleiding van je toepassing gericht aan eindgebruikers
  - Algemene omschrijving van je toepassing
  - Beschrijving van alle functionaliteit, en hoe deze gebruikt kan worden
  - Eventueel screenshots
- Een specificatiedocument van je toepassing gericht naar ontwikkelaars.
  - Algemene omschrijving van je toepassing
  - Decompositie in modules. Beschrijving van de software architectuur, inclusief schematische voorstelling.
  - Beschrijving van de communicatieprotocols van de gesimuleerde sensoren
  - Steek alles in een zip bestandje met al je documenten en alle code (ook die van derden). Steek je documenten en je code in een mapje die je naam- en voornaam draagt (bv. christophe.debruyne) en geef dezelfde naam aan je zip bestand met een ".zip" extensie. Zorg dat je toepassing uitvoerbaar is vanuit DrRacket.

Je specificatiedocument moet allesomvattend zijn: het lezen van je document zou moeten volstaan om je code te kunnen begrijpen. Stel je bij het schrijven van je document voor dat iemand die nu in derde bachelor zit in de zomervakantie aan de slag moet met jouw project.

Het is toegestaan om code van derden te gebruiken. Let op, dit omvat niet code van medestudenten! Dit is een individueel project, het overnemen van broncode van medestudenten is plagiaat en wordt niet geduld. In alle gevallen moet je echter steeds zeer duidelijk aangeven in je documenten welke code van jezelf is, en welke je gebruikt. Code van derden hoef je niet af te drukken, maar wel te beschrijven in je specificatiedocument.

Alle in te dienen documenten converteer je naar PDF formaat. Je kan dit doen met bijvoorbeeld enscript of een PDF printer. Drop je documenten elektronisch in de dropbox van de cursus van dit vak op PointCarré gericht aan alle begeleidende assistenten. We aanvaarden geen MSWord, OpenOffice of andere bestanden. Je code geef je naast de PDF versie ook af in een zip bestand.

# Belangrijke data

| Actie | Deadline |
| --- | --- |
| Indienen finale versie code + documentatie | Maandag 26-05-2012 |
| Projectverdediging | Tweede zittijd |

Over bovenvermelde deadlines wordt niet gediscussieerd! Per dag te laat gaan vier punten van je totaalscore af. Bij het overschrijden van 4 dagen te laat, wordt je afwezig gequoteerd. Dit geldt voor alle deadlines!

# Hints en Tips

- Leg een eigen deadline vast die bijvoorbeeld een week of twee weken voor de echte deadline ligt. Zo zorg je ervoor dat je project zeker op tijd af is en dat je nog tijd genoeg hebt om een goed verslag te schrijven.
- Er wordt veel aandacht besteed aan het ontwerp en het gebruik van ADT's. Een project zonder ADT's of met slechte ADT's is per definitie een slecht project.
- Druk de listings van je code af zoals ze in de editor van je programmeeromgeving voorkomen. Bijgewerkte listings maken het veel moeilijker om de samenhang van de verschillende delen te begrijpen en geven een ongeloofwaardige indruk. Voorzie regelnummers en paginanummers zodat je een goede inhoudstafel kan maken van je code.
- Laat geen debugging code in de broncode staan. Het maakt je broncode slordig en schaadt de leesbaarheid.
- Lees je verslag na! Verslagen met veel schrijffouten worden niet geduld.
- Bij vragen of problemen kan u altijd terecht bij de projectbegeleiders op project2ba@dinf.vub.ac.be.

# Appendix A: User interface gebruikers scenario's

| Interactie voorbeelden | | |
|---|---|---|
| **Doel** | **Mogelijke sequentie** | **Uitgevoerde actie** |
| Toevoegen sensor | Selecteer type sensor &<br>Selecteer protocol sensor &<br>Ingeven naam &<br>Ingeven unieke identifier (bvb.: mac adres of sensor id) | Sensor wordt toegevoegd in de lijst van sensoren |
| Visualiseren energieverbruik | Selecteer sensor in een lijst (of grondplan) | De geschiedenis van verbruik wordt opgevraagd uit de persistentie laag en gevisualiseerd in een plot |
| Aanzetten van een lamp | Selecteer de lamp in een lijst (of grondplan), klik op een 'aan' knop om de lamp aan te schakelen. | De callback vertaalt de actie naar de juiste commando's naar het domotica raamwerk. Deze commando's worden dan doorgestuurd naar de correcte lamp. |
| Visualiseren actieve actuatoren | Klik op het overzicht om gedetailleerde informatie per sensor in een tabel te kunnen bekijken. Je kan op de kolommen klikken om te sorteren. | Een lijst met alle informatie van alle sensoren en actuatoren wordt getoond. Actuatoren (en/of sensoren) die actief zijn krijgen een ander kleur. |
| Bekijken van de acties | Klik op het log venster. | De laatste 300 acties van het domotica systeem moeten in text (of grafisch) formaat getoond kunnen worden. Bvb.: "Sensor-1 [Type: Lichtsensor] werd toegevoegd", "Lamp-2 werd uitgeschakeld". |

| OPTIONELE Interactie voorbeelden | | |
|---|---|---|
| Intelligente lamp: automatische activatie bij lage lightintensiteit | Maak een nieuwe regel aan en vul de minimale waarde van de lichtsensor in. Een timeout waarde (bvb 5 minuten) en de koppeling met de lamp & lichtsensor moet ook gekozen worden. | De lamp schakelt automatisch aan en uit op basis van de lichtintensiteit (zonder acties van de gebruiker). |
| In/Uitschakelen van een lamp op basis van tijd | Maak een nieuwe geplande taak aan. Vul het start en eindstip in. Koppel de taak aan een (of meerdere) lamp(en). | De lamp schakelt automatisch aan en uit op basis van tijd (zonder acties van de gebruiker). |