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 en maar een lijst van 'minimale' vereisten. Je wordt aangemoedigd om de originele opdracht te verrijken met eigen inbreng, zolang dit in de thema van de opdracht past.

Opdracht

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.
- 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.

Gezien de complexiteit om met werkelijke hardware te werken, splitsen we de opdracht in twee. In het eerste semester leggen we de nadruk op het schrijven van je domotica energie monitoring systeem, met bijhorende grafische interface en databank. Hierbij ligt de nadruk op het toepassen van de programmeer en software engineering technieken in een vertrouwde Scheme omgeving: Racket. In dit eerste deel van de opdracht zullen jullie de gebruikte sensoren op een simpele manier simuleren. In het tweede deel wordt deze simulatie vervangen door werkelijke hardware. Hierbij komen heel wat nieuwe concepten bij kijken, zoals het programmeren van een geïntegreerde computer, door middel van een gespecialiseerde Scheme omgeving.

Dit document beschrijft het eerste deel van de opdracht. Het tweede deel bouwt logischerwijs verder op het eerste deel, daarmee is het cruciaal dat dit eerste deel in orde is. Er is een tussentijdse examinatie op het einde van de eerste zittijd, waar je punten en feedback krijgt voor het eerste deel van je project.
Het tweede deel van de opdracht zal je worden voorgesteld bij de aanvang van het tweede semester. Je zal natuurlijk nog altijd aanpassingen kunnen doen aan de design en broncode van het eerste deel tijdens het tweede semester.

Communicatie met apparaten

De virtuele domotica opstelling waarmee we werken in dit eerste deel 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.

Je broncode voor dit eerste deel van de opgave draait virtueel op de majordomo. De minimale vereiste functionaliteit van de majordomo software wordt hier vastgelegd. De communicatie met en functionaliteit van de steward en apparaten zal je zelf moeten simuleren. De functionaliteit van de steward en apparaten worden pas vastgelegd in het tweede deel van de opdracht. We leggen niet vast hoe je deze simulatie moet opstellen, het ontwikkelen van een passend ontwerp is deel van de opdracht.

We leggen wel vast hoe je raamwerk met deze simulatie moet communiceren, dit om de functionaliteit in lijn te brengen met het tweede deel van de opdracht. Eerder dan dit communicatieprotocol formeel vast te leggen, doen we dit aan de hand van een scenario. Het communicatieprotocol scenario kan je vinden in de **Appendix A** onderaan dit document. Significante aanpassingen aan het gedemonstreerde protocol bespreek je best eerst met de begeleidende assistenten.

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 B**.

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.

Lokale databank

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.

Vereiste functionaliteit en extra's

Hieronder beschrijven we de functionaliteit die je toepassing zeker moet bevatten, alsook een aantal gesuggereerde features waaruit je kan kiezen. 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.

Minimale functionaliteit:

- Een **steward** ADT dat toelaat:
  - Een apparaat toe te voegen met naam, serienummer en communicatieadres.
  - Gegeven een naam, een communicatiekanaal te openen met een toegevoegd apparaat.
- Een **simulator** die het gedrag van minstens twee soorten apparaten kan simuleren over het communicatiekanaal, zoals beschreven in **Appendix A.** (bv. schakelaar, lichtsensor)
- Een **grafische interface** die de gebruikers scenario's ondersteunt uit **Appendix B.**
  - Eenmalige informatie op te vragen en visualiseren: wat is de temperatuur in mijn kamer op dit moment.

    - Periodieke informatie op te vragen en visualiseren: een tabel of grafiek die de gemiddelde temperatuur van mijn kamer weergeeft over de hele dag.
- **Persistentie:** verzamelde informatie en configuratie wordt op een automatische en transparante manier opgeslagen in een database. Het herstarten van Racket mag geen invloed hebben op het verdere functioneren van je applicatie.

Extra functionaliteit:

- Een user account systeem met login en toegangscontrole: ouders kunnen alles veranderen, kinderen alleen informatie opvragen.
- Extra soorten apparaten simuleren: motion sensors, ramen/gordijnen/deuren, ...
- Een 'energy credits' systeem dat toelaat om energieverbruik te budgetteren over een gezien: limieten instellen, automatisch afsluiten, gelimiteerd verbruik periodes, ...
- Uitgebreide data visualisatie. Bijvoorbeeld een plattegrond waar sensoren op kunnen gevisualiseerd en aangestuurd worden.
- Uitgebreide automatische scenario's, zoals lichten die automatisch uitgaan als je een tijdje niet in de kamer bent of een anti-diefstal 'ik ben nog thuis' scenario die lichten doet aan en uit gaat alsof je nog rondloopt.
- Een web applicatie als GUI (zie bij opdracht boven).

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.

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 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.

Tussentijdse examinatie

Tijdens de eerste examenperiode organiseren we een tussentijdse examinatie die punten oplevert voor het eerste deel van het project. Hierbij krijgt elke student 15 minuten: ongeveer vijf minuten demonstratie van functionaliteiten, vijf minuten uitleg bij je ontwerp en vijf minuten vragen. Dit telt als een partieel examen, waarbij je als je niet tevreden bent met je score het examen opnieuw kan doen tijdens de tweede examenperiode. De tussentijdse examinatie laat ons ook toe om mogelijke problemen met je project te ontdekken en aan te pakken voor het tweede deel aanvat.

Belangrijke data

Actie	Deadline
Indienen tussentijdse versie code + documentatie op Pointcarré	Woensdag 08-01-2014 (23.59h)
Eerste tussentijdse evaluatie	Laatste week van de eerste zittijd
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: Apparaat Communicatieprotocol Scenario

```
;; Ik vraag een input/output port aan het steward ADT
;;  voor de slaapkamer, om met een specifieke lamp te
;;  'praten'.
> (define slaapkamer-hoofdlamp-port
    (slaapkamer-steward 'open-device-io-port 'hoofdlamp))

;; Dit zorgt ervoor dat ik niet
;;  'slaapkamer-hoofdlamp-port' moet heralen bij elke
;;  (read) en (write) commando.
> (current-input-port slaapkamer-hoofdlamp-port)
> (current-output-port slaapkamer-hoofdlamp-port)

;; Brandt het licht in de slaapkamer?
> (write '(GET POW) slaapkamer-hoofdlamp-port)
> (read)
'(ACK (POW OFF))

;; Een vraag of antwoord is een lijst van symbolen,
;;  met het eerste element een identificatie en met
;;  (PARAMETER . WAARDEN) lijsten als volgende elementen.

;; Zet het licht aan.
> (write '(PUT (POW ON)) slaapkamer-hoofdlamp-port)

;; Een commando kan een antwoord veroorzaken vanuit het
;;  apparaat, zo kan je nagaan of het commando goed is
;;  ontvangen en afgehandeld.
> (read)
'(ACK (POW ON))

;; Bij een mogelijk probleem, bijvoorbeeld een typfout,
;;  krijg je het volgende antwoord.
> (write '(PPUT OOPS TYPFOUT) slaapkamer-hoofdlamp-port)
'(NACK (PPUT OOPS TYPFOUT))

;; Sommige commandos veroorzaken een gebundeld antwoord.
> (write '(GET ALL))
'(ACK (POW ON) (LOAD 62Watt) (WORK 12.60kWh) (TEMP 21.1C)
      (FREQ 50.00Hz) (VRMS 230V) (IRMS 10mA))

;; Het volgende is _niet verplicht_,
;;  maar is een interessante uitbreiding
;; Zend me (asynchroon) een berichtje als het verbruik
;;  de grens van 15 kWh overschrijdt.
> (write '(METER (WORK 15 MSG)))

;; Een (read) zal hier blijven wachten tot er een berichtje
;;   ontvangen is, wat lang kan duren.
;; We gebruiken daarom 'thread' van Racket om dit concurrent
;;  te laten verlopen. Lees goed de Racket documentatie
;;  vooraleer je hieraan begint.
> (thread (lambda ()
          (display (read))))
> (display "waiting ...")
waiting...
(WORK)
```

Appendix B: User interface gebruikers scenario's

Interactie voorbeelden		
Doel	Mogelijke sequentie	Uitgevoerde actie
Toevoegen sensor	Selecteer type sensor & Selecteer protocol sensor & Ingeven naam & 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).