2015•2016
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
master in de industriële wetenschappen: elektronica-ICT

Masterproef

Externe sturing van de NAO-robot voor mens-robot-communicatie bij autistische kinderen

Promotor :
Prof. dr. ir. Nele MENTENS

Promotor :
dr. LUDO CUYPERS

Willem Geerts

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

KU LEUVEN | universiteit hasselt

2015•2016

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterproef

Externe sturing van de NAO-robot voor mens-robot-communicatie bij autistische kinderen

Promotor :
Prof. dr. ir. Nele MENTENS

Promotor :
dr. LUDO CUYPERS

Willem Geerts

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

universiteit hasselt | KU LEUVEN

Woord vooraf

Na mijn schakeljaar rond ik mijn opleiding tot industrieel ingenieur Elektronica-ICT aan de Universiteit Hasselt en KU Leuven af met mijn masterthesis voor COMmeto in Ham.

Tijdens deze periode heb ik veel technische en praktische kennis opgedaan. Het was een leerrijke en interessante ervaring om bij COMmeto mijn masterthesis af te leggen.

Graag zou ik de volgende personen willen bedanken voor hun hulp en goede begeleiding om deze masterthesis tot een goed einde te brengen:

- Mijn bedrijfspromotor Dr. Ludo Cuypers, voor zijn verdiepende kennis en inzicht en het feit dat ik altijd bij hem terecht kon;
- Mijn interne promotoren Prof. Dr. Nele Mentens en Ruben Smeets, voor het opvolgen van mijn masterproef;
- Michel D'havé, Dr. Todor Gerganov, Eddy Wissels, voor de aangename sfeer en steun bij COMmeto;
- Jessica De Smedt, studente aan de Universiteit Antwerpen, voor de samenwerking om deze masterproef tot een goed einde te brengen;
- Prof. Dr. Walter Daelemans en Dr. Guy de Pauw van CLiPS voor de begeleiding van Jessica De Smedt en hun vertrouwen in mij;
- Mijn ouders en mijn vriendin voor hun onvoorwaardelijke steun.

Willem Geerts

Heusden-Zolder, juni 2016

Inhoudsopgave

Lijst van tabellen

Lijst van figuren

Verklarende woordenlijst

API: Application Programming Interface

CLiPS: (Computational Linguistics & Psycholinguistics Research Center) Onderzoekscentrum in samenwerking met het departement linguïstiek van de Universiteit van Antwerpen

Compiler: Computerprogramma dat een in een brontaal geschreven programma vertaalt in een semantisch equivalent programma in een doeltaal

DCM: Device Communication Manager

FSR: Force Sensitive Resistors

IDE: Integrated Development Environment

IP: Internet Protocol

MP3: MPEG-1 Layer 3, manier om geluid te comprimeren

NAO: Zelfstandige, mensachtige robot ontworpen door Aldebaran Robotics

Ogg: Bestandsformaat dat gebruikt wordt om audio te streamen, vrij van patenten

RAM: Random-Access Memory of het werkgeheugen (vluchtig geheugen)

RGB: Is een kleurcodering op basis van de drie primaire kleuren Rood Groen Blauw

ROM: Read-only Memory (niet-vluchtige geheugen)

SDK: Software Development Kit

WAV of WAVE: Audio formaat

ZORA: Zorg Ouderen Revalidatie en (Aan)valdetectie

Abstract

De universiteit van Antwerpen heeft vorig jaar een onderzoek gedaan naar de communicatie tussen robot en mens, meer bepaald de communicatie tussen robots en autistische kinderen. Hieruit bleek dat robots een positieve bijdrage hadden tijdens therapieën. Momenteel wordt er onderzocht wanneer de robot introvert of extravert moet reageren. Om de robot in reële tijd aan te sturen doet men beroep op COMmeto.

Deze masterproef onderzoekt op welke manier de robot moet worden geprogrammeerd en of de specificaties voldoen aan de nodige vereisten. Bewegingen maken, sensoren uitlezen, audio ontvangen en verzenden zijn enkele van de aspecten die aan bod komen. Aldebaran, de ontwikkelaar van de NAO robot, ondersteunt meerdere programmeertalen voor het programmeren van de robot. Deze talen worden met elkaar vergeleken en getest om zo uit te maken met welke taal er uiteindelijk wordt geprogrammeerd. Verder moeten de gevraagde scenario's van de UAntwerpen gecodeerd worden voor hun experimenten.

Het resultaat van deze masterproef is dat de robot in reële tijd met Java door een extern werkstation wordt aangestuurd. Deze opstelling is gemakkelijk toepasbaar voor therapieën met autistische kinderen.

Summary

Last year, the university of Antwerp conducted research on the communication between robots and humans, more specifically the communication between robots and autistic children. This showed that robots have a positive contribution during therapy sessions. Currently, research is being done to know when a robot should react in an introverted or extraverted way. COMmeto programs these robots for real time behaviour.

This master's thesis explores how the robots should be programmed and whether the specifications satisfy the requirements. Making movements, reading sensors, receiving and sending audio are some of the aspects that are discussed. Aldebaran, the developer of the NAO robot, supports multiple programming languages. These languages are compared and tested in order to determine which language is best suited to be used to program. Furthermore, the scenarios of the university of Antwerp have to be coded for their experiments.

The result of this master's thesis is that the robot is controlled in real time with Java by an external workstation. This setup is easily applicable in therapies with autistic children.

1 Inleiding

1.1 Situering

De opkomst van robots in ons leven neemt steeds toe, meer nog, ze worden zelfs onmisbaar. Ook in ziekenhuizen worden ze steeds meer toegepast als extra hulp. In het Universitair Ziekenhuis van Gent maken ze gebruik van robot ZORA, die mensen interactieve zorgondersteuning biedt [1]. De universiteit van Antwerpen heeft vorig jaar een onderzoek gedaan naar de communicatie tussen de mens en een robot [2]. Voor hun onderzoek hebben ze gebruik gemaakt van een NAO robot, zoals getoond wordt in Figuur 1.

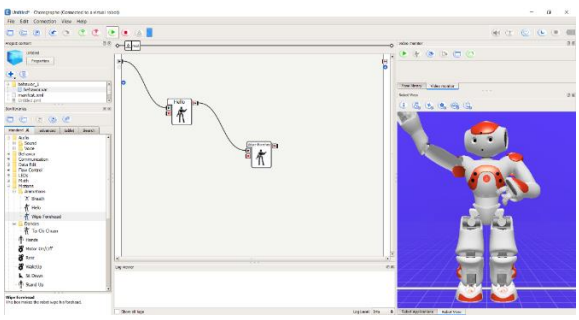
Figuur 1: NAO robot V4 Next Gen H25

De universiteit van Antwerpen nam COMmeto onder de arm om zich op het gebied van software te laten ondersteunen. COMmeto is een bedrijf dat zich o.a. toespitst op de software programmatie in projecten zoals FallRisk en AXO-SUIT. Voor dit project werd verwacht dat de NAO robot zo geprogrammeerd zou worden dat deze in reële tijd kan worden aangestuurd.

1.2 Probleemstelling

De Universiteit van Antwerpen heeft een onderzoek gedaan naar de communicatie tussen de NAO robot en de mens. Het vervolg op deze studie loopt verder en houdt in dat er onderzocht wordt of de NAO robot introvert of extravert moet reageren tegenover mensen met een ontwikkelingsstoornis, voornamelijk kinderen met autisme. Zo wordt er getracht om hun leven aangenamer te maken en hen te leren hoe ze moeten omgaan met andere mensen. Samen met COMmeto werd het software ontwerp en de programmatie voor de NAO robot gerealiseerd.

Voor het programmeren van de NAO robot is het belangrijk dat er onderzocht wordt in welke programmeertaal dit gebeurt. Choregraphe Figuur 2van Aldebaran Robotics (Figuur 2) is het meegeleverde programma van de robot. In dit computerprogramma kan men met het *drag and drop* systeem een scenario ontwikkelen voor het bewegen van de robot. Deze scenario's worden dan verzonden naar de robot; verder is het ook mogelijk om de scenario's te simuleren.



Figuur 2: Choregraphe van Aldebaran Robotics

Om in reële tijd te werk te gaan is kan er niet worden gewerkt met voorgeprogrammeerde scenario's in Choregraphe. Er is dus een andere programmeeromgeving en programmeertaal nodig om het aansturen van de robot zo vlot mogelijk te laten verlopen.

Vervolgens moeten de specificaties van de robot worden onderzocht: voldoen deze aan de eisen of moet er gewerkt worden met externe randapparatuur?

1.3 Doelstellingen

Om het project in goede banen te leiden zijn er enkele doelstellingen opgesteld die moesten worden behaald. Er zijn enkele hoofddoelstellingen die zeker moesten worden volbracht om tot een volwaardige masterproef te komen.

1.3.1 Programmeertaal

De NAO robot beschikt over een eigen platform: OpenNAO. Dit is een Linux distributie gebaseerd op Linux Gentoo. Om in reële tijd deze robot te besturen werd er onderzocht welke programmeertaal het meest geschikt is.

1.3.2 Specificaties van de NAO robot

Wat is er mogelijk met de interne sensoren, motoren, camera's? Voldoen deze zaken aan de vereisten en is het mogelijk om deze in reële tijd aan te sturen? Kunnen de inkomende data van de microfoon snel genoeg verwerkt worden zodat er een minimale vertraging is om erna de robot op de "juiste" manier te laten reageren? Met de "juiste" manier wordt in dit geval introvert of extrovert bedoeld.

1.3.3 Programmatie

Zoals eerder vermeld werkt de robot met het OpenNAO platform. Dit platform bevat een NAOqi Framework en zorgt voor de communicatie tussen de verschillende API modules. De API modules zijn verdeeld in 9 groepen:

- Core
- Motion
- Audio
- Vision
- People Perception
- Sensors
- Trackers
- Diagnosis
- DCM (Device Communication Manager) [3]

Elke van deze API modules bevat een reeks aan methodes. Met deze methodes werd de NAO robot geprogrammeerd.

Samen met de Universiteit van Antwerpen werd er bekeken hoe de robot moet reageren. Eerst worden er scenario's uitgedacht en eens dit vlot werkt gaan we over naar de volgende stap. Deze stap houdt in dat de robot geen scenario's zal gebruiken maar zelf weet hoe te reageren.

1.4 Methode en materiaal

Iedere doelstelling moest op een eigen methode worden opgelost. Hieronder wordt er besproken hoe er te werk werd gegaan.

1.4.1 Programmeertaal

Voor dit project moest er dus worden nagegaan welke programmeertaal er best kon gebruikt worden. Hier ging een literatuurstudie aan vooraf. Op de website van Aldebaran Robotics bleek dat er in totaal een viertal talen in aanmerking kwamen [4].

- C++
- Python
- Java
- Javascript

In de vorige versie 1.14 van de documentatie kwamen .NET, Urbi en Matlab nog aan bod, maar deze zijn in de huidige versie 2.1 geschrapt [5]. Eens de programmeertaal gekozen was, kon er vervolgens worden overgegaan tot het beheersen van deze taal.

1.4.2 Specificaties van de NAO robot

Een doelstelling was om de specificaties te onderzoeken en doorheen het project te bekijken of deze wel voldoen aan de vereisten. De vereisten waren dat de robot woorden in het Nederlands kon verstaan en dat dit snel genoeg zou verwerkt worden. Om een voorbeeld te geven, de Universiteit van Antwerpen vond de spraakherkenning in de robot maar matig. De robot verstond niet altijd wat men zei. Maar als deze spraakherkenning zou verbeterd worden, kan de CPU van de robot dit dan nog aan, of moet een extern werkstation dit afhandelen in communicatie met de robot? Zodra de ingebouwde elektronica van de robot niet voldoet om de taak af te handelen moet er worden gezocht naar een oplossing via een extern werkstation.

1.4.3 Programmatie

Eens de programmeertaal gekend was, kon er worden overgegaan naar de volgende stap. Hierin speelde het aanroepen van de modules een grote rol. Aan de hand van voorbeelden en met de handleiding van Aldebaran moest er resultaat worden geboekt. Eerst en vooral moest een *proof of concept* tot een recht eind worden gebracht. Eens dit gelukt was kon er worden overgegaan naar de afwerking van het spel en de verhaaltjes voor een introverte of extraverte robot.

2 Literatuurstudie

2.1 Communicatie tussen de mens en de robot

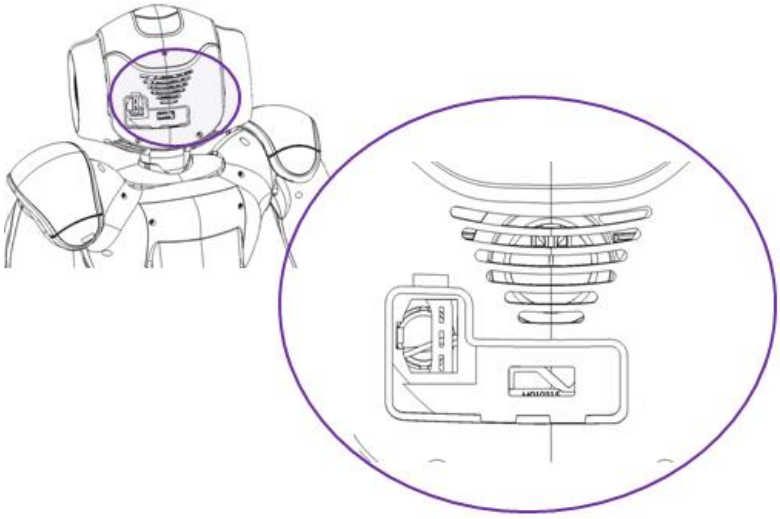
Vorig academiejaar onderzocht Jessica De Smedt voor haar bachelorthesis de interactie tussen mens en robot. In het Engels wordt dit Human-Robot Interaction (HRI) genoemd. In dit onderzoek onderzocht ze eerst het natuurlijke taalgebruik van een robot. Vervolgens onderzocht ze de HRI bij verschillende robots, onder meer bij de NAO robot. Ook werden de robots vergeleken op het gebied van emotieherkenning bij de mens. Uit een ander onderzoek bleek dat de lichaamstaal een belangrijke factor is om te communiceren met de mens en om zo toch emoties mee te geven. Uiteindelijk was haar conclusie dat de NAO robot een zeer bruikbare robot was voor therapieën met autistische kinderen of voor mensen met diabetisch. Voor haar masterthesis werd de NAO robot geprogrammeerd voor een aantal experimenten.

2.2 Specificaties van de NAO robot

Alvorens het programmeren kon beginnen werd er nagegaan wat de specificaties van de NAO robot waren. Hiervoor werd er gebruik gemaakt van de documentatie van Aldebaran Robotics.

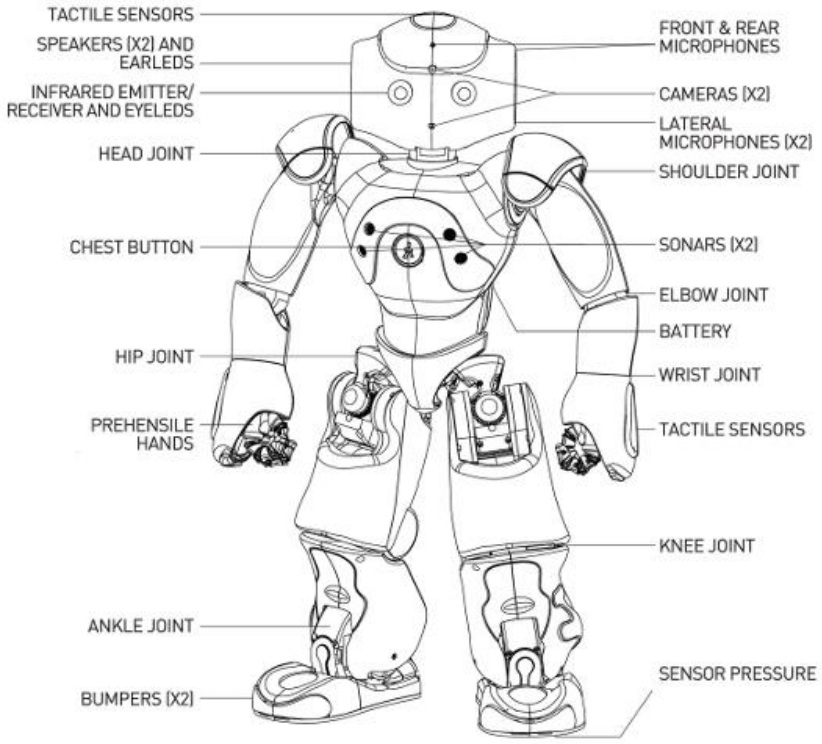
2.2.1 Versies

Sinds 2008 zijn er al enkele versies van de NAO robot op de markt verschenen. Tijdens dit project werd er gebruik gemaakt van een Next Gen V4 met als opbouw een H25. De verschillende versies zijn te onderscheiden aan de achterkant van het hoofdje (Figuur 3) [6].



Figuur 3: Gedetailleerde achterkant van versie 4 [6]

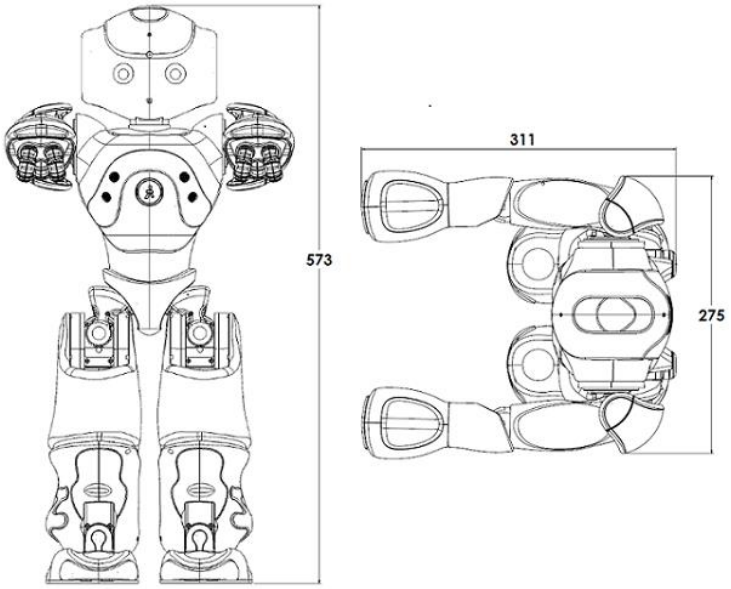
De opbouw van de gebruikte robot is een H25, deze heeft meer sensoren dan zijn vorige versie, H21. De H25 heeft als verschil dat er aanraaksensoren zijn aan de handen en dat er druksensoren zijn onder de voeten. In Figuur 4 zien we de gedetailleerde opbouw van de NAO robot [6].



Figuur 4: Gedetailleerde opbouw van de H25 [6]

2.2.2 Afmetingen en gewicht

De robot heeft een lengte van 573 mm, een breedte van 275 mm en een diepte van 311 mm zoals je kan zien in Figuur 5. Doordat de robot is gemaakt uit kunststof (ABS-PC, PA-66 en XCF-30) kon het gewicht gereduceerd worden tot 5,4 kg [7].



Figuur 5: Afmetingen van NAO robot V4 [7]

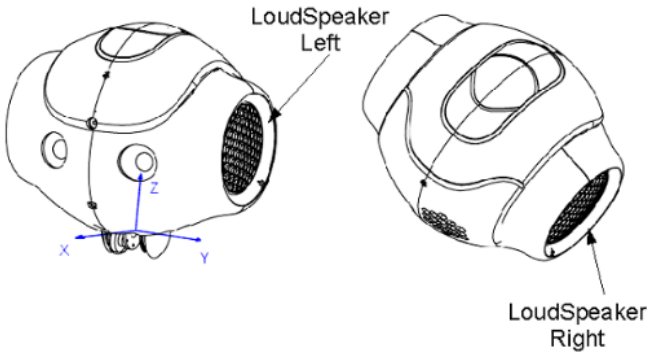
2.2.3 Processor en batterij

Aldebaran heeft voor een Intel ATOM Z530 met een kloksnelheid van 1,6 GHz gekozen. Deze 32-bit processor zorgt ervoor dat de robot de geprogrammeerde code kan verwerken [8]. Verder bevat de robot ook nog 1 GB RAM geheugen, 2 GB ROM geheugen en 8 GB Micro SDHC.

In de rug van de NAO zit de batterij. Met deze lithium ion batterij kan de robot één tot anderhalf uur actief bewegen. Het duurt ongeveer een drietal uur om de batterij volledig op te laden. De robot kan ook gebruikt worden tijdens het opladen [9].

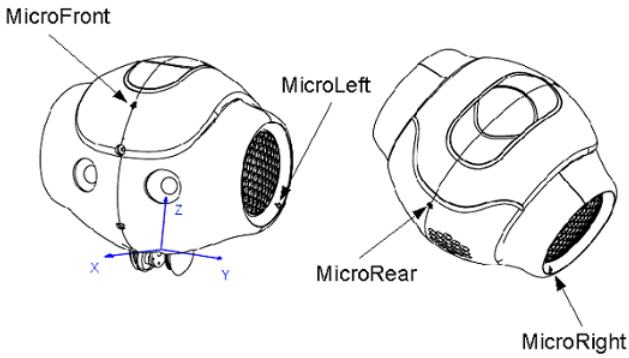
2.2.4 Luidsprekers, microfoons, camera en leds

In de oren van de NAO robot zitten er twee luidsprekers (16 bits stereo). Deze kunnen gebruikt worden om WAV, MP3 of Ogg bestanden mee af te spelen (Figuur 6) [10].



Figuur 6: Luidsprekers van de NAO robot (16 bits stereo) [10]

Er bevinden zich vier microfoons in het hoofdje, één in ieder oor en dan nog één vooraan en achteraan (Figuur 7) [11].

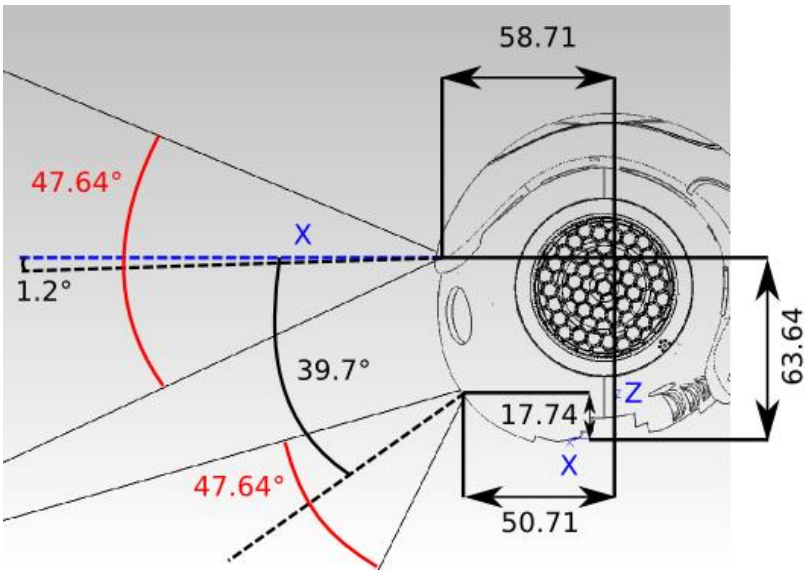


Figuur 7: Microfoons van de NAO robot [11]

De 4 kanalen kunnen zowel *interleaved* als *deinterleaved* aan 48 kHz worden ontvangen of elk kanaal apart aan 16 kHz. *Interleaved* is een techniek waarbij de data blokken van de signalen opeenvolgend in elkaar worden gevlochten. Bij *deinterleaved* worden de data blokken achter elkaar geplakt. In Figuur 8 wordt er een voorbeeld getoond.

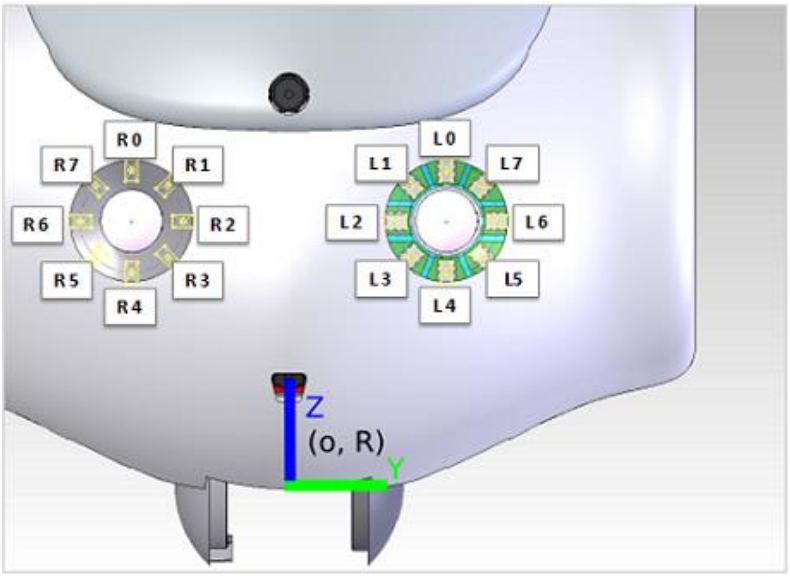
Figuur 8: Deinterleaved vs interleaved

Tussen de ogen van de robot bevinden zich twee camera's verticaal onder elkaar (Figuur 9). Deze twee camera's zijn identiek. Deze halen een resolutie tot 1280 x 960 pixels met een snelheid van 30 frames per seconde. Deze camera's worden onder andere gebruikt voor het herkennen van object of gezichten. Voor het ontvangen van de camerabeelden zijn heel wat parameters die kunnen worden ingesteld om het gewenst beeld te verkrijgen [12].



Figuur 9: Camera's van de NAO robot [12]

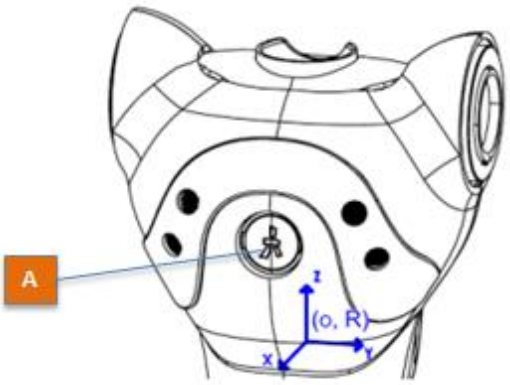
Dan zijn er nog de leds. Deze bevinden zich op het hoofd, in de oren en in de ogen van de NAO robot deze kunnen allemaal afzonderlijk worden aangestuurd. Deze leds kunnen ook van kleur veranderen door deze een RGB waarde mee te geven [13]. Elk oog bevat acht leds zoals in Figuur 10 te zien is.



Figuur 10: Leds in de ogen van de NAO robot [13]

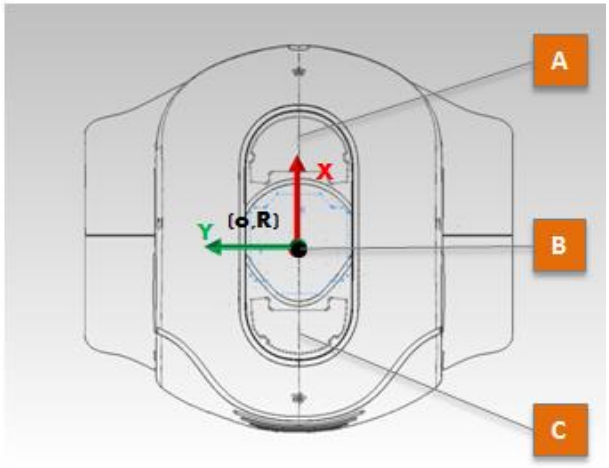
2.2.5 Drukknop en sensors

De drukknop op de borst (Figuur 11) is de belangrijkste drukknop, hiermee wordt de NAO robot aan of uitgezet. Zodra de robot is opgestart kan er op de drukknop gedrukt worden voor ander functies. Door één keer op deze knop te drukken zegt de robot zijn vooraf ingestelde naam luidop en vervolgens zegt hij zijn IP-adres. Door twee keer kort achtereen op deze knop te drukken gaat de robot wisselen van stand, ofwel gaat de robot in ruststand ofwel gaat hij rechtop staan. Door drie keer te drukken wordt het autonoom leven van de robot in- of uitgeschakeld [14].

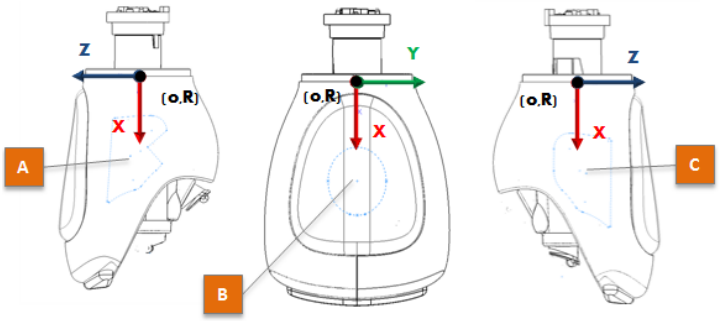


Figuur 11: Drukknop op de borst van de NAO robot [14]

Aan de bovenkant van het hoofd en rond elke hand zitten drie capacitieve sensoren (Figuur 12 en Figuur 13). Deze kunnen gebruikt worden om bepaalde acties te ondernemen door middel van aanraking, waardoor de capaciteit van deze sensor wijzigt [14].

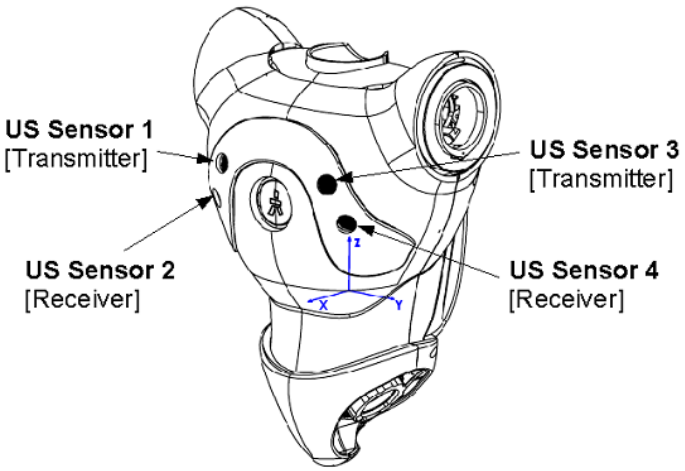


Figuur 12: Sensors op het hoofd [14]



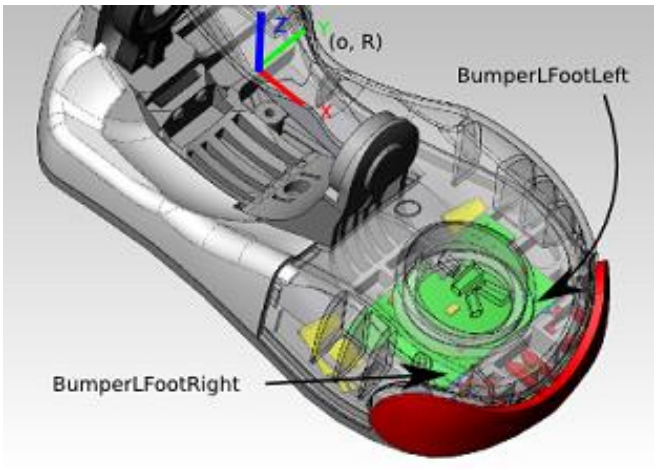
Figuur 13: Sensors rondom de hand [14]

Op de borst van de robot zitten twee ultrasone sensoren. Een ultrasoon sensor zendt geluid uit op een frequentie die niet waarneembaar is met het menselijke oor. De NAO robot zendt geluidsgolven met een frequentie van 40 kHz uit. Deze kunnen gebruikt worden om afstand van een object tot de robot te bepalen. Als de *transmitter* een geluidsgolf uitzendt tegen een object zal deze geluidsgolf weerkaatsen en ontvangen worden door de *receiver*, doormiddel van de tijd kan de afstand hierdoor worden bepaald. Het is mogelijk om afstanden te bepalen van 0,25 m tot 2,55 m. De bovenste twee sensoren zijn de *transmitters* en de twee onderste de *receivers* (Figuur 14) [15].



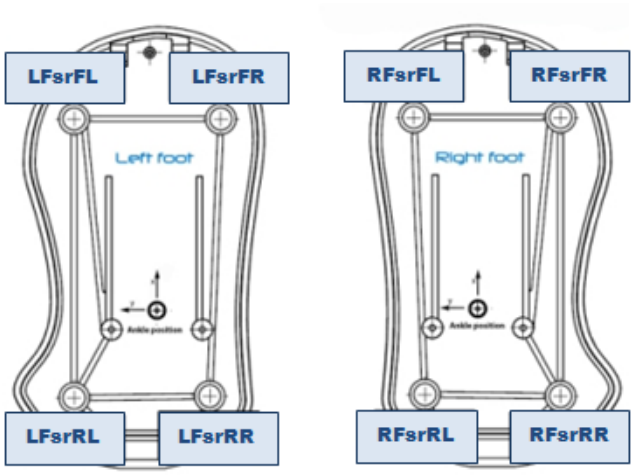
Figuur 14:Ultrasone sensoren [15]

Ten slotte aan de voeten zijn er telkens twee *bumpers*. Figuur 15 toont dit hier onder. Een *bumper* is een gewone simpele aan- en uitschakelaar. Eén aan de linkerkant en één aan de rechterkant.



Figuur 15: Bumpers aan de voeten [15]

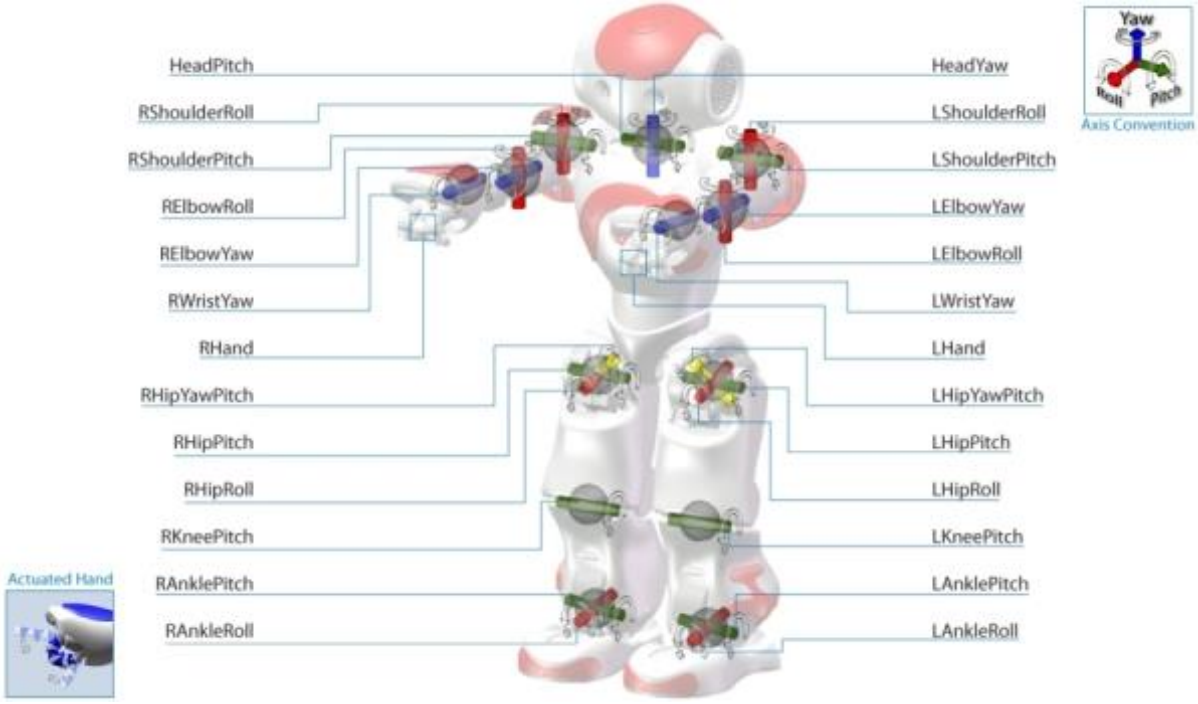
Onder de voeten zitten nog FSR's (Force Sensitive Resistors). FSR's zijn sensoren die de kracht die op de voeten wordt uitgeoefend meten. Deze FSR's kunnen van 0 tot 25 Newton meten. Aan elke voet zijn er vier zoals in onderstaande afbeelding (Figuur 16) te zien is [16].



Figuur 16: FSR sensoren onder de voeten van de NAO robot [16]

2.2.6 Actuatoren

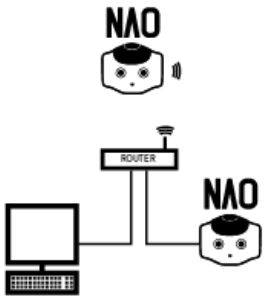
Om bewegingen te maken moeten de motoren bekrachtigd worden, waarvoor de NAO robot exact 25 actuatoren heeft (Figuur 17). Deze zijn opgedeeld in vijf groepen: het hoofd, de linker en rechter arm, en het linker en rechter been. Al deze actuatoren zorgen er voor dat de lichaamsdelen in een driedimensionale omgeving kunnen bewegen [17].



Figuur 17: Actuatoren van de NAO robot [17]

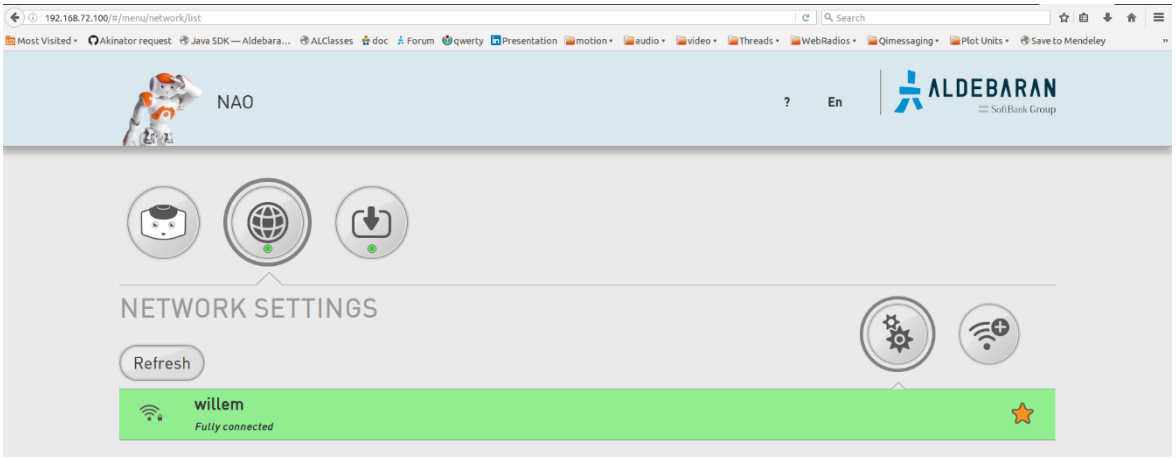
2.2.7 Connectiviteit

Er zijn twee manieren om een verbinding te maken met de robot. De eerste manier is door een ethernet kabel achteraan in het hoofdje te verbinden met de router. Zo krijgt de NAO robot een IP adres van de router en kan er een verbinding worden opgezet. Een tweede manier is via het draadloze netwerk, maar dan moet het draadloze netwerk wel worden ingesteld op de robot. Door de NAO robot eerst bekabeld aan het netwerk te hangen krijgt deze een IP adres. Door op de borstknop van de NAO robot eenmaal te drukken zegt de robot zijn gekregen IP adres. De mogelijkheden voor te connecteren met de NAO robot worden getoond in Figuur 18.



Figuur 18: Mogelijkheden voor te connecteren met de NAO robot

Vervolgens wordt een *internetbrowser* geopend en wordt er gesurft naar dit IP adres. Dan moet er worden *ingelogd*, de standaard gebruikersnaam en wachtwoord is ‘nao’. Eens *ingelogd* moet er worden verbonden met het draadloze netwerk en vanaf dan kan de robot over dit draadloos netwerk worden aangestuurd. De webpagina ziet er als volgt uit (Figuur 19).



Figuur 19: Webpagina voor het instellen van de draadloze netwerk instellingen

2.3 Programmeertalen voor de NAO robot

De NAO robot wordt aangestuurd met Choregraphe. In de bibliotheek hiervan zitten er voorgeprogrammeerde bewegingen en acties. Het is ook mogelijk om met Python in Choregraphe scenario's te programmeren. Omdat men in Choregraphe de robot niet kan aansturen in reële tijd, moest er voor een andere programmeermethode gekozen worden. In de documentatie van Aldebaran staat dat de volgende vier programmeertalen ondersteund worden. Deze worden getoond in Tabel 1 waar er ook nog een onderscheid wordt gemaakt of de code op de computer staat of op de robot. Het rechter deel van de tabel is voor dit project niet relevant.

Tabel 1:Ondersteunde programmeertalen

Programming Languages	Bindings running on		Choregraphe support	
	Computer	Robot	Build Apps	Edit code
Python	✔	✔	✔	✔
C++	✔	✔	⊘	⊘
Java	✔		⊘	⊘
JavaScript	✔	✔	✔	⊘

✔ OK
⊘ Not available
Working on it

De onderstaande programmeertalen worden besproken voor het werkstation.

2.3.1 Python

Python is de eenvoudigste programmeertaal van de vier voorgestelde talen. Om python te gebruiken moet wel eerst de Python SDK worden geïnstalleerd, deze staat op de website van Aldebaran. Deze SDK is te verkrijgen voor Windows, Linux en Mac. De python API laat ons toe om de robot op afstand aan te sturen door middel van de modules. Om hiervan gebruik te maken moet eerst de ALProxy worden geïmplementeerd zoals in de tweede regel van de afbeelding hier onderaan. De ALProxy zorgt er voor dat er een connectie met de robot kan worden opgezet. Hierbij wordt het IP adres van de robot en een poortnummer waarop deze luistert meegeven. Vervolgens moet er een ALProxy aangemaakt worden om één van de modules te gebruiken. Ten slotte wordt een methode van de aangemaakte module aangeroepen (Figuur 20) [18].

```
#Importeren van de ALProxy
from naoqi import ALProxy

#De module ALTextToSpeech aanroepen met het IP adres van de robot
tts = ALProxy("ALTextToSpeech", "<IP van de NAO robot>", 9559)

#De methode say aanroepen waardoor de robot 'Hallo' zegt
tts.say("Hallo!")
```

Figuur 20: Voorbeeldcode in python zodat de NAO robot hallo zegt

2.3.2 C++

De robot kan ook worden aangestuurd met C++. Om aan de slag te gaan met C++ moet er aan de volgende voorwaarden worden voldaan op een manier die eigen is aan elk besturingssysteem. Als eerste moet er een *compiler* en een IDE worden geïnstalleerd, voor Windows is dit bijvoorbeeld Visual Studio. Ten tweede moet de C++ SDK ondersteund worden door qiBuild. Dit wordt gebruikt om de projecten te beheren. Om qiBuild te kunnen installeren is er een CMake versie 2.8.3 of hoger vereist. CMake genereert de '*makefiles'* en de '*workspaces'* om later de code te compileren. Als dit eenmaal geïnstalleerd is gaat men over tot het instaleren van de C++ SDK, deze is te vinden is op de Aldebaran website. Uiteindelijk kan de geprogrammeerde code gecompileerd en uitgevoerd worden op de robot. In de documentatie staan er een paar voorbeelden. Figuur 21 laat een simpel voorbeeld hiervan zien.

```
**
 * Copyright (c) 2011 Aldebaran Robotics. All Rights Reserved
 * \file sayhelloworld.cpp
 * \brief Make NAO say a short phrase.
 *
 * A simple example showing how to make NAO say a short phrase using the
 * specialized proxy ALTextToSpeechProxy.
 */


#include <iostream>
#include <alerror/alerror.h>
#include <alproxies/altexttospeechproxy.h>

int main(int argc, char* argv[])
{
  if(argc != 2)
  {
    std::cerr << "Wrong number of arguments!" << std::endl;
    std::cerr << "Usage: say NAO_IP" << std::endl;
    exit(2);
  }

  /** The desired phrase to be said. */
  const std::string phraseToSay = "Hello world";
  try
  {
    /** Create an ALTextToSpeechProxy so that we can call the say method
    * Arguments for the constructor are:
    *  - IP of the robot
    *  - port on which NAOqi is listening. Default is 9559
    */
    AL::ALTextToSpeechProxy tts(argv[1], 9559);

    /** Call the say method */
    tts.say(phraseToSay);
  }
  catch (const AL::ALError& e)
  {
    std::cerr << "Caught exception: " << e.what() << std::endl;
    exit(1);
  }
  exit(0);
}
```

Figuur 21: Voorbeeldcode in C++

2.3.3 Java

In de documentatie van de Java SDK staat dat ze nog volop aan het werken zijn aan de API hiervoor en dat er nog wijzigingen kunnen zijn in de toekomst. Bij het gebruik van Eclipse, een programmeeromgeving voor onder meer Java, moet er telkens bij een nieuw project een externe '.jar' bestand worden toegevoegd zodat de API's toepasbaar zijn. Eclipse zal eerste de geprogrammeerde code compileren en dan wordt deze verzonden en gelopen op de robot. Het voordeel van Java is dat het op meerdere platformen kan uitgevoerd worden. Dus de geprogrammeerde code kan op een computer met Linux, Mac of Windows worden gedraaid of zelfs op een smartphone met Android. Om te kunnen communiceren met de robot zijn er twee belangrijke elementen. Eén de *Application* en twee de *Session*. Per programma is er maar één *application* die wordt gestart, voor elke nieuw object van de Aldebaran bibliotheek wordt er een *session* aangemaakt [19]. In onderstaande afbeelding wordt er getoond hoe men hiermee te werk gaat (Figuur 22).

```
package com.aldebaran.examples;

import com.aldebaran.qi.Application;
import com.aldebaran.qi.helper.proxies.ALTextToSpeech;

public class SayHello {

    public static void main(String[] args) throws Exception {
        String robotUrl = "tcp://192.168.72.100:9559";
        // Maakt een nieuwe 'application' aan
        Application application = new Application(args, robotUrl);
        // Start de nieuwe 'application'
        application.start();
        // Maakt een ALTextToSpeech object aan en verbind dit met de huidige 'session'
        ALTextToSpeech tts = new ALTextToSpeech(application.session());
        // Laat de robot Hallo zeggen
        tts.say("Hallo!");
    }
}
```

Figuur 22: Voorbeeldcode in Java

2.3.4 QiMessaging Javascript

Mensen uit de webdesign wereld zijn vaak beter vertrouwd met Javascript. Met QiMessaging Javascript kan er een webapplicatie op het werkstation worden geprogrammeerd voor de robot. Om dit tot een goed einde te brengen heeft Aldebaran QiMessaging ontwikkeld. Dit zorgt ervoor dat Javascript gebruik kan maken van de QiMessaging services. De QiMessaging services zijn de modules zoals in Python, C++ of Java. Als er van op het werkstation verbinding wordt gemaakt met de robot, moet de robot het qimessaging.js bestand bevatten anders worden de modules niet herkent [20]. De eerste regel van de javascript code is altijd (Figuur 23):

```
<script src="/libs/qimessaging/1.0/qimessaging.js">
```

Figuur 23: Eerste regel javascript

Vervolgens moet het object QiSession worden aangemaakt om de API's te kunnen gebruiken. Met de functie socket wordt de verbinding met de robot tot stand gebracht.

Een nadeel van de QiMessaging Javascript is dat dit niet werkt met de simulator.

2.4 Simulator

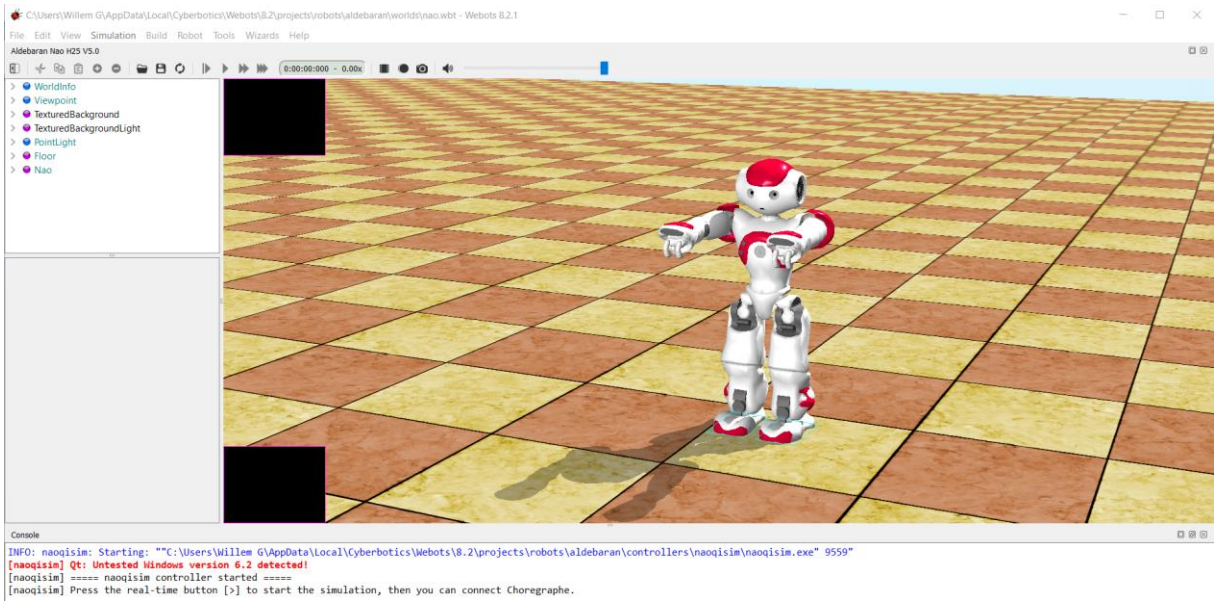
Het project was al een paar maanden bezig voordat de NAO robot bij COMmeto arriveerde. Er moest in die tussentijd een oplossing worden gezocht en dan was de simulator de ideale oplossing.

2.4.1 Webots

Met Webots is het mogelijk om de bewegingen van de robot te simuleren in een virtuele wereld. Hierdoor kan de robot niet kapot gaan en kunnen de bewegingen getest worden voordat deze worden uitgevoerd op de echte robot. Men kan ook de virtuele wereld aanpassen door er objecten in te plaatsen zoals stoelen en tafels. Buiten het simuleren van de bewegingen kunnen de sensors worden gemonitord [21].

Webots (Figuur 24) is verkrijgbaar voor meerdere platformen, onder andere voor Windows, Linux en Mac. Het programma wordt aangeboden op de website van Aldebaran en bij de NAO robot zit een licentiecode om hier gebruik van te maken.

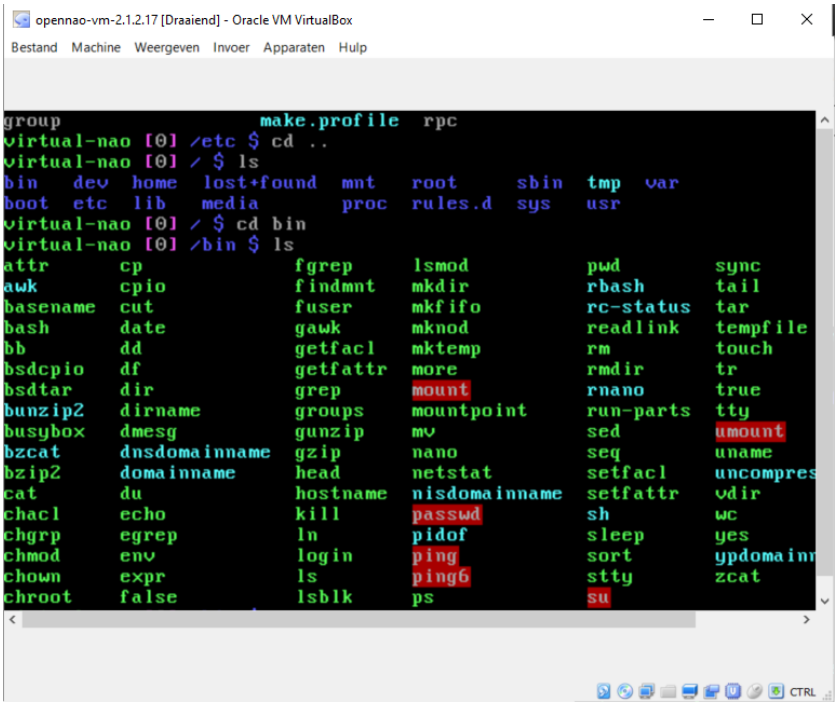
Een groot nadeel van de simulator is dat er geen geluiden gesimuleerd kunnen worden.



Figuur 24: Webots 8.2.1 voor Windows

2.4.2 OpenNAO OS VirtualBox

Van het besturingssysteem van de NAO robot bestaat er ook een simulatieomgeving. Deze moet worden opgestart met VirtualBox. Deze simulatie kan gebruikt worden om zelf services in te programmeren of aanpassingen te doen in het besturingssysteem zonder dat er schade kan worden aangericht [22]. Figuur 25 toont het OPenNAO OS in VirutalBox.



Figuur 25: OpenNAO OS in VirtualBox

2.5 Gemaakte keuze

Hier wordt omschreven waarom Java wordt gebruikt om de robot te programmeren. Java is één van de voorkeurtalen van Aldebaran Robotics. De ontwikkelaars van Aldebaran Robotics zijn druk bezig met het verbeteren van de Java *bindings* voor de robot. Omdat Java compatibel is op meerder platformen kan er in de toekomst nog veranderd worden van besturingssysteem moest dit nodig zijn. Zo zou er makkelijk een app voor Android kunnen ontwikkeld worden voor gebruik tijdens de therapieën. Op de website en het forum van Aldebaran Robotics is er veel informatie te vinden voor het programmeren in Java. Nog een voordeel is dat tijdens het schakelprogramma van de opleiding industrieel ingenieur elektronica-ICT een opleidingsonderdeel grafische applicaties in Java bevat, gegeven door dr. Kris Aerts. Hierdoor was er al een basiskennis van Java. Dr. Ludo Cuypers heeft ook ervaring in deze programmeertaal, dus als ik vragen had kon ik altijd bij hem terecht.

3 Ontwikkelingsfase

Na het afronden van de literatuurstudie kon de ontwikkelingsfase van start gaan. In deze fase van de masterproef werd de werkomgeving opgesteld en aan de technische aspecten van het project gewerkt. Met een goede planning kon deze masterproef tot een goed einde worden gebracht. Deze planning werd met Gantt Chart gemaakt. Ongeveer om de twee weken was er ook *Skype call* met de iedereen die betrokken was het project, voor het opvolgen van de vorderingen die al gemaakt werden en het bijstellen van de doelen waar en indien nodig. Samen met Dr. Ludo Cuypers werd er op regelmatige basis technische problemen besproken en opgelost. Via een facebookgesprek met Jessica De Smedt kon alles goed worden bijgesteld voor de experimenten.

3.1 Opstelling werkomgeving

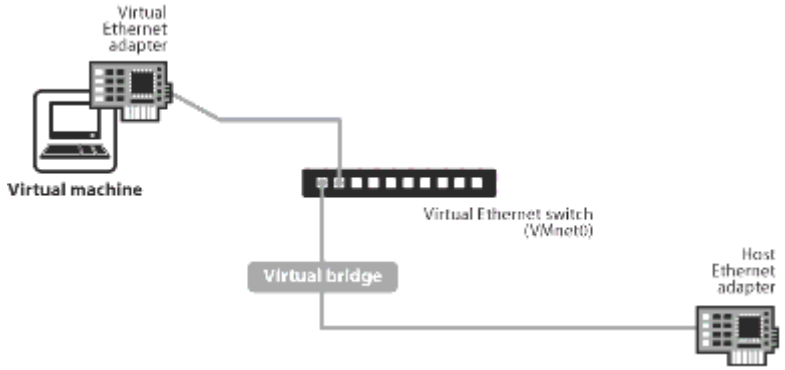
Eerst en vooral werd er een werkomgeving opgesteld voor de programmatie van de NAO robot.

3.1.1 Virtuele machine

Aan de hand van een virtuele machine is het mogelijk om de software omgeving op eender welke computer of laptop te draaien. Met VMware Workstation 12 Player is de virtuele machine tot stand gekomen. Het besturingssysteem dat voor deze virtuele machine werd gekozen is Linux Ubuntu LTS 14.04. De keuze voor dit besturingssysteem was eenvoudig; het systeem neemt weinig geheugen in en is dus eenvoudig te kopiëren. Van Linux Ubuntu weten we dat het zeer stabiel is.

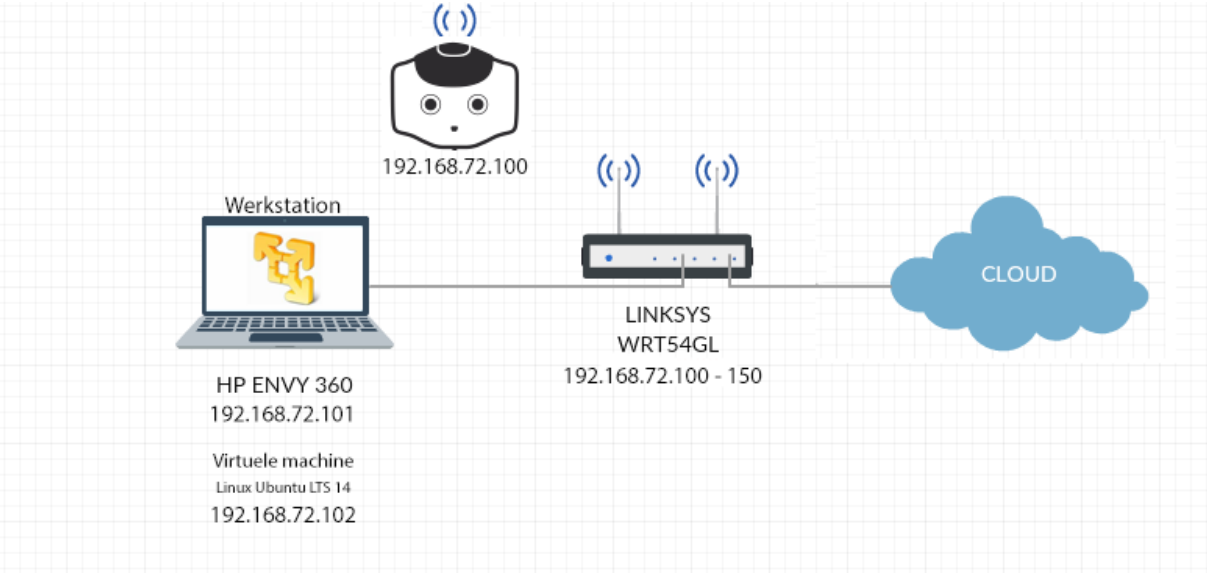
3.1.2 Netwerk

Voor de netwerk opstelling van dit project was er bij COMmeto een router beschikbaar, een Linksys WRT54GL. Deze router is zo ingesteld dat deze IP adressen verdeelt met een bereik van 192.168.72.100 tot 192.168.72.150 zowel draadloos als met de kabel. De netwerkkaart van de virtuele machine is ingesteld op *bridged* (Figuur 26). Dit wil zeggen dat de virtuele machine een IP adres krijgt uit het bereik van het netwerk waarop de laptop of computer is aangesloten [23].



Figuur 26: Bridged netwerk, met een virtuele switch [23]

Zowel de robot als de virtuele machine zitten in hetzelfde netwerk en kunnen dus via deze weg communiceren. In de onderstaande afbeelding wordt de netwerk opstelling getoond (Figuur 27).



Figuur 27: Netwerk opstelling

3.2 Eclipse

Om de NAO robot te programmeren is er gekozen voor Java. Om met Java te werken was Eclipse de meest geschikte ontwikkelomgeving. Met het opensource framework Eclipse, ontwikkeld door Eclipse Foundation, kon er eenvoudig met Java worden geprogrammeerd doordat deze een *auto IntelliSense* bevat. *Auto IntelliSense* wil zeggen dat het de functies en methodes automatisch aanvult als er nog maar een deeltje van het woord is ingetypt. Eclipse is ook beschikbaar voor de standaard besturingssystemen met de nodige plug-ins voor Java [24].

3.2.1 Java voor de NAO robot

De ontwikkelaars van Aldebaran Robotics verbeteren voortdurend de API's en hun methodes. Doordat het "java-naoqi-sdk-<version>-<platform>.jar" bestand wordt toegevoegd aan het project in Eclipse kan er gewerkt worden met de API's.

Bij het aanmaken van een nieuw programma voor de NAO robot is het essentieel dat er maar één *application* is zoals eerder vermeld. Door het ontwerppatroon van een singleton toe te passen kan er met zekerheid gezegd worden dat er maar één *application* is. Een singleton is een ontwerppatroon dat er voor zorgt dat een object van een klasse maar één keer kan worden opgeroepen. Met het object getNaoApplication kan er maar één *application* worden aangemaakt. Door een nieuwe *application* aan te maken met als argument CONNECTION wordt er een verbinding opgezet met de robot. Het argument CONNECTION bevat het internetprotocol TCP gevolgd door het IP adres en de poort. Vervolgens is de variabele *instance* niet meer gelijk aan *null* en kan deze niet meer worden aangemaakt (Figuur 28).

```
import com.aldebaran.qi.Application;

public class NaoApplication {
    // TODO Set IP adres!!
    private static String IP;
    private static String PORT = "9559"; // PORT of the robot
    private static String CONNECTION = "tcp://" + IP + ":" + PORT;

    private static Application instance = null;

    private NaoApplication() {

    };

    public static Application getNaoApplication() {
        if (instance == null) {
            CONNECTION = "tcp://" + IP + ":" + PORT;
            String[] args = new String[0];
            try {
                // Create a new application
                instance = new Application(args, CONNECTION);
                // Start your application
                instance.start();
                System.out.print("Successfully connected to the robot \n");
            } catch (Exception e) {
                System.out.print("UnSuccessfully connected to the robot");
                e.printStackTrace();
            }
        }
        return instance;
    };

    public static void setIP(String ip) {
        IP = ip;
    }
}
```

Figuur 28: De klasse NaoApplication

Door het importeren van com.aldebaran.qi.helper.proxies.* kan er gebruik worden gemaakt van al de objecten. Elk object wordt op zijn beurt gekoppeld aan de huidige *session* van de *application*. Zo wordt bijvoorbeeld het object ALTextToSpeech aangemaakt met als argument de huidige sessie. Als dit gebeurd is kunnen de methodes worden gebruikt. De tekst die wordt meegegeven met de methode "say" wordt door de robot uitgesproken. De onderstaande afbeelding (Figuur 29) is een stukje voorbeeld code dat niet gebruikt wordt in het project.

```
import com.aldebaran.qi.Application;
import com.aldebaran.qi.helper.proxies.*;

public class testSpeak {

    public static void main(String[] args) throws Exception {
        // TODO Auto-generated method stub
        //IP adres van de robot
        String robotUrl = "tcp://192.168.72.100:9559";
        // Het aanmaken van een nieuwe application
        Application application = new Application(args, robotUrl);
        // Start de application
        application.start();
        // Maakt het object ALTextToSpeech aan en linkt het aan de huidige sessie
        ALTextToSpeech tts = new ALTextToSpeech(application.session());
        // Gebruikt de methode say
        tts.say("Hallo!");
    }

}
```

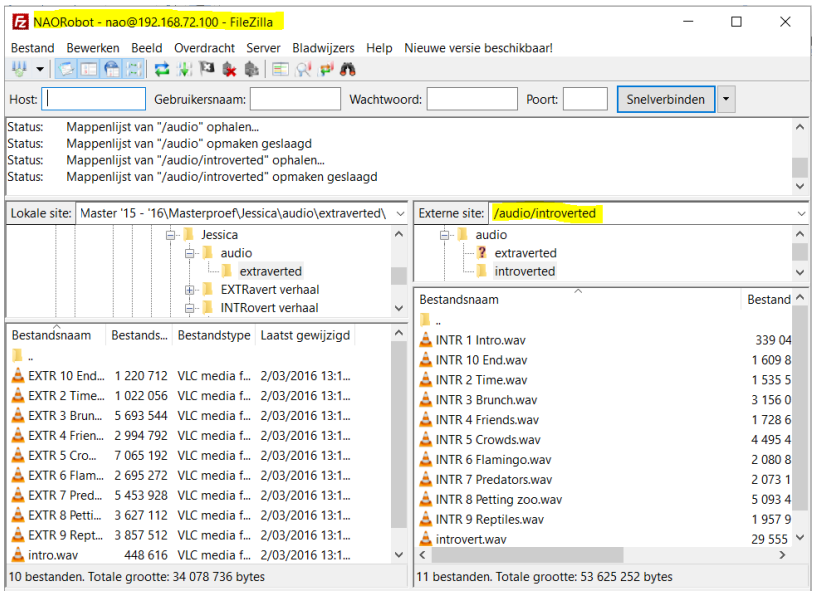
Figuur 29: Code om de NAO robot iets te laten zeggen

3.3 Het zoo verhaal

Het eerste technische doel van deze masterproef was het programmeren van het verhaal. Hierbij vertelt de NAO robot dat hij voor zijn verjaardag naar de dierentuin is gegaan. De lichaamstaal van de robot moest overeenkomen met de inhoud van het verhaal, het is belangrijk dat deze handelingen synchroon lopen.

3.3.1 Eerste versie

In de eerste versie had Jessica De Smedt tien audio bestanden ingesproken, elke bestand was een stukje van het verhaal dat dan met de NAO robot afgespeeld werd. Deze audio bestanden werden met Filezilla op de robot geüpload zoals in Figuur 30. Eens de bestanden op het geheugen van de robot stonden moesten deze via de geprogrammeerde code worden afgespeeld.



Figuur 30: Audio bestanden opgeslagen op de NAO robot via Filezilla

In deze versie van het zoo verhaal waren er maar drie klassen, ExtrZoo.java, IntrZoo.java en ConsciousnessesSettings.java. In de klassen ExtrZoo en IntrZoo werden de bewegingen synchroon met de audio geprogrammeerd zoals het voorbeeld van Jessica De Smedt in Python. De klassen ConsciousnessesSettings zorgde ervoor dat de NAO robot begon met de juiste oogkleur en volume. Bij een extraverte NAO robot is de oogkleur een felle kleur zoals geel of oranje en bij een introverte eerder zachte kleuren zoals licht blauw of bruin. Het volume van de robot is ook afhankelijk van de persoonlijkheid, zo is het volume van een extraverte luider dan die van een introverte.

Het verhaal werd dus opgedeeld in tien delen, elk deel heeft een audio bestand dat moest worden afgespeeld. Synchroon met dit bestand werden er bewegingen uitgevoerd. Voor elke specifieke beweging werd een functie geprogrammeerd. Om alles synchroon te laten verlopen werd er gewerkt met *threads*.

Threads worden gebruikt om meerdere kleine processen gelijktijdig uit te voeren. Als er meerdere *threads* worden gebruikt noemt men dit ook wel *multithreading*. Als het geluidsbestand moest worden afgespeeld en er wordt een beweging gedaan dan moet dit allebei in een aparte thread worden afgehandeld om dit synchroon te doen verlopen. Zo heeft ExtrZoo en IntroZoo een hoofdthread, ook wel de *mainthread* genoemd, en daarnaast ook enkele kortere *threads*. Om dit te doen wordt er eerst een thread aangemaakt. Vervolgens kan deze gestart worden met de functie start(). Indien we pas verder kunnen gaan als de code van deze *threads* is afgerond dan kunnen we de functie join() gebruiken.

3.3.2 Tweede versie

Toen de eerste versie op punt stond konden de doelen worden bijgesteld. Eén van de doelen was onder andere een bibliotheek maken van de specifieke bewegingen voor zowel de introverte als de extraverte robot. Voor de aanpak hiervan is er eerst nagegaan welke bewegingen eerder neutraal zijn en welke meer specifiek voor de andere twee gevallen.

In de klasse NAO.java zijn de neutrale bewegingen van de NAO robot gebundeld. De klassen EXTRNAO.java en INTRNAO.java erven van de klasse NAO. Dit wil zeggen dat als er een extraverte NAO wordt aangemaakt deze de specifieke handelingen van de klasse EXTRNAO.java bevat maar ook de neutrale bewegingen van NAO.java, dit geldt natuurlijk ook voor een introverte NAO robot. In Figuur 31 wordt het klassendiagram getoond.

Figuur 31: Klassendiagram van NAO, INTRNAO en EXTRNAO met hun methodes

Een tweede doel was om de audio bestanden van Jessica De Smedt door een actrice te laten inspreken. Dit werd niet meer gedaan in tien verschillende delen maar gewoon in één keer. Ook hielp deze actrice met de bijhorende lichaamstaal te verbeteren.

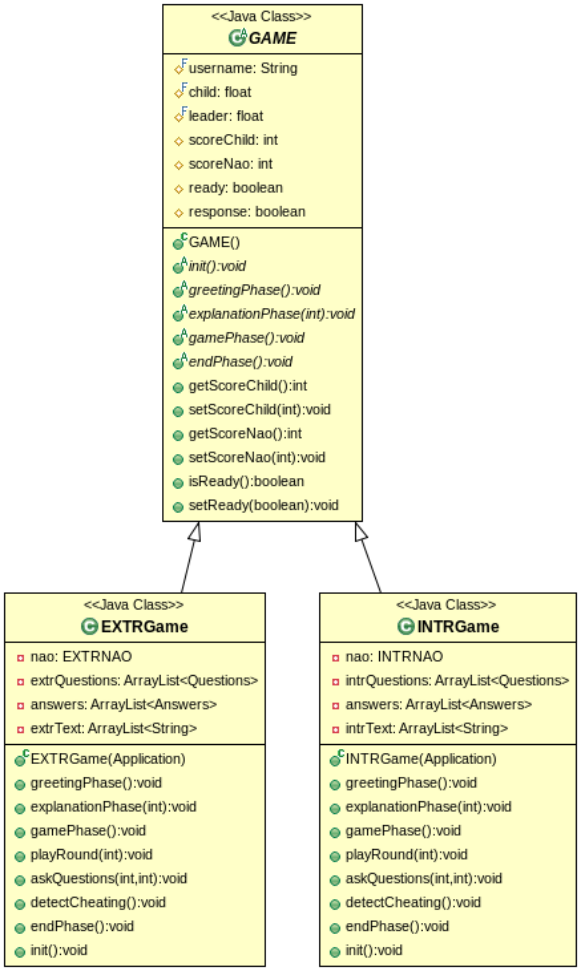
3.4 Raad het dier

Raad het dier is een spel dat de kinderen kunnen spelen met de NAO robot. Er ligt een stapel met kaartjes, met een afbeelding van allerlei dieren, voor de NAO robot (Figuur 32). De bedoeling van het spel is dat je een kaart trekt uit deze stapel en de vragen van de NAO robot beantwoordt met ja of nee. Als de robot kan raden welk dier op het kaartje staat, krijgt hij een punt en zo niet dan verdient de tegenspeler een punt. Het spel duurt drie rondes dus als de robot twee keer fout raadt dan wint de tegenspeler.

Figuur 32: NAO robot met het spel ervoor

Het spel bestaat uit vier fases. De eerste fase is de begroetingsfase, de NAO robot verwelkomt het kind, dit kan eventueel met de naam van het kind. Vervolgens is er de fase waarin het spel wordt uitgelegd en er wordt gevraagd of het kind het spel wilt spelen. Als het kind niet wil spelen dient de therapeut het kind te overtuigen. Fase drie is de spelletjesfase, hier wordt het spel gespeeld en moeten de vragen met ja of nee worden beantwoord. Ten slotte is er nog een eindfase waar er afscheid wordt genomen van het kind. De fases van het spel blijven dezelfde voor een introverte of extraverte robot maar het gedrag van de robot verschilt doordat er andere bewegingen worden uitgevoerd.

De klassen NAO, ExtrNAO en IntrNAO, zoals gebruikt bij het zoo verhaal, konden opnieuw gebruikt worden bij het programmeren van het spel. Er zijn nog drie andere klassen GAME.java, INTRGame.java en EXTRGame.java. GAME.java is een abstracte klasse met abstracte methodes. Dit wil zeggen dat de klassen die erven van de klasse GAME verplicht zijn om de abstracte methodes te implementeren. In de twee klassen die hiervan erven komen de vier fases van het spel terug. In het klassendiagram hier onder is dit duidelijk te zien (Figuur 33).



Figuur 33: Klassendiagram van GAME, EXTRGame en INTRGame

De derde fase of spelletjes-fase was het moeilijkst te programmeren. Door een boomstructuur uit te denken van de vragen en de antwoorden kon het spel worden uitgewerkt. De klasse Questions.java en Answers.java bepalen de structuur van een vraag en een antwoord. De functie listen() van de klasse NAO.java luistert telkens na een vraag of deze met ja of nee wordt beantwoord. Zo is het mogelijk om de volgende stap te bepalen. Een antwoord bevat één of meerdere dieren, als het meerdere dieren bevat gaat er een willekeurig dier uit het antwoord worden gekozen. Zo wint de robot niet altijd en dit maakt het spelen van het spel aangenaam.

3.5 Audio zenden en ontvangen

Het is niet mogelijk om met de methodes van Aldebaran Robotics een audio bestand te verzenden in reële tijd naar de robot. Daarom staan de geluidsbestanden van het verhaal op het geheugen van de robot. Nog een belangrijk probleem was dat de robot het woord “nee” niet altijd even goed verstaat en dan nog eens moet vragen wat het antwoord is. Daar was het schrijven van een eigen service voor het zenden van een WAV bestand en het ontvangen van de data van de microfoons een goed idee om dit wel te kunnen of te zien waar er zich een probleem voor doet.

3.5.1 Service voor de microfoons

Voor het ontvangen van de data van de microfoons zijn er geen methodes die gebruikt konden worden. Op het forum van Aldebaran stond de onderstaande reactie (Figuur 34). Er moest dus een eigen service gemaakt worden. Met wat hulp van de mensen van Aldebaran is het gelukt om dit te doen.

Thalia C. Mar 4

The point is to implement a 'processRemote' method that will be the callback.

For Java, you need to create your own service, advertise your 'processRemote' method, register your service and subscribe to ALAudioDevice.

Take a look at Creating a new service in the documentation to get all the details on how to create your own service, register a service and advertise one of its methods.

1. You need to create a class that inherits from QiService. This class should contain the callback method: void processRemote(Object nbOfChannels, Object nbrOfSamplesByChannel, Object aTimeStamp, Object buffer)
2. Then, advertise your method: objectBuilder.advertiseMethod("processRemote::v(mmmm)", service, "Callback for ALAudioDevice");
3. Then, register your service (for example "SoundReceiver"): session.registerService("SoundReceiver", objectBuilder.object());
4. Finally, subscribe to ALAudioDevice. Be sure to use the same name you used for registering your service: audioDevice.subscribe("SoundReceiver")

Hope this helps.

Figuur 34: Reactie op het forum voor het capteren van de microfoons

Met de methode startListening() kan er worden ingesteld hoe de microfoons gecapteerd zullen worden. Dit werd aan de hand van de onderstaande code gedaan (Figuur 35). Door nNbrChannelFlag gelijk te stellen aan 0 wordt er bepaald om naar de vier kanalen te luisteren. nDeinterleave = 0 wilt zeggen dat er gekozen wordt om de data *interleaved* te ontvangen. En de *sample rate* of bemonsteringsfrequentie van het signaal moet 16000 of 48000 Hertz zijn.

```
public void startListening() throws CallError, InterruptedException {
    nNbrChannelFlag = 0;
    nDeinterleave = 0;
    nSampleRate = 48000;
    /*
     * sampleRate - sample rate of the microphones data sent to the process function - must be 16000 or 48000
     *
     * channelsConfiguration - An int
     * ALLCHANNELS 0, LEFTCHANNEL 1, RIGHTCHANNEL 2,FRONTCHANNEL 3, REARCHANNEL 4 are the configuration currently supported.
     *
     * Deinterleaved - indicates if the microphones data sent to the process function are interleaved or not - 0 : interleaved - 1 : deinterleaved
     */

    audio.setClientPreferences(this.name, nSampleRate, nNbrChannelFlag, nDeinterleave);
```

Figuur 35: Instellen van het capteren van de microfoons

De data wordt hierna opgeslagen als byte array. De byte array moet worden omgezet naar een WAV bestand. In de onderstaande code (Figuur 36) wordt eerst het audio formaat ingesteld. Vervolgens wordt er een ByteArrayInputStream aangemaakt met als argument de ontvangen data. Deze kan dan omgezet worden om een AudioInputStream aan te maken. Eens deze *stream* is aangemaakt kan deze worden omgezet met de functie write() van de klasse AudioSystem. Het WAV bestand wordt opgeslagen op de plaats van het "*path"* [25].

```
public void ToWave() {
    try {
        int frameSize = 2;
        format = new AudioFormat(Encoding.PCM_SIGNED, nSampleRate, 16, numChannels, frameSize,nSampleRate * frameSize, false);

        ByteArrayInputStream bais = new ByteArrayInputStream(bytesToWav);//bytesToWav is een byte array
        AudioInputStream stream = new AudioInputStream(bais, format, bytesToWav.length / format.getFrameSize());

        AudioSystem.write(stream, AudioFileFormat.Type.WAVE, new File(path + "microfoon.wav"));
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Figuur 36: Byte array naar een WAV bestand

Na het beluisteren van het opgeslagen bestand werd er waargenomen dat er ruis op het signaal zat. Het ruissignaal is afkomstig van een ventilator in het hoofdje van de NAO robot. De volgende stap is dat dit audiosignaal gefilterd wordt zodat de ruis verdwijnt of vermindert.

3.5.2 Audio bestand streamen

Doordat het niet mogelijk is om met de methodes van Aldebaran Robotics audio te streamen naar de robot moest er een oplossing worden gezocht. De eerste poging voor een oplossing van het probleem was de functie playWebStream() die wel werkt. De twee belangrijkste argumenten van deze functie zijn het webadres van de webstream en het volume. Het idee was om zelf een webstream op te bouwen van het nodige audio bestand en dan af te spelen met playWebStream.

Met IceCast2, een programma om audio bestanden te *streamen*, was het gelukt om een Ogg bestand te *streamen*. Maar nu bleek dat de functie playWebSream het Ogg bestand niet ondersteunde. Na verder onderzoek bleek het enkel te werken voor MP3 bestanden.

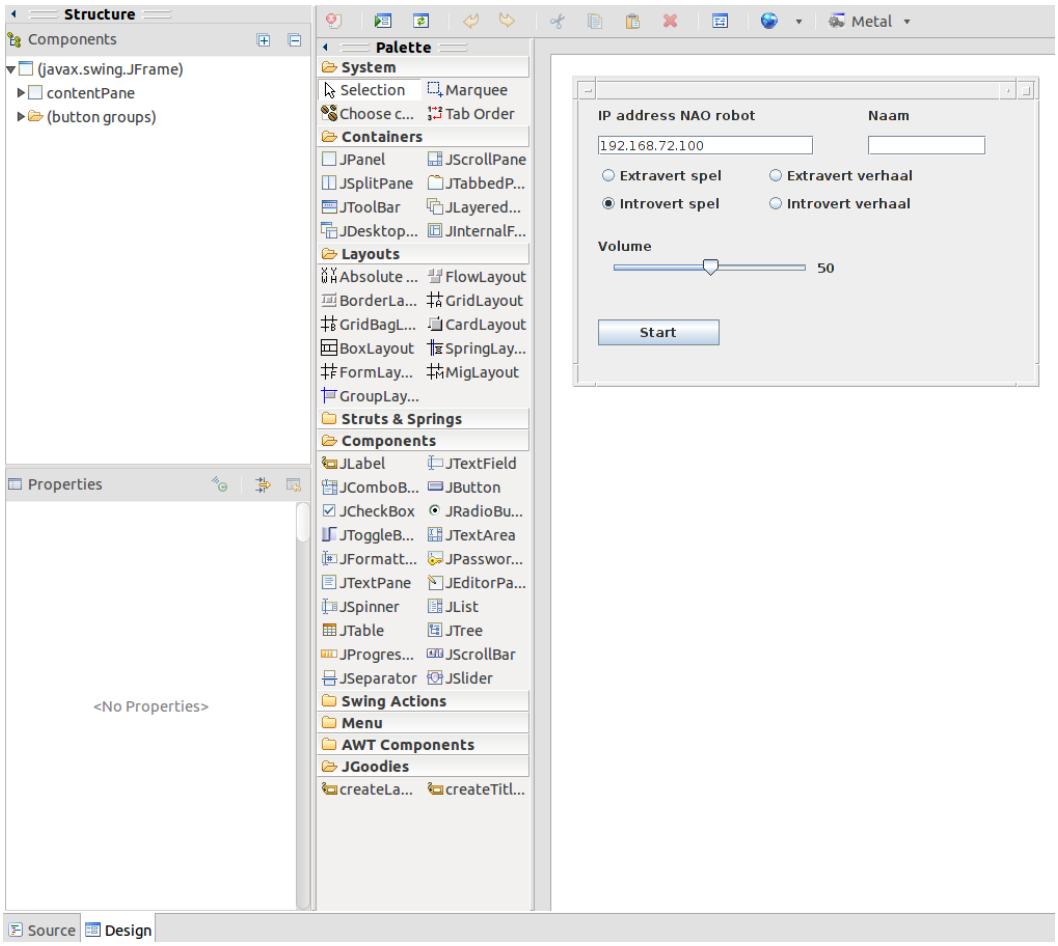
Een tweede oplossing was het schrijven van een eigen service zoals voor het ontvangen van de microfoons. Dit is nog niet gebeurd maar staat wel op de lijst als een volgend het project.

3.6 Grafische interface

Voor de gebruiker van het programma voor de NAO robot is het veel gemakkelijker om iets met een grafische interface aan te sturen. Zo kan het IP adres en de naam van het kind in een tekst vak worden ingegeven. Met een slidebar kan het volume worden ingesteld en met het keuzerondje kan er gekozen worden voor de beide versie van het spel of het verhaal.

3.6.1 WindowBuilder

Voor het maken van de grafische interface is er gebruik gemaakt van WindowBuilder. WindowBuilder is een plug-in voor Eclipse. Door te werken met WindowBuilder verliest men geen tijd door het schrijven van code. Met de *lay-out tools* is het eenvoudig om tekstvlakken, tekstvensters, knoppen, enzovoort toe te voegen. Op basis van een *drag en drop* systeem wordt alles in het hoofdvenster gesleept. Door dit te doen wordt er automatisch code gegeneerd door WindowBuilder. In Figuur 37 wordt de opbouw van de grafische interface getoond.



Figuur 37: Grafische interface met WindowBuilder

4 Resultaat

4.1 Experimenten

Door de samenwerking met CLiPS konden de experimenten worden uitgevoerd en geanalyseerd. Dit deel van het project werd gedaan door Jessica De Smedt. Zij stelde de experimenten op en haalde er de volgende resultaten uit. Voor de experimenten van start gingen werd er eerst een persoonlijkheidstest gedaan aan de hand van een “Big 5 personality test” van Metthews, Deary en Whiteman[26].

4.1.1 Experiment 1

In het eerste experiment liet men aan de proefpersonen een video zien waarin de robot het introverte of extraverte verhaal vertelt. Dit experiment werd gedaan om na te gaan wat voor de gemiddelde Vlaming, rekening houdend met de persoonlijkheid, het voorkeur gedrag van de robot was, introvert of extravert. De proefpersonen wisten niet welke van de twee ze te zien zouden krijgen, noch dat er een focus was op de extraversiedimensie. Na het filmpje bekeken te hebben moesten ze de robot evalueren aan de hand van de beweringen in de volgende tabel (Tabel 2) [26].

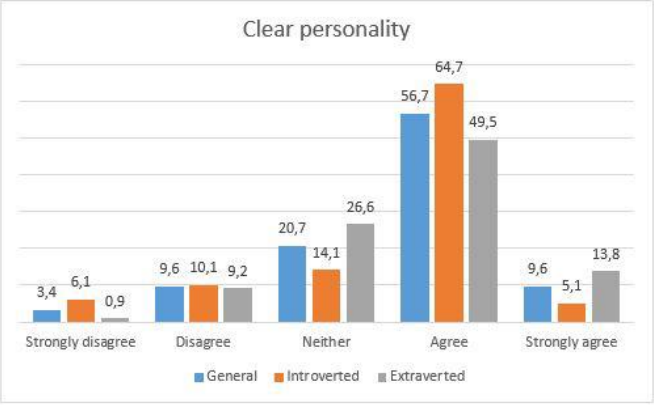
Tabel 2:Evaluatie statement [26]

Number	Statement
1	The robot has a clear personality.
2	The robot is enjoyable.
3	I would like to use the robot in the future.
4	The robot is natural.
5	I can identify with the robot's language.
6	I can identify with the robot's behaviour.

Er deden 208 proefpersonen mee aan het eerste experiment. Deze hadden een leeftijd van 18 tot 83 jaar.

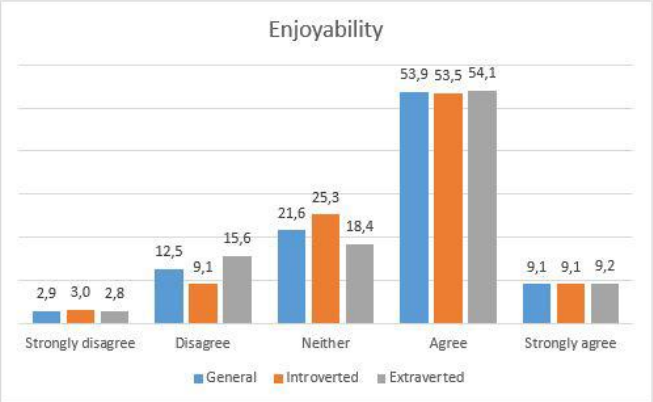
De resultaten van het eerste experiment worden weergegeven in de volgende grafieken.

De onderstaande grafiek laat ons zien dat de persoonlijkheid van de robot duidelijk was (Figuur 38).



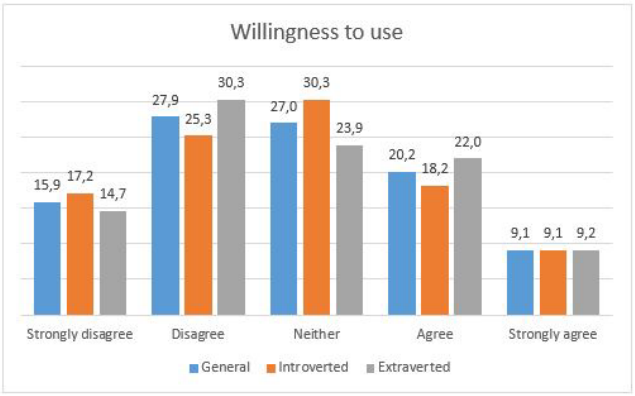
Figuur 38: Het resultaat van de persoonlijkheid van de NAO robot(in procent) [26]

Het merendeel van de proefpersonen vond het aangenaam om met de robot te werken (Figuur 39).



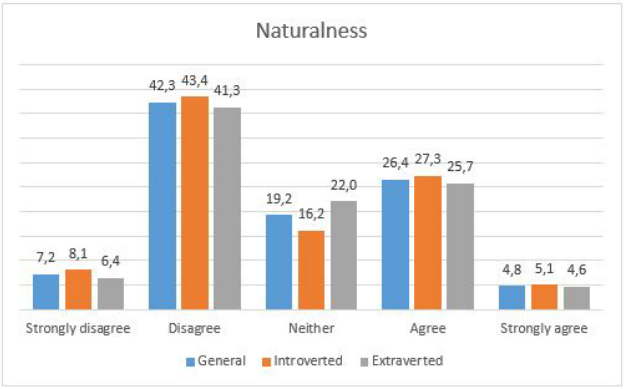
Figuur 39: Het resultaat van hoe aangenaam de robot was (in procent) [26]

Op de vraag of ze de robot ook zouden willen gebruik zijn de meningen eerder verdeeld zoals te zien is in Figuur 40.



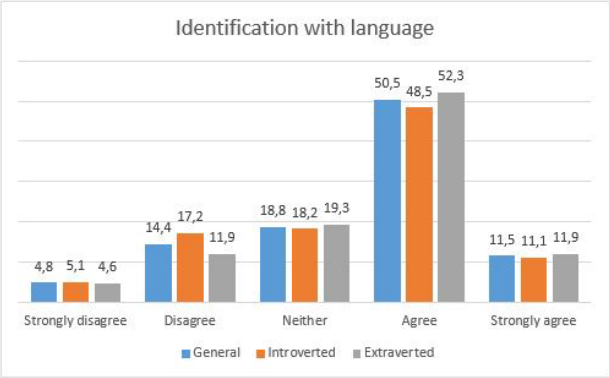
Figuur 40:Het resultaat van de bruikbaarheid van de NAO robot (in procent) [26]

Figuur 41 toont ons dat ongeveer 50 procent van de proefpersonen vond dat de robot niet natuurlijk overkwam.



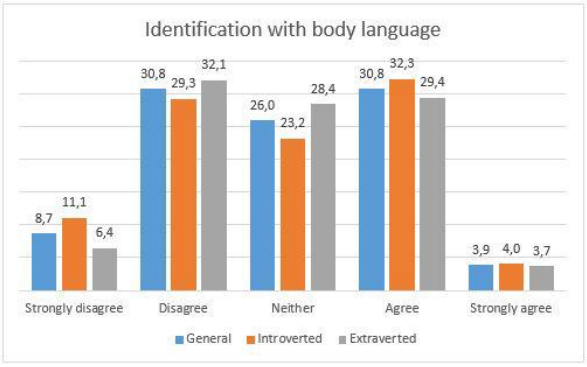
Figuur 41:Het resultaat van het natuurlijk overkomen van de NAO robot(in procent) [26]

De onderstaande grafiek laat de resultaten van de identificatie met de taal van de robot zien (Figuur 42).



Figuur 42:Het resultaat van de identificatie met taal (in procent) [26]

De resultaten van de identificatie met de taal verschillen wel degelijk met die van de lichaamstaal (Figuur 43).



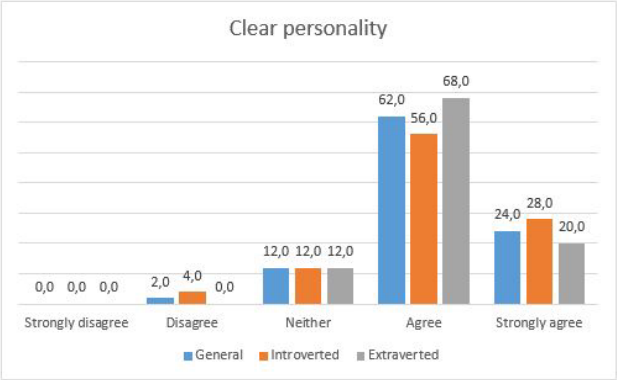
Figuur 43:Het resultaat van de identificatie met lichaamstaal (in procent) [26]

4.1.2 Experiment 2

Het tweede experiment was een vervolg op het eerste. Hierbij is het grote verschil dat er hier wel een mens-robot interactie was. Het dieren raadspel werd hiervoor gebruikt. De proefpersonen moesten enkel “ja” of “nee” antwoorden op de vragen die gesteld werden door de NAO robot.

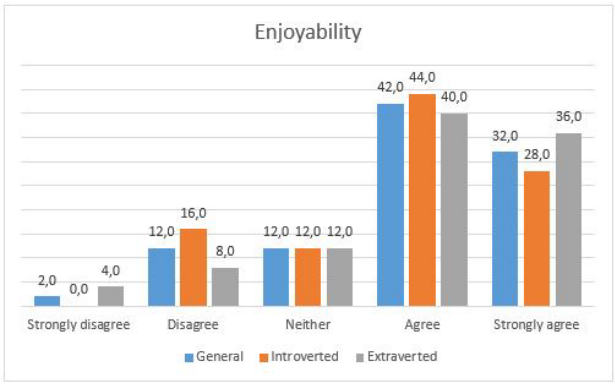
Voor dit experiment waren er 50 deelnemers met een leeftijd van 18 tot 79 jaar. De onderstaande grafieken zijn de resultaten hiervan. De vragen uit het twee experiment zij hetzelfde als het eerste experiment [26].

In Figuur 44 toont aan dat de persoonlijkheid van de twee soorten robots wordt herkent.



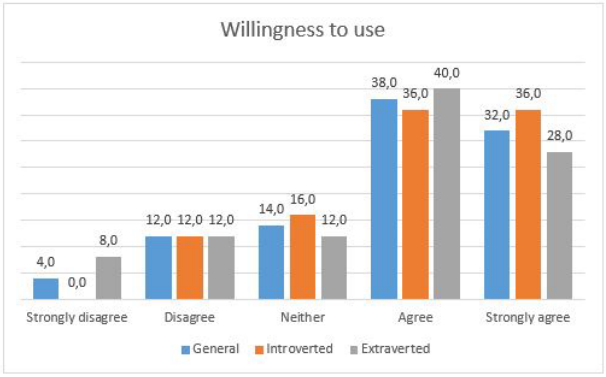
Figuur 44:Experiment 2: Het resultaat van de persoonlijkheid van de NAO robot(in procent) [26]

De NAO robot wordt bijna altijd aangenaam bevonden dit lijden we af uit Figuur 45.



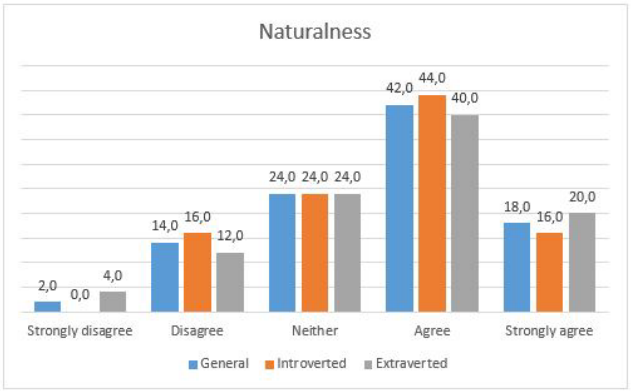
Figuur 45:Experiment 2: Het resultaat van hoe aangenaam de NAO robot was (in procent) [26]

Het grootste deel van de proefpersonen zijn overtuigd van de bruikbaarheid van een NAO robot (Figuur 46).



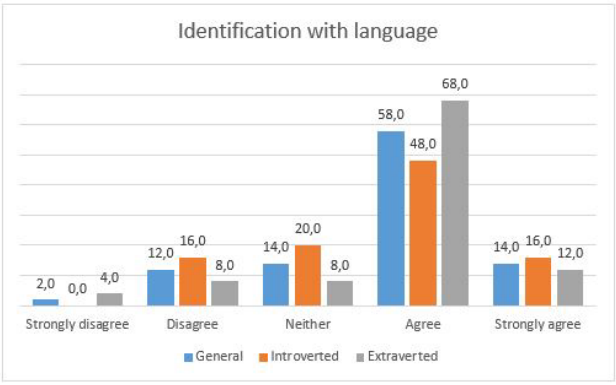
Figuur 46:Experiment 2: Het resultaat van de bruikbaarheid van de NAO robot(in procent) [26]

Het overkomen van de robot wordt in de onderstaande grafiek weergegeven (Figuur 47). Hier uit kunnen we afleiden dat de robot meestal natuurlijk overkomt.



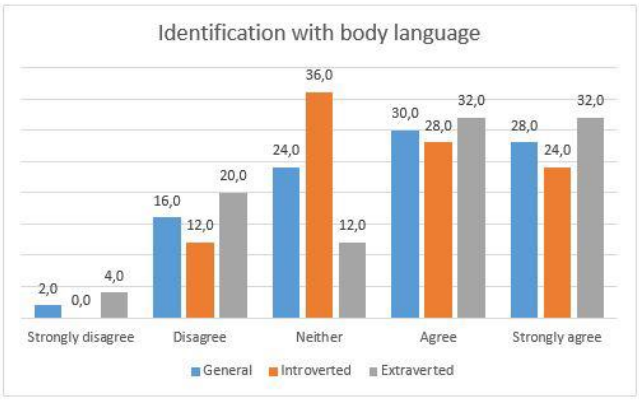
Figuur 47:Experiment 2: Het resultaat van het natuurlijk overkomen van de NAO robot(in procent) [26]

In Figuur 48 wordt het resultaat van de identificatie met taal weergegeven.



Figuur 48:Experiment 2: Het resultaat de identificatie met taal van de NAO robot(in procent) [26]

Figuur 49 toont de resultaten van de identificatie met lichaamstaal van de NAO robot.



Figuur 49:Experiment 2: Het resultaat de identificatie met lichaamstaal van de NAO robot(in procent) [26]

4.1.3 Conclusie van de experimenten

Na het analyseren van het eerste experiment hebben de introverte proefpersonen aangegeven dat ze een introverte NAO robot duidelijker vinden. En de extraverte personen vinden de extraverte robot duidelijker. Ook bleek uit dit experiment dat introverte personen over het algemeen gevoeliger zijn aan de kunstmatige persoonlijkheidskenmerken van de NAO robot dan de extraverte personen.

Het tweede experiment bevatte interactie tussen de NAO robot en de proefpersonen. Hieruit werd het volgende uit afgeleid. De conclusie uit het eerste experiment geldt ook voor het tweede experiment. Opmerkelijk is echter dat de introverte mannen zichzelf eerder herkennen in de bewegingen van de extraverte NAO robot. Maar mannen zouden wel liever een introverte robot hebben dan een extraverte [26].

5 Besluit

5.1 Conclusie en mogelijke uitbreiding

Het doel van deze masterthesis was het programmeren van de NAO robot in reële tijd. De geprogrammeerde code kon dan gebruikt worden bij de experimenten van de universiteit van Antwerpen en later worden toegepast bij therapieën met autistische kinderen.

Uit de literatuurstudie blijkt dat de NAO robot met verschillende programmeertalen kon worden geprogrammeerd zowel op de robot als op een externe computer.. Er is gekozen om de NAO robot met een extern werkstation aan te sturen. Dat werkstation is een virtuele machine van een Linux Ubuntu besturingssysteem. Uit de verschillende programmeertalen is er besloten om met Java de NAO robot te besturen. Vervolgens werd er met deze opstelling een *proof of concept* voltooid. Ten slotte werden de geprogrammeerde programma's uitgevoerd om de experimenten tot een goed einde te brengen. Deze werden dan later nog geanalyseerd door Jessica De Smedt.

De vooropgestelde doelstellingen zijn bereikt. Voor het aspect van de audio wordt er verder werk gedaan binnen COMmeto. Verder kan de grafische interface uitgebreid worden, eventueel in Java FX.

5.2 Persoonlijke ervaring

Door lange tijd aan een project te werken in een bedrijf leer je snel hoe het er in het bedrijfsleven aan toe gaat. Je leert de werknemers en stilaan ook het bedrijf kennen. Met deze ervaring ben je klaar om je carrière te starten in de bedrijfswereld.

Eén van de belangrijkste dingen die ik heb bijgeleerd tijdens deze periode is dat een software programma nooit volledig af is. Er zijn steeds dingen die verbeterd en aangepast moeten worden. Gestructureerd werken en kwalitatieve documentatie zijn dan ook erg belangrijk voor de verdere toekomst van een programma.

De grote vrijheid in het zoeken naar de beste werkmethode, gecombineerd met de hulp die ik kreeg, zorgde ervoor dat de project zich vlot ontwikkelde en snel vorm kreeg.

Literatuurlijst

[1] "Robot ZORA biedt interactieve zorgondersteuning aan patiënten in UZ Gent - UZ Gent," 2013. [Online]. Available: http://www.uzgent.be/nl/zorgaanbod/mdspecialismen/Kinderrevalidatiecentrum/nieuws/Paginas/Robot-ZORA-biedt-interactieve-zorgondersteuning-aan-pati%C3%ABnten-in-UZ-Gent.aspx. [Accessed: 18-May-2016].

[2] J. De Smedt, "Natural Language Processing Research with NAO Robots," 2015.

[3] "NAOqi APIs — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/naoqi/index.html#naoqi-api. [Accessed: 18-May-2016].

[4] "Programming — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/dev/programming_index.html. [Accessed: 18-May-2016].

[5] "Programming Guide — NAO Software 1.14.5 documentation." [Online]. Available: http://doc.aldebaran.com/1-14/dev/index.html. [Accessed: 18-May-2016].

[6] "NAO - Versions and Body Type — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/family/body_type.html#nao-version-bodytype. [Accessed: 18-May-2016].

[7] "H25 - Construction — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/family/nao_h25/dimensions_h25.html#h25-dimensions. [Accessed: 18-May-2016].

[8] D. Addendum and S. U. Addendum, "Intel ® Atom TM Processor Z5xx Series," no. March, 2009.

[9] "NAO - Battery — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/family/robots/battery_robot.html. [Accessed: 18-May-2016].

[10] "Loudspeakers — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/family/robots/loudspeaker_robot.html. [Accessed: 18-May-2016].

[11] "Microphone - Aldebaran." [Online]. Available: http://doc.aldebaran.com/2-1/family/robots/microphone_robot.html#id2. [Accessed: 18-May-2016].

[12] "NAO - Video camera — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/family/robots/video_robot.html. [Accessed: 02-Jun-2016].

[13] "LEDs — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/family/robots/leds_robot.html. [Accessed: 02-Jun-2016].

[14] "Contact and tactile sensors — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/family/robots/contact-sensors_robot.html. [Accessed: 18-May-2016].

[15] "Sonars — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/family/robots/sonar_robot.html. [Accessed: 02-Jun-2016].

[16] "FSRs — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/family/robots/fsr_robot.html. [Accessed: 02-Jun-2016].

[17] "NAO - Actuator & Sensor list — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/family/nao_dcm/actuator_sensor_names.html#actuator-sensor-list-nao. [Accessed: 18-May-2016].

[18] "Python SDK — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/dev/python/index.html. [Accessed: 19-May-2016].

[19] "Java SDK — Aldebaran 2.1.4.13 documentation." [Online]. Available: http://doc.aldebaran.com/2-1/dev/java/index_java.html. [Accessed: 19-May-2016].

[20] “QiMessaging JavaScript — Aldebaran 2.1.4.13 documentation.” [Online]. Available: http://doc.aldebaran.com/2-1/dev/js/index.html. [Accessed: 20-May-2016].

[21] “Webots — Aldebaran 2.1.4.13 documentation.” [Online]. Available: http://doc.aldebaran.com/2-1/software/webots/webots_index.html. [Accessed: 20-May-2016].

[22] “Setting up the OpenNAO virtual machine — NAO Software 1.14.5 documentation.” [Online]. Available: http://doc.aldebaran.com/1-14/dev/tools/vm-setup.html. [Accessed: 20-May-2016].

[23] “Bridged Networking.” [Online]. Available: https://www.vmware.com/support/ws4/doc/network_bridged_ws.html. [Accessed: 23-May-2016].

[24] “Eclipse - The Eclipse Foundation open source community website.” [Online]. Available: https://eclipse.org/. [Accessed: 23-May-2016].

[25] “Java byte array to wav.” [Online]. Available: http://stackoverflow.com/questions/12424659/flex-java-byte-to-mp3. [Accessed: 26-May-2016].

[26] J. De Smedt, “Robots with Personality : Introversion and Extraversion in NAO Robots,” 2016.

Bijlage

Lezing BruBotics

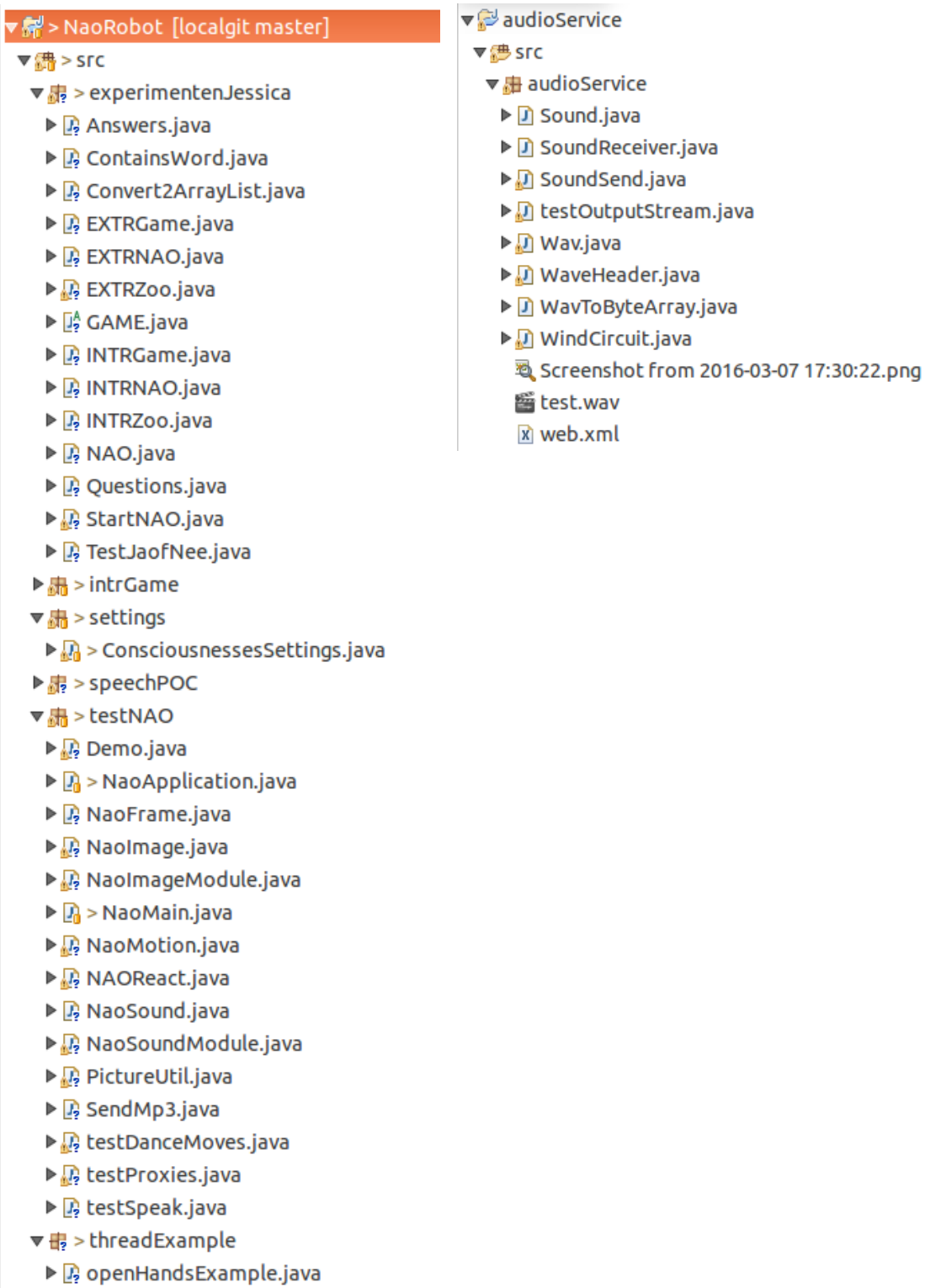
Op 27 april 2016 gingen Ludo, Todor en ik naar de Vrije Universiteit van Brussel om naar het BruBotics evenement te gaan. Het programma zag er als volgt uit:

- 16:00 - Aula Q.C.: welcome and registrations
- 16:30 (sharp) - Aula Q.C.: keynote speeches
    - Paul De Knop, rector VUB: Welcome
    - Prof. Dirk Lefeber: BruBotics
    - Patrick Danau, CEO Audi Brussels: Human-Robot collaboration
    - Marc Herremans, To Walk Again: Robots To Walk Again
    - Alexander De Croo, vice prime-minister: closing speech
- 18:00-... - U-Residence: discover our ongoing research projects, see live demonstrations and enjoy a cocktail from our cocktail robot during the accompanying reception.

Hun onderzoeksprojecten waren het interessantste. Het project “Dream: the next generation robot-enhanced therapy”, maakte ook gebruik van de NAO robot om autistische kinderen te helpen. Door met onderzoekers van dit project te praten heb ik mijn kennis van de NAO robot kunnen verruimen. Meer info over hun project kan je vinden op de website van Brubotics.eu.

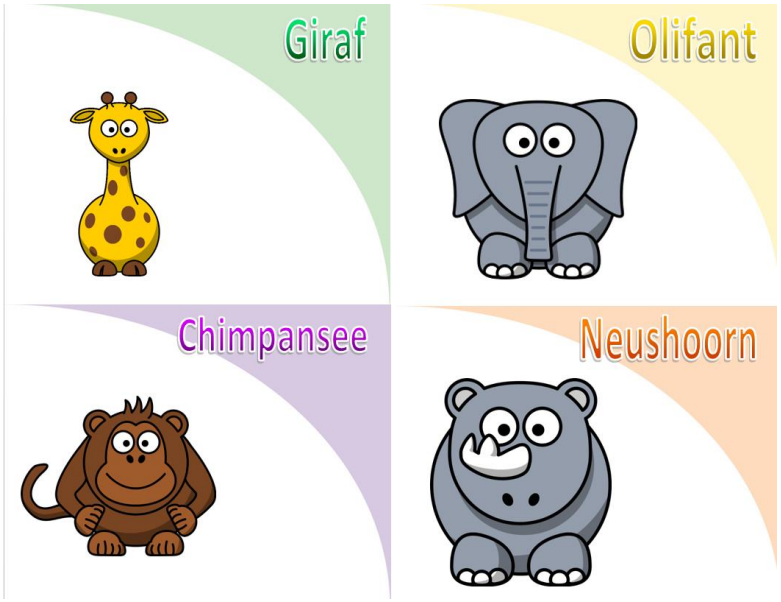
Mappenstructuur van het project

In onderstaande afbeelding (links) is de mappenstructuur van het hele project te zien inclusief de klassen waar er modules getest werden. De *package* “experimentenJessica” bevat al de code voor de experimenten. In de afbeelding rechts wordt de *package* “audioService” getoond, hier werkte ik aan het ontvangen en verzenden van de audio voor de NAO robot.

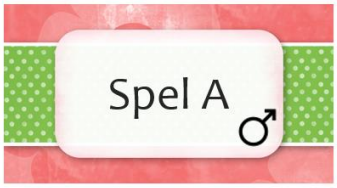


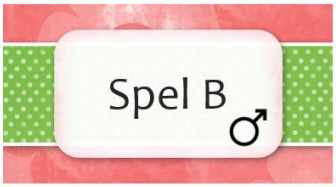
Benodigdheden voor het experiment

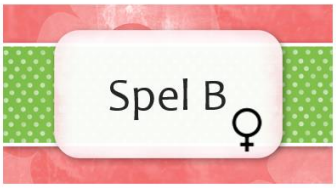
Voorbeeld van de dieren kaartjes.



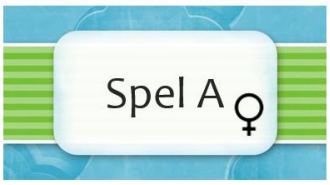
Kaartjes voor een introvert spel.

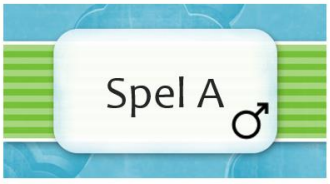


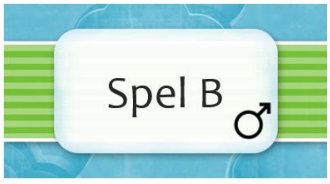




Kaartjes voor een extravert spel.







Experiment 2: Zoo Game

https://uantwerpen.eu.qualtrics.com/SE/?SID=SV_1MlRpO3ngazKcWF

Deelnemers:

In totaal 48, gebalanceerd over introvert/extravert en geslacht (CLiPS: 24 & COMmeto: 24)

- Leeftijd zo gespreid mogelijk & proefpersonen moeten niet autistisch zijn.
- De helft van elke combinatie moet het spel spelen met de introverte robot en de andere helft met de extraverte robot. Dit levert de volgende proefpersonen voor COMmeto op:

	SPEL A: INTROVERTE ROBOT	**SPEL B: EXTRAVERTE ROBOT**	**TOTAAL**
INTROVERTE VROUWEN	3	3	6
INTROVERTE MANNEN	3	3	6
EXTRAVERTE VROUWEN	3	3	6
EXTRAVERTE MANNEN	3	3	6
TOTAAL	12	12	24

-

Spel

Na de persoonlijkheidstest moet de proefpersoon een kaartje trekken, **rood** voor introverte personen, **blauw** voor extraverten. Deze kaartjes zijn gemarkeerd voor geslacht. Daarop staat spel A of spel B.

- **Spel A** is het introverte spel
- **Spel B** is het extraverte spel.

De proefpersoon mag niet weten welke versie het is, noch dat het gaat om de perceptie van introversie/extraversie. Hierna gaat dit kaartje uit het spel. Zijn alle kaartjes op, dan mogen ze natuurlijk terug in het spel komen om meer proefpersonen te verzamelen.

- Na het spel zal de proefpersoon moeten aanduiden welk spel hij of zij speelde en dit spel evalueren.

Enquêtestructuur:

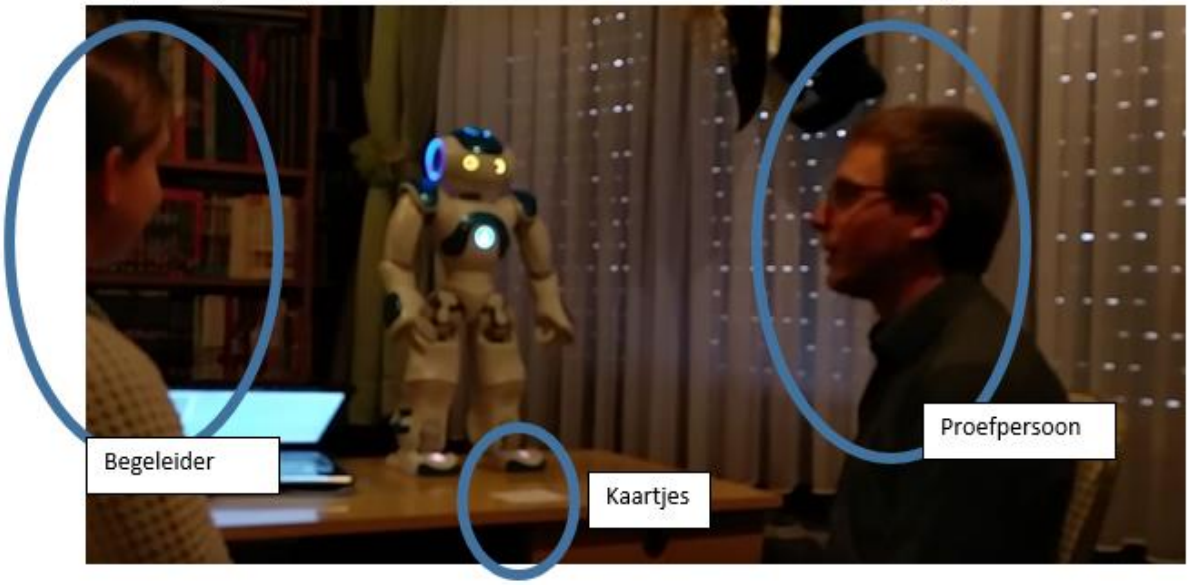
- Korte introductievragen (leeftijd, geslacht, taal etc.)
- Online persoonlijkheidstest (big 5)
- Evaluatie van het spel

Enquête

1) Proefpersoon vult enquête in:
    - https://uantwerpen.eu.qualtrics.com/SE/?SID=SV_1MlRpO3ngazKcWF
2) Op basis van de score voor extraversie trekt hij of zij een kaartje (de score bepaalt de kleur).
    a. **ROOD:** score onder de 50 (introvert)
    b. **BLAUW:** score van 50 of meer (extravert)
3) Op basis van dit kaartje wordt het spel opgestart
    a. **SPEL A:** introvert spel
    b. **SPEL B:** extravert spel
4) Als het spel afgelopen is moet de proefpersoon het spel nog evalueren.

Spel voorbereiding:

1) Vul het **IP adres van de robot en de naam van de proefpersoon** in de grafische interface
2) Het is belangrijk dat de **proefpersoon links** zit en de **begeleider rechts** (vanuit het oogpunt van NAO). De stapel kaartjes dienen ergens 20 cm voor de linkervoet van de robot te liggen



-

Gelieve de proefpersoon de instructie te geven duidelijk te articuleren en luid te spreken en steeds met ja of neen te antwoorden. De proefpersoon moet steeds in de richting van de robot spreken! Vertel hem of haar ook dat NAO enkel luistert wanneer er een beep is geweest. De robot heeft het woord verstaan als hij twee maal een beep maakt. De luisterperiode wordt ook afgesloten met een beep.

Spelverloop:

- Aan het begin van het spel zal de begeleider kort de robot moeten voorstellen aan de proefpersoon, zodat de robot verlegen/enthousiast kan reageren. De robot zal eerst wuiven en een algemene begroeting uiten. Zodra hij is uitgesproken en de arm naar beneden is heeft de begeleider 3 seconden voor NAO en de proefpersoon aan elkaar voor te stellen:

Dag, <*naam van het kind*>. Dit is NAO. NAO, dit is <*naam van het kind*>.

- Bij de extraverte robot (B) is dit genoeg. De introverte (A) gaat echter verlegen zijn en wegdraaien. Dan heeft de begeleider 2 seconden om NAO te overhalen de proefpersoon toch te groeten:

NAO! Niet verlegen zijn!

- Wanneer de introverte robot dan de proefpersoon gegroet heeft, kijkt hij terug naar voor. De begeleider heeft dan 4 seconden om een spel voor te stellen:

NAO wilt graag een spel met je spelen, nietwaar NAO?

Zoo verhaal

Introvert verhaal

Oh, wat leuk dat je dat vraagt! Wel, ik was jarig op een zaterdag, dus kon ik eens lekker genieten en uitslapen in mijn zachte, warme bed met mijn heerlijke, donzige kussens. Rond 11 uur zijn mijn vriendin en ik gaan brunchen in het hotel bij ons in de straat. Dat was echt grandioos. We kregen een uitgebreid buffet van zowel warme als koude gerechten, gevolgd door een ware suite van allerlei zoetigheden. We mochten ook zoveel koffie en thee drinken als we maar wilden. Om 14 uur zijn we dan naar de dierentuin vertrokken, waar we afgesproken hadden met twee bevriende robots die we al jaren kennen, al sinds het secundair onderwijs. Er was wel iets wat ons mateloos stoorde. Het was zo vervelend dat ik oorspronkelijk erg geïrriteerd en zelfs geërgerd was. Omdat het zo'n mooie en zonnige dag was, was het enorm druk in de dierentuin. Overal liepen en gilden kinderen en sommige mensen hadden zelfs hun irritante en stinkende hond meegebracht! Maar dankzij mijn eeuwig optimistische vriendin werd het een onvergetelijk, magisch avontuur. Onmiddellijk nadat we een kaartje gekocht hadden, zijn we naar de flamingo's gegaan. Dat vonden we zo'n sierlijke, prachtige dieren. Het is dan ook niet verwonderlijk dat het de lievelingsdieren van mijn vriendin zijn! Daarna hebben we achtereenvolgens, de leeuwen, de tijgers en de bruine beren bezocht. Het is bijna onbeschrijfelijk wat voor macht deze dieren uitstralen. Pareltjes van de natuur. Rond 16 uur zijn we gaan genieten van een drankje in het restaurant van de kinderboerderij. De dieren die daar zitten zijn misschien maar kleine boerderijdieren, maar ze zijn zo ongelooflijk schattig. We hebben een aantal konijnen kunnen aaien. We leerden er ook dat een konijn geen knaagdier is. Iedereen denkt van wel, maar het is eigenlijk een haasachtige. Niets maakt mij zo gelukkig als zo'n nieuwe, fascinerende weetjes! Het dierenrijk is dan ook zo verbazingwekkend en wonderlijk. Ten slotte zijn we nog de reptielen en de amfibieën gaan observeren. Een verzorger daagde mijn vriendin uit om een slang te aaien, en ze deed het nog ook! Dat was echt heel dapper van haar. Het was een zeer aangename dag! Ik kijk er al naar uit om opnieuw jarig te zijn! Al kies ik volgend jaar wel voor een minder drukke bestemming, denk ik.

Extravert verhaal

Ha! ’t Was een geweldige dag! Ik verjaarde dus in ’t weekend, en dat betekent uitslapen natuurlijk, nietwaar? Wat zou je zelf doen? ‘s Middags zijn mijn liefje en ik gaan ontbijten in een hotel. Of lunchen. Ah ja, ze noemen het eigenlijk brunchen. We zijn dus samen gaan brunchen. Dat was echt super, hé. We kregen daar enorm veel warm en koud eten, en daarna nog eens superveel toetjes. We hebben gesmuld! We konden niet meer denken aan eten! En koffie en thee was er zoveel je maar wou drinken. Echt fantastisch! Je zal zelf ook wel weten hoe leuk brunchen is. Dat wordt toch steeds vaker gedaan tegenwoordig. Iedereen doet het. Waarom zou je je ook niet laten verwennen en jezelf niet eens trakteren? ‘s Middags zijn we naar de dierentuin geweest met een aantal oude vrienden van ons. Ja, wij kennen die robots echt al lang, heel ons leven eigenlijk. We zijn daar vroeger nog mee naar school geweest, ’t zijn echt goede vrienden van ons. Je kent dat wel, hé, zo’n bende waar je altijd plezier mee kan beleven. Plezier verzekerd! ’t Was super goed weer. Veel zon, lekker fris briesje af en toe. Echt een mooie dag. Dan is het natuurlijk niet rustig in de dierentuin. Maar dat weet je op voorhand, toch? Zo heb ik het nog het liefst. De hele tuin bruiste van het leven. Kinderen die spelen, lachen, huppelen: zich amuseren. Honden die wat liggen te rollen in ’t gras. Snuffelen hier, graven daar. En opnieuw dicht bij hun baasje kruipen. Echt zo’n typische gezellige drukte van een zomerdag. Zoals die groep jongeren in de show met de zeehonden. Zeeleeuwen. Plezier dat die hadden. Niet de zeeleeuwen, hé. De jongeren. Die zeeleeuwen zullen ’t ook wel tof gevonden hebben. Maar die jongeren toch nog meer denk ik. Zo zag het er toch uit. We zijn eerst naar ’t hok van de flamingo’s geweest. Heel mooie beestjes, hoor! Die kleur! Van dat mooie roze. Mijn vriendin is er dol op. En dat balanceren! Geen idee hoe ze dat doen! Ik zou al lang gevallen zijn, hoor! Jij toch ook! Geef maar toe! Daarna zijn we naar de leeuwen en de tijgers geweest. En ehm. Ah ja, de beren ook. Zo’n machtige dieren dat dat zijn! De natuur wist waar ze mee bezig was. Die stralen toch zo’n macht uit! Echt prachtig gewoon. En hoe ze bewegen! Sluip links, sluip rechts, kijk om je heen, besluit niets te doen, sluip terug naar links. Ze weten dat niets of niemand hen kan aanvallen of doden en laten dat zien ook! De macht van de natuur valt niet te onderschatten, dat kan je onmogelijk ontkennen. We zijn ook gaan genieten op het terras van de kinderboerderij met een drankje. Schattig, al die beestjes! We hadden geluk en wisten een aantal konijnen te pakken te krijgen zodat we ze konden aaien. ’t Schijnt trouwens dat konijnen geen knaagdieren zijn! Ik denk dat al heel mijn leven! Geef toe, iedereen denkt dat toch! Jij dacht dat ook, toch? Maar ’t zijn hazen! Geen knaagdieren! Voor we naar huis gingen, zijn we nog gaan griezelen bij de reptielen en allerlei andere kruipertjes. Een bewaker vroeg mijn vriendin een slang aan te raken. Ik had het nooit verwacht, maar ze deed het! Ze raakte dat glibberige beestje gewoon aan. Ze tilde het op, bekeek het uitgebreid en aaide het. Daar stond ik toch wel even van te kijken, hoor! Vind jij dat ook niet indrukwekkend? ’t Was een toffe dag! Ik zal blij zijn als ik opnieuw verjaar. Weer leven in de brouwerij! Joepie! Het aftellen kan beginnen!

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Externe sturing van de NAO-robot voor mens-robot-communicatie bij autistische kinderen

Richting: **master in de industriële wetenschappen: elektronica-ICT**
Jaar: **2016**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Geerts, Willem

Datum: **6/06/2016**