Big Independent Exploration Robot

Daan de Graaf (10360093)
Matthijs Klijn (10447822)
Xander Koning (10439099)
Patrick van der Pal (6223931 / 10013555)
Mike Trieu (6366295 / 10105093)
Floris Turkenburg (10419667)

4 juli 2014

Samenvatting

Dit verslag beschrijft het experiment, waarbij meerdere Mobiele Android Apparaten, als cliënten, communiceren met een laptop, die dient als server, waarbij het de bedoeling is dat de Omnibots, die via USB verbonden zijn met de Mobiele Android Apparaten, zo snel mogelijk hun willekeurig gekozen eindpunt bereiken. Hiervoor hebben we gebruik gemaakt van MBEDs om de servo motoren van de Omnibots aan te sturen, van bluetooth om te communiceren tussen de server en de cliënten en van Python om een GUI op te zetten, zodat het geheel beter te volgen is tijdens de uitvoer van het experiment. De "wereld" is opgebouwd uit een coördinatenstelsel van QR-codes, die we scannen door middel van Mobiele Android Apparaten, bevatten de locatie van de QR-code en een willekeurig Robot-ID, die wordt gebruikt om de eindpunten te bepalen.

Inhoudsopgave

1 Inleiding

De term Net-Centric wordt gebruikt om de optimalisatie van het gebruik van hulpmiddelen en de synchronisatie van gebeurtenissen en de gevolgen binnen netwerken te beschrijven. Zo'n netwerk kan bestaan uit mensen, apparaten, informatie, maar ook diensten, die onderling verbonden zijn, waarbij de traditionele server-client relatie minder belangrijk wordt gevonden (Cole, 2014). Het is een nieuw paradigma van de server-client computing (Michael et al., 0). Oftewel, als we kijken naar de complete term Netcentric Computing, dan vervallen de mensen uit het netwerk, omdat het geheel zo veel mogelijk geautomatiseerd dient te worden om de optimalisatie te verhogen (Cole, 2014).

De opdracht bestaat uit het bedenken en het uitvoeren van een experiment, logischerwijs gebaseerd op Netcentric Computing. Wij, als groep B.I.E.R., hebben besloten om met de Omnibots te werken. De Omnibots zijn geschikt voor dit experiment, omdat deze verbonden kunnen worden met een Animatronic Lab Board en via USB ook met een Mobiel Android Apparaat. Via het Lab Board kan de Omnibot automatisch aangestuurd worden en via het Mobiele Android Apparaat kan dit geheel onderdeel zijn van een netwerk. Beide toevoegingen zorgen ervoor dat de Omnibot gebruikt kan worden voor Netcentric Computing.

Als experiment hebben wij ervoor gekozen om in een ruimte verschillende eindpunten te plaatsen, waar de verschillende Omnibots zichzelf naar toe dienen te rijden. Aangezien de Omnibot verbonden is met een Mobiel Android Apparaat, die op zijn beurt weer een camera heeft, hebben wij besloten de eindpunten te representeren als QR-codes.

2 Materiaal

2.1 Hardware

- Omnibots
- MBED NXP LPC1768 microcontroller
- Animatronic Laboratory Board
- HTC Desire C

2.2 Software

- Android Studio 0.5.2 (API19)
- Online MBED compiler
- Tera Term
- HTC Sync Manager
- pythonxy 2.7
- Android App Framework
- MBED binary Framework

2.3 Libraries

- Zbar bar code reader
- pyBluez
- Servo library

3 Methode

Om de eindpunten te vinden, moeten de Omnibots zelf binnen de ruimte zoeken. Wij hebben besloten dat zolang het eindpunt van een Omnibot nog niet is gevonden, deze in willekeurige richtingen blijft rijden. Op het moment dat zijn eindpunt is gevonden, maakt de Omnibot gebruik van een kortstepad-algoritme om zo snel mogelijk bij zijn eindpunt aan te komen. Als het eindpunt is bereikt, dan wordt de Omnibot inactief of uitgeschakeld.

Het volgende besluit dat we als groep moesten nemen, was hoe we de locatie konden representeren. Om het einde te bereiken moesten we uiteraard kunnen bepalen waar het eindpunt zich bevindt ten opzichte van de Omnibot. Ook hier hebben we ervoor gekozen gebruik te maken van QR-codes. Dit hebben we bereikt door een coördinaten-rooster op te stellen met op ieder coördinaat een QR-code. Op deze manier is het simpeler om voort te bewegen over het rooster door simpelweg een richting, noord, west, zuid of oost, te kiezen.

Om het gehele experiment succesvol af te ronden, hebben we het opgedeeld in drie hapklare brokken door het werk op te splitsen, namelijk het realiseren van de communicatie tussen de MasterApp en de SlaveApp, het installeren van een QR-scanner, het opzetten van een gebruiksvriendelijke GUI en het rijklaar maken van de Omnibots.

3.1 Communicatie

Ten eerste de communicatie tussen de MasterApp en de SlaveApp. Hiervoor hadden we de keuze uit twee types van draadloze communicatie, namelijk bluetooth en wifi. Uiteindelijk hebben we besloten voor bluetooth te kiezen, voornamelijk omdat we eerder met bluetooth gewerkt hebben en wifi nog compleet nieuw voor ons is.

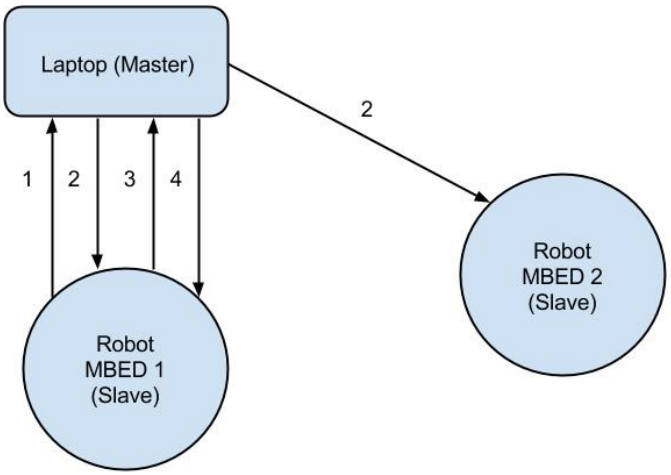
De volgende stap is te ontdekken hoe bluetooth gebruikt kan worden om data te versturen en vervolgens weer te ontvangen.

Voor de verbinding tussen een android telefoon en de masterserver die draait op de computer hebben we gezocht naar een library in Python die bluetooth verbindingen kan afhandelen. Uiteindelijk zijn we uitgekomen op PyBluez, een goed gedocumenteerde en veel gebruikte Python library. Voor Android hebben we de standaard bluetooth API gebruikt, we hebben in Android het mac-address van de bluetooth adapter van de masterserver gezet, en zo konden deze snel met elkaar verbinden. Verder hebben we in Android en in de masterserver een aparte thread aangemaakt voor bluetooth zodat de verbindingen altijd kunnen worden afgehandeld zonder de main thread onnodig te vertragen. Met write functies en read functies hebben we de communicatie tussen de master server en de android slaves laten plaatsvinden.

Om de server meerdere verbindingen te laten afhandelen gebruiken we de select functie. Deze functie zorgt ervoor dat het programma meerdere file descriptors in de gaten kan houden zonder dat het programma blokkeert (Kerrisk, 2014). In het geval van Bluetooth verbinding houdt het programma meerdere sockets in de gaten. Wanneer één van de sockets iets te doen heeft, gaat het programma de bijbehorende routine uitvoeren.

3.1.1 Blokschema

Nu we de data kunnen versturen, kunnen we specifieke berichten sturen van zowel het Mobiele Android Apparaat naar de laptop als van de laptop naar het Mobiele Android Apparaat. Welke berichten verstuurd worden tussen deze twee, wordt getoond in het onderstaande blokschema.



Figuur 1: blokschema communicatie

1. Omnibot 1 scant de QR-code en stuurt de gescande data naar de laptop. De QR-data bevat de locatie waar de QR-code zich bevindt en een Robot-ID die mogelijk overeenkomt met één van de actieve robots. Dit wordt in een aparte tabel bijgehouden.

2. De laptop analyseert de binnengekomen data en stuurt de locatie vanuit de data terug naar Omnibot 1. Mocht de Robot-ID overeenkomen met een actieve Omnibot, dan wordt de binnengekomen locatie, als eindbestemming, ook doorgestuurd naar de desbetreffende Omnibot. Dit is te zien in het voorbeeld met Omnibot 2.

3. Omnibot 1 bepaalt aan de hand van zijn locatie en zijn status een nieuwe richting. Deze richting wordt verstuurd naar de laptop.

   – Status: Als de eindbestemming van de Omnibot bekend is - een andere Omnibot heeft namelijk z'n eindbestemming gevonden - dan wordt de richting bepaald door een kortste pad oplosser. Als dit niet het geval is, lees: de Omnibot bevindt zich nog in de "roaming"-fase, dan wordt de richting willekeurig gekozen.

4. De laptop bepaalt of Omnibot 1 vanaf z'n huidige locatie in de binnengekomen richting kan rijden. Dat wil zeggen, op de beoogde locatie staat al een Omnibot of is naar die locatie onderweg, of de locatie bevindt zich buiten de opstelling. Zo ja, dan wordt een bevestiging naar Omnibot 1

gestuurd. Zo nee, dan wordt de huidige locatie van Omnibot 1 weer terug gestuurd. Dit is besloten zodat het aantal verschillende berichten dat ontvangen wordt door de Omnibot beperkt blijft.

5. Als de Omnibot een confirmation ontvangt, dan beweegt de Omnibot naar de beoogde locatie, waarna weer wordt teruggesprongen naar "Stap 1". Als de Omnibot z'n huidige locatie weer ontvangt, dan wordt direct teruggesprongen naar "Stap 3".

3.2 QR-scanner

Om het Mobiele Android Apparaat QR-codes te laten scannen en de data die het scant te gebruiken, moeten we een QR-scanner implementeren in de Omnibot applicatie. Hiervoor is gezocht naar een library, die snel QR-codes kan scannen en makkelijk te implementeren is. Uiteindelijk is gekozen voor Zbar bar code reader library. Deze library ondersteunt PC, Iphone en Android. De QR-scanner moet, nadat het een QR-code gescand heeft, de data uit de QR-code meteen door sturen naar de Bluetooth server. Daarnaast mag, wanneer de Omnibot aan het rijden is, niets gescand worden. Als de QR-scanner tijdens het rijden scant, bestaat de kans dat de QR-scanner dezelfde QR-code achterelkaar scant, omdat de Omnibot nog steeds gericht staat op de QR-code, die het al gescand heeft.

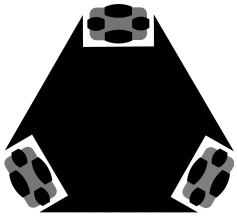
We hebben besloten om de QR-codes op de grond te plaatsen, zodat de QR-codes de Omnibots niet in de weg zitten. Om de ideale grootte van de QR-codes voor de QR-scanner te bepalen, worden QR-codes in verschillende grootte geprint. Zo kunnen we testen welke grootte het beste werkt. Ook worden de verschillende hoeken getest waaronder de QR-scanner wel of niet werkt. Dit wordt gedaan vanwege de manier waarop het Mobiele Android Apparaat, die alleen schuin naar de grond gericht kan worden, is bevestigd op de Omnibot.

3.3 GUI

Dan over de GUI. Wat houdt een GUI precies in? Wat is het verschil tussen een gebruiksvriendelijke en niet-gebruiksvriendelijke GUI? Een GUI is een Grafische User Interface, welke schermen en iconen bevat die informatie aan de gebruiker tonen, waarbij deze door zijn muis of toetsenbord te gebruiken iets hieraan kan manipuleren. Onze GUI bevat slechts een knop om hem af te sluiten en zou dus niet GUI genoemd mogen worden. Vandaar dat we deze in het vervolg een Grafische Display (GD) zullen noemen.

De GD die wij hebben gemaakt is gebruiksvriendelijk, en dan in het opzicht dat bij opstarten van de python applicatie de GD wordt opgestart en de gebruiker verder niets hoeft te doen. De GD toont eerst een lege map, omdat er nog niets is gescand door de Android toestellen die zich op de robots bevinden. Als er een QR-code wordt gescand, dan wordt deze getoond op de map. Indien een gescande QR-code een id heeft die gelijk is aan die van één van de robots, dan betekent dit dat de locatie het eindpunt van deze robot is. Dat dit een eindpunt is wordt duidelijk gemaakt door het vakje op de GD te printen met een randje in de kleur van bijbehorende robot. Normale vakjes zijn wit met een zwarte rand.

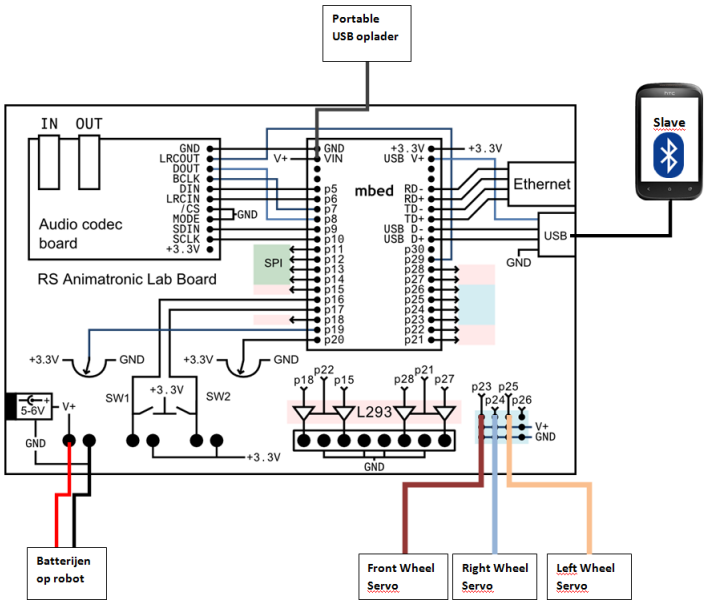
3.4 Omnibot



Figuur 2: Omnibot

De Omnibot bevat drie Servo motoren. De manier waarop de wielen hieraan zijn gekoppeld, maakt het ingewikkeld om één richting op te rijden. Om de Omnibot bijvoorbeeld naar links te laten rijden, moeten de snelheden van de drie Servo motoren op een bepaalde manier ingesteld zijn. Hiervoor kunnen snelheid-vectoren worden gebruikt, maar omdat we een vierkant coördinatenstelsel gebruiken, hebben we ervoor gekozen om de Omnibot alleen voor en achteruit te laten rijden. Voor links en rechts gaat de Omnibot eerst 90 graden draaien en dan vooruit en dan 90 graden terug draaien. Zo hoeven de Servo motoren alleen stil te staan of op volle snelheid draaien.

3.4.1 Opstelling



Figuur 3: blokschema communicatie

De Omnibot opstelling bestaat, zoals te zien is in bovenstaande figuur, uit vier verschillende onderdelen. Het eerste, meest duidelijke onderdeel is de MBED zelf. Dit is het middelpunt van de opstelling, aangezien alle overige onderdelen hier direct mee verbonden zijn.

Ten eerste is de MBED via de P23, P24 en P25-poorten verbonden met de Omnibot, meer specifiek met zijn wielen. Namelijk poort P23 met het voorwiel, poort P24 met het rechterwiel en poort P25 met het linkerwiel. Ook is de MBED verbonden met het Mobiele Android Apparaat, in ons geval de HTC Desire C, via een simpele USB-kabel. Logischerwijs heeft de gehele opstelling ook een stroomtoevoer nodig. Dit is gesplitst in twee verschillende toevoeren.

De eerste is een draagbare USB oplader die direct verbonden is met de MBED. De tweede is een set van vier batterijen die zich in de Omnibot bevinden, die verbonden zijn met de stroomtoevoer op het Animatronic Lab Board.

3.4.2 MBED en calibratie

Voor het programmeren van de MBED hebben wij het framework gebruikt dat ons werd aangeboden door de UvA. Dit framework bevat de nodige code voor het verbinden van de MBED met een Android applicatie op een mobiel. De wielen van de Omnibot werken met Servo motoren, en om deze aan te sturen hebben we gebruik gemaakt van de Servo library van Simon Ford. Door de pulsewidth van de motoren in te stellen kan men de wielen links- of rechtsom laten draaien. Het probleem wat wij hier tegen kwamen was dat de twee Omnibots een verschillende pulsewidth waarde hadden voor de neutrale stand (de wielen draaien niet). Om dit probleem te verhelpen hebben we de Servo library aangepast, zodat het mogelijk is om de neutrale stand in te stellen. Om te bepalen welke robot de code uitvoert, en welke neutrale waarde dus gebruikt moet worden, kijken wij naar het MAC-adres van de MBED. Handmatig hebben wij de neutrale waardes vastgesteld van de Omnibots en deze verbonden aan het MAC-adres van de MBED die zich op de Omnibot bevindt. Deze waardes worden ingesteld zodra de MBED wordt verbonden met een mobiel.

Van de mobiel ontvangt de MBED een richting, en aan de hand hiervan gaat hij rijden in deze richting. Aangezien de Omnibot niet zelf kan vast stellen hoe ver hij rijdt of draait, hebben wij door middel van uitproberen de waardes bepaald. Zo draait de robot niet standaard 90 graden, maar draait hij voor een bepaalde tijd een kant op. Omdat het door de tijd wordt bepaald, levert dit problemen op als de batterijen een ander vermogen geven (als zij bijvoorbeeld bijna leeg zijn), dan draaien de wielen namelijk sneller of langzamer, en wordt er respectievelijk een grotere of kleinere afstand afgelegd in dezelfde tijd.

4 Resultaten

Normaal gesproken zouden we in de sectie ‘Resultaten’ een aantal statistieken willen laten zien, waarbij duidelijk wordt hoe vaak het experiment goed is verlopen en hoe vaak bepaalde fouten voor zijn gekomen. Maar vanwege een gebrek aan tijd, hebben we het experiment niet voldoende malen uit kunnen voeren. Vandaar dat wij hebben besloten om van een aantal belangrijke stappen binnen het experiment een uitdraai en een uitleg te geven, zodat hopelijk duidelijk wordt wanneer herhalingen van dit experiment dezelfde status hebben bereikt als wij.

4.1 SlaveApp’s layout

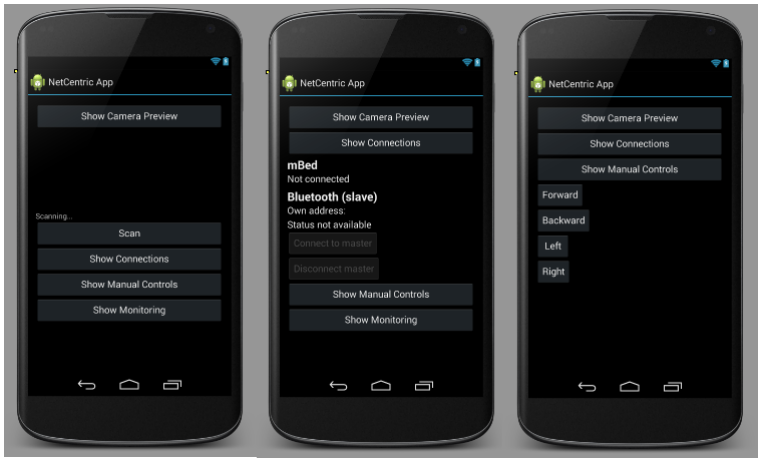
Daarentegen beginnen we met de SlaveApp, die ongeacht het aantal uitvoeringen van het experiment wel zijn bedoelde eindvorm bereikt heeft.

Onder het kopje “Show Camera Preview” bevindt zich de QR-scanner, deze wordt logischerwijs het meeste gebruikt. Op het moment dat de SlaveApp niet is verbonden met de MasterApp moet op de ‘Scan’ knop gedrukt worden om opnieuw te kunnen scannen.

Onder het kopje “Show Connections” wordt door de ‘Connect to master’ knop verbinding gemaakt met de MasterApp. Logischerwijs wordt met de knop ‘Disconnect master’ deze verbinding weer verbroken.

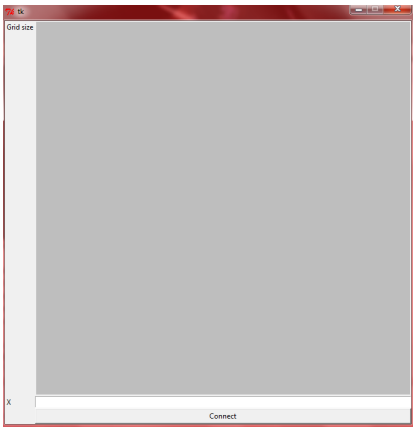
Onder het kopje “Show Manual Controls” kan de MBED handmatig bestuurd door middel van de knoppen ‘Forward’, ‘Backward’, ‘Left’ en ‘Right’.

Onder het kopje “Show Monitoring” worden ten slotte de verschillende regels geprint die gebruikt kunnen worden voor het debuggen.



Figuur 4: De layout van de SlaveApp

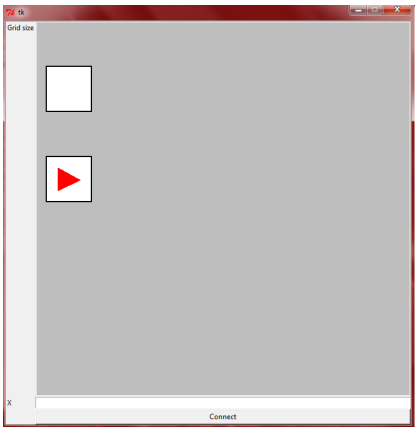
4.2 Het experiment



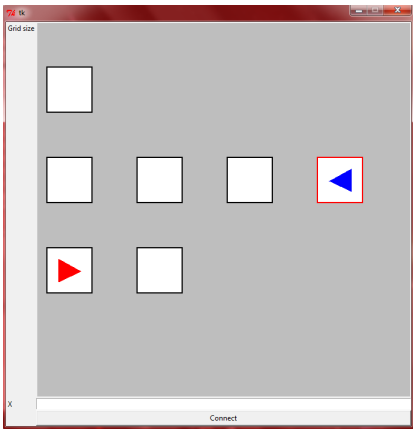
Figuur 5: De GD tijdens het opstarten

De eerste stap van het experiment is het opstarten van de GD en de server, waarna de SlaveApps verbinding kunnen maken en er een willekeurig eindpunt wordt bepaald.

Nu wordt door een SlaveApp een QR-code gescand, en dus z'n locatie bepaald. Vervolgens wordt, tijdens de 'roaming'-fase, door de SlaveApp een willekeurige richting bepaald, in Figuur 6 was dit 'East', waarna de MasterApp bepaald of de locatie ten oosten van de huidige locatie beschikbaar is voor de desbetreffende SlaveApp. Als dit niet het geval is, dan wordt een nieuwe richting gekozen. Is de locatie wel beschikbaar, zoals in dit figuur het geval is, dan rijdt de Omnibot in oostelijke richting. Op het moment dat de nieuwe locatie bereikt is, begint deze stap weer van vooraf aan.



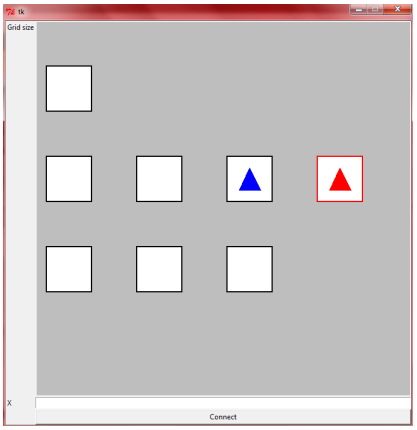
Figuur 6: De GD tijdens de 'roaming'-fase



Figuur 7: De GD op het moment dat het eindpunt is gevonden

Zolang de SlaveApp zich in de 'roaming'-fase bevindt wordt er willekeurig gemanoeuvreerd door het coördinatenstelsel, maar zodra een eindpunt wordt gevonden, zoals voor de rode Omnibot te zien is in Figuur 7, wordt de 'roaming'-fase beëindigd en treedt het kortstepad-algoritme in werking. Het kortstepad-algoritme bepaalt vanaf nu de richting in plaats van de willekeurige richtingen.

Op het moment dat de Omnibot zijn eindpunt bereikt heeft, zoals in Figuur 8 te zien is, wordt deze inactief en kan deze dus niet meer bewegen, uiteraard totdat het volgende experiment wordt opgestart met andere eindpunten voor de SlaveApps.



Figuur 8: De GD nadat het eindpunt bereikt is

5 Discussie

Tijdens de uitvoering van dit experiment, zijn we tegen een aantal punten aangelopen, waarvan wij denken dit op de volgende manieren op te kunnen lossen.

5.1 Verbeterpunten voor de Omnibots

- Stroom voor de motoren van de Omnibot via de batterijen in plaats van de stekker. Evenals stroom voor de MBED via de draagbare USB oplader in plaats van de computer door een USB kabel. Dit zorgt ervoor dat de Omnibot volledig draadloos over het rooster kan voortbewegen. Dit ontneemt de mogelijkheid dat de kabels van verschillende Omnibots in de knoop raken. Het nadeel van deze aanpassingen is dat de stroomtoevoer niet meer constant lijkt te zijn. Dit kan komen door het uitputten van de batterijen. Dit heeft als gevolg dat de Omnibot minder vermogen krijgt en minder ver rijdt en draait.

- Op dit moment hebben de Omnibots te weinig grip, dus het rijden is te wisselvallig om zeker te zijn van het resultaat. Hierdoor heeft calibreren niet of nauwelijks zin. Dit kan opgelost worden door enerzijds de ondergrond te veranderen en/of het materiaal of het profiel van de wielen van de Omnibot te veranderen. Daarnaast zitten er sensoren op de Omnibot die kunnen helpen om de rijrichting te corrigeren.

- De QR-scanner op de Omnibot werkt alleen goed als deze redelijk recht gericht staat op de QR-scanner. De grootte van de QR-code had ook invloed op het scannen. Dit zou verbeterd kunnen worden door een beeldbewerkings algoritme toe te passen. Het algoritme dat het onderzoek van Liu et al. (2008) beschrijft, werkt goed en nog steeds real-time.

5.2 Verbeterpunten voor de communicatie

- De functionaliteit van de MasterApp verspreiden over de verschillende SlaveApps. Dit verhoogt het “Netcentric Computing”-gehalte. Bijvoorbeeld door continu één van de SlaveApps te kiezen die tijdelijk wordt omgezet in een MasterApp. Dit voorkomt crashes als de huidige MasterApp wegvalt, want er kan simpelweg een nieuwe aangewezen worden. Het nadeel hiervan is dat de MasterApp niet kan rijden, dus de Omnibot die tijdelijk als MasterApp functioneert staat stil totdat een volgende SlaveApp wordt aangewezen als MasterApp.
  Een andere mogelijkheid is om de functionaliteit van de SlaveApp en de MasterApp te combineren in één enkele applicatie, zodat iedere Omnibot ook een MasterApp is, maar ondertussen nog steeds over het rooster kan voortbewegen. Hierbij moet wel veel aandacht worden geschonken aan een goede communicatie, zodat iedere Omnibot de laatste gegevens binnen krijgt voordat een beslissing genomen wordt.

- Eventueel zou er nog functionaliteit ingebouwd kunnen worden, zodat op het moment dat iedere Omnibot zijn eindpunt heeft bereikt, deze allemaal een nieuw eindpunt toegewezen krijgen, zodat het geheel weer opnieuw begint.

- Door het kortstepad-algoritme wordt op dit moment nog geen rekening gehouden met de eindpunten van de andere Omnibots. Het zou, theoretisch gezien, kunnen gebeuren dat er "starvation" optreedt, omdat één van de Omnibots zijn eindpunt heeft bereikt, terwijl een andere Omnibot door het kortstepad-algoritme langs datzelfde punt wordt gestuurd. Door een gebrekkig kortstepad-algoritme is het (nog) niet mogelijk om de Omnibot om de andere heen te laten rijden.

Bibliografie

Cole, B. (2014). What is net-centric computing? *url: http://www.embedded.com/electronics-blogs/cole-bin/4023304/What-is-Net-Centric-Computing-*.

Kerrisk, M. (2014). Select(2). *url: http://man7.org/linux/man-pages/man2/select.2.html*.

Liu, Y., Yang, J., and Liu, M. (2008). Recognition of qr code with mobile phones. In *Control and Decision Conference, 2008. CCDC 2008. Chinese*, pages 203–206. IEEE.

Michael, L. R. T., Ada, F. W. C., Sang, M. Y., and Lung, C. K. (0). Reliability analysis in net-centric computing.

Appendix 1: Uitgebreide gebruikte materialen

Hardware

- Omnibots
- MBED NXP LPC1768 microcontroller
  - High performance ARM®Cortex™-M3 Core
  - 96MHz, 32KB RAM, 512KB FLASH
  - Ethernet, USB Host/Device, 2xSPI, 2xI2C, 3xUART, CAN, 6xPWM, 6xADC, GPIO
- Animatronic Laboratory Board
- HTC Desire C
  - Android version: 4.0.3
  - Developer options:
    - USB debugging
    - Stay awake

Software

- Android Studio 0.5.2 (API19)
  - `http://developer.android.com/sdk/installing/studio.html`
- Online MBED compiler
  - `https://mbed.org/compiler/`
- Tera Term
  - `http://ttssh2.sourceforge.jp/index.html.en`
- HTC Sync Manager
  - `http://www.htc.com/us/software/htc-sync-manager/`
- pythonxy 2.7 (Volledige installatie)
  - `https://code.google.com/p/pythonxy/`
- Android App Framework
  - `http://staff.science.uva.nl/~edwin/Downloads/NCC/NC28-05-2014.zip`
- MBED binary Framework
  - Import uva_nc in the online MBED compiler. Bij het importen kan er gezocht worden naar dit programma

Libraries

- Zbar bar code reader
  - `http://sourceforge.net/projects/zbar/`
- pyBluez
  - `https://code.google.com/p/pybluez/`
- Servo library
  - Import Servo library urlhttp://mbed.org/cookbook/Servo

Appendix 2: Broncode

Zie bijlage ncc_bier.tar.gz

- MasterServer/gui.py
- MbedBinary/Servo/Servo.cpp
- MbedBinary/Servo/Servo.h
- MbedBinary/NetCentricApp.cpp
- MbedBinary/NetCentricApp.h
- MbedBinary/main.cpp
- RobotApp/app/src/main/java/uva/nc/app/BluetoothThread.java
- RobotApp/app/src/main/java/uva/nc/app/MainActivity.java
- RobotApp/app/src/main/res/layout/activity_main.xml