Robot Tag

Abdel Aberkane, Erik van der Schaaf, Yadvir Singh

Juli 2014

Abstract

Voor het vak Net-centric Computing hebben wij in een groep van drie, een spel ontworpen. De spelregels gaan als volgt; een robot is de tikker, de andere robots vluchten. Heel recht toe recht aan. Een robot variant van het ó zo bekende tikkertje. De tikker kan de andere robot's tikken door de QR-codes van de betreffende robots te scannen, deze zijn geplakt op de zijkanten van de robots. Alle robots kunnen worden bestuurd met een controller, dat is een android toestel waarop onze applicatie op werkt.

1 Inleiding

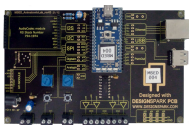
Voor het vak Net-centric Computing, moesten wij als groep zijnde een project uitvoeren. We mochten zelf bedenken wat voor project, het moest uiteraard wel net-centric computing zijn. Dat houdt in dat verschillende apparaten direct met elkaar moeten communiceren, zonder tussenkomst van een master. Aanvankelijk was ons idee om met enkele robots een spel te spelen dat sterk was gebaseerd op verstoppertje. Na een uitvoerige feedback van de docenten hadden wij echter besloten om het ietwat op een andere koers te gooien.

Om het net-centric gedeelte de boventoon te laten voeren, hebben we besloten om een soort robot tikkertje te maken. Hierbij is er een tikker, die de andere robots moet tikken. De andere robots moeten vluchten voor de tikker. Zowel de tikker als de andere robots worden bestuurd door spelers. De communicatie gebeurt onderling en wordt verzorgd door bluetooth. De besturing gebeurt via een zelf-gemaakte applicatie, genaamd ABYAER.

2 Materiaal en uitvoering

2.1 Materiaal

Voor dit project hebben we een breed scala aan materialen gebruikt. Er werden enkele omniwheel robots ter beschikking gesteld. Daar hebben we er twee van gebruikt. Deze omniwheel robots kunnen alle kanten op bewegen, ze worden per stuk aangestuurd door drie Servo motoren. Die drijven de wielen aan. Deze robots werkten op zes AA batterijen.



Figuur 1: mBed microcontroller gemonteerd op een mBed AnimatronicLab pcb

Op de robots zijn mBed's aangesloten, zie figuur 1. Deze mBed's zorgen voor de besturing van de robot's. Deze mBed's worden voorzien van stroom door onze eigen externe laders. Op de robots zijn naast de mBed's ook nog android toestellen gemonteerd. Deze android toestellen, te zien in figuur 2, zijn met een usb kabel verbonden aan de mbed. Ze ontvangen commando's van de bestuurders per bluetooth en geven dit indirect, dus via de mBed, door aan de robot. Het monteren van de telefoon en de mBed op de robot gebeurde met behulp van K'nex stukken.

Figuur 2: HTC C Android telefoon met als taak het aansturen van de mBed micro-controller. Hierbij kan de telefoon ook commando's uitvoeren die hij over Bluetooth ontvangt ("Slave")

Naast de android toestellen die gemonteerd waren op de robots, gebruiken we ook nog andere android toestellen als controller. Deze toestellen waren voorzien van een gebruikersvriendelijke applicatie waar de gebruiker de robot mee kan besturen. Deze commando's worden per bluetooth verstuurd en vervolgens ontvangen op de toestellen die gemonteerd zijn op de robot's.

Figuur 3: Htc One Android telefoon die dient als een Master telefoon waar vanaf commando's kunnen worden verstuurd aan alle Slave telefoons over Bluetooth.

2.2 De uitvoering

2.2.1 Robot besturing

Als eerst zijn we te werk gegaan met de Servo [1] bibliotheek om de robot 9 verschillende cases te geven; naar voren bewegen, naar achter bewegen, op zijn plek linksom draaien, op zijn plek rechtsom draaien, rechts naar voren, links naar voren, rechts naar achteren, links naar achteren en stil staan. Voordat de motoren van de robot worden aangestuurd, worden ze eerst gekalibreerd zodat ze werkelijk stil staan. De functie heet *control_robot(int case)* die een integer meekrijgt met de uit te voeren case.

2.2.2 Video streaming

In het eerste ontwerp is opgenomen dat er een live video stream van het gezichtsveld van de robot zichtbaar moet zijn op de controller. Hiervoor zochten wij naar een geschikte bibliotheek. Na enkele adviezen bleek het opensource project libstreaming het best geschikt voor ons project. Een populaire Android applicatie die van deze bibliotheek gebruik maakt is WIFI Camera. Voor de initiële benchmarks hebben we de applicatie op een Android telefoon geïnstalleerd om te performance te meten. Na onderzoek bleek echter dat de frame-rate dusdanig laag was (<10 FPS) dat deze methode niet geschikt is voor live-streaming.

Hierna vonden we het project Peepers [4] dat geen gebruik maakt van de libstreaming bibliotheek maar wel streeft naar hetzelfde doel. Peepers maakt hierbij gebruik van de MediaEncoder API die al in Android zit verweven. De video stream bestaat hierbij uit MJPEG (Motion-JPEG) wat in feite bestaat uit het achtereenvolgens tonen van JPEG afbeeldingen.

De prestaties hierbij waren boven verwachting. Een JPEG stream met een afmeting van 640x480 behaalden meer dan 20 FPS, wat ruim geschikt is voor een live video stream. Het nadeel van deze applicatie is dat het gebruikte video formaat alleen wordt ondersteunt door een Firefox browser.

2.2.3 Robot identificatie

Voor het te implementeren spel zochten wij naar een manier om robots te kunnen identificeren. Slechts beschikkend over sensoren, is de keus gevallen op een QR-code die op elke robot word geplaatst en door andere robots kan worden gescand. Zodoende kunnen robots elkaar herkennen en "tikken".

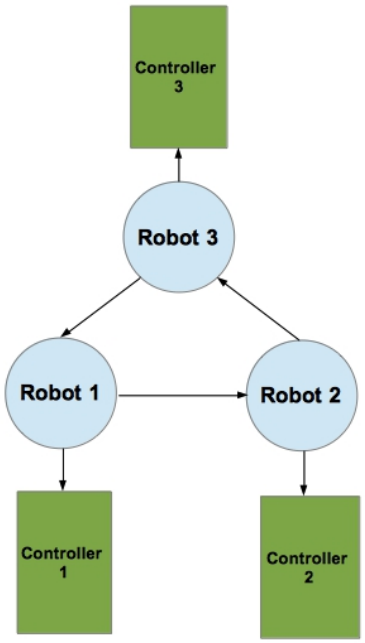
De bekendste bibliotheek voor het scannen van QR-codes is Zxing [2]. De Zxing libary is ontzettend breed en beschikbaar voor verschillende platformen. Het bijzondere aan Zxing is dat het als een normale applicatie kan worden geïnstalleerd waarna via een Intent de applicatie kan worden aangeroepen om een code te scannen. Het nadeel hiervan is echter dat de Parrent Activity wordt gepauzeerd en de robot dus niet kan bewegen. Dit kan gedeeltelijk worden opgelost door de inactivity time in InactivityTimer.java aan te passen. Mits er een barcode is gevonden, dan retourneert Zxing de controller gelijk aan de Parent applicatie. Als er binnen 3 seconden (handmatig ingesteld) geen barcode is gescand dan worden Zxing geforceerd gesloten zodat de Parrent applicatie alsnog de controle terugkrijgt en de robot weer kan bewegen.

2.2.4 Controller

Als laatst zochten wij naar een joystick waarmee de robot kan worden bewogen. De JoystickView [5] bibliotheek van Zerokol is hiervoor gebruikt, deze is naar onze eisen aangepast, namelijk de lay-out veranderd maar ook was het nodig dat wij weten welk van onze cases voor de robot moet worden uitgevoerd. Ook moest de controller applicatie een grote SCAN-knop bevatten, deze kan de user indrukken om naar een QR-code te zoeken, als deze knop is ingedrukt kunnen alle robots niet bewegen, dit hebben we gedaan omdat het anders erg lastig is om een QR-code te scannen.

2.2.5 Onderlinge communicatie

Voor de communicatie tussen de robots onderling is gebruik gemaakt van een aangeleverd framework. Voor het verzenden en ontvangen van commando's en berichten wordt gebruik gemaakt van Bluetooth. Voor de communicatie tussen de robots zijn wij opzoek gegaan naar een datastructuur zodat het spel makkelijk te schalen is. In het eerste ontwerp waren alle robots direct verbonden.



Figuur 4: Een schematische voorstelling van de datastructuur.

Iedere robot is hierbij een master van alle andere robots. Het nadeel van dit systeem is dat je aanloopt tegen een gelimiteerde aantal connectie (max 7 connecties per apparaat) mogelijkheden. Om dit probleem te verhelpen bedachten wij een datastructuur waarbij elke robot met een buur is verbonden zodat er een kring ontstaat (Daisy-chain) waarbij als iemand getikt wordt, er een bericht wordt verstuurd met de nieuwe tikker naar zijn buurman en de controller. De buurman verzendt dit bericht vervolgens weer naar zijn buurman waarna het proces zich herhaalt totdat het bericht weer bij de verzender is aangekomen waarna de verzending termineert.

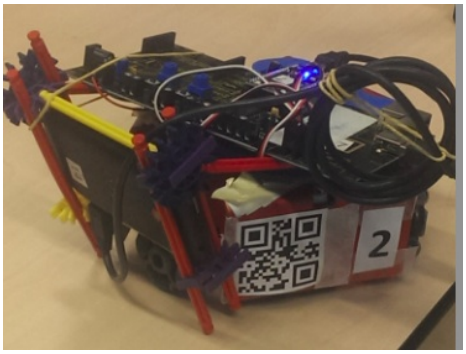
3 Resultaten

De laatste versie van de geassembleerde robot heeft op op vrijwel elk aspect gepresteerd zoals wij dit hadden verwacht bij het ontwerpen. De resultaten kunnen worden onderverdeeld in twee aparte onderdelen: Bluetooth communicatie en perceptie van de robot.

In het eerste domein heeft de robot gepresteerd zoals werd verwacht. Bluetooth blijkt uitstekend te functioneren wanneer men kleine boodschappen heen en weer stuurt waarbij er geen constante data stroom is. Door het beperken van het verkeer tussen de controller en de robot, werkt de Bluetooth binnen een tijds marge waarin de vertraging acceptabel is.

Deze beperking van het dataverkeer is gerealiseerd door alleen te communiceren naar de robot als de controller van case verandert. Zodoende wordt het berichten aantal geminimaliseerd, met als gevolg een minimale responsetijd.

De ontwerpkeuze voor de Daisy-Chain communicatie structuur werkte tevens naar verwachting. Deze datastructuur blijkt goed te werken wanneer het spel wordt geschaald met meerdere robots, waarbij een minimum geldt van twee robots en waar theoretisch geen maximum aanzit, alhoewel er wel vertraging optreedt wanneer je het spel met honderden robots zou spelen.



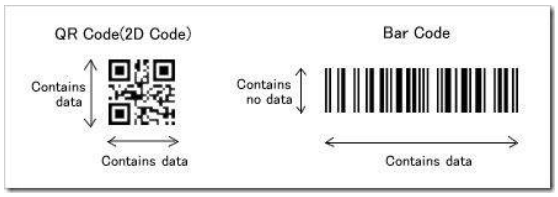
Figuur 5: De laatste versie van de robot met QR-codes op alle drie de zijdes en een Android telefoon op te voorkant gemonteerd die dient als scanner.

In het tweede domein bleek dat de identificatie van robots niet altijd te slagen. De gebruikte bibliotheek, Zxing, werkte optimaal wanneer de camera bij de eerste poging gefocust was op de barcode. Na deze “initiële” focus werden verschillende barcodes binnen een tiende van een seconden herkend. Deze “initiële” focus functioneert dus prima bij objecten die niet bewegen maar minder goed bij objecten die constant aan het verplaatsen zijn. Daarnaast heeft die bibliotheek beperkingen wanneer men een barcode wil scannen vanuit een hoek (> 20 graden).

De eerste limitatie is verholpen door alle robots in het spel te laten stilstaan wanneer er wordt getikt. Op deze wijze is er voldoende tijd voor de camera om een focus te krijgen op de robot. Verder is gebleken dat 25 centimeter de optimale afstand is voor het scannen van de QR-code.

Aanvankelijk werd er een barcode gebruikt waarbij er alleen informatie stond op de horizontale as (Code 128-B). Na enkele tests werd duidelijk dat er in een enkel geval een foute waarde werd gelezen. Alhoewel de frequentie van de foute waarden te verwaarlozen was, gingen wij toch opzoek naar een ander type code.

De keuze is uiteindelijke gevallen op QR-codes vanwege het feit dat deze beter leesbaar en onafhankelijk van oriëntatie (draaiing) zijn. Voor de constructie van de camera op de robot is gebruik gemaakt van K'nex. Ondanks de relatief instabiele constructie werkte het tikken redelijk goed.



Figuur 6: Het verschil tussen een barcode en een QR-code.[3]

Het idee voor een live video stream is bij deze test laten vallen. Het opzetten van een stream met een degelijke FPS (> 20) is hierbij wel gelukt, echter de ondersteuning voor het afspelen op een Android apparaat is niet gelukt. Hierbij is gebleken dat de video stream (Formaat: MJPEG) enkel goed werkt in een firefox browser.

Uit eindelijk is het gelukt om 2 robots samen tikkertje te laten spelen, echter door het gebrek aan meer robots zijn er geen tests uitgevoerd met meerdere robots. Wel zijn er losse telefoons gebruikt om een derde robot na te bootsen.

4 Conclusie

De basis elementen die zijn opgesteld in het ontwerp (robot besturing, communicatie onderling en identificatie) zijn allen bewerkstelligd. Zowel de robot besturing als de communicatie werkten uitstekend met de gekozen structuur. Ook de gekozen techniek, Bluetooth, bleek het best geschikt voor ons systeem met korte boodschappen over en weer.

De identificatie van robots onderling werkten in sommige situaties wel en in andere niet. Door een gebrek aan sensoren was de keuze gevallen op QR-codes die kunnen worden gescand met behulp van de telefoon op de robot. Dit werkt goed wanneer de QR-code enkele keer op een vaste afstand staat. Deze techniek is te vervangen door bijvoorbeeld RFID tags. Deze worden op de drie zijden van de robot geplaatst. Zo kunnen ze worden gescand wanneer twee robots in elkaars buurt komen.

Referenties

[1] Arduino. *Servo library.* URL: `http://arduino.cc/en/reference/servo`.

[2] Zie github pagina voor auteurs. *ZXing.* URL: `https://github.com/zxing/zxing`.

[3] Denso-Wave. URL: `http://pandazen.files.wordpress.com/2010/06/qrcode-vs-barcode.jpg`.

[4] Foxdogstudios. *Peepers: realtime video streaming from Android.* URL: `https://foxdogstudios.com/peepers.html`.

[5] Zerokol. *JoyStickView library.* URL: `https://github.com/zerokol/JoystickView`.