VPL services en sensors programmeren

Inleiding

In een eerder artikel in deze serie "Microsoft Robotics Development Studio Joystick Controlled NXT" werd beschreven hoe je robot bestuurd kon worden met een joystick en Bluetooth. Het ging daarin vooral om het aansluiten van een joystick of gamecontroller en een eenvoudig programmatje.
Dit artikel gaat over het grafisch programmeren met de Visual Programming Language van services en het gebruik van sensoren.

Voor het robotvoetbal willen we dat één programma parallel meerdere robots aanstuurt, ieder met hun eigen functie: een aanvaller krijgt andere strategische opdrachten dan de keeper maar beiden hebben dezelfde opdrachten nodig om door het veld te kunnen bewegen.
Omdat niet iedereen meerdere Mindstorm NXT's in huis heeft wordt hier beschreven hoe je één NXT meerdere functies parallel kunt laten uitvoeren.
Om te beginnen wordt een aanpak beschreven die algemeen toepasbaar is voor het schrijven van complexere programma's.

Software ontwikkel methode

Veel software ontwikkel methodes werken bij nieuwbouw volgens een vast aantal stappen:

1. Gebruikers eisen
2. Functioneel ontwerp
3. Technisch ontwerp
4. Ontwikkeling
5. Testen

In de nu volgende paragrafen worden dezelfde stappen nader uitgewerkt om het bouwen van services gestructureerd te kunnen doen.

Gebruikers eisen

Eerst wordt vastgelegd hoe de NXT als een soort ROV(Remote Operated Vehicle) en als lijnvolg autonome robot gebruikt wordt. Lijnvolgen wordt echter pas in het vervolg artikel nader uitgewerkt.
ROV's zoals die gebruikt werden bij Deep Water Horizon voeren opdrachten van gebruikers uit maar beschermen zichzelf ook tegen instructies van de gebruikers die de ROV kunnen beschadigen:

1. De robot kan met de joystick bestuurd worden
2. De robot kijkt met behulp van ultrasoon of deze tegen een object gaat botsen, weigert verder vooruit te gaan indien daarbij gevaar dreigt en staat de bestuurder alleen toe te draaien totdat er weer ruimte is om vooruit te gaan.

Vervolgens kan de robot omgeschakeld worden om autonoom een lijn volgen. Hierbij kan de bij de NXT horende test arena gebruikt worden:

1. Met de lichtsensor volgt de robot een lijn
2. Met de ultrasoon worden blikjes op de lijn gedetecteerd, die de robot automatisch ontwijkt waarna het lijnvolgen weer voortgaat
3. Door het drukken op een knop op de Joystick wordt de robot weer in ROV mode geschakeld.

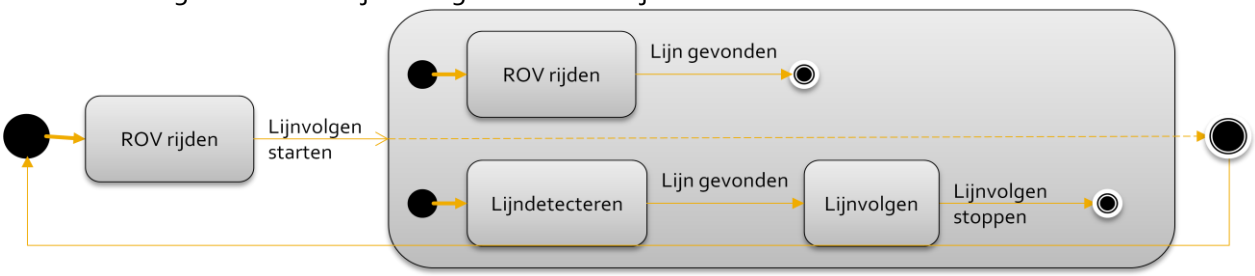
Zowel de ROV als Lijnvolg modus en alle tussenliggende statussen worden door de robot aan de gebruiker duidelijk gemaakt.

Functioneel ontwerp

Hier wordt beschreven waar de robot zoal mee bezig kan zijn. Dus in welke State bevindt de robot zich. Hoewel status nogal "onbeweegbaar" klinkt kan de robot ook een status "rijden" of "draaien" hebben. Services zorgen dat de robot in een bepaalde status terechtkomt maar hebben zelf ook allerlei eigen statussen, bijvoorbeeld een lijst met alle ingedrukte joystick buttons.
Door een gebeurtenis veroorzaakt door een service kan de robot van de ene naar de andere status overgaan: dit heet State Transition. Die gebeurtenis kan door de bestuurder veroorzaakt worden door de joystick service te gebruiken of door de ultrasoon of lichtsensor service.
De gebeurtenissen met de daaropvolgende statussen worden hieronder opgesomd. Dus steeds moet de vraag beantwoord worden: "De robot is nu aan het …"

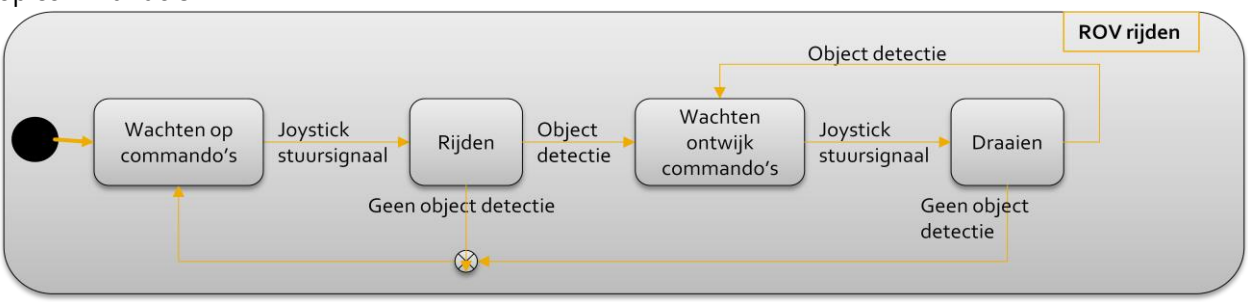
Het hoofdprogramma

In onderstaande paragrafen worden 2 superstatussen beschreven: "ROV rijden" en "Lijn volgen"
Het hoofd programma maakt van deze superstatussen gebruik.
Deze superstatussen bevatten allerlei deelstatussen.
In onderstaand State diagram is het hoofdprogramma te zien. Na het opstarten gaat de status "ROV rijden" beginnen wat betekent dat de gebruiker met de Joystick de Robot bestuurt.
Wanneer de gebruiker een button van de Joystick indrukt wordt het "Lijnvolgen starten" commando gegeven en komt het programma tegelijkertijd(dus parallel) in de status "ROV rijden" en "Lijndetecteren". De gebruiker beweegt de Robot dan nog met de Joystick totdat uiteindelijk de "Lijn gevonden" is waarna de status "ROV rijden" verlaten wordt (de Robot rijdt dus zonder Joystick) en tegelijkertijd komt de robot in status "Lijnvolgen" waardoor deze de gevonden lijn gaat volgen.
Wanneer de gebruiker dezelfde button van de Joystick wederom indrukt wordt het "Lijnvolgen stoppen" commando gegeven en wordt ook "Lijnvolgen" beeindigd, superstatus "Lijnzoeken" verlaten en begint het weer bij het begin met "ROV rijden".



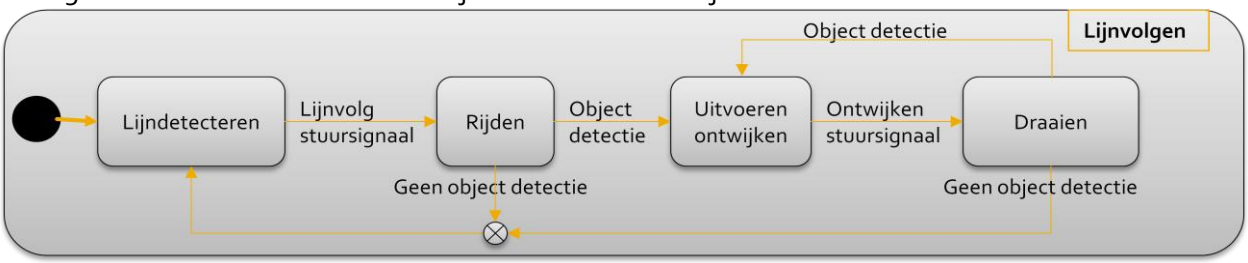
De superstatus "ROV rijden"

De superstatus "ROV rijden" behelst het besturen van de Robot door de gebruiker met een Joystick: de robot is aan het "Wachten op commando's" en voert deze uit door te "Rijden".
Wanneer de robot een object detecteert mag deze niet meer vooruit rijden en alleen nog maar: "Wachten op ontwijk commando's".
Door het aantal toegestane Joystick signalen te beperken, de robot is aan het "Wachten op ontwijk commando's"), kan de robot alleen nog maar draaien.
Zodra er geen objecten meer gedetecteerd worden gaat de robot weer terug naar status "Wachten op commando's".



Superstatus "Lijnvolgen"

De superstatus "Lijnvolgen" lijkt verdacht veel op superstatus "ROV rijden". In dit geval voert de Robot zelfstandig alle handelingen uit.
Zolang de status "Lijndetecteren" tenminste "Lijnvolg stuursignalen" tot gevolg heeft.
In dit geval zal de Robot ook zelf de objecten moeten ontwijken.



Technisch ontwerp

De statussen in het functioneel ontwerp geven aan waar het programma mee bezig is op een gegeven moment of welke service die aan het afhandelen is.
Een service is een verzameling bij elkaar horende software functies die hergebruikt kunnen worden voor verschillende doelen.
De state transitions zijn dan de uitkomsten van de services die ervoor zorgen dat nieuwe services aan het werk gaan.

Het programma bestaat dus uit deze services en reeds bestaande services, zoals die voor de joystick, motoren, ultrasoon detector en lichtsensor.
Hieronder worden de statussen van het hoofdprogramma als resultaat van een service beschreven en de services waarvan deze gebruik maken.
De services in bold zijn bestaande VPL services.

- ROV rijden(1)
    - Wachten op commando's(2)
        - **GameController** of **DesktopJoystick(3)**
    - Rijden(4)
        - **LegoNXTBrickv2(5)**
        - **LegoNXTDrivev2(6)**
    - Wachten op uitwijkcommando's(7)
        - Wachten op commando's(8)
    - Draaien(9)
        - Rijden
- Lijndetecteren(10)
    - **LegoNXTColorSensorv2** of **LegoNXTLightSensorv2(11)**
- Lijnvolgen
    - Lijndetecteren
    - Rijden
    - Uitvoeren ontwijken(12)
        - Rijden
    - Draaien
        - Rijden

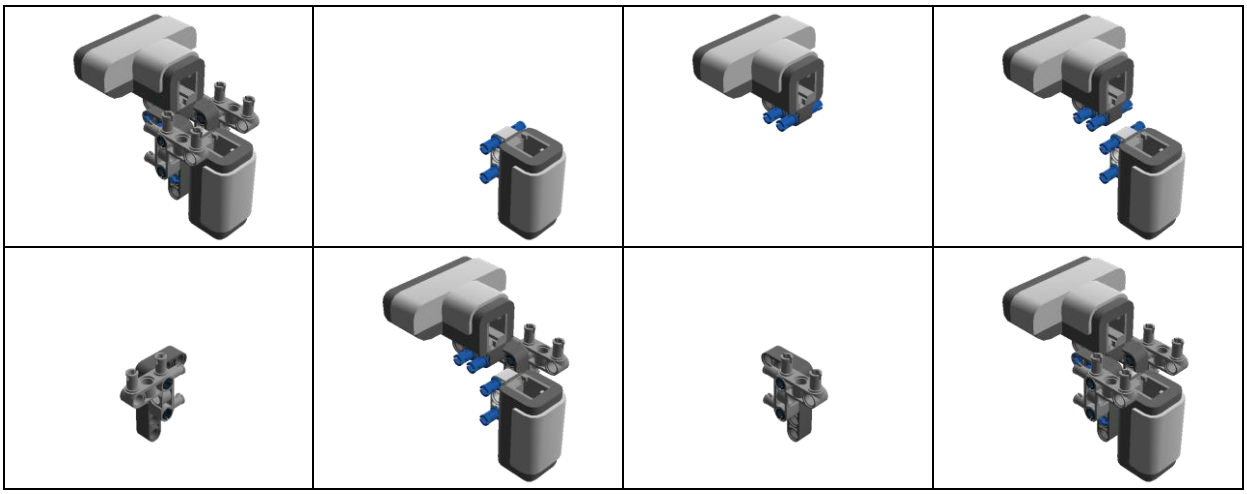
Uiteindelijk blijken er 12 services nodig te zijn waarvan er 4 al in VPL aanwezig zijn dus 8 zelf ontwikkeld moeten worden.

Ontwikkelen

Sensorarray

Eerst moeten de sensoren op de robot gemonteerd worden. Waarbij de ultrasoon rechtvooruit en de lichtsensor naar beneden gericht is.
Als je het ontwerp van de Voetbalrobot uit "3D ontwerp tools voor MRDS en Lego NXT" als basis gebruikt kun je het volgende ontwerp voor deze sensoren gebruiken:

1. Verwijder de balk van het Caster wiel
2. Verwijder aan de onderkant de balk die de twee motoren van de wielen samenhoudt
3. Maak de kabel van de middelste motor en van output poort A los
4. Verwijder de middelste motor die voor het schietmechanisme is bedoeld
5. Plaats de balk met Casterwiel weer
6. Verstevig aan de onderkant beide zijden van de motoren hetzelfde zodat de balk nu met 4 pluggen vastzit
7. Maak een vergelijkbaar balkje aan de voor/bovenkant van de motoren:
    a. Verwijder hiervoor eerst even de banden
    b. Vervang de zwarte plug door een lange blauwe en plaats de zwarte daarnaast
8. Maak onderstaande constructie
    a. Bevestig tijdig de 2 kabels
    b. Plaats de contructie onder aan de voorkant van de Brick.
    c. Verwerk de kabels op een handige manier zodat deze de wielen niet raken.
9. Test met het programmatje "View" op de NXT Brick of je de sensors volgens onderstaande lijst op de juiste standaard poort hebt aangesloten
    a. Druksensor op poort 1
    b. Geluidsensor op poort 2
    c. Licht- of Kleur sensor op poort 3
    d. Ultrasoon sensor op poort 4



De Joystick

In het document "Microsoft Robotics Development Studio Joystick Controlled NXT" paragraaf "Implementatie in VPL" staat een manier beschreven om via de JoyStick de LegoNXTDrive aan te sturen. Het is de simpelste manier om dat te doen maar hier is het de bedoeling om vooruit en achteruit te verbieden wanneer een object gedetecteerd wordt.
Omdat de Joystick Y-coordinaat in genoemde oplossing het gaspedaal representeert, zou door het negeren van dit signaal het vermogen naar de Drive altijd 0,0 zijn en de Drive zelfs niet om zijn eigen as kunnen draaien.

Daarom wordt hier een ingewikkelder methode gebruikt die door velen als intuitiever wordt beschouwd. Simpel gesteld beweegt de robot naar voren als de joystick naar voren wordt bewogen en draait deze om zijn as wanneer de joystick naar links of naar rechts bewogen wordt.
Stel dat de Joystick waarden(-1000 tot +1000) rechtstreeks, na delen door 1000, naar de motor gevoerd zouden worden. Dan ontstaat de volgende joystick tabel.

-1, 1	0, 1	1, 1
-1, 0	0,0	1, 0
-1, -1	0, -1	1, -1

De eerste waarde is steeds de X-coordinaat en de tweede de Y-coordinaat.
De lijn vanuit het midden in de richting waarin de Joystick staat noemen we de vector van de Joystick.
Dus in de rechter bovenhoek krijgen beide motoren een positieve waarde en rijdt de robot rechtvooruit. De vector heeft dan volgens de stelling van Pythagoras: √(1*1 + 1*1) = √ (2) = 1.4142136 lengte. Doordat de Joystick in een vierkant beweegt geldt die waarde voor alle hoeken maar links, rechts, voor en achter heeft de vector een lengte van 1.
De volgende tabel laat de sinus en cosinus waarden zien in deze sectoren op een cirkel met straal 1.

-0.7071068, 0.7071068	1, 0	0.7071068, 0.7071068
0, -1		0, 1
-0.7071068, -0.7071068	-1, 0	0.7071068, -0.7071068

Deze waarden komen aardig overeen met onze Joystick alleen is de vector lengte hier wel altijd 1.0.
Maar beide motoren moeten vooruit draaien als de Joystick midden voor staat. Dit kan gedaan worden door alle punten 45 graden (π/4 radialen) te roteren in positieve richting(tegen de klok in).
Met de volgende formule afkomstig van http://nl.wikipedia.org/wiki/Rotatie_(meetkunde) kan dat bewerkstelligd worden:

$$x' = x\cos\varphi - y\sin\varphi$$

$$y' = x\sin\varphi + y\cos\varphi$$

De joystick tabel komt er na rotatie dan als volgt uit te zien:

-1.4142136, 0	-0.7071068, 0.7071068	0, 1.4142136
-0.7071068, -0.7071068	0,0	0.7071068, 0.7071068
0, -1.4142136	0.7071068, -0.7071068	1.4142136, 0

Omdat de joystick waarden 1,1 (X,Y) in de hoeken heeft, is de lengte van de vector daar 1.4142136. Dat blijft zo na rotatie. Omdat dit boven de range van de LegoNXTDrive is (-1.0 tot 1.0) worden alle gevonden waarden door 1.4142136 gedeeld. Dit heeft wel tot gevolg dat de motoren meer vermogen krijgen bij draaien om 1 wiel, namelijk 1.0 dan dan bij rechtuit rijden, ieder 0.5. Deze aansturing gaat de Rijden service vormen.

Rijden activiteit en service

Start VPL. Er is een nieuwe versie "Microsoft Robotics Development Studio 4 Beta 2" beschikbaar welke je meteen even kunt installeren. Hoewel dit een Beta is, werken alle voorbeelden in dit artikel goed daarmee. Sla een nieuw programma op als "HCCRobot" in een nieuwe subfolder "HCCRobotica" onder de projectsfolder.
Let op: wanneer je een project opent in de nieuwe versie 4 Beta 2 is het daarna niet meer in de versie 2008 R3 te openen. Een mogelijke oplossing is dan handmatig diagrammen overcopieren van de nieuwe naar de oude versie.

Het bouwen van de Rijden Activity

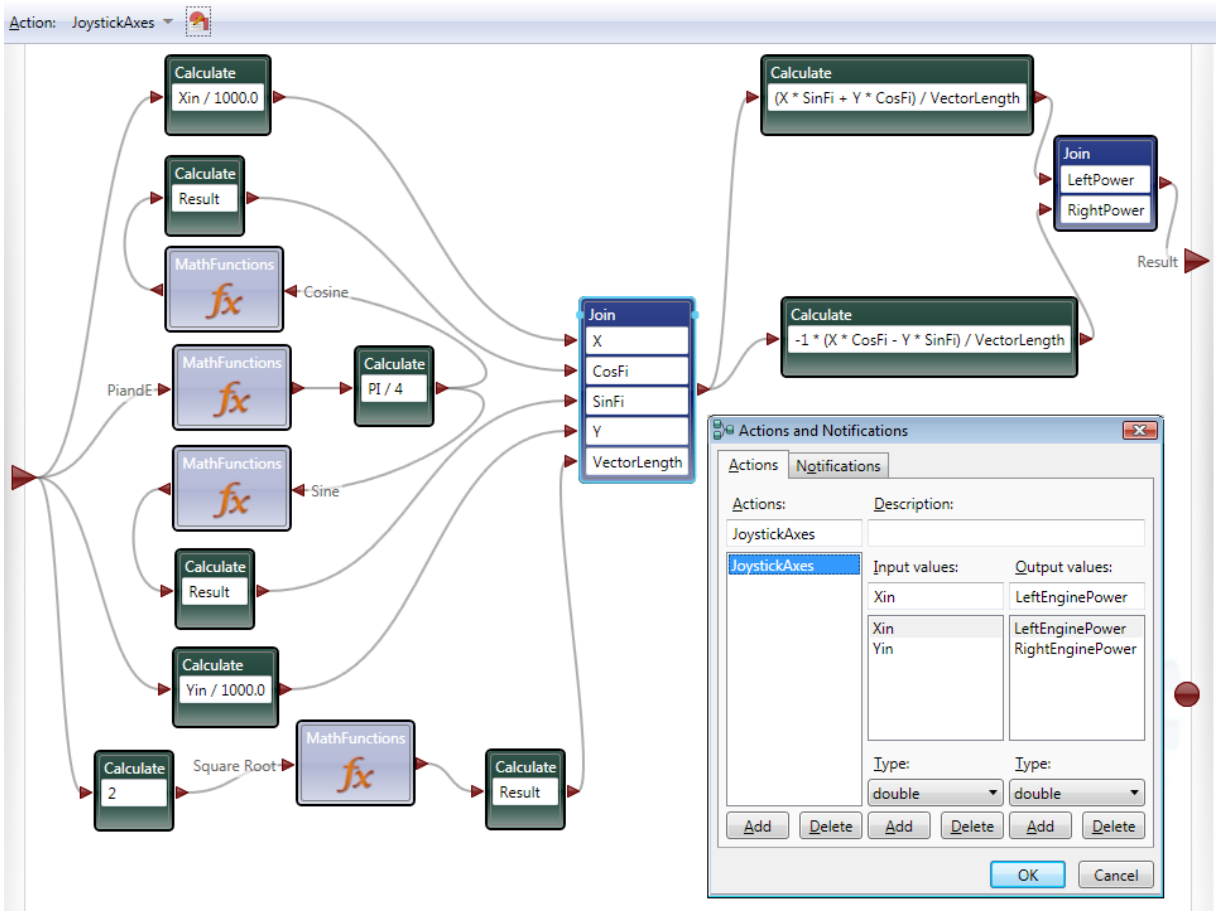
Activities vormen de basis waaruit services worden gegenereerd.

Maak een activity aan, geef deze "Rijden" als "Name" en "Friendly Name" en als "Comment": "Converteert X en Y coordinaten van een Joystick naar Power voor 2 wielen. De hoek van de Joystick vector wordt + 45 graden geroteerd zodat vooruit (sin=1, cos=0) verandert in sin=1/2 (2) en cos=1/2 wortel(2), etcetera."

Schrijf als Description "Converteert X en Y coordinaten van een Joystick naar Power voor 2 wielen.".
Let op: Dubbele quotejes in de Description van een Activity worden niet door de Service compiler gepikt. Voorbeeld: "Speelt op NXT geinstalleerde geluidjes ".rso" op naam" wordt vertaald naar in de c# code als: [Description("Speelt op NXT geinstalleerde geluidjes ".rso" op naam.")]wat het volgende in Visual C# oplevert
"Error 1 Expected class, delegate, enum, interface, or struct C:\Users\Iwan\Microsoft Robotics Dev Studio 4 Beta 2\HCCRobotica\Services\RijdenService.cs 25 17 HCCRobotDocTest".
Enkele quotejes '.rso' of \".rso\" mag wel.

Kies bij "Import Icon" een eigen gemaakte afbeelding waarmee deze Activity herkenbaar is.
Gebruik niet hetzelfde plaatje met dezelfde naam voor meerdere activities want bij het genereren van de services geeft dat problemen. Dit veroorzaakt voor alle services na de eerste met dit plaatje conflicten waardoor die services niet meer functioneren.

Dubbelklik de activity.
Klik onder de titel "Rijden" op de rode knop naast "Action". Dit kan ook via het menu "Edit\Actions and Notifications(Ctrl+Shift+A)". Hernoem "Action" naar "JoystickAxes" wat de naam is van het GameController Event. Voeg "Input Values" met de namen "Xin" en "Yin" toe, beide met "double" als "Type". Voeg "Output Values" met de namen "LeftEnginePower" en "RightEnginePower" toe, beide met "double" als "Type". Klik "Ok". Dit is als voorbeeld opgenomen in onderstaande figuur.
Bouw onderstaand schema na:



Let op dat vanuit alle Calculates steeds "value" gekozen moet worden als "Value" voor de "Data Connection".

De te kiezen functie van de "MathFunctions" staat steeds ernaast. Hergebruik steeds de eerst aangemaakte "MathFunctions". Direct na de "MathFunctions" staat steeds een calculate welke de voor de berekeningen benodigde waarde uit de resultaat message extraheerd. Kies steeds "Succes" bij de "From" connectie.
Let op dat je "-1*" in de onderste van de twee formules zet. In de paragraaf "Het genereren van de services" wordt uitgelegd waarom dit moet.
Klik op het lijntje van de Join naar de rechter grote driehoek en klik Ok. Klik op het ontstane lijntje en maak "Data Connections" tussen "LeftPower" met "LeftEnginePower" en "RightPower" met "RightEnginePower". Dit moet helaas in 2 stappen.

Bouwsuggesties

Om het bouwen van activities te vereenvoudigen volgen hier nog een paar suggestie:

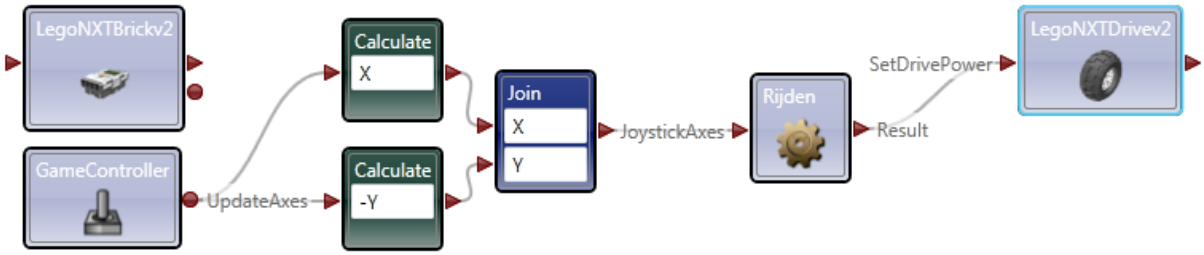
- Wanneer de ingaande waarde van een calculate een int is (zoals Xin hierboven) en de uitgaande waarde een double moet zijn deel dan door een double (zoals hierboven 1000.0).
- Door een uitgaand pijltje te slepen en ergens op het diagram te plaatsen krijg je een context menu waaruit je naast nieuwe Basic Activities ook bij het item "Activities" alle reeds in gebruik zijnde Activities en Services kunt kiezen.
- Past het diagram niet helemaal op je scherm dan is er rechtsonder een zoomfunctie waarbij het vergrootglas alles weer op 100% zet.
- Gebruik geen quotejes bij een "Data" activity van het type string
- Als een variabele naam begint met een "_" wat heel gangbaar is voor interne variabelen en je wilt controleren of de naam correct is of deze vervangen door een andere interne variabele, door in het dropdown lijstje te dubbelklikken of Enter op de naam te geven dan wordt deze vervangen met toevoeging van een extra "_".
- De errors in de volgende situaties worden veroorzaakt doordat een "Variable" rechtstreeks of via een "Join" wordt gebruikt . In dit soort situaties of eigenlijk altijd moet eerst een "Calculate" op de "Variable" volgen om de waarde echt uit te lezen:
    - Bij een "If" Activity: "The operation "!" is not valid for type "Draaien -> bool".
    - Meldingen als: "Error: The value of type 'OutsideRange -> bool' cannot be converted to the expected type 'bool'."
    - Bij het result van een Activity: "Error: The incoming message type, '(OutsideRange -> Unknown)', does not match the output type needed for this activity."
    - Ook de situatie waarbij de inkomende pijl een Set op een "Variable" doet en de uitgaande pijl direct een "Get" levert een foutief resultaat op. Een "Variable" kan maar voor een ding tegelijkertijd gebruikt worden.
- Soms is er al zoveel gewijzigd aan een activity dat er geen mogelijkheid is om een Error weg te krijgen. Een functie accepteert bijvoorbeeld geen variabele van het correcte type als Result variabele. Maak dan de functie opnieuw met een tijdelijke naam, copieer alle functionaliteit, voeg de variabelen weer toe, delete de oude functie en hernoem de nieuwe.
- In het algemeen moet goed opgelet worden dat er geen blokkades in het stroomdiagram ontstaan.

Testen

Het testen van de Rijden Activity

Ga naar het hoofddiagram en plaats een "Desktop Joystick" of "GameController", waarvan de Configuration "Instancename" gevonden kan worden met "Apparaten beheer" zoals "Port_#0001.Hub_#0005", een "LegoNXTBrickv2" waarvan de configuration "SerialPort" gevonden kan worden bij het bijbehorende Bluetooth apparaat en een "LegoNXTDrivev2" waarvan wel de "brick" en de "MotorPort" geconfigureerd moeten worden. Voor uitgebreidere uitleg zie "Microsoft Robotics Development Studio Joystick Controlled NXT".

Sluit het "GameController" notification bolletje aan op "Rijden". Kies "UpdateAxes" en paar zowel de "X"-en als de "Y"-en. Verbind het uitgaande pijltje van "Rijden" met "LegoNXTDrivev2" en "SetDrivePower" en paar de juiste "Power"-s.
Run het programma om te kijken of alles goed gegaan is. Als de robot in de verkeerde richting rijdt is het misschien noodzakelijk om met 2 calculates voor "X" en "Y" en een Join de Joystick waarden om te keren. Stop het programma.



Logging

Wanneer het programma niet helemaal werkt kan een "Log" service toegevoegd worden om waarden van variabelen te loggen. Type "Log" in het "Find Service…" vak. Om een duidelijke boodschap te krijgen is het handig om een "Calculate" in de vorm **"variabeleNaam " + variabele_waarde** voor de "Log" te zetten. Op die manier is het niet steeds nodig de velden van de Log opnieuw in te vullen bij het leggen van een verbinding ernaartoe maar volstaat het wijzigen van de Calculate. Het "Message" veld van de "Log" krijgt dan de waarde "value" en door "Edit values directly" kan in het "Category" veld een tekst (tussen "-s) geplaatst worden, hoewel deze "Category" nergens in de logging terug te vinden is.
Test Suggesties:

- Gebruik geen "User" en gewone services door elkaar (indien van toepassing)
- Wanneer een andere functie gebruikt moet worden van een reeds op het diagram aanwezige Activity of Service, dan moet om die functie te kunnen kiezen ook de eventuele uitgaande verbinding losgekoppeld worden anders blijft de oude keuze bewaard.
- Als je alle Webbrowsers afsluit dan start VPL er automatisch een op welke nadat het programma gerund heeft ook weer automatisch afgesloten wordt. Zodoende hoeven niet steeds de tabs van eerdere testruns gesloten te worden.
- Sluit VPL af voordat de computer in Sleepmode gezet wordt. VPL vertoont anders na herstarten mogelijk voor het programma desastreus gedrag.
- Soms wanneer een programma "raar" begint te doen kan na heropenen de melding "The file Diagrams.xml compressed could not be opened." This is fataal maar komt zelden voor. Toch is het verstandig regelmatig het programma met een nieuw versienummer op te slaan.

Het genereren van de service

Van het hoofd Diagram en alle activities daarin kunnen nu services gegenereerd worden.
Het is meestal pas zinvol van een MainDiagram een service te genereren als het een afgerond programma betreft dat voor bijvoorbeeld meerdere gelijkwaardige robots wordt gebruikt.
Ga naar het hoofd diagram en klik rechts onder de "Project" window op het foldertje "Diagrams". Vink in het "Properties" window "ExcludeMainDiagram" aan. Gebruik als "ContractPrefix" voor "http://schemas.hccrobotica.nl". Kies als "VisualStudioVersion" voor "Visual Studio 2010". Zonder deze instelling wil "Visual Studio" het gegenereerde project bij opening steeds converteren wat nogal vervelend is.
Kies het "Build\Compile as a Service" menu. Browse naar de "Microsoft Robotics Dev Studio 4 Beta 2" folder en maak daar een folder "HCCRobotica" aan. In deze folder komen alle gegenereerde services te staan. Klik "Ok".
De dialoog "Compile as a Service" verschijnt. Klik op "Details" waarin allerlei interessants voorbijkomt waaronder mogelijke errors.

Compile As Service Errors

De oplossing voor dit soort errors wordt beschreven in onderstaand scenario waarbij de Joystick conversie gedaan werd op de manier zoals beschreven in het document "Microsoft Robotics Development Studio Joystick Controlled NXT" paragraaf "Implementatie in VPL"
Er verschijnt de error: RijdenService.cs(164,40): error CS1059: The operand of an increment or decrement operator must be a variable, property or indexer
Even verderop staat "Failed" wat overduidelijk betekent dat het niet gelukt is.
Om te achterhalen wat er precies aan de hand is, dient het zojuist gegenereerde project "HCCRobot.csproj" in folder "Microsoft Robotics Dev Studio 2008 R3\HCCRobotica" in Visual Studio C# 2010 geopend te worden. Door daarna te dubbelklikken op "RijdenTypes.cs" en vervolgens op de error, blijkt waardoor deze error ontstaat:

```
a.RightEngine = --(DoubleCast(message.X) - 500) / 500 * message.Y / 1000;
```

De twee "-"en zijn een decrement en dat kan niet aan het begin van een formule maar alleen na een variabele zoals "mijnteller- -".
Dit lijkt bijzonder veel op de calculate in de "Rijden" activity te weten:

```
-(-(((double)X - 500) / 500) * Y) / 1000
```

Blijkbaar dacht de codegenerator dat een parenthesis meer of minder geen kwaad kon.Het toont in ieder geval aan dat gegenereerde code niet exact hetzelfde doet als wat je verwacht maar dat fouten daarin wel goed te duiden zijn.De oplossing is 2 minnen weg te laten in de formule want die blijken, na goed bestuderen van de formule, elkaar toch op te heffen.
*Dit is ook de reden om in het diagram die "-1 * " te plaatsen en niet gewoon een "-".*

Wanneer het programma weer gedebugged moet worden, moet wel eerst "ExcludeMainDiagram" weer uit. **ECHT WAAR!** Anders worden wel alle activities en gebruikte services opgestart maar wordt de onderlinge communicatie op het hoofddiagram genegeerd.

Gebruiken van de services

Door in het services paneel aan de linkerkant "Hcc" als filter te kiezen is te zien dat er nu een "HCCRobot Rijden" service is toegevoegd. Eventueel kan met menu "View\Reload Services" deze lijst geupdate worden.
Om bepaalde services eruit te filteren kan tegelijkertijd een uitzonderings filter gebruikt worden bijvoorbeeld "-(User) NXT". Met het plusje naast "All Found" wordt het filter als snelkeuze bewaard.
Door de Description van een Activity te beginnen met een nummer en dat op te hogen iedere keer dat de services gebuild worden, is het in het Services paneel eenvoudig te zien of de laatst gegenereerde service al beschikbaar is door de muis over de Service te bewegen.

Wanneer nu een nieuw programma gemaakt wordt kan deze service gekozen worden voor het vertalen van JoystickAxes naar Engine Power.

Omdat de service gebruikt maakt van "MathFunctions" moet in het hoofdprogramma wel een "MathFunctions" aanwezig zijn waarmee de "HCCRobot Rijden" service kan Partneren.
Stel in de Properties van "HCCRobotRijden" de "Configuration" in op "Set initial configuration" en kies onder "Partners" de "MathFunctions" van het hoofdprogramma.
Ook de eerder genoemde "Log" service (of een "Text" service) die mogelijk in een Service gebruikt wordt moet dus toegevoegd worden en in de properties van de service geconfigureerd.
Wanneer in de service bovendien een NXT service gebruikt wordt moet deze op het hoofddiagram ook volledig geconfigureerd worden.

Het programma zoals beschreven in "Het testen van de activity" kan nu dus met de "HCCRobot Rijden" service gebouwd worden.

Service suggesties

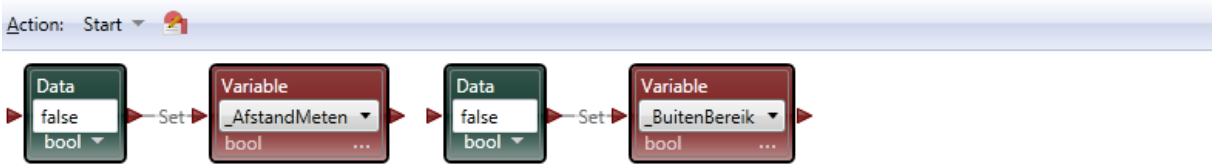
Hier nog een aantal suggesties ten aanzien van het gebruik van Services:

- De source codes worden wel in HCCRobotica folder opgeslagen maar de services staan in de "Microsoft Robotics Dev Studio 4 Beta 2\bin" folder. Door te filteren op datum kun je niet bruikbare services daar vinden en eventueel verwijderen.
- Wanneer services ontwikkeld en getest moeten worden is het gebruik van 2 VPL omgevingen tegelijkertijd het handigst. Een voor het ontwikkelen van de activities en compileren tot services en een voor uitproberen van de services.
- Let dan wel op dat niet zowel de activity applicatie en de service gebruikende applicatie tegelijkertijd runnen want dan volgt er een "Failed to connect " error. Mogelijk moet daarna de NXT opnieuw opgestart worden.

Ontwikkelen fase 2

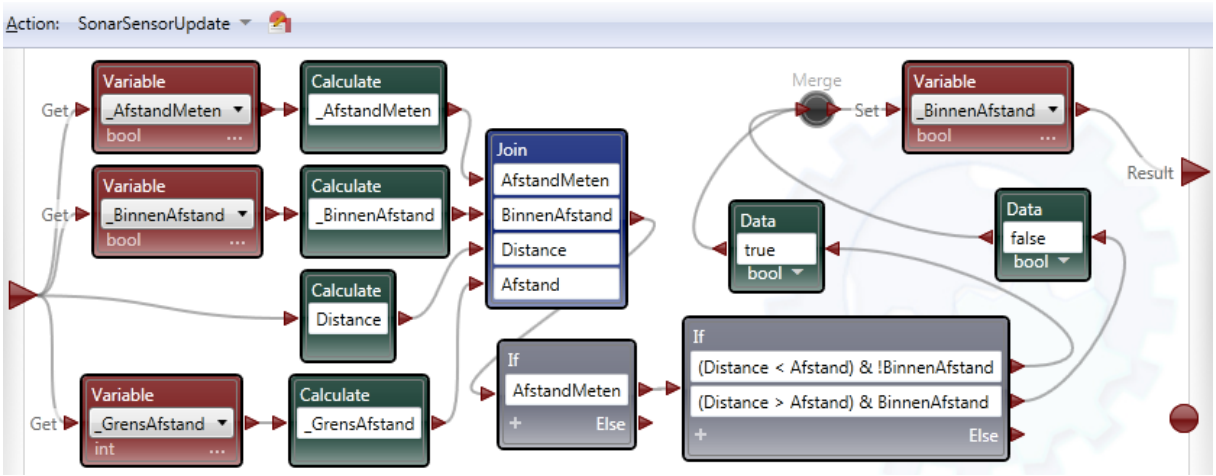
De Afstandmeter Activity

Maak een Activity met Name en Friendly Name "AfstandMeter" en Description "Bepaalt of de Distance van een sensor boven of onder een bepaald bereik valt.". Maak ook onderstaande functies:



"Start" heeft nooit Input of Output values omdat het alleen dient voor Activiteit initialisatie.

"StelAfstandIn" heeft 1 " Input values" genaamd "Afstand" van het type int.

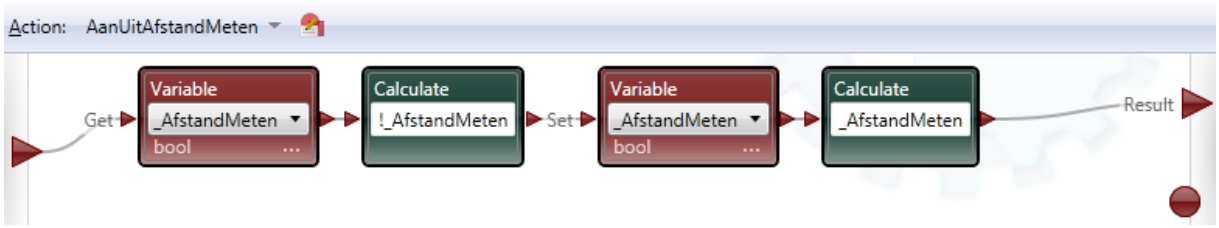


"SonarSensorUpdate" heeft 1 "Input values" genaamd "Distance" van het type int en 1 "Output values" genaamd "BinnenAfstand" van het type bool.
Deze functie checkt of AfstandMeten gedaan moet worden en vergelijkt dan de Distance afkomstig van de UltraSoon sensor BinnenAfstand. Is deze kleiner dan wordt BinnenAfstand true en omgekeerd.

Om niet te vaak BinnenAfstand te hoeven wijzigen wordt eerst gekeken naar de huidige waarde. Daar zijn 2 redenen voor:

1. Om het aantal messages en interne acties te beperken
2. De waarde van _AfstandMeten moet ook gelezen worden als gevolg van events door de ene service en bijna tegelijkertijd geschreven worden als gevolg van events van een andere.

Meer over dit paralelle gedrag bij de beschrijving van het hoofdprogramma.

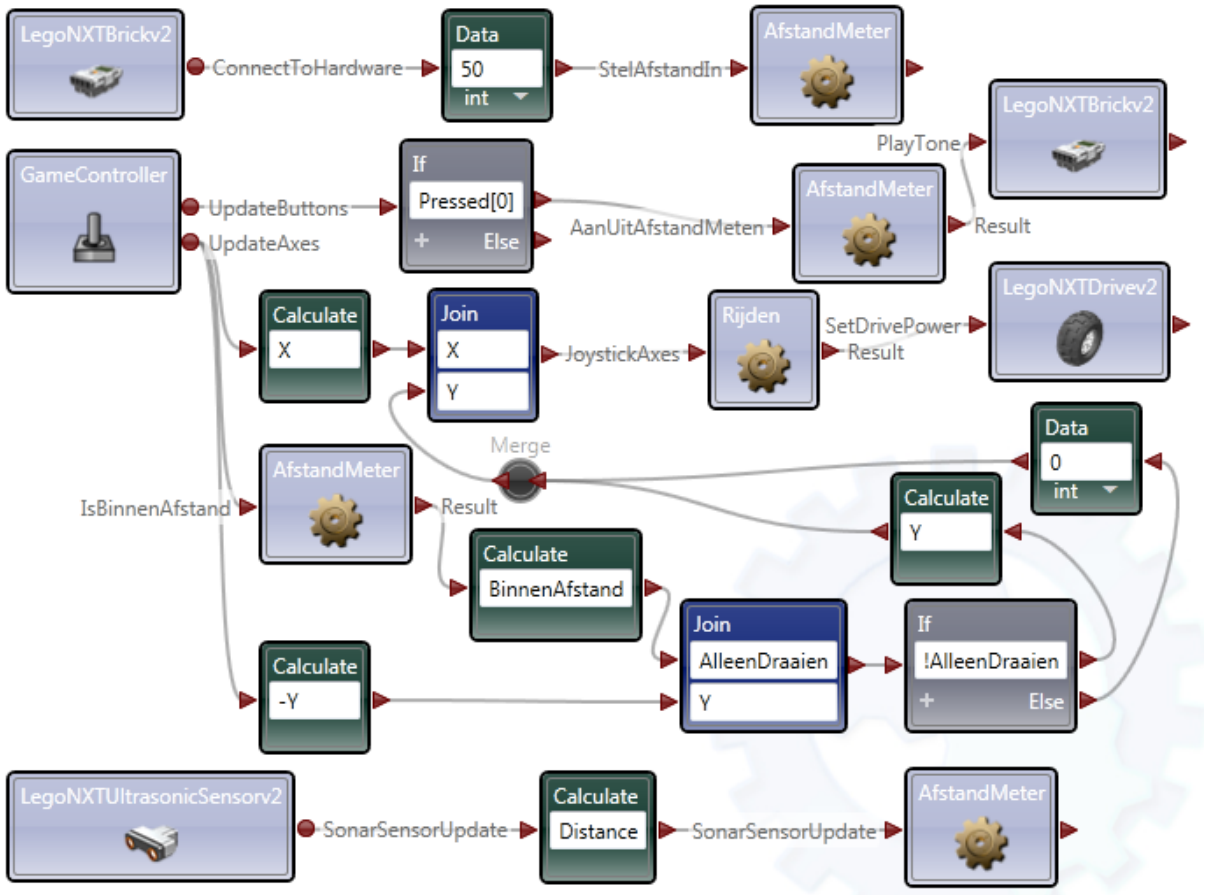


"AanUitAfstandmeten" heeft 1 "Output values" genaamd "AfstandMeten" van het type bool.

"IsBinnenAfstand" heeft 1 "Output values" genaamd "BinnenAfstand" van het type bool.

Het hoofdprogramma

Onderstaand is het hoofdprogramma afgebeeld.



De werking is als volgt:

1. Zodra de NXT met zijn Hardware connect wordt de GrensAfstand op 50 ingesteld
2. De UltrasonicSensor geeft voortdurend de Distance door aan AfstandMeter maar AfstandMeter handelt deze alleen af wanneer AfstandMeten op Aan staat.
3. De gebruiker kan nu met de GameController de AfstandMeter Aan of Uit schakelen (wat herkenbaar is door een piep) waarbij geldt:
    a. Uit: De X en Y coordinaten van de Joystick worden normaal doorgegeven
    b. Aan: De X coordinaat wordt normaal doorgegeven maar voor de Y coordinaat geldt:
        i. Als de robot meer dan GrensAfstand van een voorwerp vandaan is wordt de Y coordinaat gewoon doorgegeven
        ii. Als de robot minder dan GrensAfstand van een voorwerp vandaan is wordt de Y coordinaat genegeerd en kan de robot alleen nog rondjes draaien

Het bepalen of BinnenAfstand true of false is moet dusdanig opgelost worden dat het programma niet vastloopt.
In een eerdere versie gebeurde het volgende: iedere keer wanneer de UltrasoonSensor een verandering van afstand waarnam werd AfstandMeter.SonarSensorUpdate() aangeroepen en werd het resultaat naar een lokale variabele "BinnenAfstand" in het hoofdprogramma geschreven. Deze variabele werd dan weer door GameController events in het "!AlleenDraaien" if-statement uitgelezen.
Door de vele events van de UltrasonicSensor was de Variable steeds bezet door het geschrijf van AfstandMeter zodat de GameController events mogelijkheid kregen deze uit te lezen. Op een Variable is het namelijk niet mogelijk een wachtrij te zetten maar op een Object zoals de "AfstandMeter" Activity kan dat wel.
Door de variabele "_BinnenAfstand" intern te plaatsen worden aanroepen veroorzaakt door zowel de GameController als de UltrasonicSensor netjes op volgorde van binnenkomst afgehandeld.

Volgende keer

Volgende keer wordt er een lijnvolg functie toegevoegd welke ook om blikjes heen kan rijden. Zodoende worden de statediagrammen gecomplementeerd.
Verder wordt er een voorbeeld gegeven dat gebruik maakt van de HCCSoccerPlayer service.

Colofon

Kijk ook eens op het HCC Soccer Project http://hccsoccer.codeplex.com/ waar de server software gedownload kan worden en op http://www.twintellect.com waar dit en andere documenten te lezen zijn.

Technisch adviseur	Bert Berrevoets
Auteur	Iwan Tolboom

Email: iwan.tolboom@chello.nl