Realisatie van een
Virtueel Huisdier
in Soft - en Hardware

Tweede Deel Programmeerproject Eerste Bachelor 2007-2008

Introductie

Situering

Waar het eerste deel van het programmeerproject voornamelijk steunde op de vakken "Structuur I" en "Algoritmen en datastructuren I", leunt dit deel van het programmeerproject eerder aan bij de cursus "Interpretatie I" waarin recent het programmeren van **geïntegreerde systemen** aan bod gekomen is. Deze bestaan uit op maat ontwikkelde hardware én software die samen één bepaalde taak optimaal uitvoeren: van het doorlopen van magnetron- of wasprogramma's tot het controleren van loopbanden in fitnesscentra.

Praktijktoepassing

Dit deel van het programmeerproject beoogt het realiseren van een geïntegreerd systeem op maat van de opdracht. Een door middel van **Scheme** programmeerbare **microcontroller** enerzijds en een op de opdracht af te stemmen assortiment **sensoren** anderzijds vormen de hardware-componenten van dit systeem. Samen realiseren zij de brug tussen de ons omringende **fysieke wereld** en de abstracte wereld van het Scheme programma.

Een goede programmeerstijl, doordacht ontwerp en een verantwoorde keuze van datastructuren en algoritmen blijft ook voor dit deel van het project uitermate belangrijk. Meer nog, de inherent **beperkte reken -en opslagcapaciteit** van de microcontroller zet eventuele tekortkomingen alleen maar in de verf.

Individuele **creativiteit** bij het samenstellen van het systeem wordt aangemoedigd, waardoor deze opdracht een unieke kans biedt om de in het afgelopen jaar verworven vaardigheden op een ludieke manier in de praktijk toe te passen.

Ondersteuning

In de practica van het vak "Interpretatie I" wordt kort de werking van enkele **prototypische sensoren** verduidelijkt, wat voor elektrische **schakelingen** deze met de microcontroller verbinden en welke **Scheme-procedures** sensorwaarden uitlezen.

Alhoewel deze technieken met het nodige doorzettingsvermogen makkelijk te veralgemenen zijn tot andere sensoren van hetzelfde type, is dit soort hardware gevoelig en kan het soms ook eigenwijs zijn. **Aarzel daarom niet om bij langdurige problemen contact op te nemen met een van de assistenten.** Alhoewel creatief omspringen met de verstrekte hardware aangemoedigd wordt, blijft de nadruk van dit project wel degelijk bij het programmeren liggen.

Opdracht

Concreet bestaat de opdracht eruit een autonoom **virtueel huisdier** (zie inzet) te realiseren en dit te te **verankeren** in de **fysieke wereld** door het van de nodige sensoren te voorzien.

Terwijl een klassiek virtueel huisdier gaat slapen wanneer zijn eigenaar te kennen heeft gegeven dat het donker is door een knopje in te drukken, gaat zijn in de fysieke wereld verankerde tegenhanger bijvoorbeeld slapen wanneer zijn lichtsensor duisternis waarneemt en zijn bewegingssensor stilstand aangeeft.

Eind jaren '90 waren virtuele huisdieren, met de "Tamagotchi" als protaganist van deze rage, onlosmakelijk verbonden met de Belgische speelplaatstaferelen. Het gaat telkens om **autonome** toestelletjes met een **scherm** waarvan de huidige **gemoedstoestand** en **vitale kenmerken** van het virtuele huisdier af te lezen zijn. Deze variëren niet alleen met de leeftijd van het huisdier, maar zijn ook afhankelijk van **interacties** met de eigenaar. Hiertoe zijn de toestelletjes doorgaans voorzien van een drietal knopjes waarmee het dier onder andere gevoed, verzorgd en beloond kan worden.

Minimale functionaliteit

Actuatoren en sensoren

Het virtuele huisdier moet **minstens** over de volgende **sensoren** beschikken zodat deze zowel op invoer van zijn eigenaar als veranderingen in de fysieke wereld kan reageren: drie drukknopjes, een lichtsensor en een temperatuursensor. De invoer van deze sensoren beïnvloedt de gemoedstoestand en vitale kenmerken van het huisdier.

Het huisdier moet eveneens over de volgende **actuatoren** beschikken om zijn gemoedstoestand kenbaar te maken: drie LEDjes, een buzzer en een grafisch LCD-scherm.

Vitale kenmerken

Het gedrag van een virtueel huisdier is in grote mate afhankelijk van zijn **vitale kenmerken** die **gradueel** met de tijd kunnen **variëren** tussen twee extremen. Zo zal een rebels huisdier bijvoorbeeld vaker ongewenst verdrag vertonen dan een volgzaam huisdier. De eigenlijke waarde van een vitaal kenmerk wordt bepaald door interacties met de gebruiker. Zo wordt het virtuele huisdier bijvoorbeeld volgzamer door ongewenst gedrag te bestraffen en gelukkiger door er spelletjes mee te spelen.

Volgende vitale kenmerken moeten minstens gemodelleerd zijn:

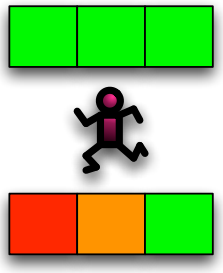
gemoedstoestand: tussen gelukkig en ongelukkig

volgzaamheid: tussen gehoorzaam en rebels

hongergevoel: tussen voldaan en hongerig

gezondheid: tussen gezond en ziek

vermoeidheid: tussen uitgeslapen en bekaf



De vitale kenmerken moeten in een oogopslag van het **LCD-scherm** af te lezen zijn. Dit kan bijvoorbeeld door een rij weer te geven waarvan elke cel overeenkomt met een kenmerk en de kleur van de cel met de huidige waarde van het desbetreffende kenmerk. Esthetischere oplossingen (naar de originele "Tamagotchi") zijn echter welkom op voorwaarde dat de kwaliteit van de code hier niet onder lijdt.

Interacties met de omgeving

Het virtuele huisdier moet minimaal volgende interacties met zijn omgeving en zijn eigenaar ondersteunen.

om aandacht vragen: wanneer een van de vitale kenmerken kritiek wordt, verwittigt het virtuele huisdier zijn eigenaar door een melodietje te spelen via de buzzer. Een rebels huisdier misbruikt deze buzzer echter ook om aandacht te vragen wanneer er niets mis is.

spelletje spelen: de eigenaar kan een eenvoudig spelletje met het virtuele huisdier spelen wanneer dit zich ongelukkig voelt. Het doel van het spel bestaat eruit te raden welk van de drie LEDjes het virtuele huisdier zal laten branden door het corresponderende knopje in te drukken. Bij een correcte gok verbetert de gemoedstoestand van het huisdier.

verzorgd worden: op regelmatige basis introduceert het virtuele huisdier in zijn omgeving uitwerpselen die de eigenaar moet opruimen. Gebeurt dit niet, verslechtert de gezondheid van het huisdier door een gebrek aan hygiëne. Eens ziek, kan het huisdier verzorgd worden door medicijnen toe te dienen en het warm in te duffelen. Dit laatste doet de eigenaar door de warmtesensor te verwarmen.

gevoederd worden: wanneer het dier hongerig is, kan het gevoederd worden. Na verloop van tijd wordt het dier echter opnieuw hongerig. Langdurig hongerlijden leidt onvermijdelijk tot de dood. Overvoedering leidt, net als een gebrek aan hygiëne, tot ziekte.

slapen: na verloop van tijd slaat de vermoeidheid toe en moet het dier slapen. Dit kan het dier echter alleen in een donkere omgeving die de eigenaar moet creëren door de lichtsensor af te dekken. Langdurige oververmoeidheid leidt tot de dood.

rebelleren: een virtueel huisdier kan op drie manieren rebelleren. Het dier kan voedsel weigeren wanneer het hongerig is, het kan spelletjes weigeren te spelen wanneer het ongelukkig is of het kan om aandacht vragen wanneer er geen enkel vitaal kenmerk kritiek is. Deze willekeurige vormen van ongehoorzaamheid treden vaker op wanneer het dier rebels is. Door ongepast gedrag te bestraffen wordt het dier volgzamer. Door ongepast gedrag ongemoeid te laten, wordt het dier rebelser.

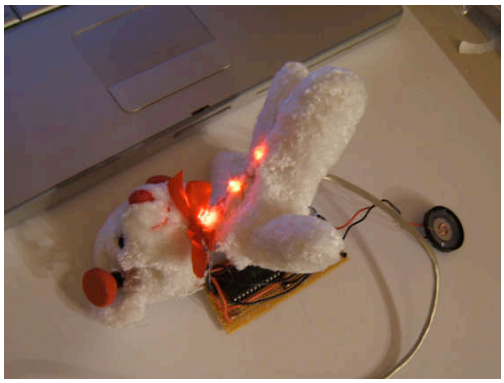
Een geïntegreerd, autonoom systeem

Het virtueel huisdier moet **autonoom** werken en dus gebruiksklaar zijn van zodra het van stroom voorzien wordt. Er mag **geen interactie via de read-eval-print loop** nodig zijn om het programma te starten.

Suggesties voor extra functionaliteit

Hierboven werd de minimale functionaliteit opgesomd die je implementatie moet voorzien. Wat volgt zijn **suggesties** voor extra functionaliteit waarmee je je implementatie kan uitbreiden eens deze aan de gebruikelijke kwaliteitseisen voldoet.

Zoals voor het eerste deel van het project geldt ook nu nog steeds dat wie meer hooi op zijn vork neemt, daarvoor beloond wordt. Bovendien hebben we er bewust voor gekozen jullie van meer sensoren te voorzien dan strikt nodig is voor dit project. We hopen dan ook op enkele originele en zelfs ludieke creaties.



Naast het toevoegen van een **extra sensor** of actuator vormt het **periodiek opslaan** van de vitale kenmerken van het huisdier zodat deze niet verloren gaan reeds een interessante uitbreiding op de minimale functionaliteit. Het huisdier met een (op het internet aangesloten) **computer** laten **communiceren** om hiervan data voor vitale kenmerken te bekomen een andere. Maar net zo goed kijken we al uit naar virtuele huisdieren die als teddybeer vermomd zijn.

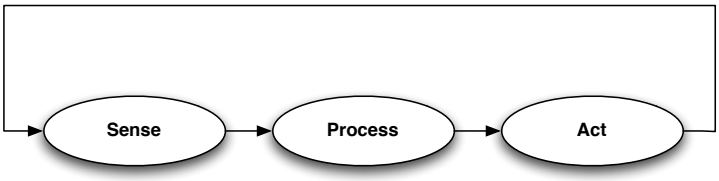
Implementatierichtlijnen

Simulatie voor alles

We raden je aan de software module per module te ontwikkelen in je vertrouwde **desktop Scheme-omgeving** (zoals DrScheme). Eventuele sensorinvoer zal hierbij gesimuleerd moeten worden. Wanneer alles naar behoren werkt, kan je de module via de Scheme **read-eval-print** loop van de microcontroller in de fysieke wereld testen.

Een eenvoudige architectuur

De software van veel eenvoudige geïntegreerde systemen is opgebouwd rond een zogenaamde **sense-process-act** loop. Deze leest achtereenvolgens sensorgegevens uit, verwerkt de uitgelezen sensorwaarden en voert ten slotte een gepaste actie uit.



De keuze van de actie is vaak echter niet alleen afhankelijk van de huidige sensorwaarden, maar ook van acties die ondernomen zijn in het verleden (een vorige iteratie van de lus).

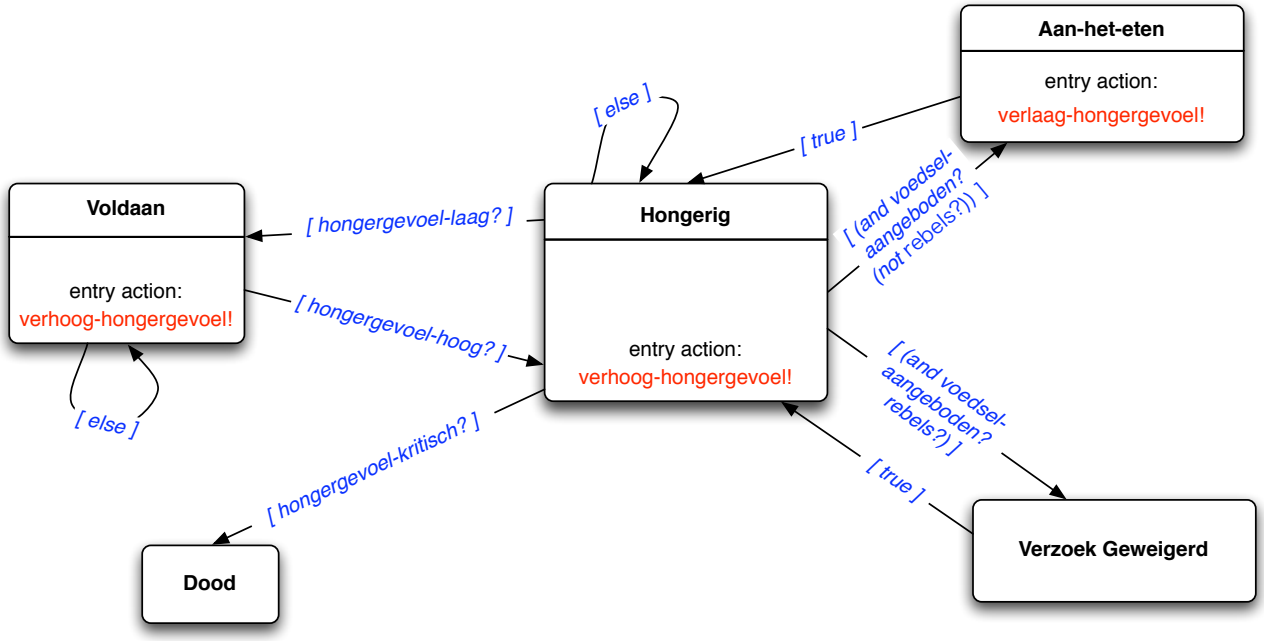
> Zo is de functie van een knopje op een horloge bijvoorbeeld afhankelijk van de **toestand** waarin het horloge verkeert. Eenzelfde knopje kan zowel gebruikt worden voor het instellen van de uren in de tijd-instel-toestand als het starten van een chronometer in de chronometer-toestand. Het horloge kan van de ene naar de andere toestand overgaan mits het correcte knopje ingedrukt wordt.

Om dit soort complexiteit de baas te kunnen, wordt het *process*-gedeelte van de sense-process-act loop vaak geïmplementeerd als een **eindige toestandsmachine** (*finite state machine*).

> De achterliggende idee is dat het systeem telkens in één toestand verkeert en slechts naar een nieuwe toestand overgaat wanneer aan een bepaalde **transitieconditie** voldaan is. Deze condities kunnen over de ingelezen sensorwaarden gaan, maar bijvoorbeeld ook over de toestand waarin een andere toestandsmachine op hetzelfde moment verkeert. Aan het **betreden en/of verlaten van een toestand** kunnen tot slot acties verbonden worden die door het *act*-gedeelte van de loop uitgevoerd moeten worden.

Een concrete toestandsmachine als voorbeeld

Beschouw volgende grafische voorstelling van een toestandsmachine die een deel van de interacties van het virtueel huisdier implementeert. Toestanden worden voorgesteld als rechthoeken met daarin in het vet een naam en de met het betreden van de toestand verbonden actie in het rood. Transities worden voorgesteld als pijlen met een eventuele transitieconditie in het blauw.



> In de eerste iteratie door de sense–process-act loop start het virtuele huisdier in de toestand **Voldaan**. Vanuit deze toestand zijn er twee transities naar een volgende toestand mogelijk. Het huisdier kan naar de toestand **Hongerig** overgaan wanneer aan de transitieconditie *hongergevoel-hoog?* voldaan is. Anders neemt het huisdier de transitie met label *else* waardoor het opnieuw in de toestand **Voldaan** terecht komt. Het voert dan de actie verhoog-hongergevoel! uit omdat dit de met het betreden van de toestand verbonden actie is. In de tweede iteratie van de sense-process-act loop zal dus ofwel de toestand **Hongerig** ofwel de toestand **Voldaan** de ingelezen sensorwaarden verwerken.

Het huisdier kan slechts in een toestand tegelijkertijd verkeren. Bij elke iteratie moet de toestandsmachine daarom ofwel in dezelfde toestand blijven (wanneer aan geen enkele transitieconditie voldaan is), ofwel naar slechts 1 andere toestand overgaan. Er zijn twee manieren om dit te verzekeren. Ofwel stel je als ontwerper van de toestandsmachine alle transitieconditions zo op dat ze mutueel exclusief zijn, ofwel maakt je toestandsmachine een willekeurige keuze wanneer verschillende overgangen tegelijkertijd mogelijk zijn.

Het virtuele huisdier als een verzameling van toestandsmachines

Bovenstaande toestandsmachine realiseert reeds een deel van de interacties die afhankelijk zijn van het vitale kenmerk “hongergevoel”. Ook voor de overige vitale

kenmerken kan een gelijkaardige toestandsmachine opgesteld worden. Aangezien het virtuele huisdier daardoor te herleiden valt tot een verzameling toestandsmachines, smeekt dit project dan ook om een goed ontworpen, algemeen inzetbaar ADT voor toestandsmachines.

> Merk op dat tussen deze machines toestanden en transities (deelmachines) gedeeld kunnen worden. Denk maar aan het al dan niet bestraffen van rebels gedrag dat zich bij verschillende vitale kenmerken manifesteert. Bovendien bestaan er veel gelijkenissen tussen deelmachines. Zo gebeurt veel van de interacties met de gebruiker via een met behulp van knopjes navigeerbare menustructuur. Denk maar aan bepaalde spelletjes, het voederen of het straffen van het dier. Het spreekt voor zich dat dit niet tot code-duplicatie mag leiden.

Het kan handig zijn om niet elke toestandsmachine te betrekken bij elke iteratie van de sense-process-act loop. Het zou bijvoorbeeld ongepast zijn dat een machine het LCD-scherm overneemt wanneer de eigenaar een spelletje aan het spelen is. Hiervoor kan je zelf een mechanisme voorzien waarmee machines andere machines in of uit de loop kunnen brengen.

Implementatiefasen

Aangezien dit project parallel loopt aan het vak “Interpretatie I”, raden we je aan het in fasen te implementeren.

> De eerste en belangrijkste fase omvat de implementatie van het ADT toestandsmachine in je vertrouwde desktop Scheme-omgeving. Voeg vervolgens een sense-process-act loop toe waarbij je het uitlezen van sensors en het aansturen van actuatoren simuleert. Implementeer daarna, gebruik makende van je ADT toestandsmachine, de interacties rond een eenvoudig vitaal kenmerk zoals “slaperigheid”. Op voorwaarde dat je geen implementatie-specifieke Scheme functies gebruikt hebt, kan je deze code zonder aanpassingen gebruiken op de microcontroller. Parallel met de lessen “Interpretatie I” kan je de simulaties geleidelijk vervangen door fysieke implementaties.

Op deze manier hoef je niet te wachten tot een bepaalde sensor in het vak aan bod gekomen is.