Joystickstyrning av robothunden AIBO

Daniel Karlsson

Hösten 2003

Examensarbete för 10 p, Institutionen för datavetenskap,
Naturvetenskapliga fakulteten, Lunds Universitet

Thesis for diploma in computer science, 10 credit points,
Department of Computer Science, Faculty of Science, Lund University

Joystickstyrning av robothunden AIBO

Sammanfattning

Till grund för denna rapport ligger en undersökning där en joystick används för att styra robothunden AIBO:s rörelser. Hunden har fyra ben, tio lysdioder och två mikrofoner vilket ger den stora möjligheter att uttrycka sig. Då en relativt avancerad joystick (Microsoft Sidewinder Force Feedback 2) används, blir möjligheten att styra hunden med ett rikt rörelsemönster mycket god.

I rapporten redovisas dels detaljer på hur systemet är sammankopplat och dels hur rörelser på joysticken överförs till rörelser på hunden.

En interaktiv möjlighet att styra hunden har skapats och en mängd rörelser och ljud kan genereras från hunden med programvaran som redovisas i denna rapport. Då ett stort antal rörelser har kopplats in i systemet finns möjligheten att relativt enkelt bygga om programmet enligt en egen idé till joystickstyrning av hunden.

Joystick control of robotic dog AIBO

Abstract

This report investigates the possibility of controling the robotic dog AIBO´s movement using a joystick. The dog has four legs, ten diodes and two microphones which contribute to many possibilties for it to express itself. With a quite advanced joystick the possibility to control the dog with a rich movement pattern will be good.

The report presents details of how the system is put together and how movements from the joystick have been transferred to movements of the dog.

An interactive possiblity to control the dog has been created and a significant amount of movements and sounds can be generated with the software presented in this report. Because a large amount of movements is avaliable in the system, the possibility of creating a solution of one´s own to control the dog can be done quite easily.

Innehållsförteckning

1 Inledning

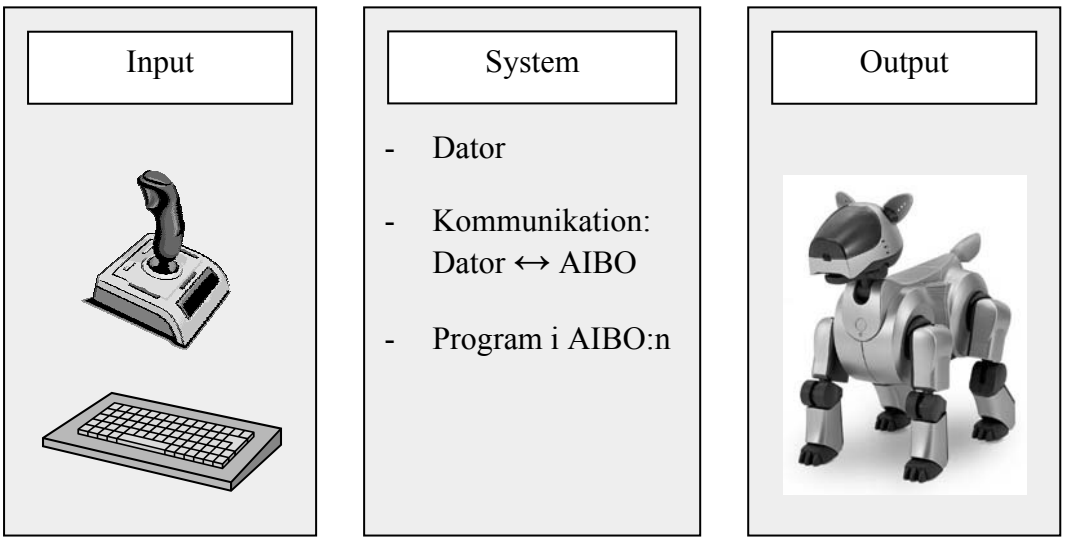
1.1 Bakgrund

AIBO-hundarna på Lunds Universitet används inom forskning samt som redskap vid utbildning inom Artificiell Intelligens (AI). En AIBO [2] är en robot som ser ut som en hund. Inom utbildning används AIBO:n av studenterna för att laborera med, medan den inom forskningen används för att testa AI-intressanta områden. RoboCup [3] är ett internationellt forsknings- och utbildningsinitiativ där AI och robotutveckling sätts i fokus. I RoboCup används AIBO-hundar för att tävla mot varandra i ett fotbollsliknande bollspel.

Det är viktigt att kunna visa upp områden som kan främja intresse och engagemang för studenter. En AIBO skapar intresse i sig, men det är när man ser dess möjligheter som intresset kommer upp på en helt annan nivå. Därför är ett interaktivt alternativ till att visa upp möjligheterna med en AIBO viktigt. De verktyg som används för att styra AIBO:n är ett tangentbord och en joystick. Joysticken är av modellen ”Microsoft Sidewinder Force Feedback 2” [4] och den har tre axlar och åtta knappar. AIBO:n går att styra med både en joystick och ett tangentbord. Det primära i det här exjobbet är att styra AIBO:n med en joystick, medan ett tangentbord är tänkt att användas främst vid utveckling av programvaran i AIBO:n, samt som ersättning till joysticken.

1.2 Input till output

För att styra AIBO:n med en joystick krävs en dator att koppla joysticken till, samt en kommunikationsväg från denna dator till AIBO:n. I AIBO:n behövs sedan ett program för att den ska kunna utföra önskade rörelser. Inputs till systemet blir en joystick och/eller ett tangentbord. Outputs från systemet blir rörelser och ljud ifrån AIBO:n.



Figur 1.2.1 Inputs, outputs och system.

1.3 Problemformulering

De problem som ska lösas för att få AIBO:n att röra sig efter en joystick är:

1. Läsa av Joystickens rörelser.

Drivrutiner tillgängliga från leverantören till joysticken finns endast till Windows. Då operativsystemet Windows inte ska användas så behövs drivrutiner till joysticken för aktuellt operativsystem tas fram. När detta är gjort ska sedan rörelserna och knapptryckningarna på joysticken kunna läsas in.

2. Upprätta kommunikation mellan AIBO:n och datorn.

AIBO:n har ett s.k. AIBO LAN Card och kan då kommunicera via ett trådlöst nätverk. En vanlig TCP/IP-kommunikation kan vara lämplig.

3. Ett program på AIBO:n som tar emot styrkommandon och utför dessa.

AIBO:n ska röra sig utifrån styrkommandon som skickats från joysticken. Vilka rörelser som AIBO:n ska kunna utföra har inte fastslagits definitivt utan lämnats relativt fritt. I AIBO:n finns ett operativsystem med en speciell struktur och denna struktur måste betraktas för att sedan kunna bygga upp programmet.

2 Information om AIBO

En AIBO är en robot som ser ut som en hund. Det finns ett flertal olika modeller och den som använts i detta examensarbete är ERS-210. Figur 2.1 visar några av de AIBO-modeller som finns.

Figur 2.1 De olika AIBO-modellerna f.v. ERS-110, ERS-210, ERS-220 och ERS-311.

Denna del av rapporten kommer att ta upp specifika detaljer om hunden (AIBO:n) såsom hur den rör sig, uppfattar sin omgivning och hur den är uppbyggd både mjukvaru- och hårdvarumässigt.

Hunden har ett antal effektorer och sensorer. En effektor är något som ger uttryck utåt såsom en rörelse eller ett ljud från hunden. En sensor är något som hunden tar in information med t.ex. en kamera. Hundens effektorer och sensorer kommer att beskrivas mer i detalj i kapitel 2.1.1 och 2.1.2.

Hundens mjukvara har en lite speciell struktur. Den använder Aperios som är ett operativsystem skapat specifikt för inbyggda system. Ett inbyggt system kan i stort beskrivas som något som styrs av en inbyggd processor. Det kan t.ex. vara en tv, en mikrovågsugn eller en AIBO. Till operativsystemet Aperios används ett programmeringsgränssnitt med namnet OPEN-R [1]. Mer information om mjukvaran i hunden finns i kapitel 2.2.

2.1 Hårdvaran

2.1.1 Effektorer

En detaljerad beskrivning av AIBO ERS-210 finns i bilaga B. Nedan följer en mer övergripande beskrivning av hunden.

- Hunden har måtten: bredd: 152 mm, höjd: 250 mm, längd: 281 mm, och väger 1.5 kg.
- Den har fyra ben där varje ben kan böjas i knäna (framåt/bakåt) och i höften (utåt/framåt/bakåt).
- Huvudet är rörligt i tre led (vänster-höger / upp-ner / lutning vänster-höger).
- Svansen är rörlig i två led. Munnen och öronen i en led.
- Totalt finns tio lysdioder i färgerna röd, grön, blå och orange. Då vissa av dessa är placerade som ögon kan de användas för att utrycka känslor hos hunden.
- Högtalare för ljud såsom hundskall eller musik.

Hunden kommunicerar via ett trådlöst nätverk med ett s.k. AIBO LAN Card. Hunden drivs dessutom av ett batteri vilket medför att inga sladdar behövs.

2.1.2 Sensorer

Hundens möjligheter att uppfatta världen möjliggörs/begränsas av dess sensorer. Människan har fem olika sinnen: syn, hörsel, känsel, smak och lukt. Hundens sensorer är något liknande. Den har: en kamera (syn), mikrofoner (hörsel) och trycksensorer (känsel) men dock ingen möjlighet att ta in smak och lukt.

Människan kan bedöma avstånd med hjälp av sina två ögon och bestämma ljudriktningar med sina två öron. Hunden har en infraröd sensor för att bestämma avstånd och två mikrofoner för att kunna bestämma ljudriktningar. Hunden har även en accelerationssensor så att den t.ex. kan känna av ifall den trillat omkull eller tappat balansen.

2.2 Mjukvaran

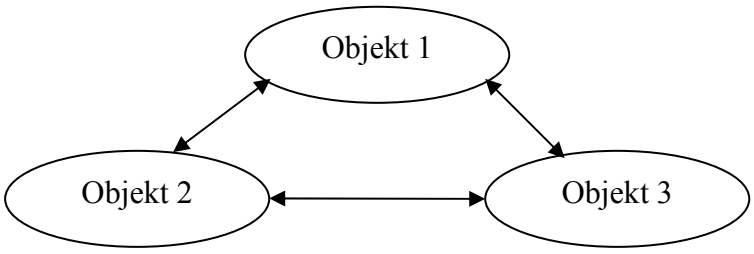
2.2.1 Operativsystemet Aperios

Aperios är ett operativsystem skapat specifikt för inbyggda system såsom AIBO. Den har en relativt liten storlek (100 kb), vilket gör operativsystemet lämplig då resurserna (minnesutrymme och även fysiskt utrymme) är mer begränsade i inbyggda system.

Aperios är ett objektorienterat, distribuerat realtidsoperativsystem. Operativsystemet är uppbyggt såsom ett nätverk med objekt där operativsystemet känner av, kontinuerligt uppdaterar sig själv och lägger till/tar bort objekt när det behövs.

2.2.2 Programmeringsgränssnittet OPEN-R

OPEN-R är ett programmeringsgränssnitt som är optimerat för effektiv utveckling av robotars mjukvara och hårdvara. En OPEN-R applikation består av ett antal objekt. Ett objekt här är inte ett objekt såsom i ett objektorienterat språk, utan det är mer som en egen process. I OPEN-R kommunicerar dessa objekt med varandra med hjälp av gemensamma minnesutrymmen. Denna typ av kommunikation i OPEN-R kallas för interobjektkommunikation. Specifikt för ett OPEN-R-objekt är att den har en egen tråd som exekveras jämsides med en massa andra objekt. Ytterligare information om OPEN-R finns i dess medföljande dokumentation [7] och handledning [8].



Figur 2.2.2.1 Objekt i OPEN-R.

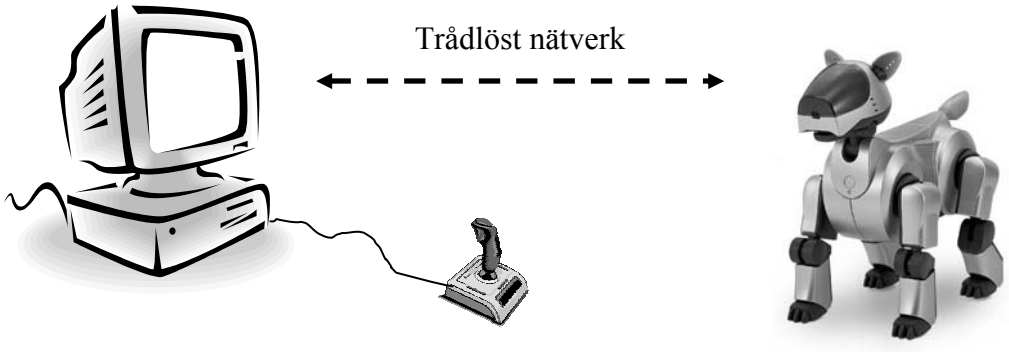
En typisk livscykel för ett OPEN-R-objekt är:

1. Laddas av systemet.
2. Väntar på ett meddelande.
3. Utför åtgärder utifrån mottaget meddelande.
4. Gå tillbaka till steg 2.

Specifikt för ett OPEN-R-objekt är också att:

- Det kan inte avsluta sig självt utan finns kvar så länge systemet finns kvar.
- Det kan, olikt från andra programmeringsmiljöer, startas från flera olika håll (det har ingen main-funktion).
- Det brukar definieras för en specifik uppgift. Ett objekt kan t.ex. sköta robotens ben, medan ett annat blinkar med robotens ögon.

3 Systemet



3.1 Översikt

För att hunden ska kunna styras med en joystick behövs en dator eftersom joysticken inte direkt kan kopplas till hunden. Det blir även smidigare på detta sätt eftersom inga sladdar då behöver vara kopplade till hunden. Om man kopplar joysticken till en dator så behövs en kommunikationsväg mellan denna dator och hunden för att få dit joystickens kommandon. Detta sker med ett trådlöst nätverk som kommunicerar mellan datorn och hunden i båda riktningar. Kommunikationsvägen tillbaka till hunden används för att skicka en mottagandebekräftelse, men den behövs även då en framtida vidareutveckling av systemet troligtvis kräver det.

En första del i systemet är ett program som tar emot joystickkommandon och skickar dessa till en nätverksadress (till hunden). Detta program som finns i datorn är skrivet i C under Linuxmiljö. En mer detaljerad beskrivning finns i kapitel 3.2.

En andra del i systemet är ett program som utför rörelser m.m. på hunden utifrån de signaler som mottas ifrån datorn och i sin tur joysticken. De saker som detta program ska klara av blir i stort:

- Ta emot meddelanden via det trådlösa nätverket.
- Utföra rörelser utifrån dessa meddelanden.

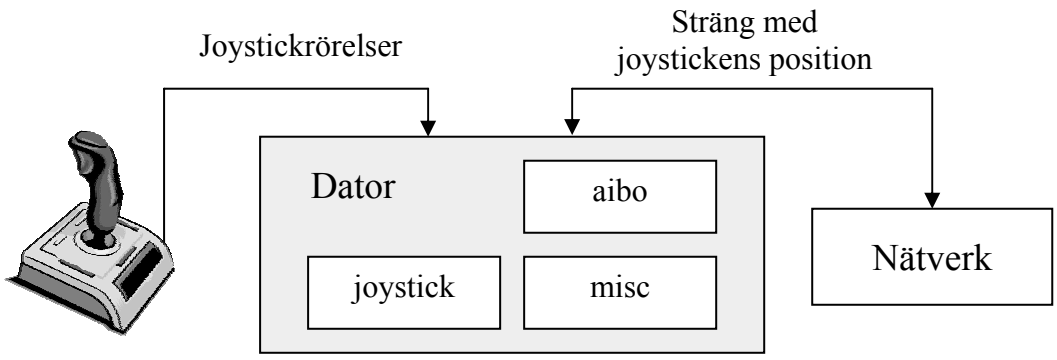
Programmet är skrivet med OPEN-R i C++ under operativsystemet Aperios. En mer detaljerad beskrivning finns i kapitel 3.3.

I den senare delen av detta kapitel kommer strukturen över hur hunden rör sig i förhållande till joysticken att beskrivas. Kapitel 3.4 beskriver alla rörelser som hunden kan utföra med detta system. Kapitel 3.5 beskriver joysticken med dess axlar och knappar. Kapitel 3.6 kommer sedan att beskriva hur hunden rör sig i förhållande till joysticken efter dessa förutsättningar.

En bildtagningsfunktion har lagts till systemet. Vid begäran tar AIBO:n en bild som returneras. Detta beskrivs närmare i kapitel 3.7.

3.2 Systemet i Datorn

I datorn finns ett C-program vars syfte är att läsa av en joysticks rörelser och skicka dessa till hunden. Informationen om joystickens rörelser läses kontinuerligt in och för att smidigt kunna skicka iväg denna information så packas den ihop till en sträng som känns igen av mottagaren (hunden). I programmet finns en enkel tangentbordsstyrning implementerad som främst är avsedd för testning, men kan även användas då en joystick inte finns tillgänglig. Datorn har ett operativsystem av Linuxtyp. Här finns tre filer skapade specifikt för uppgiften: *misc*, *joystick* och *aibo*. Dessa tre filer ingår alla i samma program.



Figur 3.2.1 Systemet i datorn, översikt.

Misc
Denna fil innehåller funktioner som används av filen *aibo*. Dessa är:

- `void delay(float seconds);` // Skapar en fördröjning på valt antal sekunder.
- `void clrscr(void);` // Rensar skärmen och flyttar markören upp till vänstra hörnet.
- `char getKey();` // Tar emot ett tecken från tangentbordet utan att enter- tryckning krävs.

Joystick
Denna fil har till uppgift att läsa av joysticken. Det finns här två funktioner:

- `void open_joystick(void);` // Initierar joysticken till programmet.
- `void read_joystick(void);` // Läser av joystickkommandon.

Aibo
Här finns main-funktionen. Programmet startar genom att joysticken initieras och kommunikation till hunden etableras. Detta följs av ett textbaserat menysystem där val av joystick- och tangentbordsstyrning kan väljas. Här finns en publik funktion:

- `void send_command(char *command);` // Skickar iväg en sträng till nätverket och väntar in en mottagandebekräftelse innan programmet sedan fortsätter.

3.2.1 Meddelandesträngen

För att smidigt skicka iväg joystickens/tangentbordets styrkommandon samlas hela informationen i en sträng. Denna sträng startar med ett tecken (’J’ för Joystick och ’K’ för Keyboard), som sedan följs av data i ett format som känns igen av hundens programvara. Exempel på meddelanden:

- `K07` // Skickar meddelande om att utföra kommando 7 till hunden.
- `J0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0` // Joysticken står helt still och inga knappar är nedtryckta.

Vid tangentbordsstyrning representerar tecknen vilken åtgärd som ska vidtas på hunden. I exemplet ovan ska åtgärd nummer 7 utföras på hunden. Vad detta innebär bestäms av hundens programvara.

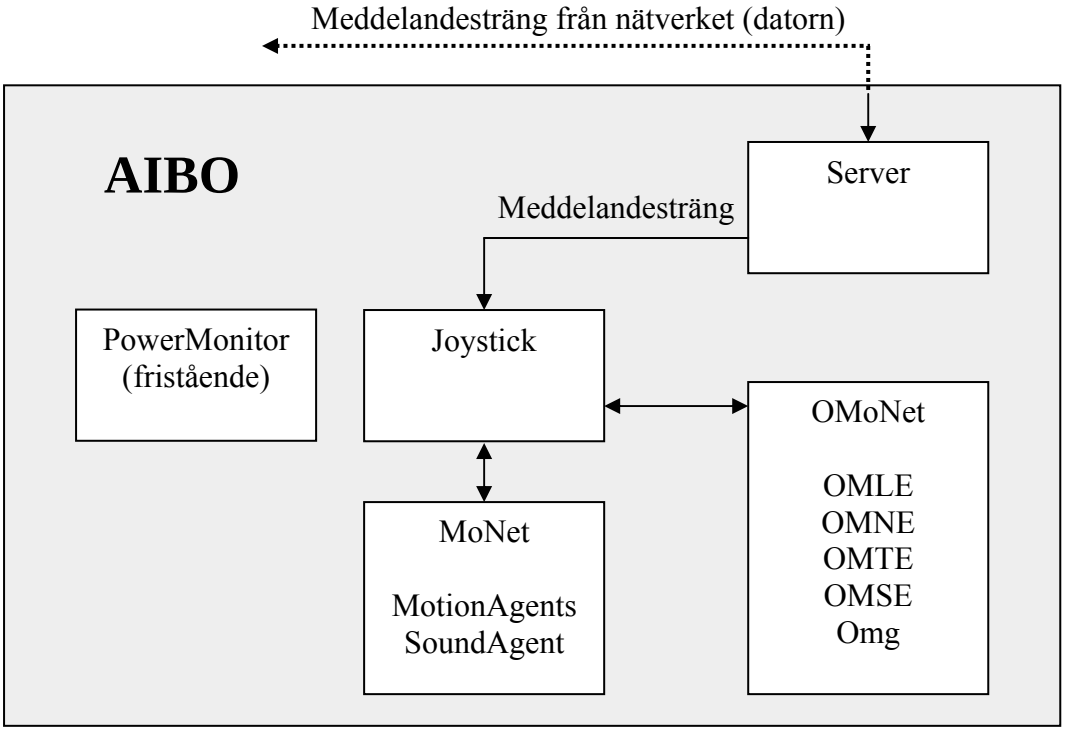
Vid joystickstyrning representerar de 6 första värdena axelriktningar och de följande 8 knapplägen (#-tecknet skiljer värdena åt). Axelriktningarna (för aktuell joystick) har värden mellan ±32767. Dessa är uppdelade till ±3 dels för att ytterligare noggrannhet i nuläget inte behövs och dels för att minska belastningen på programvaran i hunden.

3.3 Systemet i AIBO:n

I AIBO:n finns ett C++-program vars syfte är att ta emot en meddelandesträng och utföra rörelser m.m. utifrån denna. Programmet består av 12 stycken OPER-R-objekt (se kapitel 2.2.2):

- Joystick // Styr vad som ska utföras på hunden.
- Server // Kommunicerar med programmet på datorn.
- MoNet // Har speciella rörelser för hunden definierade.
- MotionAgents // Används av MoNet för att generera rörelser.
- SoundAgent // Används av MoNet för att generera ljud.
- PowerMonitor // Bevakar på/av-knapp och strömförsörjning.
- OMoNet, OMLE, OMNE, OMTE, OMSE, Omg // 6 OPEN-R-objekt som används för rörelser på hunden.

Programmet använder sig av två olika system för att röra sig. Det ena systemet heter OMoNet och det andra heter MoNet. Dessa två rörelsesystem har sina egna speciella rörelser och är sammankopplade i detta program för att få ett så stort rörelseschema som möjligt. Rörelsesystemen som är hämtade från andra program har modifierats för att passa in i detta joystickstyrningsprogram. Strukturen på programmet visas nedan med de väsentligaste delarna utsatta.



Figur 3.3.1 Systemet i AIBO:n, översikt.

PowerMonitor
PowerMonitors enda syfte är att bevaka hundens på/av-knapp och strömförsörjning.

Server
Servern kan både skicka och ta emot data. Servern är inställd på att ta emot textsträngar, men även andra dataformer går med små modifikationer att ta hand om.

Joystick
Det är detta objekt som utför rörelserna på hunden utifrån inkommen meddelandesträng. Hur hundens rörelser är sammankopplade med meddelandesträngen kan läsas i kapitel 3.6. Hur denna fil är uppbyggd står beskrivet mer utförligt i kapitel 3.3.1.

OMoNet & MoNet
Dessa två objekt med dess tillhörande objekt (se figur 3.3.1) kallas i rapporten för de två rörelsesystemen. Rörelsesystemen kräver en del förklaring så de tas upp i mer detalj i kapitel 3.4.

3.3.1 Beskrivning av OPEN-R-objektet Joystick

Eftersom Joystick-objektet är den centrala delen i joystickprogrammet beskrivs det objektet här lite mer utförligt. Ett OPEN-R-objekt kan som tidigare nämnts ha fler ställen varifrån den startas. Joystick-objektet har 11 sådana här ställen:

1. virtual OStatus DoInit (const OSystemEvent& event)
2. virtual OStatus DoStart (const OSystemEvent& event)
3. virtual OStatus DoStop (const OSystemEvent& event)
4. virtual OStatus DoDestroy (const OSystemEvent& event)
5. void ReadyMotion (const OReadyEvent& event)
6. void ReadySpecialMotion (const OReadyEvent& event)
7. void NotifySpecialMotion (const ONotifyEvent& event)
8. void NotifyMotion (const ONotifyEvent& event)
9. void ResultGSensor (const ONotifyEvent& event)
10. void ResultSensor (const ONotifyEvent& event)
11. void MakeDecision (const ONotifyEvent& event)

Funktionerna startas beroende på events (händelser). Vilket event som startar vilken funktion beror på en koppling objekten emellan. Denna koppling finns konfigurerad i en fil tillhörande objektet med namnet stub.cfg.

De fyra första funktionerna är obligatoriska och måste alltid finnas. Dessa anropas vid uppstart och nedkoppling. De övriga är specifika för detta objekt, där 5 och 6 är s.k. subjects och resterande är s.k. observers. Med subject menas att funktionen sänder iväg data, medan observer betyder att det är en funktion som tar emot data. Event inträffar när dessa funktioner får meddelande om att data är redo att sändas eller om data finns redo att tas emot. Objekten använder gemensamma minnesutrymmen när de sänder data och det finns ingen direkt kommunikation objekten emellan. Kommunikationen sker istället i form av publika meddelanden om var i minnesutrymmet och till vem som datan är ämnad. Nedan följer en liten beskrivning av vilka objekt funktionerna kommunicerar med.

ReadyMotion & NotifyMotion
Kommunicerar med objektet OMoNet för att generera hundens rörelser.

ReadySpecialMotion & NotifySpecialMotion
Kommunicerar med objektet MoNet för att generera hundens rörelser.

ResultGSensor
Bevakar accelerationssensorn och ser om den ger utslag för att hunden faller.

ResultSensor
Ger konstant uppdatering av sensorers värden på hunden.

MakeDecision
Tar emot ny information ifrån joysticken (ny meddelandesträng).

Joystick-objektet måste hålla reda på i vilket tillstånd den befinner sig i. Det finns i objektet 5 sådana här tillstånd:

J_BOOTING
Objektet är under uppstart. Den måste utföra vissa rutinrörelser innan den kan göra något annat. Dessa rutinrörelser är att gå från oidentifierat läge till standardläge i båda rörelsesystemen.

J_ONLINE
När objektet är i detta tillstånd utför den rörelser enligt kommandon från meddelandesträngen. Det är ett s.k. standardtillstånd då inget annat speciellt utförs på hunden. I detta tillstånd är det rörelser från OMoNet som utförs.

J_PREP_SM (prepare special motion)
Förbereder en rörelse från MoNet genom att gå till standardpositionen (stående).

J_EXEC_SM (execute special motion)
Tillståndet då en MoNet-rörelse utförs. Joystick-objektet väntar i princip bara på att rörelsen ska bli klar. När rörelsen är klar utförs ytterligare en MoNet-rörelse där AIBO:n går tillbaka till standardpositionen (stående).

J_RECOVER
Ifall AIBO:n av någon anledning trillar (t.ex. när den sparkar en boll), så utförs en rörelse som får AIBO:n att resa på sig igen. Det är i detta tillstånd den befinner sig då.

- För att inte missa några rörelsekommandon vid exekvering används en rörelsemeddelandekö. Denna kö sparar undan rörelsekommandon från OMoNet (ej MoNet) för att utföras senare.

- För att utföra en rörelse i Joystick-objektet, sätts först variabeln: `jmotion_` till önskad rörelse, för att sedan anropa funktionen: `Move()`. Tillståndet måste vara `J_ONLINE` annars utförs inte rörelsen, men om det är en OMoNet-rörelse kan den läggas in i kön för att utföras senare.

- Rörelserna av hundens huvud styrs med två av joystickens axelleder. Hundens rörelsesystem för huvudet är uppbyggt på ett sätt som innebär att huvudet vid kommando sätts till en viss koordinat. För att få en mjuk rörelse uppdateras huvudets koordinat i funktionen `ResultSensor` då denna anropas med relativt jämna mellanrum. Hastigheten på hur mycket huvudet rör sig vid varje uppdatering bestäms av två variabler: `PAN_GAIN` och `TILT_GAIN`.

3.4 AIBO:ns möjliga rörelser

De rörelser som hunden med detta program kan utföra kommer att beskrivas här. Som nämnts tidigare används två olika rörelsesystem med sina egna speciella rörelser. I figur 3.4.1 nedan finns en bild över vilka rörelser som kan genereras från vilket system.

Figur 3.4.1 De två rörelsesystemen.

De två rörelsesystemen använder sig båda av en variabel som håller reda på vilket tillstånd den senast genererade rörelsen slutade vid (stående, sittande eller sovande). Detta så att de vet hur de smidigt kan röra sig när de ska göra sin nästa rörelse. Dessa två variabler krockar därmed när det andra systemet utför en rörelse och variabeln inte längre anger hundens nuvarande position.

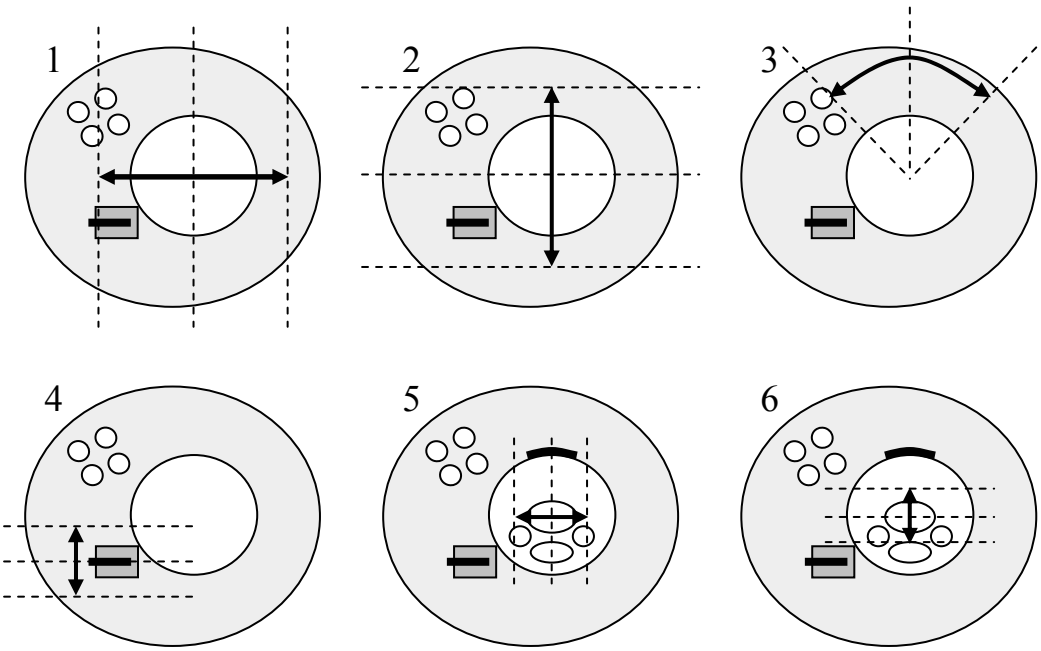
Detta är löst genom att, när en övergång sker mellan de olika rörelsesystemen, går hunden till en viss standardposition som är likadan i det båda systemen (positionen stående). I och med detta börjar och slutar båda rörelsesystemens rörelser från denna standardposition och därmed undviks de ryckiga rörelser som annars hade uppkommit.

3.5 Joysticken

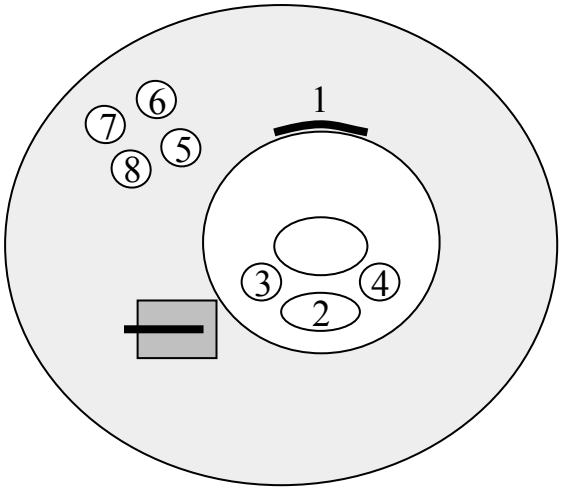
Joysticken som använts för att styra roboten är av modellen:

Microsoft Sidewinder Force Feedback 2.

Den har axlar som går att röra i 6 olika leder samt 8 knappar. Var dessa finns illustreras i figurerna 3.5.1 och 3.5.2. Då drivrutiner inte finns tillgängliga från leverantören annat än till Windows, har drivrutiner som tar tillvara på nästan alla funktioner i joysticken laddats ner från en nätadress [5]. Det finns i nuläget (november 2003) ett projekt [6] som arbetar för att kunna använda alla joystickens funktioner.



Figur 3.5.1 Joystickens 6 olika axelleder.



Figur 3.5.2 Joystickens 8 knappar.

3.6 Rörelsemappning

Detta kapitel beskriver hur hunden rör sig i förhållande till joystickens rörelser i det program som skapats. Det är detta som i rapporten refereras som rörelsemappning. I kapitel 3.4 och 3.5 beskrivs de möjligheter och förutsättningar som ligger till grund för denna mappning. Den mappning som finns nu är något som enkelt går att utvidga i ett eventuellt framtida arbete. Tabell 3.6.1 och 3.6.2 visar hur joysticken och hunden samverkar.

Joystickknapp	AIBO-rörelse
1	Utför en slumpvis vald rörelse ur MoNet-paketet.
2	Återgå till standardläge (stå upp, huvudet fram).
3	Spark med vänster ben (i huvudets riktning).
4	Spark med höger ben (i huvudets riktning).
5	Används inte.
6	Vickar på svansen.
7	Frikoppling. Kopplar ur övriga knappar (förutom avslutaknappen) och axlar så ingenting händer när dessa rörs.
8	Avslutaknapp.

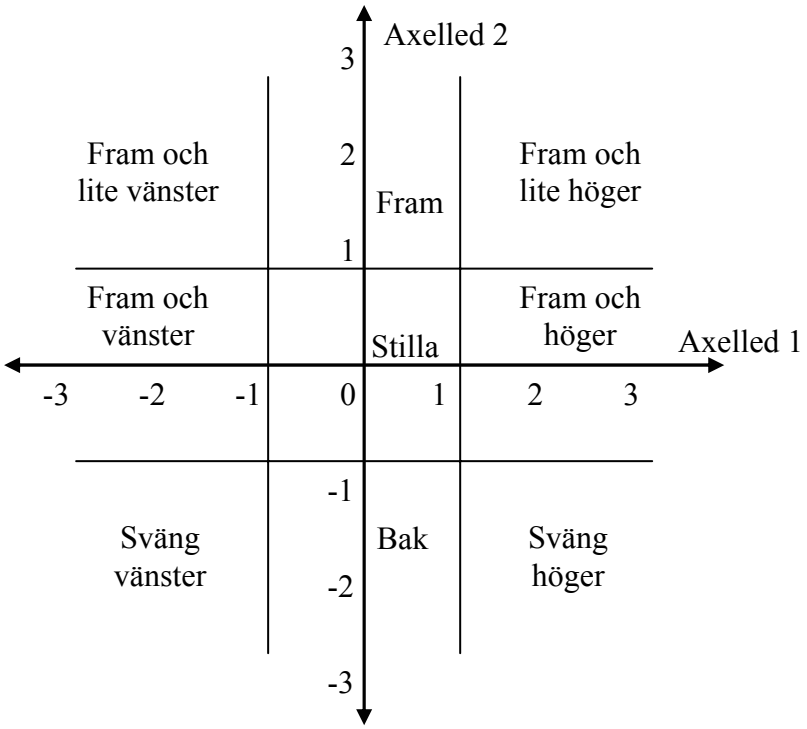
Tabell 3.6.1 Mappningen Joystick -> AIBO-rörelse. Knappar.

Axelled	AIBO-rörelse
1	Gång vänster/höger med gångstilen default.
2	Gång fram/bak med gångstilen default.
3	Används inte.
4	Används inte.
5	Huvudrörelse vänster/höger.
6	Huvudrörelse upp/ner.

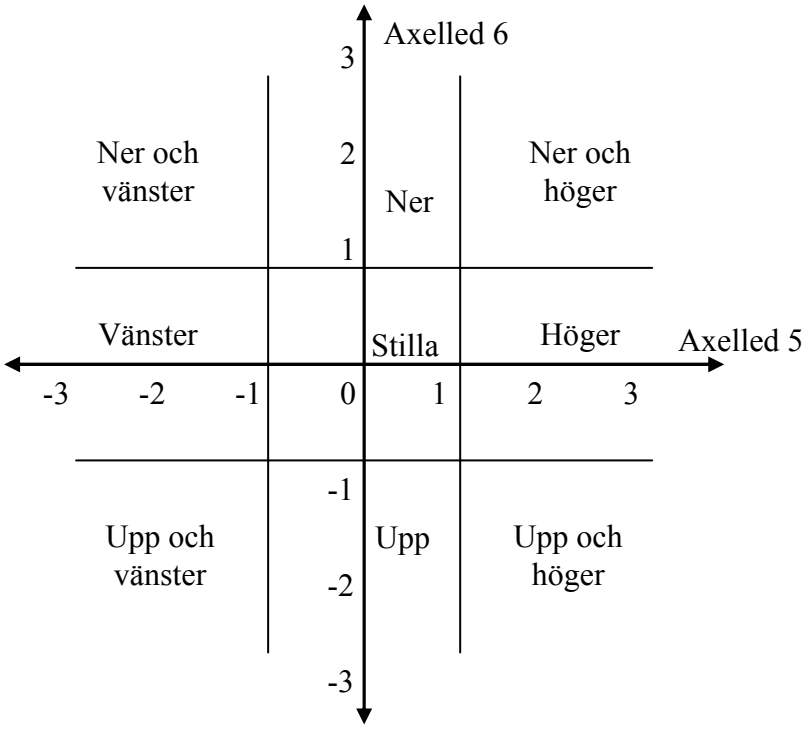
Tabell 3.6.2 Mappningen Joystick -> AIBO-rörelse. Axlar.

Värt att notera är att axlarnas värden inte spelar någon roll när de väl kommit över ett visst tröskelvärde. I kapitel 3.2.1 står det att axlarnas värden förkortats ner till ±3. Tröskelvärdet ligger i nuläget vid ±1, så om joystickens axel över-/understiger 1/-1 så utförs rörelsen. Figur 3.6.3 och 3.6.4 illustrerar bl.a. detta.

En extrafunktion som lagts till är att ifall hunden ramlar omkull kommer den automatiskt att kunna resa på sig igen. Hunden kan uppfatta fall och fallriktningar med hjälp av sin accelerationssensor.



Figur 3.6.3 Resultat av axelvärden för gång.



Figur 3.6.4 Resultat av axelvärden för rörelser av huvud.

3.7 Bildtagning

Då tid fanns över skapades en extrafunktion till programmet. En bildtagningsfunktion där OPEN-R objektet Server returnerar en bild ifall inkommen meddelandesträng börjar med ett X. Denna funktion används av ett externt program [9] som kopplar upp sig mot servern och kontinuerligt tar emot bilder från programmet. Det externa programmet visar upp bilder i snabb följd i realtid från AIBO:ns kamera. Detta program kan koppla upp sig mot servern samtidigt som PC-programmet (för joystick och tangentbordstyrning) också är uppkopplat. Kombinationen av dessa två program ger en joystickstyrning där användaren i realtid kan se vart han styr från AIBO:ns kamera.

4 Slutsatser och kommentarer

4.1 Bedömning av lösningen

I problemformuleringen togs tre punkter upp som skulle utföras för att hunden skulle kunna röra på sig:

1. Läsa av Joystickens rörelser.
2. Upprätta kommunikation mellan AIBO:n och datorn.
3. Ett program på AIBO:n som tar emot styrkommandon och utför dessa.

Dessa tre punkter har utförts och hunden förflyttar sig efter joystickens rörelser på ett helt godtagbart sätt. Vad gäller drivrutiner så används en generell drivrutin för joysticks under Linux. Denna drivrutin har stöd för alla knappar och axlar, men har ingen möjlighet att tolka kraftåterkopplingsfunktionen i joysticken. Då examensarbetet i stort utgick från att få fram ett resultat så fullföljdes detta genom att många redan färdiga moduler sattes ihop. Det finns i OPEN-R-paketet en massa exempelprogram m.m. som med framgång modifierats och satts ihop för att nå fram till resultatet.

Då användningssyftet med joystickstyrningen är lite oklart så har inte så stort arbete lagts ner på att göra en slutgiltig lösning på rörelsemappningen. Arbetet har istället mer koncentrerats på att skapa så stora möjligheter som möjligt att koppla in de rörelser som man vill ha till hunden. Med detta finns möjligheten att skapa flera olika sorters rörelsemappning genom att modifiera OPEN-R-objektet Joystick.

4.2 Förbättringsmöjligheter

Det har tidigare nämnts att möjligheten finns att skapa en bättre rörelsemappning än den nuvarande. I nuvarande rörelsemappning används endast en gångstil av 10 möjliga. Som beskrivits i kapitel 3.6 antar axlarna värden på ±3, vilket innebär att 3 olika gångstilar kan användas beroende på hur mycket axeln rörs. T.ex. kan axelvärde innebära:

0 → stilla.
1 → använd gångstilen slow.
2 → använd gångstilen stable.
3 → använd gångstilen fast.

På detta sätt kan en lite intressantare gång erhållas.

En ytterliggare förbättring är att använda en egen gångstil med mer jämn gång och där fart och vinkel är mer analoga. För att kunna göra detta krävs först att signalerna om joystickens axelvärden får en högre upplösning, vilket kan höjas i programmet på datorn upp till ±32767, samt att denna nya gångstil måste läggas till i programmet på hunden. En extra fördel som detta medför är att hunden då även eventuellt kan kunna gå snett bakåt vilket inte är möjligt med nuvarande system.

Joysticken som används har en kraftåterkopplingsfunktion vilket innebär att motorer på joysticken kan ge en känsla av motstånd när axeln rörs. Detta ger möjlighet att från hunden skicka signaler till joysticken om något speciellt inträffar. Vad detta skulle kunna tänkas vara överlåter jag till läsarens fantasi.

Som beskrivs i kapitel 3.4 används en standardposition för övergångar mellan de två olika rörelsesystemen. Det kan uppfattas som omständigt att hunden måste gå via en position när den byter rörelsesystem. För att slippa detta kan variablerna som håller reda på denna position tas till vara och uppdatera det andra inaktiva rörelsesystemet i fråga. Variabeln som håller reda på rörelsesystemet MoNet:s nuvarande position heter `currentPosture`.

Referenser

[1] Officiell hemsida för OPEN-R SDK:
http://openr.aibo.com/
(Länken verifierad december 2003)

[2] En av Sonys sidor med information om AIBO:
http://www.aibo-europe.com/
(Länken verifierad december 2003)

[3] Hemsida för RoboCup.:
http://www.robocup.org/
(Länken verifierad december 2003)

[4] Microsofts informationssida om joysticken som använts:
http://www.microsoft.com/hardware/sidewinder/FFB2.asp
(Länken verifierad december 2003)

[5] Sida där drivrutinen till joysticken finns.
http://www.linuxgames.com/joystick/
(länken verifierad december 2003)

[6] Sida där utveckling av drivrutin till joysticken bedrivs.
http://madfab.free.fr/ff/
(Länken verifierad december 2003)

[7] OPEN-R SDK documentation English - 1.1.3 - r4. Manualer som bifogas OPEN-R SDK.
- InstallationGuide
- InternetProtocolVersion4
- Level2ReferenceGuide
- ModelInformation_210
- ProgrammersGuide
- RevisionRecord

[8] OPEN-R SDK tutorials English. Handledning som bifogas OPEN-R SDK.
- SDK_Tutorial1Web
- SDK_Tutorial2Web
- SDK_Tutorial3Web
- SDK_Tutorial4Web
- SDK_Tutorial5Web
- SDK_Tutorial6Web
- SDK_Tutorial7Web
- tutorial_OPENR_ENSTA-1.0

[9] Colorcalibration Java program. Program som kan kopplas ihop med joystickprogrammet för att visa bild i realtid från AIBO:ns kamera. http://www.efd.lth.se/~e97ca/xj/Aibo_final/
(Länken verifierad december 2003)

Bilaga A – Ordlista

AI: Artificiell Intelligens.

AIBO: Artificially Intelligent roBOts. En serie robotar från Sony där namnet också är det japanska ordet för: sällskap/vän.

Aperios: Ett objektorienterat realtidsoperativsystem.

Effektor: Något som manipulerar den fysiska verkligheten, t.ex. en lampa eller en rörelse.

ERS: Entertainment RobotS.

Force Feedback: En funktion som ger tillbaka en respons i form av en kraft. Direkt översättning: kraftåterkoppling/kraftrespons.

Gränssnitt: Kontaktyta mellan olika funktioner eller delar i ett system.

Interobjektkommunikation: Den kommunikation som används mellan objekt i OPEN-R.

Observer: Den sida som tar emot data vid interobjektkommunikation.

OPEN-R (SDK): Ett C++-baserat programmeringsgränssnitt som används för utveckling av mjukvara till AIBO-robotar.

OPEN-R-objekt: I denna rapport, ett objekt skapat i OPEN-R SDK.

Rörelsemappning: I denna rapport, förhållandet mellan rörelsen på joysticken och rörelsen på AIBO:n.

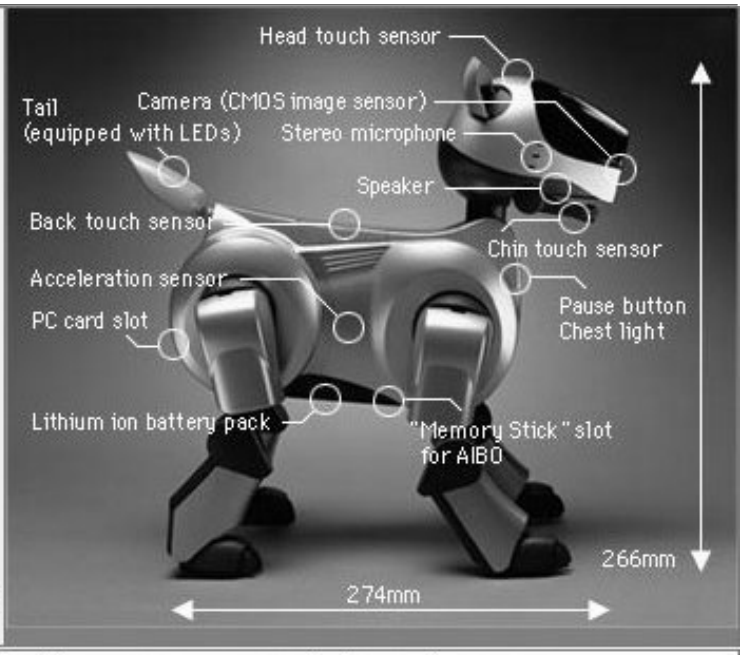
SDK: Software Development Kit.

Sensor: Något som observerar den fysiska verkligheten, t.ex. en mikrofon eller en kamera.

Subject: Den sida som sänder data vid interobjektkommunikation.

Bilaga B – AIBO-data

Dessa två bilder är hämtade från: http://www.aibo-friends.com/contentid-33.html (länken verifierad november 2003) och sammanställer teknisk data om AIBO av modell ERS-210.

<table>
<tr>
<td>
- Head touch sensor
- Camera (CMOS image sensor)
- Stereo microphone
- Speaker
- Chin touch sensor
- Pause button Chest light
- Back touch sensor
- Acceleration sensor
- "Memory Stick" slot for AIBO
- Lithium ion battery pack
- PC card slot
- Tail (equipped with LEDs)
- Joints (20 "degrees of freedom")
Weights (inclusing a battery and a memory stick) : Approx. 1.5kg (Approx.3.3lb)
Dimensions (W/H/D) (not including ears and tail) : Approx. 152 x 281 x 250 mm (Approx. 5.98 x 11.06 x 9.84 inches)
</td>
<td>


</td>
</tr>
<tr><td>CPU</td><td>64-bit RISC processor. Clock speed 384 MHz</td></tr>
<tr><td>Components</td><td>Body, Head, Tail, Leg x 4, "Removable"</td></tr>
<tr><td>Program Storage Medium</td><td>Memory Stick for AIBO</td></tr>
<tr><td>Movable Parts</td><td>Mouth: 1 degree of freedom
Head: 3 degrees of freedom
Legs: 3 degrees of freedom x 4 legs
Ears: 1 degree of freedom x 2 ears
Tail: 2 degrees of freedom
Total: 20 degrees of freedom</td></tr>
</table>

Figur B.1 Data om AIBO ERS-210.

Input/Output	PC Card slot Type2 In/Out Memory Stick slot In/Out AC IN Power Supply connector Input
Image Input	CMOS Image sensor
Audio Input	Miniature Microphones
Audio Output	Miniature Speaker
LCD Display	Time, Volume, Battery condition
Built-in Sensors	Temperature Sensor Infrared Distance Sensor Acceleration Sensor Pressure Sensors (Head, the Back, Chin & Legs) Vibration Sensor
Built-in Clock	Date & Time
Power Consumption	Approx. 9W (tentative) Standard operation in autonomous mode
Operating Time	Approx. 1.5Hours (tentative) Standard operation in autonomous mode
Charging Time	Approx. 2 hours (with a supplied AC Power Adaptor and the "Lithium Ion Battery pack" ERA-201B1)
Dimensions(W/H/D) (not including ears and tail)	Approx. 152 x 281 x 250 mm (Approx. 5.98 x 11.06 x 9.84 inches)
Weights(including a battery and a memory stick)	Approx. 1.5kg (Approx.3.3lb)
Color	Gold/Silver/Black
Supplied Accessories	AC Adapter, Lithium Ion Battery Pack ERA-201B1(1), Ball, Documentation, etc
Operating Temperature	41 F to 95 F (5C to 35C)
Operating Humidity	10% to 80%

Figur B.2 Data om AIBO ERS-210.

Bilaga C – Specifik programinformation

Här visas och kommenteras programmets struktur. Denna bilaga är tänkt ge instruktioner till den som vill testa och modifiera programmet.

C.1 Kompileringsinstruktioner

1. Gå till katalogen: `cd Joystick/`
2. Skriv: `make`
3. Vänta tills klar, skriv sedan: `make install`

C.2 Minneskortinstruktioner

Minneskortet behöver förberedas genom att vissa filer till operativsystem m.m. laddas ner. Detta behöver bara göras en gång och den version som rekommenderas är: wconsole, nomemprot.

Sedan ska följande utföras vid varje modifikation av programmet:

1. Montera minneskortläsaren: `mount /mnt/memory_card/`
2. Kopiera filerna till minneskortet:
   `cp -Rf MS/OPEN-R/ /mnt/memoty_card/`
3. Avmontera minneskortläsaren: `umount /mnt/memory_card/`
4. Vänta tills avmonteringen är klar.

C.3 Exekveringsinstruktioner

1. Starta AIBO:n med På/Av-knappen.
2. Starta PC-programmet: `./aibo` (finns i katalogen: `Joystick/PC/`)
3. Vänta tills AIBO:n startat upp och menysystemet i PC-programmet kommer fram.
4. Välj tangentbordstyrning, joystickstyrning eller att avsluta.

C.4 Programstruktur och information

Nedan finns katalogstrukturen i programmet (visas i grått) med kommentarer till en del av filerna/katalogerna. Kommentarerna kommer ovanför filen/katalogen som dem syftar till.

```
Katalogen med all mjukvara för joystickstyrning av AIBO.
Joystick/
```

Katalogen där programmet, ämnat att köra i datorn, finns.
Exekverbara filer är:

- ./aibo -> Kör programmet (inga argument).
- ./clean -> Rensar katalogen från onödiga filer.
- make -> kompilerar programmet (makefile).
- ./jstest -> Kör joysticktestprogram. Hjälp om argumenten kommer upp vid körning.

Joystick/PC/

Skriv här in port och adress till AIBO:n.
Joystick/PC/aibo.h

Lite speciella kommandon som används visas här.
Joystick/PC/misc.h

Joystick- och tangentbordstyrningen sköts härifrån. Upprättar även kommunikationen.
Joystick/PC/aibo.c
Joystick/PC/misc.c
Joystick/PC/makefile
Joystick/PC/jstest
Joystick/PC/aibo
Joystick/PC/clean
Joystick/PC/jstest.c

Skriv här in sökvägen till joysticken.
Joystick/PC/joystick.h

Här finns programmerat vad som händer vid joystickstyrning. Bara skicka iväg joystickens position eller utföra något annat på datorn.
Joystick/PC/joystick.c
Joystick/Makefile

Bibliotek och filer för att få gångkommandon m.m. från Soccerlion-paketet att fungera.
Joystick/OMWares/
Joystick/OMWares/MS/
Joystick/OMWares/MS/nomemprot/
Joystick/OMWares/MS/nomemprot/OPEN-R/
Joystick/OMWares/MS/nomemprot/OPEN-R/MW/
Joystick/OMWares/MS/nomemprot/OPEN-R/MW/DATA/
Joystick/OMWares/MS/nomemprot/OPEN-R/MW/DATA/P/
Joystick/OMWares/MS/nomemprot/OPEN-R/MW/DATA/P/BASEPOS.CFG
Joystick/OMWares/MS/nomemprot/OPEN-R/MW/OBJS/
Joystick/OMWares/MS/nomemprot/OPEN-R/MW/OBJS/OMCDT.BIN
Joystick/OMWares/MS/nomemprot/OPEN-R/MW/OBJS/OMG.BIN
Joystick/OMWares/MS/nomemprot/OPEN-R/MW/OBJS/OMLE.BIN
Joystick/OMWares/MS/nomemprot/OPEN-R/MW/OBJS/OMNE.BIN
Joystick/OMWares/MS/nomemprot/OPEN-R/MW/OBJS/OMONET.BIN
Joystick/OMWares/MS/nomemprot/OPEN-R/MW/OBJS/OMPSD.BIN
Joystick/OMWares/MS/nomemprot/OPEN-R/MW/OBJS/OMSE.BIN
Joystick/OMWares/MS/nomemprot/OPEN-R/MW/OBJS/OMTE.BIN
Joystick/OMWares/include/
Joystick/OMWares/include/OMWares/
Joystick/OMWares/include/OMWares/CdtStruct.h
Joystick/OMWares/include/OMWares/GInfo.h
Joystick/OMWares/include/OMWares/LE2CmdList.h
Joystick/OMWares/include/OMWares/LE2CmdStruct.h
Joystick/OMWares/include/OMWares/LE2_S_CmdList.h
Joystick/OMWares/include/OMWares/LocomoParamMaker.h

```
Joystick/OMWares/include/OMWares/MMM_CommandList.h
Joystick/OMWares/include/OMWares/MoNetMessageMaker.h
Joystick/OMWares/include/OMWares/Name.h
Joystick/OMWares/include/OMWares/OMGsensorData.h
Joystick/OMWares/include/OMWares/OMLE2Commander.h
Joystick/OMWares/include/OMWares/OMPsdCPG.h
Joystick/OMWares/include/OMWares/OMPsdRH.h
Joystick/OMWares/include/OMWares/OMPsdStatus.h
Joystick/OMWares/include/OMWares/OMTECommander.h
Joystick/OMWares/include/OMWares/OMTouchSensorData.h
Joystick/OMWares/include/OMWares/OMoNetMessage.h
Joystick/OMWares/include/OMWares/SECmdStruct.h
Joystick/OMWares/include/OMWares/SE_CmdList.h
Joystick/OMWares/include/OMWares/TECmdList.h
Joystick/OMWares/include/OMWares/TE_CmdList.h
Joystick/OMWares/lib/
Joystick/OMWares/lib/libOMWares.a

OPEN-R-objektet Joystick tar emot ett joystick- eller
tangentbordskommando från OPEN-R-objektet Server och utför dessa
med hjälp av diverse andra OPEN-R-objekt.
Joystick/Joystick/
Joystick/Joystick/Makefile
Joystick/Joystick/Joystick.cc
Joystick/Joystick/Joystick.h
Joystick/Joystick/joystick.ocf
Joystick/Joystick/MNetMsgQ.cc
Joystick/Joystick/MNetMsgQ.h
Joystick/Joystick/stub.cfg

MS-katalogen (MS = Memory stick).
Joystick/MS/
Joystick/MS/OPEN-R/
Joystick/MS/OPEN-R/MW/
Joystick/MS/OPEN-R/MW/CONF/

Konfigurationen över hur alla OPEN-R-objekten är sammankopplade
Joystick/MS/OPEN-R/MW/CONF/CONNECT.CFG

Konfigurationen över vilka OPEN-R-objekt som finns med.
Joystick/MS/OPEN-R/MW/CONF/OBJECT.CFG

MoNet-paketets rörelser finns definierade här. Här går att lägga
till kombinationer av rörelser utifrån befintliga rörelser.
Joystick/MS/OPEN-R/MW/CONF/MONETCMD.CFG
Joystick/MS/OPEN-R/MW/CONF/MONET.CFG
Joystick/MS/OPEN-R/MW/CONF/DESIGNDB.CFG
Joystick/MS/OPEN-R/MW/DATA/
Joystick/MS/OPEN-R/MW/DATA/P/

Konfigurationen över hur AIBO:n ska bete sig när den hamnat i ett
odefinierat tillstånd (t.ex. vid programstart). Används för
Soccerlion-paketets rörelser.
Joystick/MS/OPEN-R/MW/DATA/P/BASEPOS.CFG
Joystick/MS/OPEN-R/MW/DATA/P/MOTION.ODA
Joystick/MS/OPEN-R/MW/DATA/P/SOUND.ODA
Joystick/MS/OPEN-R/MW/OBJS/


Konfigurationen på AIBO:ns nätverksinställningar.
Joystick/MS/OPEN-R/System/Conf/wlanconf.txt
```

OPEN-R-objektet MoNet. Använder sig av OPEN-R-objektet MotionAgents för att generera rörelser, och OPEN-R-objektet SoundAgent för att generera ljud. Rörelser och ljud genereras genom att ett id, som finns specificerat i filen: Joystick/MS/OPEN-R/MW/CONF/MONETCMD.CFG, skickas till MoNet.

```
Joystick/MoNet/
Joystick/MoNet/CommandArc.h
Joystick/MoNet/CommandNode.h
Joystick/MoNet/DirectedGraph.h
Joystick/MoNet/MoNet.cc
Joystick/MoNet/MoNet.h
Joystick/MoNet/moNet.ocf
Joystick/MoNet/MoNetCommandInfo.cc
Joystick/MoNet/MoNetCommandInfo.h
Joystick/MoNet/MoNetCommandInfoManager.cc
Joystick/MoNet/MoNetCommandInfoManager.h
Joystick/MoNet/stub.cfg
Joystick/MoNet/Makefile
```

OPEN-R-objektet PowerMonitor. Lyssnar av På/Av-knappen samt kollar strömförbrukningen.

```
Joystick/PowerMonitor/
Joystick/PowerMonitor/Makefile
Joystick/PowerMonitor/PowerMonitor.cc
Joystick/PowerMonitor/PowerMonitor.h
Joystick/PowerMonitor/powerMonitor.ocf
Joystick/PowerMonitor/stub.cfg
```

OPEN-R-objektet Server. Tar emot skickad data från PC-programmet aibo (i katalogen: Joystick/PC/aibo), skickar denna information till OPEN-R-objektet Joystick samt skickar en bekräftelse tillbaka till PC-programmet.

```
Joystick/Server/
Joystick/Server/Makefile
Joystick/Server/TCPConnection.h
Joystick/Server/server.ocf
Joystick/Server/stub.cfg
Joystick/Server/Server.cc
Joystick/Server/Server.h
```

Här definieras vilken port som Server ska lyssna på.

```
Joystick/Server/ServerConfig.h
```

OPEN-R-objektet MotionAgents används av OPEN-R-objektet MoNet för att generera rörelser.

```
Joystick/MotionAgents/
Joystick/MotionAgents/MoNetAgent.cc
Joystick/MotionAgents/MoNetAgent.h
Joystick/MotionAgents/MoNetAgentManager.cc
Joystick/MotionAgents/MoNetAgentManager.h
Joystick/MotionAgents/MotionAgents.cc
Joystick/MotionAgents/MotionAgents.h
Joystick/MotionAgents/motionAgents.ocf
Joystick/MotionAgents/MTNAgent.cc
Joystick/MotionAgents/MTNAgent.h
Joystick/MotionAgents/NeutralAgent.cc
Joystick/MotionAgents/NeutralAgent.h
Joystick/MotionAgents/stub.cfg
Joystick/MotionAgents/Makefile
```

```
OPEN-R-objektet SoundAgent används av OPEN-R-objektet MoNet för
att generera ljud.
Joystick/SoundAgent/
Joystick/SoundAgent/SoundAgent.cc
Joystick/SoundAgent/SoundAgent.h
Joystick/SoundAgent/soundAgent.ocf
Joystick/SoundAgent/stub.cfg
Joystick/SoundAgent/WAV.cc
Joystick/SoundAgent/WAV.h
Joystick/SoundAgent/Makefile

Bibliotek och filer som får kommandon från MoNet-paketet att
fungera.
Joystick/common/
Joystick/common/include/
Joystick/common/include/MoNetData.h
Joystick/common/include/MTN.h
Joystick/common/include/MTNFile.h
Joystick/common/include/ODA.h
Joystick/common/libMTN/
Joystick/common/libMTN/MTN-FFORM-E.txt
Joystick/common/libMTN/MTN-FFORM-J.txt
Joystick/common/libMTN/MTN.cc
Joystick/common/libMTN/MTNFile.cc
Joystick/common/libMTN/MTNFile_Print.cc
Joystick/common/libMTN/Makefile
Joystick/common/libODA/
Joystick/common/libODA/ODA.cc
Joystick/common/libODA/Makefile
```

C.5 Modifieringsinstruktioner

Det är främst i OPEN-R-objektet Joystick som modifieringar kan göras.

Joystick.h
Här finns bl.a. möjligheter att ändra knappar och axelriktningar på joysticken. T.ex. kan man byta utförandet av knapp 3 och knapp 5, väldigt lätt genom att just byta plats på dessa.

Joystick.cc
Denna fil innehåller nästan all kod som används för rörelsemappningen.

Funktionen MakeDecision anropas då ett nytt kommando tagits emot från PC-programmet (via OPEN-R-objektet Server). Här kan logik angående olika knapptryckningar och axelrörelser skrivas.

Funktionen ResultSensor används nu endast för att röra på huvudet. Denna är programmerad så att den anropas kontinuerligt. Skriv till kod här ifall något behöver anropas kontinuerligt.

Server.cc
I funktionen ReceiveCont tas meddelanden emot och skickas vidare till OPEN-R-objektet Joystick. Här kan saker som inte behöver involvera joysticken, men som behöver skickas/tas emot, läggas till.