

Testing OPEN-R Samples for SONY AIBO ERS-7

Miguel Silva Rentes

`<miguel.rentes@fe.up.pt>`

LIACC - Artificial Intelligence and Computer Science Laboratory
University of Oporto, Portugal



March 22, 2005

Contents

1	Samples for ERS-7	4
1.1	BallTrackingHead7	4
1.2	BlinkingLED7	5
1.3	MovingHead7	5
1.4	MovingLegs7	5
1.5	SensorObserver7	5
1.6	PIDControl7	7
1.7	LMasterRSlave7	8
2	Common samples for ERS-7 and ERS-200 series	9
2.1	BattChecker	9
2.2	Crash	10
2.3	DNSLookUp	11
2.4	EchoClient	11
2.5	EchoServer	12
2.6	ERA201D1Info	12
2.7	HelloWorld	12
2.8	HelloWorld-stubgen	13
2.9	ImageCapture	14
2.10	ImageObserver	15
2.11	MoNet	15
2.12	NTP	16
2.13	ObjectComm	16
2.14	ObjectComm-multi	17
2.15	PowerMonitor	17
2.16	RobotDesign	17
2.17	SoundPlay	18
2.18	SoundRec	18
2.19	TinyFTPD	19
2.20	UDPEchoServer	19
2.21	W3AIBO	20

List of Figures

1	Joints and Sensors indexes and values (part 1 of 3).	6
2	Joints and Sensors indexes and values (part 2 of 3).	7
3	Joints and Sensors indexes and values (part 3 of 3).	8
4	The PIDControl7 sample interface.	9
5	The BattChecker wireless console interface.	10
6	The BattChecker MFC window.	10
7	The Crash wireless console interface.	11
8	The DNSLookUp wireless console interface.	11
9	The EchoClient wireless console interface.	12
10	The EchoServer wireless console interface.	13
11	The ERA201D1Info wireless console interface.	13
12	The HelloWorld wireless console interface.	14
13	The ImageCapture sample.	14
14	A picture taken with ImageCapture sample.	15
15	The ImageObserver wireless console interface.	15
16	The MoNet wireless console interface.	16
17	The NTP wireless console interface.	16
18	Communication between 2 Subjects and 3 Objects.	17
19	The RobotDesign wireless console interface.	18
20	The SoundRec wireless console interface.	18
21	The TinyFTPD wireless console interface.	19
22	The UDPEchoServer wireless console interface.	20
23	Accessing a picture and all its layers with a web browser. . .	20

1 Samples for ERS-7

1.1 BallTrackingHead7

Let's start with something easy and fun at the same time. Give the following commands at the command line prompt:

1. `$ cd /sample/ers7/BallTrackingHead7/BallTrackingHead7/`
2. `$ make`
3. `$ make install`
4. `$ cd ..`
5. `$ make`
6. `$ make install`

This is pretty much the usual procedure for compiling all the necessary files to test almost any Open-R code on Aibo. The other thing to do is to check for errors when compiling all the necessary files. If there were no errors at compile time, all the necessary files should be inside

```
/sample/ers7/BallTrackingHead7/MS/
```

and there should be 5 .BIN files inside

```
/MS/OPEN-R/MW/OBJS/
```

Now, all we have to do is to copy two folders to a blank SONY Open-R Memory Stick: first, we copy the folder

```
/usr/local/OPEN-R.SDK/OPEN-R/MS_ERS7/BASIC/memprot/OPEN-R
```

and then the folder

```
/sample/ers7/BallTrackingHead7/MS/OPEN-R/
```

Once that is accomplished¹, we just need to insert the memory stick in the AIBO and boot it.

AIBO will start looking for the ball, and when it finds one, it plays a sound file. When it loses track of the ball, it will play another sound file, different of that used when it finds a ball, so we know what's happening just by hearing the sound it plays.

¹NEVER delete the file MEMSTICK.IND inside the SONY Open-R memory stick, just work with the OPEN-R folder inside the memory stick

1.2 BlinkingLED7

In this sample, AIBO will turn on its back and head leds (while blinking and changing the leds colors), and move its ears.

The procedure to follow is simpler to the previous one:

1. `$ cd /sample/ers7/BlinkingLED7/`
2. `$ make`
3. `$ make install`

After checking that no compile erros occurred, we just have to copy the following two folders to a blank SONY Open-R Memory Stick:

```
/usr/local/OPEN_R_SDK/OPEN_R/MS_ERS7/BASIC/memprot/OPEN-R  
/sample/ers7/BlinkingLED7/MS/OPEN-R/
```

Finally, we only have to insert the memory stick in the AIBO and boot it.

1.3 MovingHead7

In this sample, AIBO will perform the same movements as in the previous sample while moving its head in the horizontal plane, from -93° to 93° . To put this sample working, we just have to follow the same steps as we did in the BallTrackingHead7 sample (see section 1.1).

1.4 MovingLegs7

In this sample, AIBO will do the same tricks as in the previous sample (blinking its leds, moving its ears and head), and also slightly moving its legs. Again, the procedure to follow is identical as taken in the BallTrackingHead7 sample (see section 1.1).

1.5 SensorObserver7

Let's now get to more serious business. In this sample, the wireless console will show the current values of the sensors and the joints of the AIBO (see Figure 1, Figure 2 and Figure 3). If we move any joint or touch any sensor of AIBO, the wireless console will update the joints and sensors values and display them in the wireless console, after we press the return key (Enter key). The procedure to test this sample is quite easy:

1. `$ cd /sample/ers7/SensorObserver/SensorObserver/`
2. `$ make`

ERS-7 nunData 34 frameNumber 1017		
ACC X	[27]	val -350237 175118 -525356 -525356 sig 491 488 492 492
ACC Y	[26]	val 251452 251452 251452 251452 sig 468 468 468 468
ACC Z	[28]	val -9637569 -9468489 -9468489 -9806650 sig 433 434 434 432
BODY PSD	[29]	val 107344 107627 107344 107344 sig 556 555 556 556
WLAN SW	[30]	val 1 1 1 1 sig 18 18 18 18
BACK SW F	[33]	val 0 0 0 0 sig 0 0 0 0
BACK SW M	[32]	val 0 0 0 0 sig 0 0 0 0
BACK SW R	[31]	val 0 0 0 0 sig 0 0 0 0
HEAD SENSOR	[3]	val 0 0 0 0 sig 0 0 0 0
CHIN SW	[1]	val 0 0 0 0 sig 161 161 161 161
PSD NEAR	[4]	val 500000 500000 500000 500000 sig 355 355 355 355
PSD FAR	[5]	val 0 0 0 0 sig 0 0 0 0
HEAD TILT1	[7]	val 52359 52359 45377 41887 sig 662 664 660 659 pwm 0 0 0 0 refval -757010 -757010 -757010 -757010 refsig 512 512 512 512

Figure 1: Joints and Sensors indexes and values (part 1 of 3).

3. \$ make install
4. \$ cd ..
5. \$ make
6. \$ make install
7. Copy the folder <MS_ERS7>²/WCONSOLE/memprot/OPEN-R/ and the folder /sample/ers7/SensorObserver/MS/OPEN-R/ to a blank SONY Open-R memory stick
8. Edit the file WLANDFLT.txt in the folder /OPEN-R/SYSTEM/CONF/ that is on the memory stick and change it accordingly to your network environment. It should look something like this:

```

HOSTNAME=AIBO1
ETHER_IP=192.168.102.235
ETHER_NETMASK=255.255.255.0
IP_GATEWAY=192.168.102.37
ESSID=AIBONET
WEPENABLE=1
WEPKEY=SUPER

```

²from this point forward in the text, <MS_ERS7> should mean /usr/local/OPEN_R.SDK/OPEN-R/MS-ERS7

HEAD PAN	[6]	val	3655 18278 3655 3655
		sig	504 508 504 504
		pwm	0 0 0 0
		refval	32901 32901 32901 32901
		refsig	512 512 512 512
HEAD TILT2	[2]	val	-118432 -124666 -118432 -124666
		sig	496 495 496 495
		pwm	0 0 0 0
		refval	-18699 -18699 -18699 -18699
		refsig	512 512 512 512
MOUTH	[0]	val	-201656 -201656 -206137 -201656
		sig	471 471 470 471
		pwm	0 0 0 0
		refval	-17925 -17925 -17925 -17925
		refsig	512 512 512 512
RFLEG J1	[19]	val	826121 826121 814486 814486
		sig	417 417 419 419
		pwm	0 0 0 0
		refval	273434 273434 273434 273434
		refsig	512 512 512 512
RFLEG J2	[18]	val	0 0 0 0
		sig	512 512 512 512
		pwm	0 0 0 0
		refval	0 0 0 0
		refsig	512 512 512 512
RFLEG J3	[17]	val	1838242 1838242 1838242 1838242
		sig	365 365 365 365
		pwm	0 0 0 0
		refval	1008183 1008183 1008183 1008183
		refsig	512 512 512 512
RFLEG SW	[16]	val	0 0 0 0
		sig	10 10 10 10
LFLEG J1	[11]	val	835790 841936 835790 848081
		sig	607 608 607 609
		pwm	0 0 0 0
		refval	251966 251966 251966 251966
		refsig	512 512 512 512

Figure 2: Joints and Sensors indexes and values (part 2 of 3).

```

APMODE=2
CHANNEL=3
DNS_SERVER_1=193.136.28.138
DNS_DEFDNAME=fe.up.pt
USE_DHCP=1

```

9. Place the memory stick in the AIBO and boot it
10. Open a wireless console³ and give the following command:

```
telnet 192.168.102.235 59000
```

to establish a connection with AIBO at the 59000 port. After a few seconds, it should appear the values of the sensors and joints, like in Figures 1, 2 and 3.

1.6 PIDControl7

This next sample has the same procedure than the previous sample, and allows us to show and test the PID (Proportional Integral Derivative) Control. In the wireless console is shown the gains (p, i, d, and desired degree) and the index that we want to choose for the joints in the AIBO's legs, and as a result of that, the

³In this text, it's used the Cygwin wireless console

```

sig      178 177 178 177
pwm      0 0 0 0
refval   -227131 -227131 -227131 -227131
refsig   512 512 512 512
-----
RRLEG J2 [22] val   182249 182249 182249 182249
          sig     541 541 541 541
          pwm     0 0 0 0
          refval  17085 17085 17085 17085
          refsig  512 512 512 512
-----
RRLEG J3 [21] val   2076578 2076578 2076578 2076578
          sig     325 325 325 325
          pwm     0 0 0 0
          refval  1029452 1029452 1029452 1029452
          refsig  512 512 512 512
-----
RRLEG SW [20] val   0 0 0 0
          sig     10 10 10 10
-----
LRLEG J1 [15] val   -2163169 -2168684 -2163169 -2152139
          sig     879 880 879 877
          pwm     0 0 0 0
          refval  -72220 -72220 -72220 -72220
          refsig  512 512 512 512
-----
LRLEG J2 [14] val   170858 170858 170858 170858
          sig     545 545 545 545
          pwm     0 0 0 0
          refval  -17073 -17073 -17073 -17073
          refsig  512 512 512 512
-----
LRLEG J3 [13] val   2194446 2194446 2194446 2194446
          sig     301 301 301 301
          pwm     0 0 0 0
          refval  1027517 1027517 1027517 1027517
          refsig  512 512 512 512
-----
LRLEG SW [12] val   1 1 1 1
          sig     10 10 10 10
-----
TAIL TILT [24] val   133125 133125 127337 133125
          sig     488 488 489 488
          pwm     0 0 0 0
          refval  0 0 0 0
          refsig  512 512 512 512
-----
TAIL PAN [25] val   -744393 -782243 -757009 -757009
          sig     571 574 572 572
          pwm     0 0 0 0
          refval  0 0 0 0
          refsig  512 512 512 512
-----
Hit return key>

```

Figure 3: Joints and Sensors indexes and values (part 3 of 3).

AIBO will create a text file containing the values necessary to put in motion the desired movement, and will save it in a file in the folder /OPEN-R/MW/DATA/P of the memory stick (see Figure 4).

1.7 LMasterRSlave7

See pages 13-15 of [1].


```

PIDControl7 continue? (y/n) > y
[ 0] RFLEG_J1
[ 1] RFLEG_J2
[ 2] RFLEG_J3
[ 3] LFLEG_J1
[ 4] LFLEG_J2
[ 5] LFLEG_J3
[ 6] RRLEG_J1
[ 7] RRLEG_J2
[ 8] RRLEG_J3
[ 9] LRLEG_J1
[10] LRLEG_J2
[11] LRLEG_J3
jointIndex > 2
2
pGain > 0
0
iGain > 0
0
dGain > 0
0
desiredValue [deg] > 30
30
filename > hohoho.txt
hohoho.txt
PIDControl7 continue? (y/n) > n
n
Connection closed by foreign host.

```

Figure 4: The PIDControl7 sample interface.

2 Common samples for ERS-7 and ERS-200 series

2.1 BattChecker

For this sample, we need to take a few more steps, that are explained in the README file in this sample folder. This sample shows the level of the AIBO's battery using three different methods: with the help of the wireless console, a Win32 Window, and with a MFC Windows Application with shared memory.

So, we have three different ways of getting the level of AIBO's battery:

1. For the first one, we only need to use the wireless console. It's very simple, and all we have to do is to follow the instructions in the README file in the sample folder (see Figure 5). Note: the OPEN-R folder that is necessary to copy to a blank Sony Open-R memory stick is the folder
`<MS_ERS7>/WCONSOLE/nomemprot/OPEN-R/`
2. For the second method, we need to follow the README instructions and also to create the file WLANCONF.txt with the same contents as the file WLANDFLT.txt and in the same folder as the WLANDFLT.txt, in the memory stick, and to change the local file HOSTGW.CFG where it has "10.0.1.100" to the AIBO's IP address. After that, it's just a matter of following the instructions in the README file. The result is a pop-up window that shows the same information as in the previous method.
3. Finally, for the last method of viewing the battery level, all we have to do is to change the local file HOSTGW.CFG in
`/sample/common/BattChecker/RP/host/MFC/MS/OPEN-R/MW/CONF`
 where it has "10.0.1.100" to the AIBO's IP address. If everything went ok,

there should appear a window showing AIBO's battery level like in Figure 6.



Figure 5: The BattChecker wireless console interface.

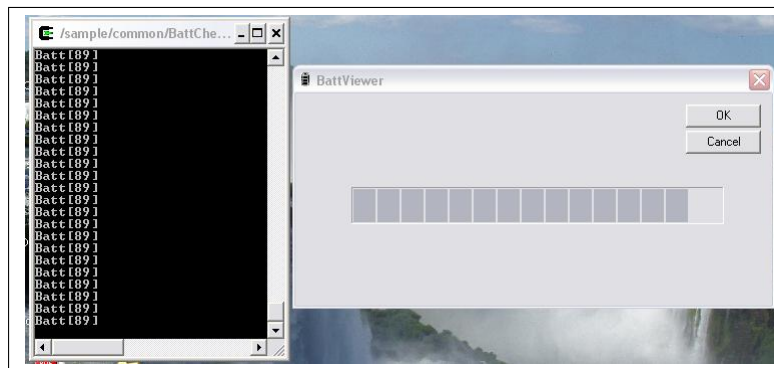


Figure 6: The BattChecker MFC window.

2.2 Crash

In this sample, we can see what happens to AIBO when, for some reason, it crashes. This sample forces AIBO to crash in a variety of different situations and contexts (division by zero, stack destroyed, jump to broken text, etc. ...). While testing the sample, and choosing the error you want to make AIBO to commit, we will hear a sound file indicating a crash occurred, and that AIBO will shut down (see Figure 7). After that, we can see in `/OPEN-R/EMON.LOG` what was the error and the running object that caused the crash. We can also use the script `emonLogParser` (in the sample folder) to obtain more data about the crash that occurred (see Chapter 5 of [2]).

```

50 ovirtualRobotCom 0x80291200 0x8000003a
51 ovirtualRobotAud 0x802957c0 0x8000003b
52 IPStack 0x80295500 0x8000003c
53 OrinocoDriver 0x802953a0 0x8000003d
54 (Handler) 0x80296500
55 OrinocoEnabler 0x80295240 0x8000003e
56 hookConsoleIO 0x802950e0 0x8000003f
57 emergencyMonitor 0x80294f80 0x80000040
58 netconf 0x80294e20 0x80000041
59 anttcpio 0x80294cc0 0x80000042
60 hookConsoleIOAct 0x80294b60 0x80000043
61 crash 0x802963a0 0x80000044

=== Crash Menu ===
0. access null data pointer
1. access null text pointer
2. destroy stack
3. cause address miss alignment
4. use unusable coprocessor
5. jump to broken text
6. cause ILB modification error (memprot only)
7. overflow on convert
8. use not-a-number source
9. use denormalized number source

select: 0

```

Figure 7: The Crash wireless console interface.

2.3 DNSLookUp

This sample attempts to get the hostname and the IP Address of a computer inside the network where AIBO is. The wireless console prompts for the name of a computer in the network, and AIBO will give the complete hostname inside the network, and the computer's IP Address (see Figure 8). The procedure to compile all the sample's files is the same as in the BallTrackingHead7 sample (see section 1.1).

```

46 oserviceManager 0x80290240 0x80000036
47 ovirtualRobot 0x802900e0 0x80000037
48 odesignedRobot 0x802914c0 0x80000038
49 osystemLogger 0x80291360 0x80000039
50 ovirtualRobotCom 0x80291200 0x8000003a
51 ovirtualRobotAud 0x802957c0 0x8000003b
52 IPStack 0x80295500 0x8000003c
53 OrinocoDriver 0x802953a0 0x8000003d
54 (Handler) 0x80296500
55 OrinocoEnabler 0x80295240 0x8000003e
56 hookConsoleIO 0x802950e0 0x8000003f
57 emergencyMonitor 0x80294f80 0x80000040
58 netconf 0x80294e20 0x80000041
59 anttcpio 0x80294cc0 0x80000042
60 hookConsoleIOAct 0x80294b60 0x80000043
61 powerMonitor 0x802963a0 0x80000044
62 dnslookUp 0x80296240 0x80000045

Default Server : ns1.fe.up.pt
Address        : 193.136.20.10
Default domain : fe.up.pt
DNSlookUp> mitchell
mitchell
Name   : mitchell.fe.up.pt
Address : 192.168.102.169
DNSlookUp>

```

Figure 8: The DNSLookUp wireless console interface.

2.4 EchoClient

In this sample, AIBO is acting as an echo client and a remote computer acts like an echo server. It's very simple: AIBO sends a test message and the remote computer receives it and echoes it back to AIBO. The result can be seen in Figure 9. The procedure to compile and run all the necessary files is the same as in the

BallTrackingHead7 sample (see section 1.1), with two exceptions: first, we need to change the file `EchoClientConfig.h` to include the IP of the remote computer that will act as an echo server (the variable to change is `ECHOSERVER_IP`), and second, we need to compile and execute the `echo_server.exe` in the `echo_server` folder before doing `telnet` to AIBO.

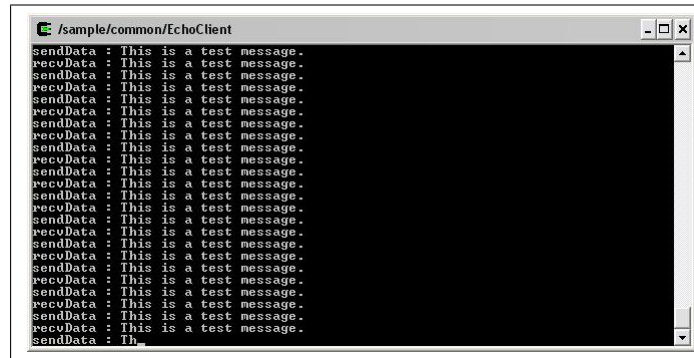


Figure 9: The EchoClient wireless console interface.

2.5 EchoServer

In this sample, the roles switch comparing to the previous sample: the AIBO acts like an echo server, and a remote computer acts like an echo client. The procedure for compiling and running this sample is the same as for the BallTrackingHead7 sample (see section 1.1). The `echo_client.exe` takes as arguments the IP of AIBO. After that, the wireless console takes the string that we want to send to AIBO, and after pressing the ENTER button, AIBO gets the string and echoes back to our remote computer (see Figure 10).

2.6 ERA201D1Info

This sample shows a set of information related to AIBO's MAC Address, Ether Statistics, WLAN Settings, WLAN Statistics and IP Address (see Figure 11). It's a very simple sample to test and we just have to follow the same procedure as the BallTrackingHead7 sample (see section 1.1).

2.7 HelloWorld

See pages 8-11 of [1].

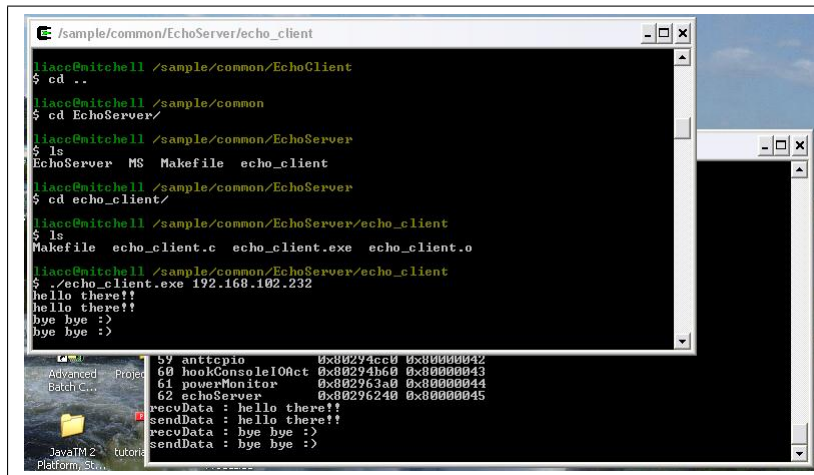


Figure 10: The EchoServer wireless console interface.

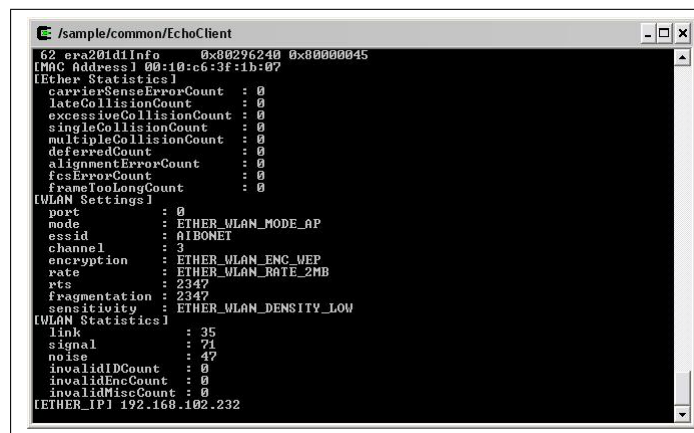


Figure 11: The ERA201D1Info wireless console interface.

2.8 HelloWorld-stubgen

This samples provides the same results as in the previous sample, but it does that in two different ways: using the StubGenerator (see section 3.2 of [2]) and using remote processing. The procedure to follow is the same as in the BallTrackingHead7 sample if we want to take advantage of the StubGenerator, and is identical to the ObjectComm sample if we want to use remote processing (see chapter 4 of [1]). Either the way we choose to run this sample, the results can be seen in Figure 12.

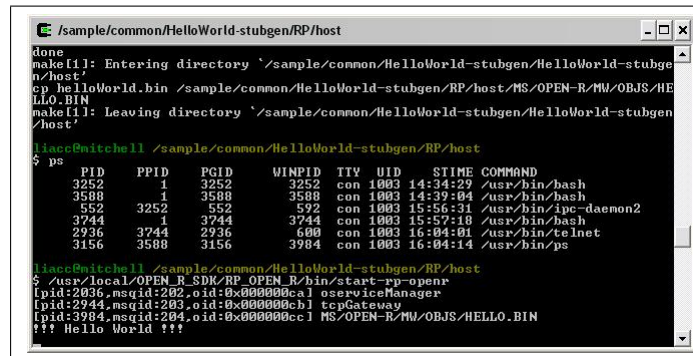


Figure 12: The HelloWorld wireless console interface.

2.9 ImageCapture

This sample takes pictures with the AIBO's camera and stores them as an .BMP file in the AIBO's /OPEN-R/MW/DATA/P/ folder (see Figure 13 and Figure 14). We can access to the folder via FTP (see TinyFTPD sample) so we don't have to take the memory stick out of AIBO everytime we want to see the pictures we took. Whenever we are ready to take a new picture, we just have to touch AIBO's back sensors and to access the folder were AIBO stores them to see the pictures. The procedure to follow is the same as in the BallTrackingHead7 sample (see section 1.1)⁴.

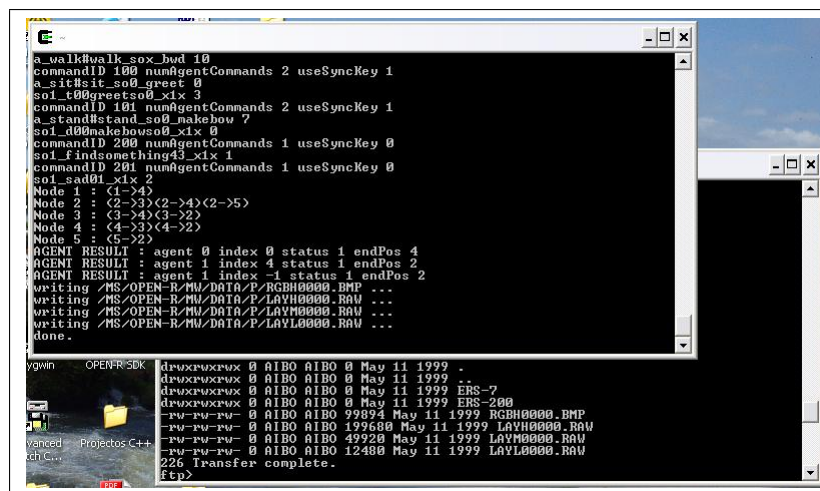


Figure 13: The ImageCapture sample.

⁴In this sample, as AIBO boots it stands up and moves his legs a little bit before staring at whatever you want to take a picture. So, be carefull not to put AIBO in a place where it can fall down.



Figure 14: A picture taken with ImageCapture sample.

2.10 ImageObserver

This sample does something similar to the previous sample, but it shows more data to the wireless console, like the frame number, color frequency, etc. It also saves several layers of a picture as .BMP files, so we can see all the layers that compound one particular picture that AIBO took. As in the previous sample, we can access all pictures via FTP by knowing AIBO's IP Address (see Figure 15).

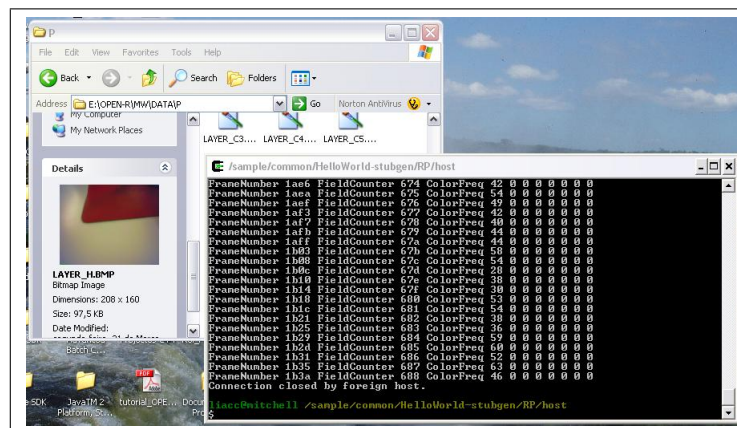
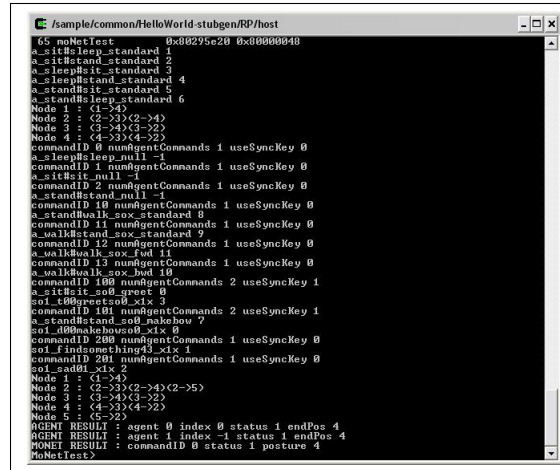


Figure 15: The ImageObserver wireless console interface.

2.11 MoNet

This sample shows some movements AIBO can do by providing a number in the wireless console interface for this sample (see Figure 16). After giving a command for a specific action, we can observe which agents were responsible for the action chosen, and the current status of the sample. The procedure to compile and run the necessary files is the same as the usual, but we also need to compile all files in the folder `MoNetTest` prior to compiling all files in the folder

MoNet⁵.



```

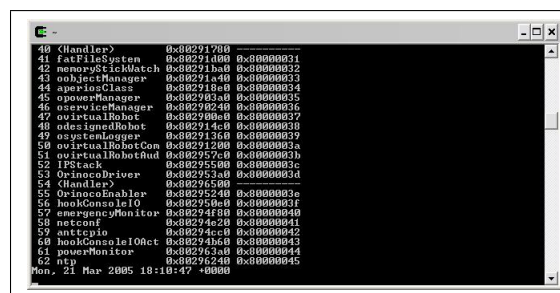
C:\sample\common\HelloWorld-stubgen\RP\host
65 moNetTest 0x80295e20 0x00000040
a_sit$leep_standard 1
a_sit$stand_standard 2
a_sit$leep$it_standard 3
a_sit$leep$it_standard 4
a_sit$leep$it_standard 5
a_sit$leep$it_standard 6
Node 1 : (1->)
Node 2 : (2->)(2->4)
Node 3 : (3->)(3->2)
Node 4 : (4->)(4->)
commandID 0 numAgentCommands 1 useSyncKey 0
a_sit$leep$leep_null -1
commandID 1 numAgentCommands 1 useSyncKey 0
a_sit$ic_null -1
commandID 2 numAgentCommands 1 useSyncKey 0
a_sit$stand_stand_null -1
commandID 10 numAgentCommands 1 useSyncKey 0
a_sit$stand_walk_sox_standard 0
commandID 11 numAgentCommands 1 useSyncKey 0
a_sit$stand_walk_sox_standard 9
commandID 12 numAgentCommands 1 useSyncKey 0
a_sit$walk$sox$nd 11
commandID 13 numAgentCommands 1 useSyncKey 0
a_sit$walk$sox$nd 10
commandID 100 numAgentCommands 2 useSyncKey 1
a_sit$it_sob_greet 0
soi_sad$sox$ix 3
commandID 101 numAgentCommands 2 useSyncKey 1
a_sit$stand_sob_makehov 7
soi_sad$sox$ix 0
commandID 200 numAgentCommands 1 useSyncKey 0
soi_sad$sox$ix 1
commandID 201 numAgentCommands 1 useSyncKey 0
soi_sad$ix 2
Node 1 : (1->)
Node 2 : (2->)(2->4)(2->5)
Node 3 : (3->)(3->2)
Node 4 : (4->)(4->)
Node 5 : (5->)
AGENT RESULT : agent 0 index 0 status 1 endPos 4
AGENT RESULT : agent 1 index -1 status 1 endPos 4
MONET RESULT : commandID 0 status 1 posture 4
MoNetTest>

```

Figure 16: The MoNet wireless console interface.

2.12 NTP

In this sample, that stands for Network Time Protocol, AIBO attempts to synchronize its internal clock time with the one in a given computer. The procedure to test this sample is the same as in the BallTrackingHead7 sample, but is necessary to change the file NTP .CFG to include the IP Address of the computer that AIBO will connect and synchronize to (see Figure 17).



```

C:\
40 (Handler) 0x80291780
41 fatFileSystem 0x80291400 0x00000031
42 memorySystickWatch 0x802918a0 0x00000032
43 objectManager 0x80291a40 0x00000033
44 aperiosClass 0x802918e0 0x00000034
45 powerManager 0x802903a0 0x00000035
46 oServiceManager 0x80290240 0x00000036
47 virtualRobot 0x802940e0 0x00000037
48 odesignedRobot 0x802914e0 0x00000038
49 oSystemLogger 0x802913e0 0x00000039
50 virtualRobotCom 0x80291400 0x0000003a
51 virtualRobotAud 0x802957c0 0x0000003b
52 IPStack 0x80295500 0x0000003c
53 OrinocoDriver 0x802953a0 0x0000003d
54 (Handler) 0x80295500
55 OrinocoEnabler 0x80295440 0x0000003e
56 hookConsoleIO 0x802950e0 0x0000003f
57 emergencyMonitor 0x80294180 0x00000040
58 netconf 0x80294e20 0x00000041
59 anttcpio 0x80294cc0 0x00000042
60 hookConsoleIOact 0x80294060 0x00000043
61 powerMonitor 0x802963a0 0x00000044
62 ntp 0x80296240 0x00000045
Mon, 21 Mar 2005 10:10:47 +0000

```

Figure 17: The NTP wireless console interface.

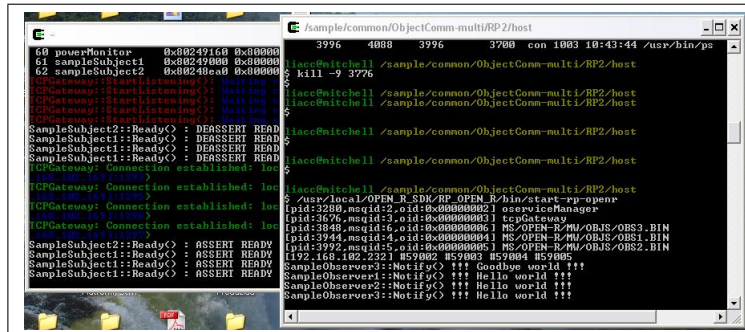
2.13 ObjectComm

See pages 12 and 13 of [1].

⁵In this sample, as AIBO boots it stretches his legs before starting the sample. So, be careful not to put AIBO in a place where it can fall down.

2.14 ObjectComm-multi

In this sample we can see again how the process of several objects communicating with themselves works, how to do this communication, and some mistakes one should avoid when programming a multi-object communication (see `ObjectComm-multi.pdf` inside this sample folder). The procedure for this sample is the same when compiling and running programs that require remote processing (see chapter 4 of [1]). As an example, in Figure 18 is shown the result of having 2 Subjects communicating with 3 Objects with several TCP connections between them (making easy the exchange of messages). This example is in the folder `RP2` of the sample folder.



```
68 powerMonitor 0x80249160 0x800000
61 sampleSubject1 0x80249000 0x800000
62 sampleSubject2 0x80248ea0 0x800000

TCPGateway: startListening(): fail log
TCPGateway: startListening(): fail log
TCPGateway: startListening(): fail log
SampleSubject2::Ready() : DESSERT READ
SampleSubject1::Ready() : DESSERT READ
SampleSubject1::Ready() : DESSERT READ
SampleSubject1::Ready() : DESSERT READ
SampleSubject1::Ready() : DESSERT READ
TCPGateway: connection established: log
TCPGateway: Connection established: log
TCPGateway: Connection established: log
TCPGateway: Connection established: log
SampleSubject2::Ready() : ASSERT READ
SampleSubject1::Ready() : ASSERT READ
SampleSubject1::Ready() : ASSERT READ
SampleSubject1::Ready() : ASSERT READ
SampleObserver3:Notif() !!! Goodbye world !!!
SampleObserver1:Notif() !!! Hello world !!!
SampleObserver2:Notif() !!! Hello world !!!
SampleObserver3:Notif() !!! Hello world !!!
```

Figure 18: Communication between 2 Subjects and 3 Objects.

2.15 PowerMonitor

This samples monitorees the level of AIBO's battery with the help of the `powerMonitor` Object. This Object is always present in every code we test on AIBO since it's essential for the task of monitoring the power level of AIBO's battery. It's the simplest sample and allows us to see how the power monitoring is done in C++ code. The procedure to follow is the same as in the `BallTrackingHead7` sample (see section 1.1).

2.16 RobotDesign

The `RobotDesign` sample shows the AIBO version that is currently being used by the program, i.e., shows on the wireless console which AIBO is being used: ERS-7, ERS-210 or ERS-220 (see Figure 19). The procedure to follow to test this sample is the same as in the `BallTrackingHead7` sample (see section 1.1).

```

43 objectManager 0x80291a40 0x80000033
44 apertisClass 0x802918e0 0x80000034
45 powerManager 0x802908a0 0x80000035
46 oserviceManager 0x80290240 0x80000036
47 ovirtualRobot 0x802900e0 0x80000037
48 odesignRobot 0x802914e0 0x80000038
49 osystemLogger 0x80291360 0x80000039
50 ovirtualRobotCom 0x80291200 0x8000003a
51 ovirtualRobotAud 0x802957c0 0x8000003b
52 IPStack 0x80295500 0x8000003c
53 OrinocoDriver 0x802953a0 0x8000003d
54 (Handler) 0x80295500 -
55 OrinocoEnabler 0x80295240 0x8000003e
56 hookConsoleIO 0x802950a0 0x8000003f
57 emergencyMonitor 0x80294f80 0x80000040
58 netconf 0x80294e20 0x80000041
59 anttcpio 0x80294cc0 0x80000042
60 hookConsoleIOct 0x80294b60 0x80000043
61 powerMonitor 0x802963a0 0x80000044
62 robotDesign 0x80296240 0x80000045
RobotDesign::DoInit()
RobotDesign::DoStart()
OPENR::GetRobotDesign() : ERS-7
on ERS-7.

```

Figure 19: The RobotDesign wireless console interface.

2.17 SoundPlay

This sample plays a sound file that is in the folder /OPEN-R/MW/DATA/P/ in the memory stick. AIBO will select the boot sound file accordingly to its version (in ERS-7, AIBO will play the sound file BOOT.WAV in /ERS-7/ inside the above folder). The procedure to follow to run and test this sample is the same as in the BallTrackingHead7 sample (see section 1.1).

2.18 SoundRec

This sample records the surrounding sounds that AIBO captures from its environment and stores it in a .WAV file (approximately 16 seconds) inside the /OPEN-R/MW/DATA/P/ folder in the memory stick (see Figure 20). Once the sample is finished we can access the WAV file by getting the file directly from the memory stick or via FTP (see TinyFTPD sample).

```

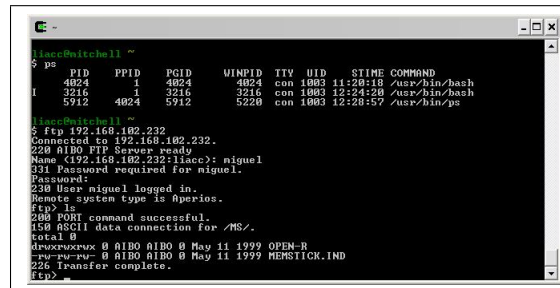
42 memoryStickWatch 0x80291ba0 0x80000032
43 objectManager 0x80291a40 0x80000033
44 apertisClass 0x802918e0 0x80000034
45 powerManager 0x802908a0 0x80000035
46 oserviceManager 0x80290240 0x80000036
47 ovirtualRobot 0x802900e0 0x80000037
48 odesignRobot 0x802914c0 0x80000038
49 osystemLogger 0x80291360 0x80000039
50 ovirtualRobotCom 0x80291200 0x8000003a
51 ovirtualRobotAud 0x802957c0 0x8000003b
52 IPStack 0x80295500 0x8000003c
53 OrinocoDriver 0x802953a0 0x8000003d
54 (Handler) 0x80295500 -
55 OrinocoEnabler 0x80295240 0x8000003e
56 hookConsoleIO 0x802950a0 0x8000003f
57 emergencyMonitor 0x80294f80 0x80000040
58 netconf 0x80294e20 0x80000041
59 anttcpio 0x80294cc0 0x80000042
60 hookConsoleIOct 0x80294b60 0x80000043
61 powerMonitor 0x802963a0 0x80000044
62 soundRec 0x80296240 0x80000045
63 tinyFTPD 0x802960e0 0x80000046
START RECORDING (about 16sec)
SAVE SOUNDREC.UAU. WAIT ... DONE
also build data connection for /RS-OPEN-R/MW/DATA/P/
total 0
drwxr-xr-x 0 AIBO AIBO 0 May 11 1999 .
drwxr-xr-x 0 AIBO AIBO 0 May 11 1999 ..
-rw-rw-rw- 0 AIBO AIBO 1048620 May 11 1999 SOUNDREC.UAU
226 Transfer complete.
ftp>

```

Figure 20: The SoundRec wireless console interface.

2.19 TinyFTPD

This sample allows us to get the files stored in AIBO's memory stick via FTP without having to shut down AIBO everytime we want to access the memory stick files. It's a very simple sample to test and run (the procedure is the same as in the BallTrackingHead7 sample), and once we run the sample we can access all AIBO's files just like in a FTP server. A prompt asks you for a username and a password to access AIBO (all username and passwords are stored in the `PASSWD` file in `/OPEN-R/MW/CONF/`, so we can change it to any users and passwords we would like⁶), and after that we have a set of commands to get files, put files, etc., that can be seen with the `help` command (see Figure 21). One extra advantage is that we can use this sample with any other sample we would like, so we don't have to take out the memory stick from AIBO whenever we want to test different samples. All we have to do is to include the `TINYFTPD.BIN` file in `/OPEN-R/MW/OBJS/` and change the file `/OPEN-R/MW/CONF/OBJECT.CFG` to include the path to the `TINYFTPD.BIN`. Once that is done, we can access the memory stick, change the object files to test other samples (making of course, the necessary changes in configuration files), and then give the command `QUOTE REBT` to reboot AIBO with the new sample we want to test. It's a very useful procedure of we want to debug and test some code and don't want to waste time taking out and putting in the memory stick in AIBO.



```
liac@hitchell ~
$ ps
  PID  PPID  PGID  WNPID  TTY  UID  STIME  COMMAND
  ---  ---  ---  ---  ---  ---  ---  ---
  4024   1  4024  4024  con  1003  11:20:18 /usr/bin/bash
  3216   1  3216  3216  con  1003  12:24:20 /usr/bin/bash
  5912  4024  5912  5220  con  1003  12:28:57 /usr/bin/ps
liac@hitchell ~
$ ftp 192.168.102.232
Connected to 192.168.102.232.
220 AIBO FTP Server ready
Name (192.168.102.232:liac): niquel
331 Password required for niquel.
Password:
230 User niquel logged in.
Remote system type is Apeiros.
ftp> ls
200 PORT command successful.
150 ASCII data connection for /MS/.
total 0
drwxrwxrwx 0 AIBO AIBO 0 May 11 1999 OPEN-R
-rw-rw-rw- 0 AIBO AIBO 0 May 11 1999 MEMSTICK.IND
226 Transfer complete.
ftp>
```

Figure 21: The TinyFTPD wireless console interface.

2.20 UDPEchoServer

This sample is very similar to the EchoServer sample, except it uses UDP protocol instead of TCP for the connections between the echo server and client. AIBO plays the role as an echo server and waits for a request to echoe a given string the client sends (see Figure 22). The procedure to test and run this sample is the same as in the BallTrackingHead7 sample, with a small exception: it's necessary to compile all files in the folder `/sample/common/UDPEchoServer/udp_echo_cli-`

⁶don't forget to give an extra blank line in the end of this file

ent and to run the `udp_echo_client` with the AIBO's IP Address as its argument.

```
~/sample/common/UDPEchoServer/udp_echo_client
$ cd /sample/common/UDPEchoServer/
~/sample/common/UDPEchoServer
$ ls
WC Makefile UDPEchoServer udp_echo_client
~/sample/common/UDPEchoServer
$ cd udp_echo_client/
~/sample/common/UDPEchoServer/udp_echo_client
$ ls
Makefile udp_echo_client.c udp_echo_client.exe udp_echo_client.o
~/sample/common/UDPEchoServer/udp_echo_client
$ ./udp_echo_client.exe 192.168.102.232
Enter message to AIBO>hello there you!
Receive from AIBO : hello there you!
Enter message to AIBO>bye
Receive from AIBO : bye :)
Enter message to AIBO>
~/sample/common/UDPEchoServer/udp_echo_client
```

Figure 22: The UDPEchoServer wireless console interface.

2.21 W3AIBO

This sample takes pictures in continuous way, and allows us to view them with a web browser and to all the layers that compound a given picture. The procedure to follow is well explained in the README file in this sample folder. The results of this sample can be seen in Figure 23.

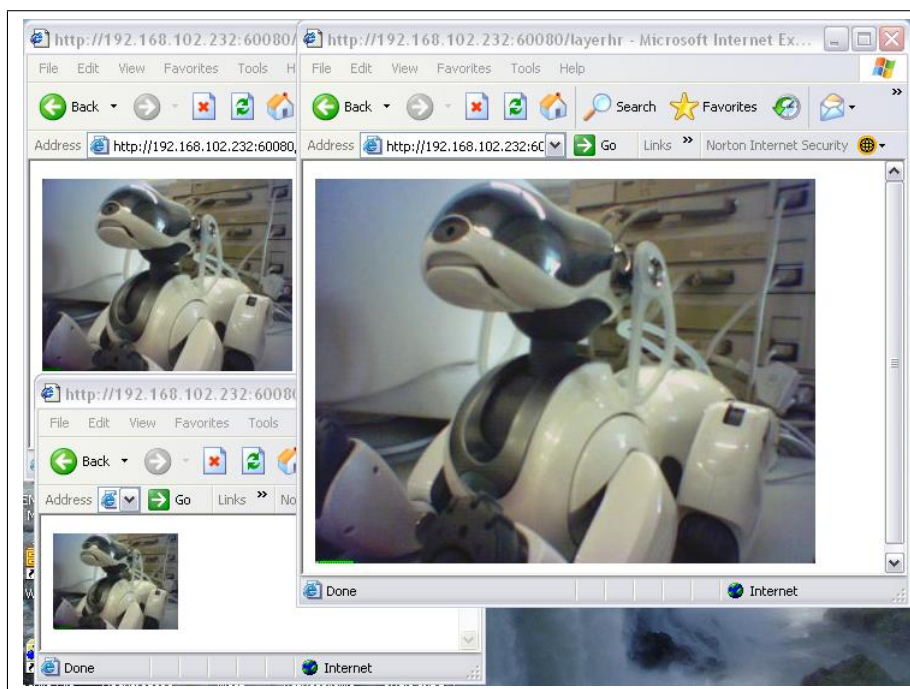


Figure 23: Accessing a picture and all its layers with a web browser.

References

- [1] Sony Corporation. *Open-R SDK Installation Guide*. Sony Corporation, 2004.
- [2] Sony Corporation. *Open-R SDK Programmer's Guide*. Sony Corporation, 2004.