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

List of Figures

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 precedure for compiling all the necessary files to test almost any Open-R code on Aibo. The other thing to do is to check for erros when compiling all the necessary files. If there were no erros 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 accomplished[1], 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 looses 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.

[1]NEVER 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^o to 93^o. 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`

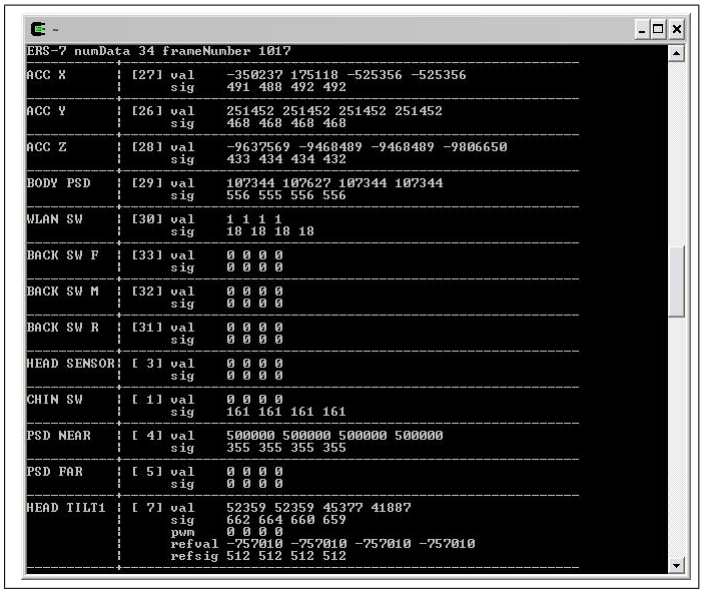

Figure 1: Joints and Sensors indexes and values (part 1 of 3).

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
```

[2]from this point forward in the text, <MS_ERS7> should mean `/usr/local/OPEN_R_SDK/OPEN-R/MS-ERS7`

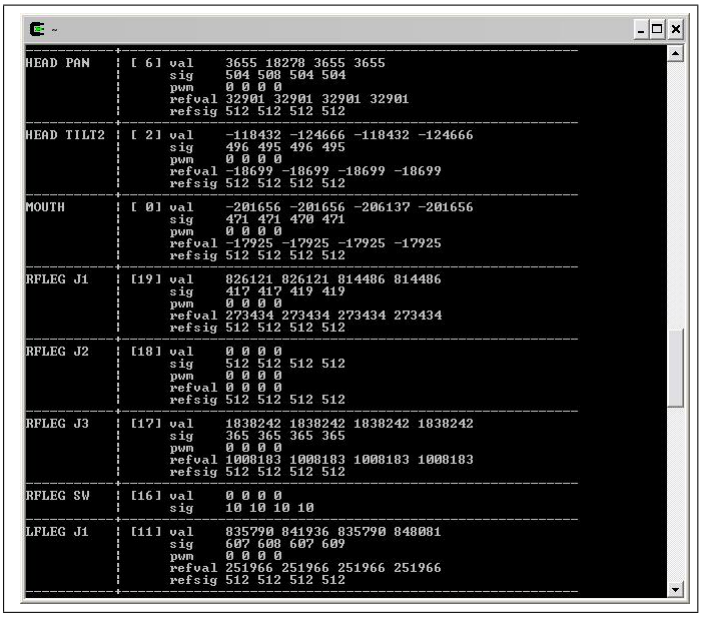

Figure 2: Joints and Sensors indexes and values (part 2 of 3).

```
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 console[3] 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

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

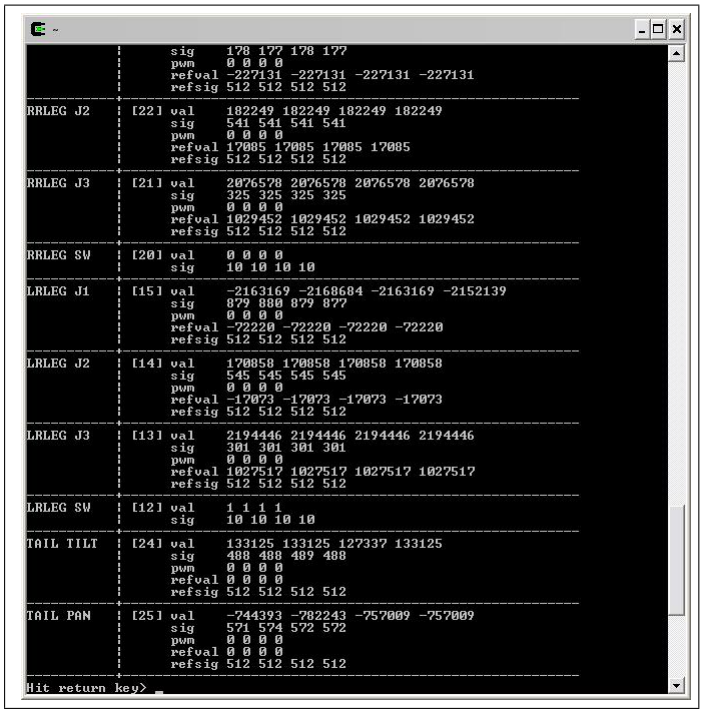

Figure 3: Joints and Sensors indexes and values (part 3 of 3).

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].

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.

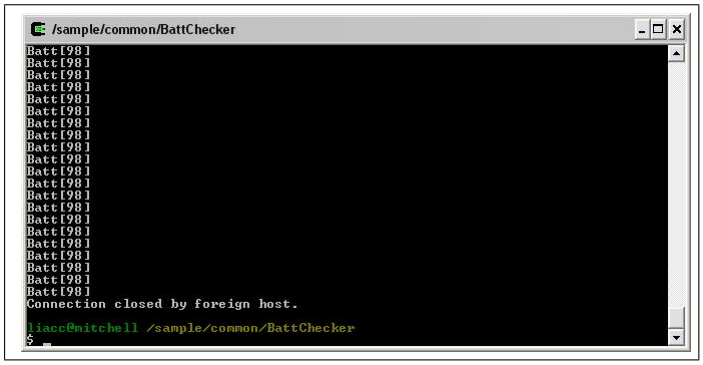

Figure 5: The BattChecker wireless console interface.

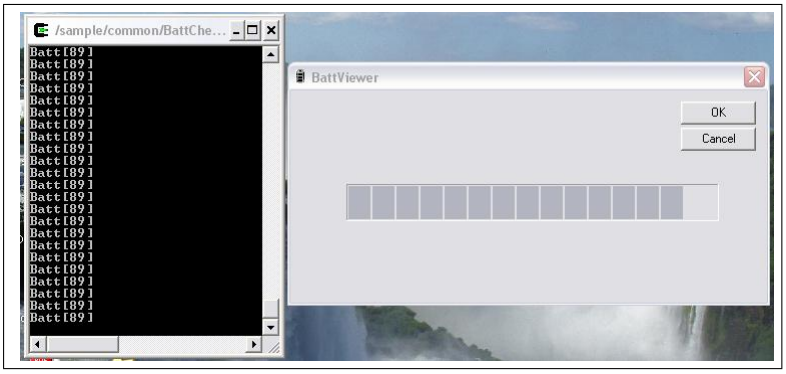

Figure 6: The BattChecker MFC window.

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]).

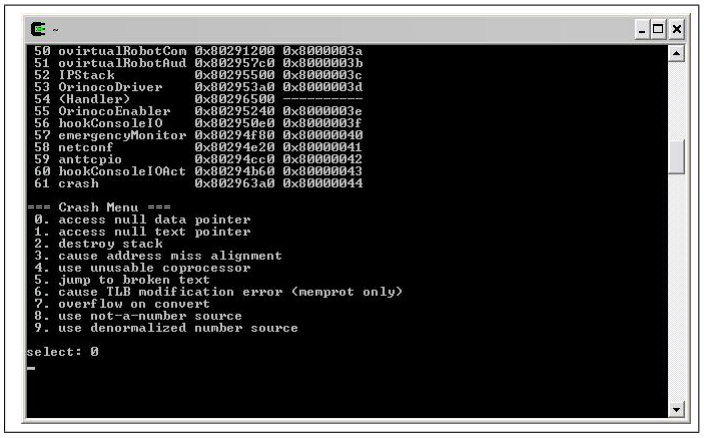

Figure 7: The Crash wireless console interface.

2.3 DNSLookUp

This sample attemps 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).

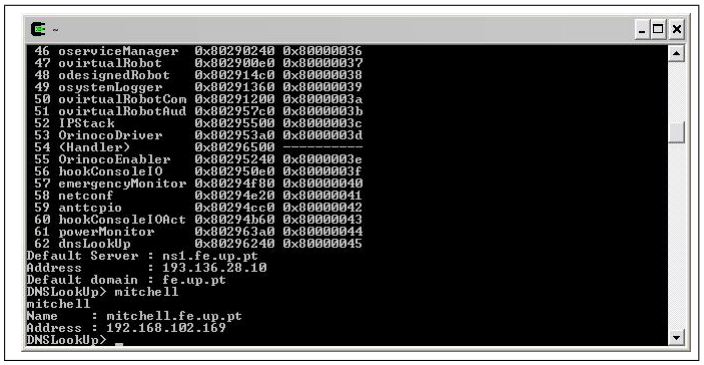

Figure 8: The DNSLookUp wireless console interface.

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.

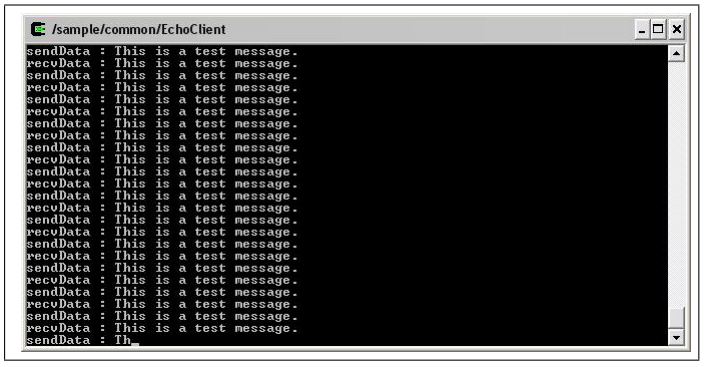

Figure 9: The EchoClient wireless console interface.

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 BallTracking-Head7 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].

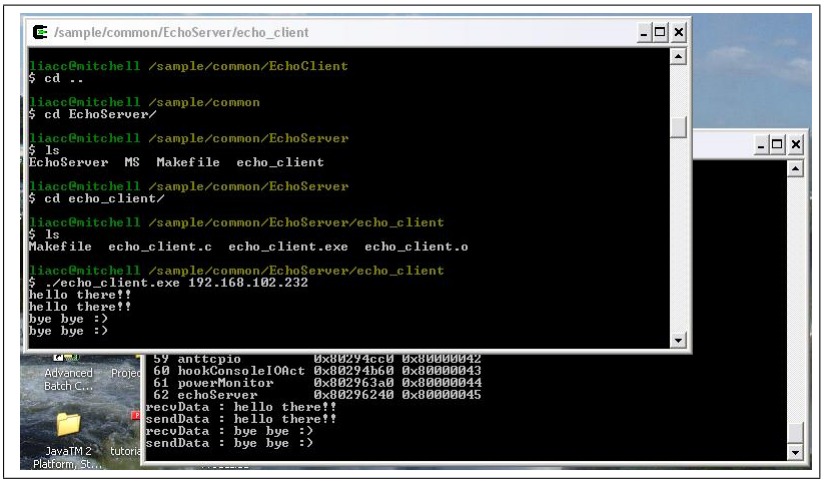

Figure 10: The EchoServer wireless console interface.

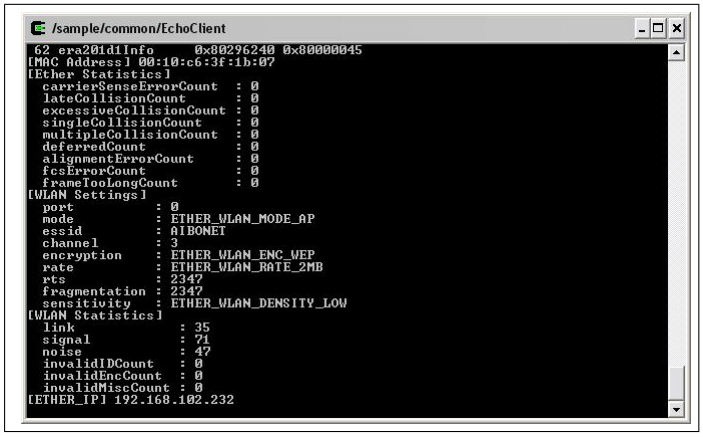

Figure 11: The ERA201D1Info wireless console interface.

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.

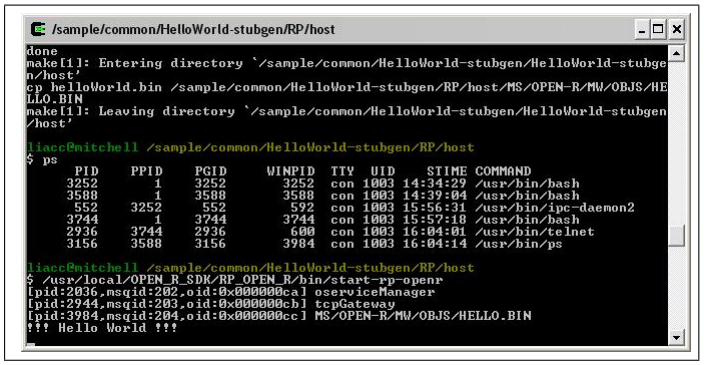

Figure 12: The HelloWorld wireless console interface.

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].

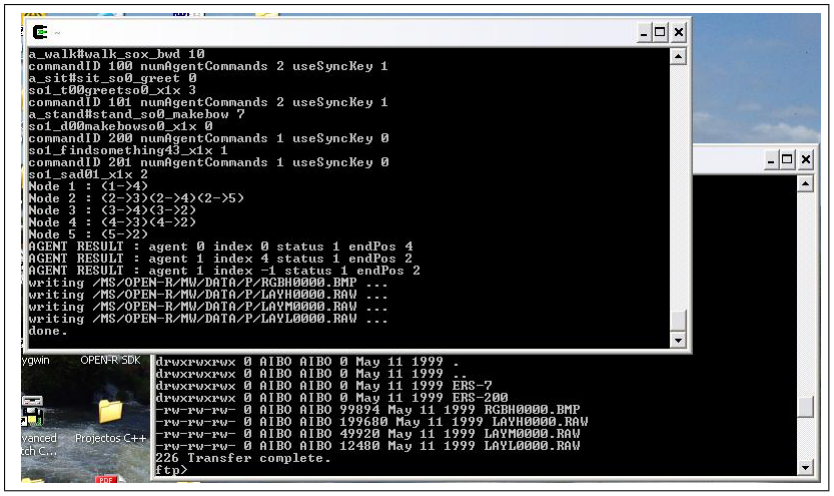

Figure 13: The ImageCapture sample.

[4]In 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 compund 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).

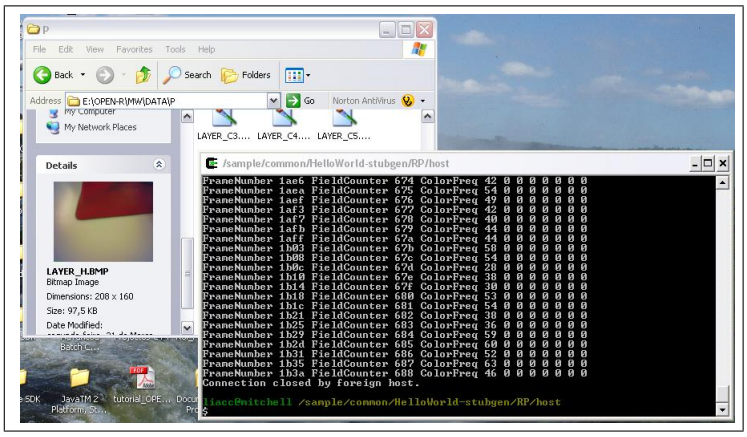

Figure 15: The ImageObserver wireless console interface.

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 choosen, 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

MoNet[5].

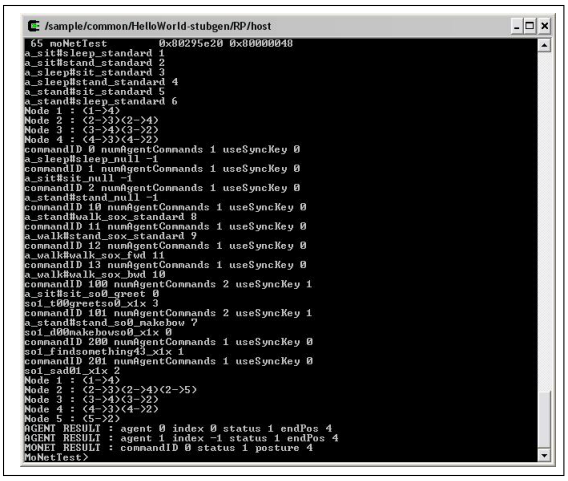

Figure 16: The MoNet wireless console interface.

2.12 NTP

In this sample, that stands for Network Time Protocol, AIBO attemps 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).

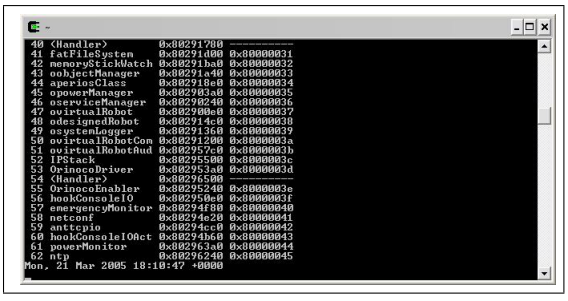

Figure 17: The NTP wireless console interface.

2.13 ObjectComm

See pages 12 and 13 of [1].

[5]In this sample, as AIBO boots it stretches his legs before starting the sample. So, be carefull 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.

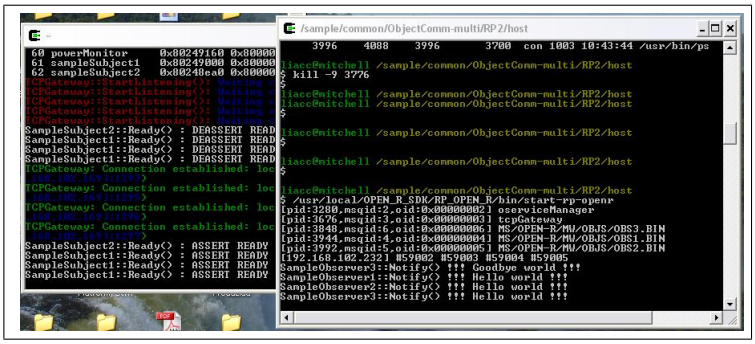

Figure 18: Communication between 2 Subjects and 3 Objects.

2.15 PowerMonitor

This samples monitores 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 essencial 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).

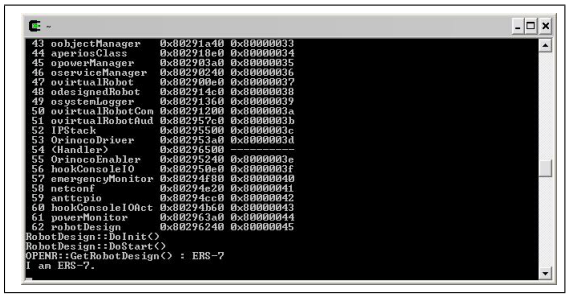

Figure 19: The RobotDesign wireless console interface.

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).

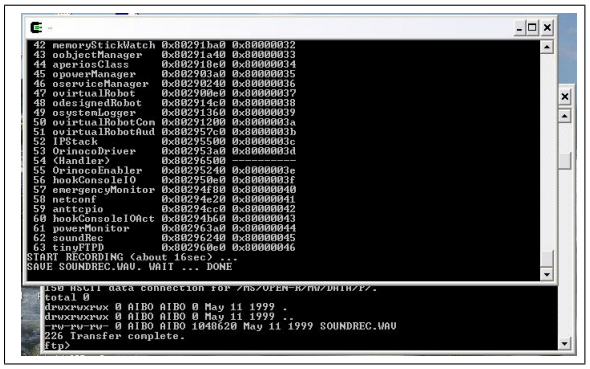

Figure 20: The SoundRec wireless console interface.

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 like[6]), 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 `TINYFTP.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.

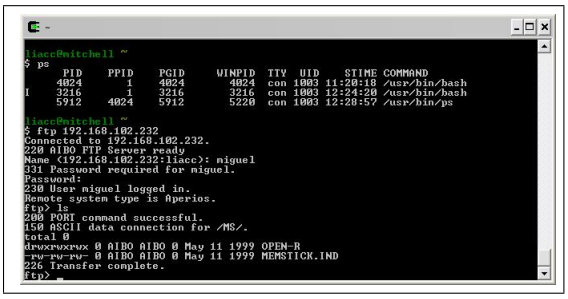

Figure 21: The TinyFTPD wireless console interface.

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-`

[6] don'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.

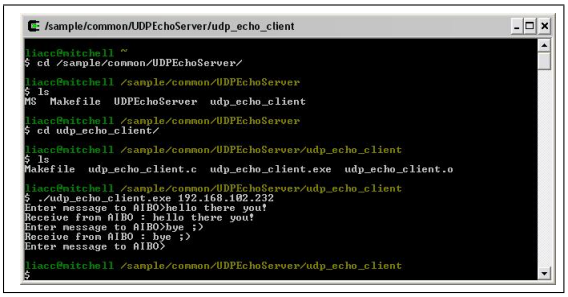

Figure 22: The UDPEchoServer wireless console interface.

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.

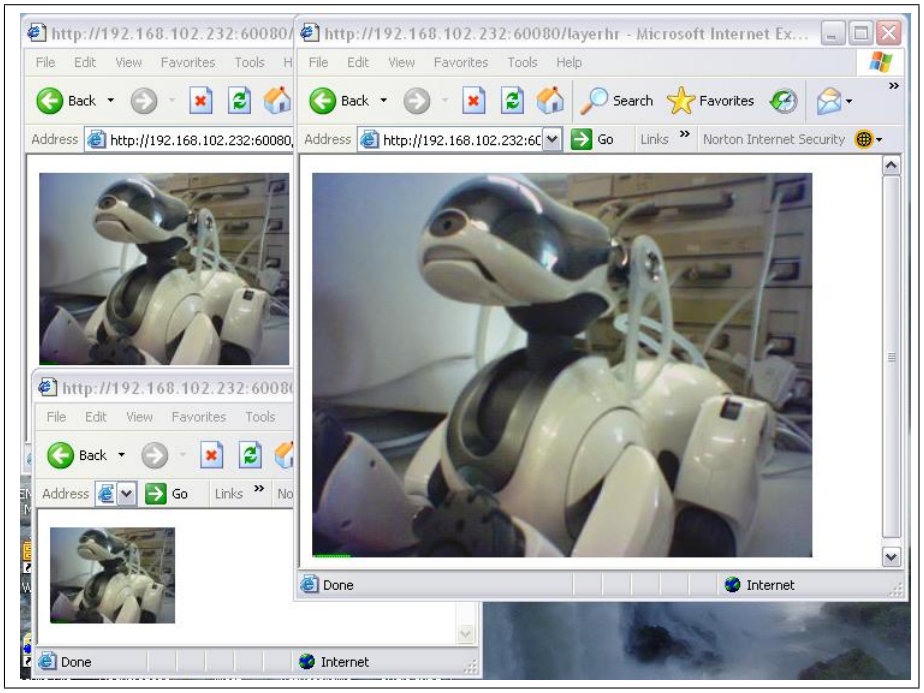

Figure 23: Accessing a picture and all its layers with a web browser.

References

[1] Sony Corporation. *Open-R SDK Installation Guide*. Sony Corporation, 2004.

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