ROBotic Open-architecture Technology for
Cognition, Understanding, and Behavior

Project No. 004370

RobotCub

Development of a Cognitive Humanoid Cub

Instrument: Integrated Project
Thematic Priority: IST – Cognitive Systems

D8.5 The iCub manual

Due Date: Month 65
Submission date: Month 65

Start date of project: **01/09/2004** Duration: **65 months**

Organisation name of lead contractor for this deliverable: DIST, University of Genova

Responsible Person: Giorgio Metta

Revision: **2.0**

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including the Commission Service)	
RE	Restricted to a group specified by the consortium (including the Commission Service)	
CO	Confidential, only for members of the consortium (including the Commission Service)	

Table of Contents

1 Introduction....................3
2 iCub manual....................3
3 Annex....................3

1 Introduction

This deliverable is an attempt to create the iCub user manual/technical manual. This has been implemented as a set of Wiki pages since the updates are still frequent. It has been decided to keep a linear structure with chapters and a limited number of hyperlinks and cross-chapter references. The aim is to provide operational guidelines to the iCub users by building a set of how-to's. The second release of this manual is now available. We have cleaned the content considerably during the last period.

2 iCub manual

The iCub manual is meant to become the main entry point for the robot documentation. It can be accessed online at: http://eris.liralab.it/wiki/Manual
Please refer to the online version for the most complete and up to date documentation of the iCub.

3 Annex

A tentative "print" on a PDF file of the Wiki is included as Annex to the present document. This has been composed manually from the existing manual which is much bigger. Unfortunately, at the moment we don't have an automatic tool to produce a printable manual from the set of Wiki pages. This problem will be addressed subsequently when the important information has been stabilized.

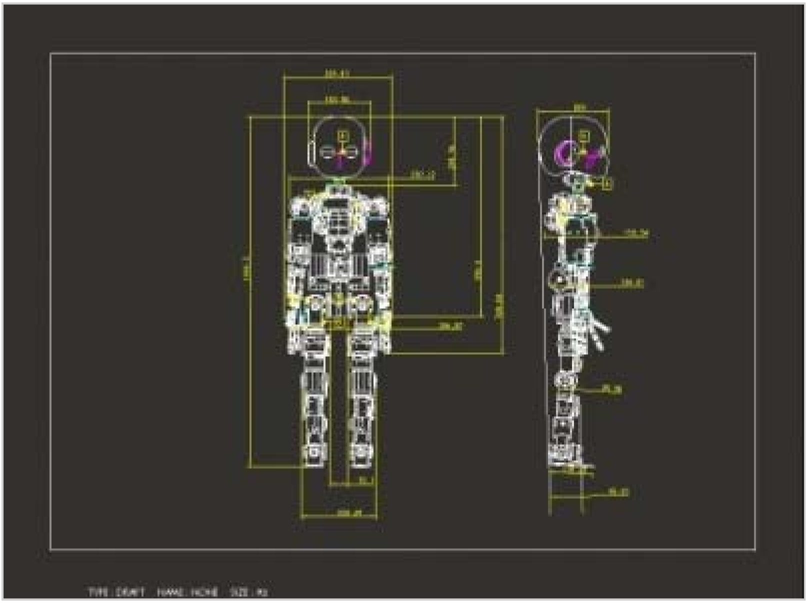
The iCub manual

Manual

From Wiki for RobotCub and Friends

Motivations: This is the first draft of the user-manual of the iCub. It includes description of the procedures to install and maintain the platform, both at the hardware and software level. The sections describing the software will also describe how to install and use what is available with the robot and provide guidelines on how to develop new capabilities and algorithms.

This document is an attempt in coalescing the robot knowledge into a **linear** document/manual. Emphasis on linear.


The iCub

How to contribute: we welcome contributions and suggestions, but please add new pages to Section 15. From time to time we will incorporate new pages into the other sections of the manual.

Contents

- 1 One. Hardware of the iCub
- 2 Two. Troubleshooting of the hardware
- 3 Three. Calibration
- 4 Four. Protocols
- 5 Five. Kinematics and Dynamics
- 6 Six. Software, Compiling YARP and iCub
- 7 Seven. Software, YARP
- 8 Eight. Software, dependencies
- 9 Nine. Software, iCub
- 10 Ten. Standardization of methods
- 11 Eleven. Guidelines
- 12 Twelve. Documentation
- 13 Thirteen. Committing changes
- 14 Fourteen. How to install the robot
- 15 Fifteeen. Unofficial documentation
- 16 Acknowledgments

One. Hardware of the iCub

To obtain the CAD and 2D drawings describing the mechanical and electronic parts which are presented in this chapter please **follow this link**. This section overlaps consistently with the Deliverable 8.1 (specifications of the iCub open system). This section of the manual is meant to be used by first opening the bill of materials, reading the component type and then consulting the corresponding description in the subsections below. For certain component, a link to the vendor website is available.

The philosophy of this chapter is to provide links and references to the technical documentation and not necessarily to substitute it.

1. Parts and specifications: see the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) extracted from the CAD
2. Brushless motors
3. DC motors
4. Controller cards
5. Motorola DSP and CodeWarrior
6. Other boards datasheets
7. Cameras
8. Microphones
9. Inertial sensing
10. CAN bus interface for debugging (ESD)
11. Quad-CAN bus interface
12. Hall-effect readings, electronics
13. Force/torque sensors, electronics
14. Face specifications and control
15. Cables
16. Power supply
17. Wiring, general diagrams and details of the connections
18. Encoder magnets
19. Springs, belts and cables
20. CPU board PC104
21. Ball bearings
22. Commercial mechanical parts
23. Commercial electronic parts
24. iCub stand
25. Other components: see the bill of materials (above)

Two. Troubleshooting of the hardware

1. Connectors
2. Cabling tools required: list as xls file (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/CablingTools.xls)
3. Tendons and replacement
4. Crimps
5. Common problems and solutions
   - Assembly instructions
   - Electrical wiring

Three. Calibration

1. DSP code: firmware versions (http://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCubPlatform/doc/manuals/RC_DIST_100_
2. Other firmware code: more firmware in repository
3. Initial calibration of the iCub, purpose
   - Legs Calibration Procedure (http://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCubPlatform/doc/manuals/RC_DIS
   - Arm Calibration Procedure (http://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCubPlatform/doc/manuals/ST2009_
   - Head Torso Calibration Procedure (http://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCubPlatform/doc/manuals/RC_DIS
4. Calibration files, details

5. Available sensors, encoders, currents

Four. Protocols

1. CAN bus protocol and messages: see here
2. YARP protocols: port protocol (http://eris.liralab.it/yarpdoc/yarp_protocol.html) , name server protocol (http://eris.liralab.it/yarpdoc/name_server.html)
3. Other protocols
4. Software and hardware tools to analyse the protocols

Five. Kinematics and Dynamics

1. iCub joints specification: naming, conventions
   - Note: the kinematic is not final yet. The robot is of course final, the documentation is to be improved here.
2. iCub kinematics ICubForwardKinematics, D-H parameters (unmodified convention):
3. Vergence, version and binocular disparity quantities for motor control Vergence, Version and Disparity.
4. iCub dynamics
5. Simulators
6. iKin (http://eris.liralab.it/iCub/dox/html/group__iKin.html) : library for forward\inverse kinematic and control tasks; some iKin-based modules (e.g. iKinArmCtrl (http://eris.liralab.it/iCub/dox/html/group__iKinArmCtrl.html)) are available as well. iKin requires IPOPT (https://projects.coin-or.org/Ipopt) . Look how to install IPOPT here: Installing IPOPT.

Six. Software, Compiling YARP and iCub

In this section we guide you through the installation process of the YARP and iCub software.

Important: the software can compile on different platform, however we support only Windows and Linux Debian/Ubuntu. On Windows we support Visual Studio (>=8.0), on Linux gcc. See Section 15 for (experimental) instructions for Mac OS X.

Follow these steps:

1. Prepare your system
   - Libraries and Development Environment: Linux
   - Libraries and Development Environment: Windows
   - Get a Subversion client
   - Client typical flags for subversion
   - Check your system (optional)
2. Getting the software
   - Getting YARP
   - Getting the iCub software
3. Setup your environment
   - Compile environment (Linux, Windows)
4. CMake utility
   - Some info about CMake and tips about CMake problems
5. How to compile everything:
   1. Compilation in Linux
   2. Compilation in Windows
6. Compile YARP and the iCub software on the pc104
   - Compilation on the pc104

Seven. Software, YARP

1. The architecture (http://eris.liralab.it/yarpdoc/what_is_yarp.html)

2. The YARP companion and YARP executables (http://eris.liralab.it/yarpdoc/yarp.html)
3. Scriptable stuff
4. CMake files, preparation (http://eris.liralab.it/yarpdoc/using_cmake.html)
5. Basic OS classes (http://eris.liralab.it/yarpdoc/yarp_os.html)
6. Basic communication classes (http://eris.liralab.it/yarpdoc/note_ports.html)
7. Advanced OS classes
8. Advanced communication classes (http://eris.liralab.it/yarpdoc/port_expert.html)
9. Device drivers, existing (http://eris.liralab.it/yarpdoc/note_devices.html)
10. Device drivers, how to write a new one (http://eris.liralab.it/yarpdoc/add_a_device.html) , tutorial on building a new device in Yarp. Important: see also how to compile the iCub devices
11. How to include a new device driver into YARP (and iCub)
12. How to add a new carrier
13. How to create a new Portable
14. YARP documentation (http://eris.liralab.it/yarpdoc/index.html) , a general tutorial, and additional Wiki pages on Yarp

Eight. Software, dependencies

Compiling YARP and iCub requires some dependencies are met.

1. Device drivers
2. Libraries, supported compilers and tools:
    - List of Dependencies
    - Installation instructions: see Section 6.1 (Prepare your system).

Nine. Software, iCub

This section explains how the software is organized. Here you will find more details about how we the repository, modules and applications.

1. The Linux on the pc104
2. iCub architecture, an introduction
    - iCub Software Architecture
    - iCub Cognitive Architecture
    - Software Implementation of the iCub Cognitive Architecture (version 1.0)
3. Automation
    - Running applications
4. Software interface: standard port names for hardware devices
5. Documentation of key modules
    1. iCubInterface2 (http://eris.liralab.it/iCub/dox/html/group__icub__iCubInterface.html) documentation and configuration file description.
    2. Framegrabber parameters: Dragonfly Parameters
    3. Running devices for the iCub
    4. What runs on the PC104 CPU
    5. ControlBoard config file, interfaces and examples: motor control in yarp (http://eris.liralab.it/yarpdoc/yarp_motor_control.html) and a simple tutorial on motor control
    6. Getting sensory data: inertia sensor, sound, images, encoders, forces, etc.
6. Debugging tools
7. Firmware: update tools and version descriptions.
8. Starting up the iCub: see here

Ten. Standardization of methods

1. Organization of the repository:
    - Organization of sources, binaries, applications and config files
    - Here is a summary of the software development guidelines of the cognitive architecture
2. Cluster, example configuration and networking

- Some old material: click here
3. Modules, standardization, configuration
    - How to write a module
    - Configuration and resource files
    - Module standards -- see http://eris.liralab.it/iCub/dox/html/module_standards.html and http://eris.liralab.it/iCub/dox/html/icub_tutorial_module.html
4. Preparing scripts for an application
5. Licensing

Eleven. Guidelines

Here we describe better practices for software development on the iCub.

1. Coding styles
2. Things to avoid
3. Naming of ports, variables, scripts, modules, etc
    - Module names in the iCub repository (and therefore module directory names as well) conform to the convention 'wordOneWordTwo'. Changing the CVS structures requires some updates (SUBDIRS, CMakeLists.txt's etc.) if they're not already that way. We moved the "qGui"s to the gui folder, naming them 'guiXy' so if a user wants to look for a gui it is easy to type gui+<tab> and all gui's show up (dropping furthermore the 'q' as it is not interesting for a user what library is used).
4. Other conventions

Twelve. Documentation

1. Compiling the documentation
2. Writing new documentation
    - About modules: http://eris.liralab.it/iCub/dox/html/module_documentation.html
    - About applications: http://eris.liralab.it/iCub/dox/html/application_documentation.html
3. Servers, online material
    - Yarp: http://eris.liralab.it/yarp
    - iCub: http://eris.liralab.it/iCub
4. Documents that aren't connected to the source code
    - Document location

Thirteen. Committing changes

1. Software
2. Hardware
3. Contributing to the Manual

Fourteen. How to install the robot

1. How to install the robot: installing_icub
2. Installation videos:
    - video_1: How to remove the robot from the box and fix it on a table: http://eris.liralab.it/misc/videos/video_1.wmv
    - video_2: How to connect the power suppliers: http://eris.liralab.it/misc/videos/video_2.wmv
    - video_3: how to set the power suppliers and how to connect the robot: http://eris.liralab.it/misc/videos/video_3.wmv

- Temporary page: upgrading repositories to svn: Upgrading the pc104 software repositories to subversion

Fifteeen. Unofficial documentation

Place here pages contributed by users.

1. Testing the robot. Getting the attention system running
2. Installation of the software on Mac OS X:
    - Libraries and Development Environment: Mac OS X
    - Compile environment (Mac OS X)
    - Compilation in Mac OS X (experimental)
3. Tweaking particular modules for MacOS X:
    - Getting the ARToolKit module to work on the Mac
4. SSH remote calls. Getting SSH to read your environment variables

Acknowledgments

List people who contributed to this manual.

1. Hardware of the iCub

Brushless motors

From Wiki for RobotCub and Friends

The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code and Data sheet	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
RBE-01210_ROTOR	RBE-01210-A rotor, Kollmorgen - Rotor for frameless and brushless motor, Tc=0.115Nm, Ic=5.41A	Kollmorgen	RBE-01210-A rotor	Danahermotion	Subassembly of RBE-01210-A	+39 0362594260	Danahermotion (http://www.danahermotion.it/dove.html) info@danahermotion.it
RBE-01210_STATOR	RBE-01210-A stator, Kollmorgen - Stator for frameless and brushless motor, Tc=0.115Nm, Ic=5.41A	Kollmorgen	RBE-01210-A stator	Danahermotion	Subassembly of RBE-01210-A	+39 0362 594260	Danahermotion (http://www.danahermotion.it/dove.html) info@danahermotion.it
RBE-01211_ROTOR	RBE-01211-A, Kollmorgen - Rotor for frameless and brushless motor, Tc=0.223 Nm, Ic=5.81 A	Kollmorgen	RBE-01211-A rotor	Danahermotion	Subassembly of RBE-01211-A	+39 0362 594260	Danahermotion (http://www.danahermotion.it/dove.html) info@danahermotion.it
RBE-01211_STATOR	RBE-01211-A, Kollmorgen - Stator for frameless and brushless motor, Tc=0.223 Nm, Ic=5.81 A	Kollmorgen	RBE-01211-A stator	Danahermotion	Subassembly of RBE-01211-A	+39 0362 594260	Danahermotion (http://www.danahermotion.it/dove.html) info@danahermotion.it
RBE513	RBE-00513-A, Kollmorgen - Frameless and brushless motor	Kollmorgen	RBE-00513-A	Danahermotion	RBE00513-A	+39 0362 594260	Danahermotion (http://www.danahermotion.it/dove.html) info@danahermotion.it

More information on motors

- The catalog with other specs of the RBE motors is available here

Gears


An example of the Brushless motors used in the iCub

The iCub mounts Harmonic Drive gears coupled with the Kollmorgen motors, both in frameless version.
The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
CSD-14-100-2A-R	CSD-14-100-2A-R, Harmonic drive - Gearbox	Harmonic drive	604808	Harmonic drive Italia S.r.l.	604808	+39 030 772 1588	Harmonic Drive (http://www.harmonicdrive.de/contenido/cms/front_content.php?idart=22&changelang=3)
CSD-17-100-2A-R	CSD-17-100-2A-R, Harmonic drive - Gearbox	Harmonic drive	405618	Harmonic drive Italia S.r.l.	405618	+39 030 772 1588	Harmonic Drive (http://www.harmonicdrive.de/contenido/cms/front_content.php?idart=22&changelang=3)

Back to the table of contents

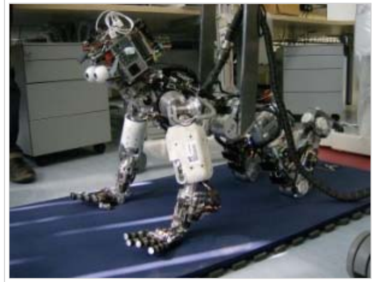
DC motors

From Wiki for RobotCub and Friends

The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities

RobotCub code	Description	Manufacturer	Manuf. code and Data sheet	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
1319T012SR_IE2-400_14-1-246	1319T012SR-IE2-400+14/1-246:1+2082, Faulhaber - Brush motor, gear box, encoder with 500 mm cable length	Faulhaber (http://www.faulhaber.com/n41659/n.html) .	1319T012SR-IE2-400+14/1-246:1+2082	Servotecnica	1319T012SR-IE2-400+14/1-246:1+2082	+39 0362 4921	Servotecnica (http://www.servotecnica.it/ingles
1319T012SR_IE2-400_14-1-66	1319T012SR-IE2-400+14/1-66:1+2082, Faulhaber - Brush motor, gear box, encoder with 500 mm cable length	Faulhaber (http://www.faulhaber.com/n41659/n.html) .	1319T012SR-IE2-400+14/1-66:1+2082	Servotecnica	1319T012SR-IE2-400+14/1-66:1+2082	+39 0362 4921	Servotecnica (http://www.servotecnica.it/ingles
1331T012SR_IE2-400_14-1	1331T012SR-IE2-400+14/1-159:1+2082, Faulhaber - Brush motor, gear box, encoder with 500 mm cable length	Faulhaber (http://www.faulhaber.com/n41659/n.html) .	1331T012SR-IE2-400+14/1-159:1+2082	Servotecnica	1331T012SR-IE2-400+14/1-159:1+2082	+39 0362 4921	Servotecnica (http://www.servotecnica.it/ingles
1717012SR_IE2-512_16-7-246	1717T012SR-IE2-512+16/7-246:1+2082, Faulhaber - Brush motor, gear box, encoder with 500 mm cable length	Faulhaber (http://www.faulhaber.com/n41659/n.html) .	1717T012SR-IE2-512+16/7-246:1+2082	Servotecnica	1717T012SR-IE2-512+16/7-246:1+2082	+39 0362 4921	Servotecnica (http://www.servotecnica.it/ingles
1724T012SR_IE2-512	1724T012SR-IE2-512+2082, Faulhaber - Brush motor, encoder with 500 mm cable length	Faulhaber (http://www.faulhaber.com/n41659/n.html) .	1724T012SR-IE2-512+2082	Servotecnica	1724T012SR-IE2-512+2082	+39 0362 4921	Servotecnica (http://www.servotecnica.it/ingles
RC_TLR_007_P_006_00	1224M012S-K380-30B20+10/1-256:1+K262, Faulhaber - Brush motor, gear box, encoder with 450 mm cable length - With modified shaft (hole)	Faulhaber (http://www.faulhaber.com/n41659/n.html) .	1224M012S-K380-30B20+10/1-256:1+K262	Servotecnica	1224M012S-K380-30B20+10/1-256:1+K262	+39 0362 4921	Servotecnica (http://www.servotecnica.it/ingles
RC_TLR_007_P_064_00	1016M012G-K380-30B19+10/1-256:1+K262, Faulhaber - Brush motor, gear box, encoder with 450 mm cable length - With modified shaft (hole)	Faulhaber (http://www.faulhaber.com/n41659/n.html) .	1016M012G-K380-30B19+10/1-256:1+K262	Servotecnica	1016M012G-K380-30B19+10/1-256:1+K262	+39 0362 4921	Servotecnica (http://www.servotecnica.it/ingles
RC_TLR_007_P_066_00	1331T012SR-IE2-400+14/1-159:1+2082, Faulhaber - Brush motor, gear box, encoder with 500 mm cable length - With modified shaft (hole)	Faulhaber (http://www.faulhaber.com/n41659/n.html) .	1331T012SR-IE2-400+14/1-159:1+2082	Servotecnica	1331T012SR-IE2-400+14/1-159:1+2082	+39 0362 4921	Servotecnica (http://www.servotecnica.it/ingles
RC_TLR_009_P_050_00	1219M012G+10/1-256:1, Faulhaber - Brush motor and gear box - With modified shaft (hole)	Faulhaber (http://www.faulhaber.com/n41659/n.html) .	1219M012G+10/1-256:1	Servotecnica	1219M012G+10/1-256:1	+39 0362 4921	Servotecnica (http://www.servotecnica.it/ingles
RC_TLR_009_P_051_00	1016M012G+10/1-256:1, Faulhaber - Brush motor and gear box - With modified shaft (hole)	Faulhaber (http://www.faulhaber.com/n41659/n.html) .	1016M012G+10/1-256:1	Servotecnica	1016M012G+10/1-256:1	+39 0362 4921	Servotecnica (http://www.servotecnica.it/ingles

RC_TLR_010_P_009_00	1016M012G-K380-30B19+10/1-1024:1+K262, Faulhaber - Brush motor, gear box, encoder with 450 mm cable length - With modified shaft (hole)	Faulhaber (http://www.faulhaber.com/n41659/n.html) .	1016M012G-K380-30B19+10/1-1024:1+K262	Servotecnica	1016M012G-K380-30B19+10/1-1024:1+K262	+39 0362 4921	Servotecnica (http://www.servotecnica.it/ingles
GPL22R-343	GPL22R 3-343:1, Gysin - Planetary gearbox	Gysin (http://www.gysin.com/en/about-us/address.html)	GPL22R 3-343:1	Servotecnica	GPL22R 3-343:1	+39 0362 4921	Servotecnica (http://www.servotecnica.it/ingles



Example of the use of DC motors on the iCub

Servo motor

There is also a Futaba servo motor that actuates the eyelids The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities

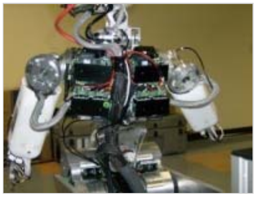
RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
FUTABA_S3111	S3111, Futaba - Micro Servo motor with J Connector	Futaba	FUTM0047 (http://www.gpdealera.com/cgi-bin/wgainf100p.pgm?I=FUTM0047)	Tower Hobbies	S3111 (http://www3.towerhobbies.com/cgi-bin/wti0001p?&I=LXLLA8)	+39 0362 4921	Tower hobbies (http://www.towerhobbies.com/)

Back to the table of contents

Controller cards

From Wiki for RobotCub and Friends

The controller cards are custom components. The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.



Example of mounting of the BLL/BLP control cards

The MCP and MC4 control pair

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
RC_DIST_001_P_202_00	BLL-001, IIT - Electronic Board with logic section for BLP-001 power driver	IIT (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bll/)	BLL-001	MicroDesign	BLL-001	+39 010 6972471	Micro design (http://www.micro-design.it/)
RC_DIST_001_P_203_00	BLP-001, IIT - Electronic Board with power driver for brushless motor, power supply 24-48V	IIT (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/blp/)	BLP-001	MicroDesign	BLP-001	+39 010 6972471	Micro design (http://www.micro-design.it/)
RC_DIST_001_P_200_00	MC4, IIT - Controller electronic board	IIT (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/mc4/)	MC4-001	MicroDesign	MC4-001	+39 010 6972471	Micro design (http://www.micro-design.it/)
RC_DIST_001_P_201_00	MCP, IIT - Power electronic board	IIT (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/mcp/)	MCP-001	MicroDesign	MCP-001	+39 010 6972471	Micro design (http://www.micro-design.it/)

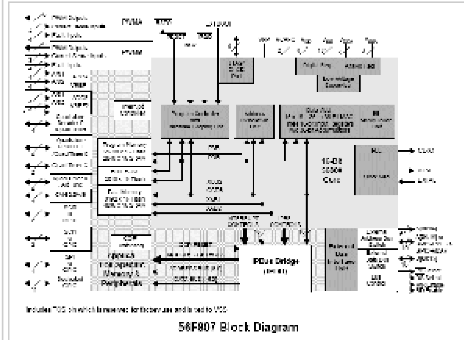
- An initial manual of the controllers is available from:iCub manuals repository (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/manuals/) in various formats (WARNING: some text is in Italian)
- The photo on the right shows the MC4 and MCP cards. In this configuration each MCP can power four MC4 which in turn can control four motors each.

Back to the table of contents

Motorola

From Wiki for RobotCub and Friends

The controller cards are built around the Freescale DSP 56F807 whose block diagram is shown on the right.



The block diagram of the DSP56F807

Additional components

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
360	TTL-232R-3V3, FTDI - Converter cable from USB B to serial, L=350mm	FTDI	TTL-232R-3V3	Farnell	1329311 (http://it.farnell.com/ftdi/ttl-232r-3v3/cavo-usb-livello-ttl-convertitore/dp/1329311?_requestid=2379181329311)	-	Farnell (http://www.farnell.com/)
361	CWH-UTP-ONCE-HE, Freescale - CodeWarrior USB TAP, Run control for freescale processors	Freescale	CWH-UTP-ONCE-HE	Mouser	841-CWH-UTP-ONCE-HE (http://it.mouser.com/Search/Refine.aspx?Keyword=841-CWH-UTP-ONCE-HE)	-	Mouser (http://it.mouser.com/contact/)

More information

- The compiler of the current firmware is the CodeWarrior compiler: follow this link (http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=CW-56800E-DSC&parentCode=null&nodeId=0127262E703BC3) for more information
- The DSP specs can be found here (http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=DSP56F807&webpageId=1143155762630726076045&nodeId=01624686365dlq62926045&fromPage=tax)
- Plenty of documentation is also available on the Freescale website. For example the data sheet can be downloaded directly from: here (http://www.freescale.com/files/dsp/doc/data_sheet/DSP56F807.pdf)

Back to the table of contents

Other boards

From Wiki for RobotCub and Friends

Each control card requires the acquisition of numerous commercial components. The bills of material are available here (http://www.robotcub.org/iCubPlatform) from the SVN repository (this file is not uploaded yet).

the MCP and MC4 control pair

Electronic boards

Several additional PCBs have been designed to fit the iCub. The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf?revision=750) for exact quantities.

More information

A collection of datasheets is available here (http://eris.liralab.it/misc/datasheets) for download.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
363	MAIS right hand, IIT - Electronic board, 32-channel miniature ADC card	IIT	MAIS_000 (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/mais/)	EES	MAIS_000	+39 0106140492	EES (http://www.ees.it/contatti.html)
369	MAIS right hand, IIT - Electronic board, 32-channel miniature ADC card	IIT	MAIS_000 (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/mais/)	EES	MAIS_000	+39 0106140492	EES (http://www.ees.it/contatti.html)
669	MAIS finger connector, IIT - Connector electronic board for Icub hands	IIT	P2007_0000_RBCS_RC_MAIS_FC	EES	P2007_0000_RBCS_RC_MAIS_FC	+39 0106140492	EES (http://www.ees.it/contatti.html)
362	STRAIN, IIT - Electronic board, 6-axial strain gauges amplification card	IIT	STRAIN_000.pdf (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/manuals/STRAIN_0000.pdf) MU2007_RBCS_RC_STRAIN_0002.pdf (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/manuals/MU2007_RBCS_RC_STRAIN_0002.pdf)	EES	STRAIN_000	+39 0106140492	EES (http://www.ees.it/contatti.html)
AEA	AEA-001, IIT - Electronic board, 12-bit digital absolute encoder with spi interface, power supply 5V/3.3V	IIT	AEA-001 (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/aea/)	EES	AEA-001	+39 0106140492	EES (http://www.ees.it/contatti.html)

Back to the table of contents

Cameras

From Wiki for RobotCub and Friends

iCub mounts two Dragonfly 2 cameras by PointGrey (http://www.ptgrey.com/products/dragonfly2/index.asp) The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
DR2-03S2C-EX-CS	DR2-03S2C-EX-CS, PointGrey - Videocam Dragonfly2 color extended version, 640x480, 1/3, *CCD*	Point Gray Research	DR2-03S2C-EX-CS	PointGrey (http://www.ptgrey.com/products/dragonfly2/index.asp) sales@ptgrey.com	DR2-03S2C-EX-CS	-	Point Gray Research (http://www.ptgrey.com/contact.asp)
348	ACC-01-2004, PointGrey - Ultra thin firewire cable, 6-6 pin, L=300mm	Point Gray Research	ACC-01-2004	PointGrey (http://www.ptgrey.com/products/dragonfly2/index.asp) sales@ptgrey.com	ACC-01-2004	-	Point Gray Research (http://www.ptgrey.com/contact.asp)
345	ACC-01-5001, PointGrey - M12 Micro Lens Holder with IR Filter support	Point Gray Research	ACC-01-5001	PointGrey (http://www.ptgrey.com/products/dragonfly2/index.asp) sales@ptgrey.com	ACC-01-5001	-	Point Gray Research (http://www.ptgrey.com/contact.asp)
347	ACC-01-4000, PointGrey - Optics Micro Lens 4MM	Point Gray Research	ACC-01-4000	PointGrey (http://www.ptgrey.com/products/dragonfly2/index.asp) sales@ptgrey.com	ACC-01-4000	-	Point Gray Research (http://www.ptgrey.com/contact.asp)
356	ACC-01-9000, PointGrey - Ribbon flat cable, 20 poles, 0.5mm	Point Gray Research	ACC-01-9000	PointGrey (http://www.ptgrey.com/products/dragonfly2/index.asp) sales@ptgrey.com	ACC-01-9000	-	Point Gray Research (http://www.ptgrey.com/contact.asp)
217	82220-CAML12, Futura elettronica - Mini lens CAML 12, Focus 2.8mm, Aperture 2.0	FuturaElettronica	82220-CAML12 (http://www.futurashop.it/allegato/8220-CAML12.asp?L2=ACCESSORI&L1=CCTV%20-%20VIDEOSORVEGLIANZA&L3=OTTICHE&cd=8220-CAML12&nVt=&d=21,00)	FuturaElettronica	82220-CAML12	+39 0331 799775	Futura elettronica (http://www.futurashop.it/index.html) futuranet@futuranet.it

More information

- The cameras are available in kit version which must be purchased at least once to get the device drivers for Windows.

the Dragonfly 2 camera

Back to the table of contents

Microphones

From Wiki for RobotCub and Friends

The iCub mounts two microphones in the ear. These are fixed to the external structure of the face. The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
MICRO_POM-2746L	POM-2746L-R, ProjectsUnlimited - Omnidirectional microphone, 2V, 500uA, 60dB, Diam.6mm H= 2.7mm	ProjectsUnlimited	POM-2746L-R	Digikey	668-1163-ND (http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=668-1163-ND)	-	Digikey (http://dkc1.digikey.com/us/en/mkt/Contact.html?WT.mc_id=hp_ContactUsButton)

More information

- More information are available from the IST report

Back to the table of contents

Inertial sensing

From Wiki for RobotCub and Friends


The MTX inertial box

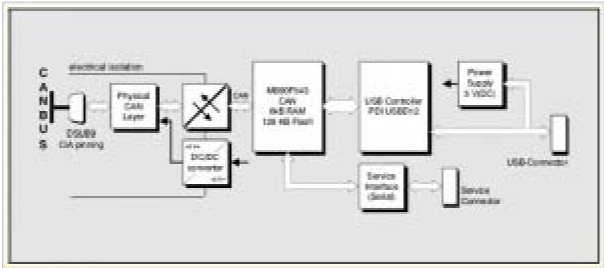
The inertial sensors of the iCub is the MTx from xsense. The cable provided by the vendor is here replaced by a custom cable.
The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
MTX	MTx-28A33G25, Xsens - Miniature 3D inertial tracker, motion control system	XSENSE (http://www.xsens.com/en/company-pages/company/distributors)	MTx	Leane NET S.r.l.	MTx-28A33G25	+39 0187 692070	Leane NET Srl (http://www.leanenet.it/contattaci.php) info.sarzana@leane.it

Back to the table of contents

CAN interface

From Wiki for RobotCub and Friends


The CAN-USB interface schematics

The CAN-USB interface

Although the iCub comes with a custom quad-can interface card, for debugging it is sometime convenient to have an extra can bus interface: Yarp has a device driver for the ESD CAN bus mini controller.

The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
91	ESD - USB to CAN Interface, from USB 1.1 to 9-pole male DSUB, supply via USB, dimension 55x55x25 mm	ESD	C.2064.02 (http://www.usbtocan.com/ProductInfo.htm)	ESD	C.2064.02	+49 511 372980	ESD Inc. (http://www.esd-electronics.com/)

More information

Information about the Yarp device driver and motor control interface can be found in the software documentation. For example see:

- Here (http://eris.liralab.it/iCub/dox/html/group__ecan.html) for the Yarp device driver
- Here (http://eris.liralab.it/yarpdoc/yarp_motor_control.html) for the motor control interface

Back to the table of contents

CFW card

From Wiki for RobotCub and Friends


The CFW-001 quad-can PC104 card

The PC104 electronics is completed by a Quad-port CAN bus interface realized specifically for iCub (called CFW-001 card) which comprises:

- Four CAN bus ports
- Two Firewire ports
- Audio amplifier

The CFW-001 design and manual are in the CVS repository.

The major components of the CFW card are:

- CAN chip: SJA 1000, (see data sheet here)
- PCI-bridge: PLX chip (see data sheet here)
- Firewire: Texas (see datasheet here)
- Amplifiers (see datasheet here). For two electret condenser microphones see here

More information

- The CFW-001 card is part of a wider set of components that provides a Pentium CPU to the iCub (embedded).
  The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	
163	SDCZ6-4096-E11, Sandisk - USB pen, 4GB	Sandisk	SDCZ6-4096-E11	Distrelec	865161 (https://www.distrelec.it/ishopWebFront/catalog/product.do /para/keywords/is/SanDisk_Cruzer_Micro_Titanium/and/language/is/it/and/shop /is/IT/and/series/is/1/and/id/is/02/and/node/is/DD-24218/and/artView/is/true /and/productNr /is/865161.html;jsessionid=F62387956B325B8D2086F1376945350F.chdist144)	-	Distrelec (
90	PicoPSU-80-WI-32, Minibox - DC-DC ATX power supply unit, 14-32V, 80W	Mini-box.com	PicoPSU-80-WI-32	Mini-box.com	PicoPSU-80-WI-32 (http://www.mini-box.com/PicoPSU-80-WI-32V)	-	/site/resellers.html;jsessionid=0a(
212	CK-PB945+, Embedded logic - Cable set for PC104 Motherboard PB945+	Embedded logic	CK-PB945+ (http://www.sisav.it /schede/datasheet /pb945plus.pdf)	Sistemi avanzati elettronici	CK-PB945+	+39 015 983206	Sistemi avanzati elettro
208	PB945+T7400, Embedded logic - PC104 Motherboard PB-945+, with Celeron Core 2Duo 2,16 Mhz	Embedded logic	PB945+T740	Sistemi avanzati elettronici	PB945+ T740 (http://www.embedded-logic.com /index.php?MENUE=Produkte&SUB=5&TYPE=PB945Plus)	+39 015 983206	Sistemi avanzati elettro

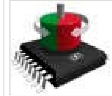
- PB945+ wiki webpage: CPU board PC104

Back to the table of contents

Hall-effect sensors

From Wiki for RobotCub and Friends

Brushless motors position sensors



Hall effect sensors in the iCub brushless motors

Hall effect sensors are used in the iCub to measure the position of those joints controlled with brushless motors.
The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
SS495A	SS495A1, Honeywell - Hall effect sensor, 8.7mA, 4.5-10.5Vcc, p.1.3mm, THD	Honeywell	SS495A1	RS	216-6247 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=216-6247&x=0&y=0)	-	RS (http://www.rs-components.com/index.html)

Hand position sensors

Hall effect sensors in the iCub hand

Hall-effect sensors are used in the iCub to measure the position of the finger joint angles. Signals are acquired by the MAIS card
The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
363	MAIS right hand, IIT - Electronic board, 32-channel miniature ADC card	IIT	MAIS_000 (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/mais/)	EES	MAIS_000	+39 0106140492	EES (http://www.ees.it/contatti.html)

369	MAIS right hand, IIT - Electronic board, 32-channel miniature ADC card	IIT	MAIS_000 (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/mais/)	EES	MAIS_000	+39 0106140492	EES (http://www.ees.it/contatti.html)
669	MAIS finger connector, IIT - Connector electronic board for Icub hands	IIT	P2007_0000_RBCS_RC_MAIS_FC	EES	P2007_0000_RBCS_RC_MAIS_FC	+39 0106140492	EES (http://www.ees.it/contatti.html)

More informations

The picture below shows the layout of the MAIS card rendered on top of the CAD of the hand.
The size and shape of the MAIS PCB has been specifically trimmed to fit the available space on the back of the hand.

Placement of the MAIS card on the hand of the iCub

Back to the table of contents

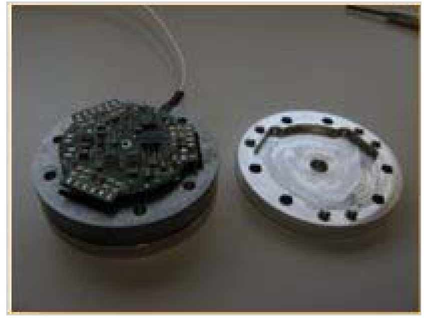
FT sensor

From Wiki for RobotCub and Friends

- The F/T sensor (6-dof) has been also design to fit the iCub. In particular, the size of the sensor has been made compatible with an existing commercial product.
  On the other hand the electronics has been made to fit the sensor itself and consequently reducing space.

- The F/T sensor is based on semiconductor-based strain gauges.

- The F/T sensor electronics is called STRAIN and it is available from the iCubPlatform CVS module. In particular see here (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/strain/)



The F/T sensor of the iCub

- More documents are available here:
  - STRAIN_000.pdf (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/manuals/STRAIN_0000.pdf)
  - MU2007_RBCS_RC_STRAIN_0002.pdf (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/manuals/MU2007_RBCS_RC_STRAIN_0002.pdf)

- There are four F/T sensors on the iCub located in the upper arms and upper legs respectively. This is an unconventional placement that requires some extra computation to map the sensor's measurement to joint level torque control.

More information

- Semiconductor stain gauges (http://www.microninstruments.com/store/ushapedgage.aspx)
- F/T sensors are mounted by Micron Instruments: (http://www.microninstruments.com)

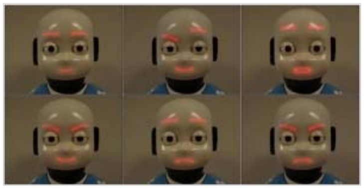
Back to the table of contents

Face specs

From Wiki for RobotCub and Friends

Details of the face design and control are described in these two documents:

- Expression control (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/manuals/RC_IST_110_D_000_07_EXPRESSIONS_CONTROL.pdf)
- Face design (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/manuals/RC_IST_111_D_000_05_FACE_EXPRESSIONS_DESIGN.pdf)



The iCub face

Electronic boards

Following the links in the table below you will find informations, schematics, and gerber files of the expression control printed circuit board and of the LED printed circuit boards.
The table reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
EXPRESSIONS_CONTROL_BOARD	EXPRESSION, IIT- Facial expression electronic control board	IIT	EXPRESSIONS_CONTROL_BOARD (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/expression/)	EES	EXPRESSIONS_CONTROL_BOARD	+39 0106140492	EES (http://www.ees.it/contatti.html)
EXPRESSIONS_L_EYEBROW_BOARD	LED LEFT EYEBROW, IIT - Facial expression electronic led board for left eyebrow	IIT	EXPRESSIONS_L_EYEBROW_BOARD (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/led/)	EES	EXPRESSIONS_L_EYEBROW_BOARD	+39 0106140492	EES (http://www.ees.it/contatti.html)
EXPRESSIONS_R_EYEBROW_BOARD	LED RIGHT EYEBROW, IIT - Facial expression electronic led board for right eyebrow	IIT	EXPRESSIONS_R_EYEBROW_BOARD (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/led/)	EES	EXPRESSIONS_R_EYEBROW_BOARD	+39 0106140492	EES (http://www.ees.it/contatti.html)
EXPRESSIONS_MOUTH_BOARDS	LED MOUTH, IIT - Facial expression electronic led board for mouth	IIT	EXPRESSIONS_MOUTH_BOARDS (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/led/)	EES	EXPRESSIONS_MOUTH_BOARDS	+39 0106140492	EES (http://www.ees.it/contatti.html)

Back to the table of contents

Cables

From Wiki for RobotCub and Friends

Special tiny wires have been used almost everywhere in the iCub. Please refer to the following table and the cabling bill of materials for further details.
The cabling bill of materials can be downloaded in Microsoft Excel format from here (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/) .

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
66	5853-1 WH005, Alpha wire - Teflon insulated cable, 1 pole, AWG26, 600V, white, UL E20042	Alpha Wire	5853-1 WH005	RS	177-0839 (http://it.rs-online.com/web/search /searchBrowseAction.html?method=searchProducts& searchTerm=1770839)	-	RS (http://www.rs-components.com /index.html)
67	5853-7 BR005, Alpha Wire - Teflon insulated cable, 1 pole, AWG26, 600V, brown, UL E20042	Alpha Wire	5853-7 BR005	RS	177-0902 (http://it.rs-online.com/web/search /searchBrowseAction.html?method=searchProducts& searchTerm=5853-7+BR005&x=16&y=16)	-	RS (http://www.rs-components.com /index.html)
72	5853-3 RD005, Alpha Wire - Teflon insulated cable, 1 pole, AWG26, 600V, red, UL E20042	Alpha Wire	5853-3 RD005	RS	177-0851 (http://it.rs-online.com/web/search /searchBrowseAction.html?method=searchProducts& searchTerm=1770851&x=17&y=7)	-	RS (http://www.rs-components.com /index.html)
85	4xAWG32 TPU, Industrifil - Teflon insulated cable, 4 poles, AWG32, 250V, with braid, tape and jacket	Industrifil	4xAWG32 TPU (http://www.industrifil.com /En/TPU32.php)	Industrifil	4xAWG32 TPU	+33 046 6886600	info@industrifil.com
70	5853-5 YL005, Alpha Wire - Teflon insulated cable, 1 pole, AWG26, 600V, yellow, UL E20042	Alpha Wire	5853-5 YL005	RS	177-0889 (http://it.rs-online.com/web/search /searchBrowseAction.html?method=searchProducts& searchTerm=5853-5+YL005&x=35&y=8)	-	RS (http://www.rs-components.com /index.html)
519	60.7001-22, Multicontact - PVC insulated cable, 1 pole, AWG27, 150V, red	Multicontact	60.7001-22	Farnell	4326714 (http://it.farnell.com/jsp/search /productdetail.jsp?sku=4326714)	-	Farnell (http://www.farnell.com/)
518	60.7001-21, Multicontact - PVC insulated cable, 1 pole, AWG27, 150V, black	Multicontact	60.7001-21	Farnell	4326702 (http://it.farnell.com/jsp/search /productdetail.jsp?sku=4326702)	-	Farnell (http://www.farnell.com/)
16	8xAWG32 TES, Industrifil -	Industrifil	8xAWG32 TES (http://www.industrifil.com /En/TES32.php)	Industrifil	8xAWG32 TES	+33 046 6886600	info@industrifil.com

	Teflon insulated cable, 8 poles, AWG32, 150V, with braid, tape and jacket						
17	6xAWG32 TES, Industrifil - Teflon insulated cable, 6 poles, AWG32, 150V, with braid, tape and jacket	Industrifil	6xAWG32 TES (http://www.industrifil.com/En/TES32.php)	Industrifil	6xAWG32 TES	+33 046 6886600	info@industrifil.com
23	053787, Jst - Ribbon flat cable, 26 poles, pitch 1.27mm, AWG30, 150V	JST	53787	RS	424-2012 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=424-2012)	-	RS (http://www.rs-components.com/index.html)
310	61.7605-21, Multi contact - Silicon insulated cable, 1 pole, 256x0.05mm, AWG20, 1500V, black, UL E120880, L=25m	Multicontact	61.7605-21	Farnell	4326908 (http://it.farnell.com/jsp/search/productdetail.jsp?sku=4326908)	-	Farnell (http://www.farnell.com/)
311	61.7605-22, Multi contact - Silicon insulated cable, 1 pole, 256x0.05mm, AWG20, 1500V, red, UL E120880, L=25m	Multicontact	61.7605-22	Farnell	4326910 (http://it.farnell.com/jsp/search/productdetail.jsp?sku=4326910)	-	Farnell (http://www.farnell.com/)
328	60.7003-21, Multi contact - PVC insulated cable, 1 pole, 66x0.07mm, AWG23, 500V, black, L=100000mm	Multicontact	60.7003-21	Farnell	4326751 (http://it.farnell.com/jsp/Cable/Single+Wire/MC+(MULTI-CONTACT)/60.7003-21/displayProduct.jsp?sku=4326751)	-	Farnell (http://www.farnell.com/)
329	60.7003-22, Multi contact - PVC insulated cable, 1 pole, 66x0.07mm, AWG23, 500V, red, L=100000mm	Multicontact	61.7605-22	Farnell	4326751 (http://it.farnell.com/jsp/Cable/Single+Wire/MC+(MULTI-CONTACT)/60.7003-21/displayProduct.jsp?sku=4326751)	-	Farnell (http://www.farnell.com/)
513	RJCAB-V-1000, Woodhead - FTP-LAN cable, AWG26, category 5e	Woodhead	RJCAB-V-1000	RS	451-7035 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=451-7035&x=26&y=11)	-	RS (http://www.rs-components.com/index.html)
520	60.7001-24, Multicontact - PVC	Woodhead	60.7001-24	Farnell	4326738 (http://it.farnell.com/jsp/search/productdetail.jsp?sku=4326738)	-	Farnell (http://www.farnell.com/)

	insulated cable, 1 pole, AWG27, 150V, yellow						
521	60.7001-25, Multicontact - PVC insulated cable, 1 pole, AWG27, 150V, green	Woodhead	60.7001-25	Farnell	4326740 (http://it.farnell.com/jsp/search/productdetail.jsp?sku=4326740)	-	Farnell (http://www.farnell.com/)
69	5853-4 GR005, Alpha Wire - Teflon insulated cable, 1 pole, AWG26, 600V, green, UL E20042	Alpha Wire	5853-4 GR005	RS	177-0867 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=1770867&x=32&y=15)	-	RS (http://www.rs-components.com/index.html)
71	5853-10 VI005, Alpha Wire - Teflon insulated cable, 1 pole, AWG26, 600V, violet, UL E20042	Alpha Wire	5853-10 VI005	RS	177-0946 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=1770946&x=25&y=9)	-	RS (http://www.rs-components.com/index.html)
434	12xAWG38 TES, Industrifil - Teflon insulated cable, 12 poles, AWG38, 150V, with braid, tape and jacket	Industrifil	12xAWG38 TES	Industrifil	12xAWG38 TES	+33 046 6886600	info@industrifil.com

Back to the table of contents

Power supply

From Wiki for RobotCub and Friends

iCub uses two power supplies by Xantrex for the 12V and the 48V supplies respectively. Any other voltage required is generated internally (e.g. 5 and 3.3V)
The following table summarizes the specific power supplies currently employed.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
205	XFR-1.2Kw-35-35, Xantrex - Power supply, 1,2 Kw, 0-35V, 0-35A, 44x430x509mm, 8.2kg, 19"rack mount, 1U	Xantrex (http://www.xantrex.com/web/id/6/corp.asp)	XFR-1.2Kw-35-35	DQM-Yokogawa	XFR-1.2 Kw-35-35	+39 0125 564051	DQM s.r.l. (http://www.dqm.it/contacts.asp)
206	XFR-2.8Kw-60-46, Xantrex, - Power supply, 2,8 Kw, 0-60V, 0-46A, 89x430x534mm, 15kg, 19*rack mount, 2U*	Xantrex (http://www.xantrex.com/web/id/6/corp.asp)	XFR-2.8Kw-60-46	DQM-Yokogawa	XFR-2.8Kw-60-46	+39 0125 564051	DQM s.r.l. (http://www.dqm.it/contacts.asp)

More information

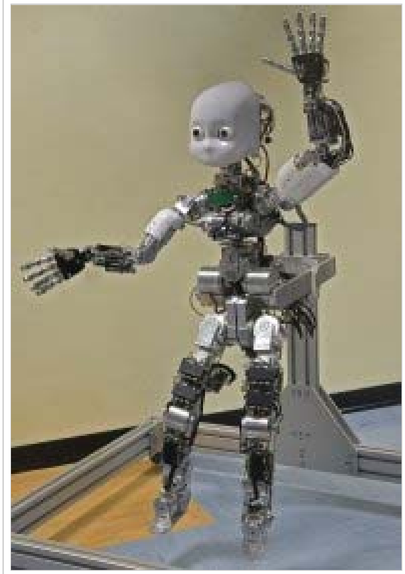
- More information on these power supplies can be found on Xantrex website (http://www.xantrexpowersupplies.com/)
- See also this general datasheet
- Here you can find a more powerful version

Back to the table of contents

Electrical wiring

From Wiki for RobotCub and Friends

Wiring documentation is available from the SVN repository. There are typically folders called "cabling" inside the main subassemblies (e.g. iCubPlatform/hardware/head/cabling). The bill of materials is also available from the "bom" folder (e.g. iCubPlatform/hardware/head/cabling/bom).



The iCub

Wiring manual

- The starting point of the wiring manual is the general wiring schematics (https://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCubPlatform/hardware/cabling/RobotCubCabling.pdf)

- The BOM files are available here (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/) (In particular check the file CablingBOM.xls)

And also:

- Here (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/head/cabling/) for head cabling
- Here (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/upperbody/cabling/) for upperbody cabling
- Here (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/lowerbody/cabling/) for lowerbody cabling

All these documents are GPL/FDL as per the entire RobotCub documentation even if at the moment a proper header is not available.

Back to the table of contents

Magnets

From Wiki for RobotCub and Friends

Encoder magnets are custom made. For size and dimensions please see the corresponding parts in the CVS repository (DXF files).
The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
MAGNET6MM	MPI - Magnetic NdFeB cylinder, MPN35H, Diam.6x2,5mm, Toll.+/-0.1 mm, Diametral magnetized	MPI	Cilindri in MPN35H nichelati, Dim.6x2.5mm, Toll.+/-0.1 mm, Magnetizzati diametralmente	MPI	Cilindri in MPN35H nichelati, Dim.6x2.5mm, Toll.+/-0.1 mm, Magnetizzati diametralmente	+39.02.93566034	MPI (http://www.mpi.it/contattaci.htm) info@mpi.it
RC_TLR_009_P_011_00	MPI - Magnetic NdFeB ring, MPN35H, Diam.7.5x2.39x1.5mm, Toll.+/-0.1 mm, Diametral magnetized	MPI	Anelli in MPN35H nichelati, Dim.7.5x2.39x1.5mm, Toll.+/-0.1 mm, Magnetizzati diametralmente	MPI	Anelli in MPN35H nichelati, Dim.7.5x2.39x1.5mm, Toll.+/-0.1 mm, Magnetizzati diametralmente	+39.02.93566034	MPI (http://www.mpi.it/contattaci.htm) info@mpi.it

Back to the table of contents

Springs, belts and cables

From Wiki for RobotCub and Friends

The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
TEFLON_SHEATH	KT- 80 0025 4473, Angst&Pfister - Insulating sheat, AWG22, inner dia 0,709 -0/+0,102 mm, material thickness 0,152 -0/+0,102mm, teflon, transparent	Angst&Pfister	KT- 80 0025 4473	Angst&Pfister	KT- 80 0025 4473 (It is not a standard P/N. To purchase it, directly contact manufacturer at the phone number or e-mail here on the right)	+39 02300871	Angst&Pfister (http://www.angst-pfister.com/en/DesktopDefault.aspx/tabid-1/) i.canese@angst-pfister.it
LT017-180-156	T017180156 L, MeterSpring - Left torsion spring	Spec	T017180156 L	Meterspring	T017180156 L	+39 01523581	Meterspring (http://www.meterspec.it/spec-nel-mondo.php) meter@meterspec.it
LT020-180-140	T020180140 L, MeterSpring - Left torsion spring	Spec	T020180140 L	Meterspring	T020180140 L	+39 01523581	Meterspring (http://www.meterspec.it/spec-nel-mondo.php) meter@meterspec.it
LT14180_109	T014180109 L, MeterSpring - Left torsion spring	Spec	T014180109 L	Meterspring	T014180109 L	+39 01523581	Meterspring (http://www.meterspec.it/spec-nel-mondo.php) meter@meterspec.it
RT017-180-156	T017180156 R, MeterSpring - Right torsion spring	Spec	T017180156 R	Meterspring	T017180156 R	+39 01523581	Meterspring (http://www.meterspec.it/spec-nel-mondo.php) meter@meterspec.it
RT020-180-140	T020180140 R, MeterSpring - Right torsion spring	Spec	T020180140 R	Meterspring	T020180140 R	+39 01523581	Meterspring (http://www.meterspec.it/spec-nel-mondo.php) meter@meterspec.it
RT14180-109	T014180109 R, MeterSpring - Right torsion spring	Spec	T014180109 R	Meterspring	T014180109 R	+39 01523581	Meterspring (http://www.meterspec.it/spec-nel-mondo.php) meter@meterspec.it
SHEATH	0443/0.25/1.6/500, Mollificio Astigiano - Open spiral spring AISI302 inox steel D ext= 1.6mm D wire= 0.25mm L=500mm	Mollificio Astigiano	0443/0.25/1.6/500	Mollificio Astigiano	0443/0.25/1.6/500	+39 0141959623 +39 0141959904	info@mollificioastigiano.com
03M036001	CD360MXL025, Taraffo - Toothed belt, plastic, Primitive	Sitspa	CD360MXL025	Taraffo	360MXL025	+39 010713076	Sitspa (http://www.sitspa.com/en-US/Sales_network_en.html)

	lenght= 360 inch dec, Pitch= MXL, Width= 0.25 inch cent						
03M053601	CD536MXL025, Taraffo - Toothed belt, plastic, Primitive lenght= 536 inch dec, Pitch= MXL, Width= 0.25 inch cent	Sitspa	CD536MXL025	Taraffo	536MXL025	+39 010713076	Sitspa (http://www.sitspa.com/en-US/Sales_network_en.html)
B73	B73, Taraffo - Flange	Sitspa	B73	Taraffo	B73	+39 010713076	taraffo@taraffo.it
CG077063	CG077063, Carlsthal - Stainless steel microcable, costr. 0.63mm, not coated	Carlstahl	CG077063	Carlstahl	CG077063	+49 7162 40072220	Carlstahl (http://www.carlstahl.de/welcome/) joachim.frank@carlstahl.com
U7191215	U7191215, Carlsthal - Stainless steel microcable, costr. 7x19mm, dia 1.2/1.5mm, PA12, trasparent coated, AISI 316, t/s 1770 N/mm4	Carlstahl	U7191215	Carlstahl	U7191215	+49 7162 40072220	Carlstahl (http://www.carlstahl.de/welcome/) joachim.frank@carlstahl.com
U7191517	U7191517, Carlsthal - Stainless steel microcable, costr. 7x19mm, dia 1.5/1.75mm, PA12, trasparent coated, AISI 316, t/s 1770 N/mm3	Carlstahl	U7191517	CarlstahL	U7191517	+49 7162 40072220	Carlstahl (http://www.carlstahl.de/welcome/) joachim.frank@carlstahl.com
U7194561	U7194561, Carlsthal - Stainless steel microcable, costr. 7x19mm, dia 0.45/0.61mm, PA12, trasparent coated, AISI 316, t/s 1770 N/mm2	Carlstahl	U7194561	Carlstahl	U7194561	+49 7162 40072220	Carlstahl (http://www.carlstahl.de/welcome/) joachim.frank@carlstahl.com

Back to the table of contents

- Disclaimers

CPU board PC104

From Wiki for RobotCub and Friends

Here there are more information about PC104 CPU board manufacturers and suppliers.
The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

The PB945+T7400 board

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
208	PB945+T7400, Embedded logic - PC104 Motherboard PB-945+, with Celeron Core 2Duo 2,16 Mhz	Embedded logic	PB945+T740	Sistemi avanzati elettronici	PB945+ T740 (http://www.embedded-logic.com/index.php?MENUE=Produkte&SUB=5&TYPE=PB945Plus)	+39 015 983206	Sistemi avanzati elettronici (http://www.embedded-logic.com/index.php?MENUE=Contact&SUB=1) sales@sisav.it
211	PB-1GB Sdram, Embedded logic - 1GB DDR-RAM So-DIMM / 200 pins for PB945+	Embedded logic	PB-1GB Sdram	Sistemi avanzati elettronici	PB-1GB Sdram	+39 015 983206	Sistemi avanzati elettronici (http://www.embedded-logic.com/index.php?MENUE=Contact&SUB=1) sales@sisav.it
212	CK-PB945+, Embedded logic - Cable set for PC104 Motherboard PB945+	Embedded logic	CK-PB945+	Sistemi avanzati elettronici	CK-PB945+	+39 015 983206	Sistemi avanzati elettronici (http://www.embedded-logic.com/index.php?MENUE=Contact&SUB=1) sales@sisav.it
633	CHSSPB945, Embedded logic - Chassis mounting kit for PC104 Motherboard PB945+	Embedded logic	CHSSPB945	Sistemi avanzati elettronici	CHSSPB945	+39 015 983206	Sistemi avanzati elettronici (http://www.embedded-logic.com/index.php?MENUE=Contact&SUB=1) sales@sisav.it
277	ICK PGA 17X17, Fischer Elektronic - Heatsink, PGA 8.6°C/W, Black anodized aluminium, Adhesive mount, 43.1x43.1x16.51mm	Fischer elekronik	ICK PGA 17 X17 (http://www.farnell.com/datasheets/17557.pdf)	Farnell	4620963 (http://it.farnell.com/jsp/search/productdetail.jsp?sku=4620963&_requestid=507111)	+39 02 93995200	Farnell (http://www.farnell.com/)
278	2321B-TCM42S-TACH, Aavid Thermalloy - Fan for heatsink, Adhesive mount, 43.2x41.3mm	Aavid Thermalloy	2321B-TCM42S-TACH	Farnell	430730 (http://it.farnell.com/jsp/search/productdetail.jsp?sku=430730&_requestid=507237)	+39 02 93995200	Farnell (http://www.farnell.com/)

Back to the table of contents

Ball bearings

From Wiki for RobotCub and Friends

Here there are more informations about ball bearings manufacturers and suppliers.
The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
604	604 2Z, Skf - Radial ball bearing 4x12x4mm	Skf (http://www.skf.com/portal/skf/home)	604 2Z (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=true&windowName=null&perfid=101008&prodid=1010086040)	Bierredi	604 2Z SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
61800	61800, Skf - Radial ball bearing 10x19x5mm	Skf (http://www.skf.com/portal/skf/home)	61800 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=true&windowName=null&perfid=101002&prodid=1010021800)	Bierredi	61800 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
61801	61801, Skf - Radial ball bearing 12x21x5mm	Skf (http://www.skf.com/portal/skf/home)	61801 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101002&prodid=1010021801)	Bierredi	61801 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
61803	61803, Skf - Radial ball bearing 17x26x5mm	Skf (http://www.skf.com/portal/skf/home)	61803 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101002&prodid=1010021803)	Bierredi	61803 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
61804	61804, Skf - Radial ball bearing 20x32x7mm	Skf (http://www.skf.com/portal/skf/home)	61804 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101002&prodid=1010021804)	Bierredi	61804 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
61900	61900, Skf - Radial ball bearing 10x22x6mm	SSkf (http://www.skf.com/portal/skf/home)	61900 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101001&prodid=1010011900)	Bierredi	61900 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
61901	61901, Skf - Radial ball bearing 12x24x6mm	Skf (http://www.skf.com/portal/skf/home)	61901 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101001&prodid=1010011901)	Bierredi	61901 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it

6001-QE6	6001Q, Skf - Radial ball bearing 12x28x8mm	Skf (http://www.skf.com/portal/skf/home)	6001Q (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=105001&prodid=1050010001)	Bierredi	6001Q SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
607-QE6	607QE6, Skf - Radial ball bearing 7x19x6mm	Skf (http://www.skf.com/portal/skf/home)	607QE6 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=105001&prodid=1050016070)	Bierredi	607QE6 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
61805-Y	61805, Skf - Radial ball bearing 25x37x7mm	Skf (http://www.skf.com/portal/skf/home)	61805 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101002&prodid=1010021805)	Bierredi	61805 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
61806-Y	61806, Skf - Radial ball bearing 30x42x7mm	Skf (http://www.skf.com/portal/skf/home)	61806 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101002&prodid=1010021806)	Bierredi	61806 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
618-4	618/4, Skf - Radial ball bearing 4x9x3mm	Skf (http://www.skf.com/portal/skf/home)	618/4 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101002&prodid=1010026184)	Bierredi	618/4 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
618-5	618/5, Skf - Radial ball bearing 5x11x3mm	Skf (http://www.skf.com/portal/skf/home)	618/5 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101002&prodid=1010026185)	Bierredi	618/5 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
618-6	618/6, Skf - Micro radial ball bearing 6x13x4mm	Skf (http://www.skf.com/portal/skf/home)	618/6 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101002&prodid=1010026186)	Bierredi	618/6 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
618-8	618/8, Skf - Micro radial ball bearing 8x16x4mm	Skf (http://www.skf.com/portal/skf/home)	618/8 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101002&prodid=1010026188)	Bierredi	618/8 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
618-9	618/9, Skf - Micro radial ball bearing 9x17x4mm	Skf (http://www.skf.com/portal/skf/home)	618/9 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101002&prodid=1010026189)	Bierredi	618/9 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
626-QE6	626QE6, Skf - Radial ball bearing 6x19x6mm	Skf (http://www.skf.com/portal/skf/home)	626QE6 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&	Bierredi	626QE6 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it

			windowName=null&perfid=105001&prodid=1050016260)				
HK-1210_INA	HK1210, Skf - Drawn cup needle roller bearings, 6x12x10mm, open ends, unsealed	Skf (http://www.skf.com/portal/skf/home)	HK1210 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=146111&prodid=146111015)	Bierredi	HK1210 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
W_628_5_2Z	W628/5 2Z Skf - Radial ball bearing, 5x11x4mm	Skf (http://www.skf.com/portal/skf/home)	W628/5 2Z (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101047&prodid=1010476285)	Bierredi	W628/5 2Z	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
W_628_8_2Z	W628/8 2Z Skf - Radial ball bearing 8x16x5mm	Skf (http://www.skf.com/portal/skf/home)	W628/8 2Z (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=101048&prodid=1010486288)	Bierredi	W628/8 2Z	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
SKF_QJ205N2MA	QJ205 N2MA/C2L, Skf - Four point contact ball bearing, 25x52x15mm, oblique	Skf (http://www.skf.com/portal/skf/home)	QJ205 N2MA/C2L (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=129002&prodid=1290020205)	Bierredi	QJ205 N2MA/C2L	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
RMB_UL_255X	UL255X-480-P5P-6/15-L23, Myonic - Radial ball bearing, 5x2,5x1,5mm, open ends	Myonic (http://www.myonic.com/index.php?lang=2&idcatside=13)	UL 255X-480-P5P-6/15-L23	GMN	UL 255X-480-P5P-6/15-L23	+39 02 76003865	GMN (http://www.gmn.de/front_content.php?idcat=&idart=90&lang=2) info@gmnitalia.it
RMB_UL_407	UL407X-480-P5P-6/15-L23, Myonic - Radial ball bearing, 7x4x2mm, open ends	Myonic (http://www.myonic.com/index.php?lang=2&idcatside=13)	UL407X-480-P5P-6/15-L23	GMN	UL407X-480-P5P-6/15-L23	+39 02 76003865	GMN (http://www.gmn.de/front_content.php?idcat=&idart=90&lang=2) info@gmnitalia.it
RMB_ULKU_3006X	ULKU3006X-480-P5P-6/15-L23, Myonic - Radial ball bearing, 4,76x2,38x1,5mm, open ends, extended inner ring	Myonic (http://www.myonic.com/index.php?lang=2&idcatside=13)	RMB_ULKU_3006X	GMN	RMB_ULKU_3006X	+39 02 76003865	GMN (http://www.gmn.de/front_content.php?idcat=&idart=90&lang=2) info@gmnitalia.it
RMB_ULU_3006X	ULU3006X-480-P5P-6/15-L23, Myonic - Radial ball bearing, 4,76x2,38x1,5mm, open ends, extended inner ring	Myonic (http://www.myonic.com/index.php?lang=2&idcatside=13)	RMB_ULU_3006X	GMN	RMB_ULU_3006X	+39 02 76003865	GMN (http://www.gmn.de/front_content.php?idcat=&idart=90&lang=2) info@gmnitalia.it
RMB_ULU_3006X	ULU3006X-480-P5P-6/15-L23, Myonic - Radial ball bearing, 4,76x2,38x1,5mm, open ends, extended inner ring	Myonic (http://www.myonic.com/index.php?lang=2&idcatside=13)	RMB_ULU_3006X	GMN	RMB_ULU_3006X	+39 02 76003865	GMN (http://www.gmn.de/front_content.php?idcat=&idart=90&lang=2) info@gmnitalia.it

KAA-10-XLO	KAA10XLO, Kaydon - Four point contact ball bearing, 35x25,5x4,76mm	Kaydon bearings (http://www.kaydonbearings.com/international.php)	KAA10XLO	MAGI Srl	KAA10XLO	+39 02 55194708	MAGI Srl (http://www.magicuscinetti.com/contact.php) info@magicuscinetti.com
KO2508XP0	K02508XPO, Kaydon - Four point contact ball bearing, 41x25x8mm	Kaydon bearings (http://www.kaydonbearings.com/international.php)	K02508XPO	MAGI Srl	K02508XPO	+39 02 55194708	MAGI Srl (http://www.magicuscinetti.com/contact.php) info@magicuscinetti.com

Back to the table of contents

Commercial mechanical parts

From Wiki for RobotCub and Friends

Here there are more informations about commercial mechanical parts manufacturers and suppliers.
The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
H10--_-_NKM	KM0 Skf, Bi.erre.di - Locking nut M10x0.75mm	Skf (http://www.skf.com/portal/skf/home)	KM0 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=267601&prodid=267601000)	Bierredi	KM0 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
R10--_-_NMB	MB0 Skf, Bi.erre.di - Locking washer, Int. diameter=10mm, Ext. diameter=21mm	Skf (http://www.skf.com/portal/skf/home)	KM0 (http://www.skf.com/skf/productcatalogue/Forwarder?action=PPP&lang=en&imperial=false&windowName=null&perfid=267601&prodid=267601000)	Bierredi	KM0 SKF	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
RC_TLR_010_P_010_00	Cod 20 punzone victoria 0.5X100, Bi.erre.di. - Dowel pin, Diam.0.5x5mm, from 0.5x100mm	Bierredi	Cod 20 punzone victoria 0.5X100	Bierredi	Cod 20 punzone victoria 0.5X100	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
RC_TLR_010_P_027_00	Cod 20 punzone victoria 0.5X100, Bi.erre.di. - Dowel pin, Diam.0.5x5mm, from 0.5x100mm	Bierredi	Cod 20 punzone victoria 0.5X100	Bierredi	Cod 20 punzone victoria 0.5X100	+39 010 6592011	Bierredi (http://www.bierredi.it/bierredi/contatti.htm) info@bierredi.it
351	Spacer, hexagonal, M3, M/F, h=5mm, stainless steel	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it
352	Spacer, hexagonal, M3, M/F, h=15mm, stainless steel	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it
357	Spacer, hexagonal, M3, M/F, h=10mm, stainless steel	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it
19	Spacer, hexagonal, M3, M/F, h=20mm, stainless steel	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it
399	Spacer, hexagonal, M2,5, M/F, h=5mm, stainless steel, shortened male	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it
549	Screw, hexagon socket countersunk head, M3x23,5, stainless steel, DIN7991, ISO 10642, UNI5933	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it
558	Spacer, hexagonal, M2,5, M/F, h=8mm, stainless steel, shortened male	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it
47	Spacer, hexagonal, M2,5, F/F, h=8mm, PVC	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it
503	Spacer, hexagonal, M3, F/F, h=3mm, PVC	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it
690	Spacer, hexagonal, M2,5, M/F, h=5,5mm, teflon	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it

691	Spacer, hexagonal, M2,5, M/F, h=6mm, teflon	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it
692	Spacer, hexagonal, M2,5, M/F, h=8mm, teflon	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it
693	Screw, slotted cheese head, M2,5x4, teflon, DIN84 A, ISO1207, UNI6107	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it
688	Spacer, hexagonal, M3 F/F, h=20mm, stainless steel	Mitor system S.r.l.	Part made with custom drawing	Mitor system S.r.l.	-	+39 077 4635162	mitorsystem@tiscali.it
33	Hexagonal nut, M2,5, steel A4, DIN934, ISO4032, UNI5588	RS	248-4567	RS	248-4567 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=248-4567&x=0&y=0)	-	RS (http://www.rs-components.com/index.html)
V2_5-6--_-_I1207_NYLON	Screw, slotted countersunk head, M2,5x6, nylon, DIN84 A, ISO1207, UNI6107	Richco (http://www.richco.se/)	NSE 1207 M2.5 6	Farnell	1261858 (http://it.farnell.com/jsp/search/productdetail.jsp?sku=1261858&_requestid=121882)	-	Farnell (http://www.farnell.com/)
D2_5--_-_I4032_NYLON	Hexagonal nut, M2,5, nylon, DIN934, ISO4032, UNI5588	Duratool	1110025	Farnell	7016931 (http://it.farnell.com/jsp/search/productdetail.jsp?sku=7016931&_requestid=252314)	-	Farnell (http://www.farnell.com/)
D2_5--_-_I4032_NYLON	Hexagonal nut, M2,5, nylon, DIN934, ISO4032, UNI5588	Richco (http://www.richco.se/)	NSE 1207 M3 8	Farnell	1261872 (http://it.farnell.com/richco/nse-1207-m3-8/set-vite-con-slot-testa-svasata/dp/1261872?Ntt=1261872)	-	Farnell (http://www.farnell.com/)
ORM0020_10	ORM 0020-10, Monti&Barabino - O-ring, 2x1mm, plastic	Monti&Barabino	ORM 0020-10	Monti&Barabino	ORM 0020-10	+39 010 413341	info@montiebarabino.it

Back to the table of contents

Commercial electronic parts

From Wiki for RobotCub and Friends

Here there are more informations about commercial electronic parts manufacturers and suppliers.
The table below reports the details. See the bill of materials (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/iCubBom_1.8.pdf) for exact quantities.

RobotCub code	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Phone number	Suppl. Email/Website
360	TTL-232R-3V3, FTDI - Converter cable from USB B to serial, L=350mm	FTDI	TTL-232R-3V3 (http://www.farnell.com/datasheets/81225.pdf)	Farnell	1329311 (http://it.farnell.com/ftdi/ttl-232r-3v3/cavo-usb-livello-ttl-convertitore/dp/1329311?_requestid=237918)	-	Farnell (http://www.farnell.com/)
161	UPM1E222MHD, Nichicon - Elettrolithic Capacitor 2200uF, 25V, THD, Diam.12.5x18mm	Nichicon	UPM1E222MHD (http://www.farnell.com/datasheets/59460.pdf)	Farnell	8812519 (http://it.farnell.com/jsp/Passive+Components/Capacitors/NICHICON/UPM1E222MHD/displayProduct.jsp?sku=8812519)	-	Farnell (http://www.farnell.com/)
427	DS18S20, Maxim - 1-Wire Parasite-Power Digital Thermometer, TO-92	Maxim	DS18S20 (http://www.farnell.com/datasheets/76928.pdf)	Farnell	9724761 (http://it.farnell.com/maxim/ds18s20/termometro-digitale-18s20-to-92/dp/9724761?_requestid=23547)	-	Farnell (http://www.farnell.com/)
57	ECOS1JP222BA, Panasonic - Elettrolithic Capacitor, 2200uF, 63V, 20%, THD, Diam.22x30mm	Panasonic	ECOS1JP222BA (http://docs-europe.electrocomponents.com/webdocs/00ba/0900766b800badcc.pdf)	RS	127-509 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=127509)	-	RS (http://www.rs-components.com/index.html)

Back to the table of contents

ICub stand

From Wiki for RobotCub and Friends

The I-cub stand is composed of two parts:
a vertical one, made of custom pieces (here (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/tools/stand/) you can find its mechanical drawings)
and a moving platform on wheels: the table below reports its commercial details. See the assembly drawing file for the items positions and aluminium profiles lenght.

Item number	Q.ty to order	Description	Manufacturer	Manuf. code	Supplier	Suppl. Order code	Suppl. Email/Website
1	3 pcs	Al profile 2mt lenght	Bosch Rexroth	3842528040	RS	390-0133 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=390-0133&x=0&y=0)	RS (http://www.rs-components.com/index.html)
2	1x (100 pcs pack)	M8x30 Hexagon socket head cap screws	Bossard	BN610 Art.1233459	Bossard	BN610 Art.1233459 (https://shop.bossard.com/it/index.cfm?app_page=0:31004:30002:8382:1#)	Bossard (http://www.bossard.com/)
3	1x (10 pcs pack)	10mm sliding block M8 thread	Bosch Rexroth	3842528735	RS	390-0414 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=390-0414&x=0&y=0)	RS (http://www.rs-components.com/index.html)
4	2 pcs	Wheel with brake diam.80mm	Bosch Rexroth	3842521686	RS	390-2268 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=390-2268&x=0&y=0)	RS (http://www.rs-components.com/index.html)
5	2 pcs	Wheel diam.80mm	Bosch Rexroth	3842521684	RS	390-2274 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=390-2274&x=0&y=0)	RS (http://www.rs-components.com/index.html)
6	12 pcs	Angle bracket 45x45x45mm with bolts	Bosch Rexroth	3842523561	RS	390-1805 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=390-1805&x=0&y=0)	RS (http://www.rs-components.com/index.html)
7	4 pcs	Angle bracket 45x45x90mm with bolts	Bosch Rexroth	3842523570	RS	390-1811 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=390-1811&x=0&y=0)	RS (http://www.rs-components.com/index.html)
8	1x (10 pcs pack)	Plastic cap	Bosch Rexroth	3842511783	RS	390-0206 (http://it.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=390-0206&x=0&y=0)	RS (http://www.rs-components.com/index.html)

Back to the table of contents

2. Troubleshooting of the hardware

Tendons and replacement

From Wiki for RobotCub and Friends

Contents

- 1 Tendons
- 2 Building a tendon (replacement or spare)
  - 2.1 Material for producing a brushless tendons (crimped)
  - 2.2 Material and instructions for producing hand tendons
  - 2.3 Material and instructions for producing wirst tendons
  - 2.4 ICUB0 cables broken

Tendons

At the moment tendons can be seen as a sort of mechanical fuses: when stressed with high forces, they break so as to prevent other (more serious) damages to motors, gearboxes and other parts. You can find a list of all tendons length and details on how to replace them at the following wiki page: Assembly instructions. In the present wiki page we describe how to create new cables and suggest the proper tools to use.

Building a tendon (replacement or spare)

There are three main types of tendons in the iCub. The first type of tendons are used in the bigger joints and are characterized by two crimps on the two ends of the cable. They are typically used in the brushless motor (shoulders, elbows, torso and legs). The second type of tendons are much smaller and characterized by the fact that they slide insdide a tube coated with teflon. They are used to actuate the hand degrees of freedom. A third type of tendon is used to actuate the wrist.

Material for producing a brushless tendons (crimped)

Replacing a brushless tendon with a new one is **not that complicated** and can be done by a "non-necessrily-skilled" person. The following is needed:

- a little bit of training (if you have an iCub then you have a skilled technician which has been trained at the IIT)
- a press with suitable specs: oleodynamic press (15 ton. pressure)
- crimps dxf_crimp1 (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/upperbody/mechanics/shoulders/dxf/rc_tlr_003_p_101_00.dxf) proe_crimp2 (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/upperbody/mechanics/shoulders/proe/rc_tlr_003_p_101_00.drw.1) proe_crimp3 (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/upperbody/mechanics/shoulders/proe/rc_tlr_003_p_101_00.prt.1)
- coated steel wires:
  - Producer: Carlstahl. Code: U7191215 stainless steel microcable, costr. 7x19, dia 1,2/1,5 mm PA 12 trasparent coated.
  - Producer: Carlstahl. Code: U7191517 stainless steel microcable, costr. 7x19, dia 1,5/1,75 mm PA 12 trasparent coated.
- a mold of matrix for pressing the crimps dxf_mold1 (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform

/hardware/tools/crimping_tool/dxf/rc_tlr_999_a_001_00.dxf) dxf_mold2 (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/tools/crimping_tool/dxf/rc_tlr_999_p_001_00.dxf) dxf_mold3 (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/tools/crimping_tool/dxf/rc_tlr_999_p_002_00.dxf)

Unfortunately, instructions on how to replace these tendons are only in italian: italianCableManual.pdf (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/assembly/AssemblyInstructions_Italian.pdf) but there are nice drawings that can be really helpful shoulderWiring.pdf (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/assembly/ShoulderWiring_English.pdf) .

Material and instructions for producing hand tendons

Replacing an hand tendon is quite complicated and in order to perform a good replacement quite a lof of practice is required. The following is needed:

- coated steel wires:
  - Producer: Carlstahl. Code: CG077063 stainless steel microcable, costr. 0,63 mm, not coated Carlstahl
  - Producer: Carlstahl. Code: U7194561 stainless steel microcable, costr. 7x19, dia 0,45/0,61 mm PA 12 trasparent coated.
- teflon tube:
  - Producer: Angst-pfister. Product: TEFLON_SHEATH, Sheath teflon Zeus.
- tube to be used as a guide for the tendon:
  - Producer: Mollifico Astigiano. SHEATH FINGERS CABLES SHEATHS

Instructions on how to replace hand tendons can be found here: tendonsHand2007.pdf (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/assembly/tendonsHand2007.pdf)

Material and instructions for producing wirst tendons

Replacing an wrist tendon is modestly complicated. The following is needed:

- coated steel wires:
  - Producer: Carlstahl. Code: CG077063 stainless steel microcable, costr. 0,63 mm, not coated Carlstahl

Instructions on how to replace wrist tendons can be found here: tendonsWrist.pdf (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/assembly/tendonsWrist.pdf)

ICUB0 cables broken

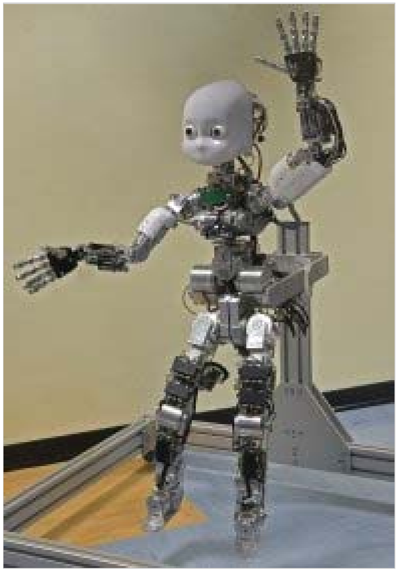
This is the history of the cables broken while debugging the prototype robot in Genoa. It is here for the records (and affective reasons) since it is no longer updated.

- ICUB0 cables broken Image:ICUB0 cables substitution.zip

Assembly instructions

From Wiki for RobotCub and Friends

Assembly manual (in Italian). This is a detailed assembly manual which still needs to be translated (sorry). It consists of several PDFs and videos illustrating the assembly of the various parts. These files are all available from the CVS repository under the 'doc' section (root/doc).


The iCub

Assembly manual

The general assembly instructions are reported in: Assembly instruction (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/assembly/AssemblyInstructions_Italian.pdf)

This is complemented by a set of photos and videos of the various stages of the assembly. These can be browsed at: assembly manual link (http://eris.liralab.it/misc/assemblymanual)

The length of the tendons cables is reported in: iCubCablesLength_Italian.pdf (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/assembly/iCubCablesLength_Italian.pdf)

Details of the wiring of the shoulder are described in: ShoulderWiring_English (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/assembly/ShoulderWiring_English.pdf)

The description of the wiring of the hand cables is reported in: tendonsHand2007.pdf (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/assembly/tendonsHand2007.pdf)

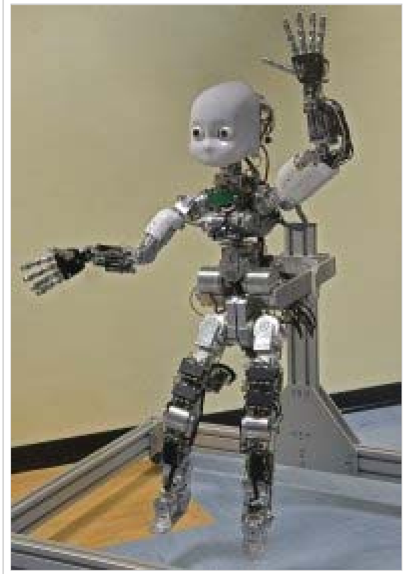
The description of the wiring of the wrist cables is reported in: tendonsWrist.pdf (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/assembly/tendonsWrist.pdf)

All these documents are GPL/FDL as per the entire RobotCub documentation even if at the moment a proper header is not available.

Back to the table of contents

Electrical wiring

From Wiki for RobotCub and Friends


The iCub

Wiring documentation is available from the SVN repository. There are typically folders called "cabling" inside the main subassemblies (e.g. iCubPlatform/hardware/head/cabling). The bill of materials is also available from the "bom" folder (e.g. iCubPlatform/hardware/head/cabling/bom).

Wiring manual

- The starting point of the wiring manual is the general wiring schematics (https://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCubPlatform/hardware/cabling/RobotCubCabling.pdf)

- The BOM files are available here (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/bom/) (In particular check the file CablingBOM.xls)

And also:

- Here (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/head/cabling/) for head cabling
- Here (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/upperbody/cabling/) for upperbody cabling
- Here (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/hardware/lowerbody/cabling/) for lowerbody cabling

All these documents are GPL/FDL as per the entire RobotCub documentation even if at the moment a proper header is not available.

Back to the table of contents

3. Calibration

More firmware

From Wiki for RobotCub and Friends

Here is a short list of the existing firmware for the various control and sensor cards of the iCub.

List of firmware source code repositories

The complete firmware of the iCub consists of:

- control firmware for the BLL and MC4 control cards: see this page and the manual here (http://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCubPlatform/doc/manuals/RC_DIST_100_D_12_01_firmware_versions.doc) .
- firmware of the STRAIN sensor card is available here (http://robotcub.svn.sourceforge.net/viewvc/robotcub/trunk/iCub/src/firmware/strain/) (svn browsing).
- firmware of the MAIS sensor card is availablehere (http://robotcub.svn.sourceforge.net/viewvc/robotcub/trunk/iCub/src/firmware/mais/) (svn browsing).
- firmware for the facial expressions control card.

Please note that the firmware folder (http://robotcub.svn.sourceforge.net/viewvc/robotcub/trunk/iCub/src/firmware/) in the repository contains more code including libraries and loaders for the various cards.

Precompiled version

Precompiled versions of the firmware (ready to upload via the CanLoader) can be found in here (http://robotcub.svn.sourceforge.net/viewvc/robotcub/trunk/iCub/src/firmware/build/) (svn repository).

4. Protocols

CAN protocol

From Wiki for RobotCub and Friends

Control CAN bus protocol description

CAN bus protocol from the CVS repository (http://www.robotcub.org/cvsweb/cvsweb.cgi/~checkout~/iCubPlatform /doc/manuals/RC_DIST_100_D_12_01_canbus_protocol.doc?rev=1.6;content-type=application%2Fx-msword) .

CAN_BCAST Specifications: doc file

Detailed CAN bus protocol: doc file

Detailed CANLoader protocol: from the CVS repository (http://www.robotcub.org/cvsweb/cvsweb.cgi/iCubPlatform /doc/manuals/RC_DIST_100_D_15_01_CANLOADER_PROTOCOL.doc?rev=1.1;content-type=application%2Fx-msword) .

STRAIN board CAN bus protocol description

STRAIN card protocol: STRAIN_000.pdf (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform/doc/manuals /STRAIN_0000.pdf)

5. Kinematics and Dynamics

ICub joints

From Wiki for RobotCub and Friends

Contents

- 1 Naming convention
- 2 Head group
- 3 Left arm
- 4 Right arm
- 5 Torso
- 6 Left leg
- 7 Right leg
- 8 Device and Port view
- 9 Units

Naming convention

The iCub joints are organized into six sub-systems: the head, left arm, right arm, torso, left leg, and right leg.

The joints are numbered to give a natural open kinematic chain, with the base reference frame on the torso. 0 is the most proximal joint, N_max the most distal joint. The key reference point on the body is the base of the neck.

The joint numbers are used when calling methods of the motor control device interfaces. A mechanism will be available for mapping from joint identifiers to numbers. Joint identifiers are unique across the body, except for bilateral symmetry - left or right is specified separately.

Head group

The head has 6 joints in the standard configuration (without the facial features).

Joint number	Can Address	Identifier	Description	Notes
0	3	neck_pitch	Neck pitch	Assuming the standard definition of roll, pitch and yaw with respect to a gravity oriented reference frame aligned with the torso main dimensions
1	3	neck_roll	Neck roll	--
2	3	neck_yaw	Neck yaw	--
3	1	eyes_tilt	Eyes tilt	Common tilt of the eyes
4	1	eyes_version	Eyes version	Common version, the eyes move together, synchronized in the DSP controller (see also VergenceVersion)

5	1	eyes_vergence	Eyes vergence	Vergence control, the eyes move together, synchronized in the DSP controller (see also VergenceVersion)

Left arm

The arm includes the hand for a total of 16 controlled degrees of freedom.

Joint Number	Can Address	Identifier	Description	Notes
0	x	shoulder_pitch	Shoulder pitch	Front-back movement when the arm is aligned with gravity (post decoupling in firmware)
1	x	shoulder_roll	Shoulder roll	Adduction-abduction movement of the arm (post decoupling in firmware)
2	x	shoulder_yaw	Shoulder yaw	Yaw movement when the arm principal axis is aligned with gravity (post decoupling in firmware)
3	x	elbow	Elbow	--
4	x	wrist_prosup	Wrist pronosupination	Forearm rotation along the arm principal axis
5	x	wrist_pitch	Wrist pitch	when hand-wrist aligned with the arm principal axis: i.e. this is relative to the forearm (not necessarily to gravity). Decoupling made in firmware
6	x	wrist_yaw	Wrist yaw	Decoupling made in firmware
7	x	hand_finger	Hand finger adduction/abduction	--
8	x	thumb_oppose	Thumb opposition	--
9	x	thumb_proximal	Thumb proximal flexion/extension	Single tendon looped
10	x	thumb_distal	Thumb distal flexion	Single tendon + return spring for extension spanning two physical joints
11	x	index_proximal	Index proximal flexion/extension	Single tendon looped
12	x	index_distal	Index distal flexion	Single tendon + return spring for extension spanning two physical joints
13	x	middle_proximal	Middle proximal flexion/extension	Single tendon looped
14	x	middle_distal	Middle distal flexion	Single tendon + return spring for extension spanning two physical joints
15	x	pinky	Ring and little finger flexion	Single tendon + return spring spanning six joints on two fingers

Right arm

The arm includes the hand for a total of 16 controlled degrees of freedom. The structure is identical to the left arm.

Torso

Joint number	Identifier	Description	Notes
0	torso_yaw	Torso yaw	With respect to gravity
1	torso_roll	Torso roll	Lateral movement (after decoupling in firmware - differential joint)
2	torso_pitch	Torso pitch	Front-back movement (after decoupling in firmware - differential joint)

Left leg

This refers to the new version which is not yet fully designed.

Joint number	Identifier	Description	Notes
0	hip_pitch	Hip pitch	When the leg principal axis is aligned with gravity (front-back movement)
1	hip_roll	Hip roll	Lateral movement (when leg aligned with gravity)
2	hip_yaw	Hip yaw	Rotation along the leg/tight principal axis
3	knee	Knee	--
4	ankle_pitch	Ankle pitch	When the calf is aligned with gravity
5	ankle_roll	Ankle roll	When the calf is aligned with gravity

Right leg

The structure is the same as the left leg.

Device and Port view

The iCub user will be able to control all joints as they wish, if they wish. For each sub-system, the programmer will be able to retrieve the control interfaces (http://eris.liralab.it/yarp/specs/dox/user/html/d5/d63/group__dev__iface__motor.html) they need.

We expect that these control interfaces will normally be accessed remotely, for all but the tightest control loops. There will be three ports per control group. For example, the head group will have three ports named as follows:

port name	purpose
/icub/head/rpc:i	commands that require replies
/icub/head/command:i	streaming commands
/icub/head/state:o	motor state information

A user can see the commands that the "rpc:i" port supports by using "yarp rpc" to send it the message "help", or browsing to it via the address reported by "yarp where" and clicking "help". For example, typing:

```
yarp rpc /icub/head/rpc:i
```

followed by:

```
[set] [pos] 0 45
```

will command axis 0 of the head (neck_pitch) to 45 degrees. If no motion is seen, it may be necessary to "enable" the axis:

```
[set] [aen] 0
```

where "aen" means "axis enable". The braces are optional. Of course, there are APIs for all such operations.

Here are the identifiers used in port names for the different control groups:

Part	Side	Identifier
head	--	head
arm	left	left_arm
arm	right	right_arm
torso	--	torso
leg	left	left_leg
leg	right	right_leg

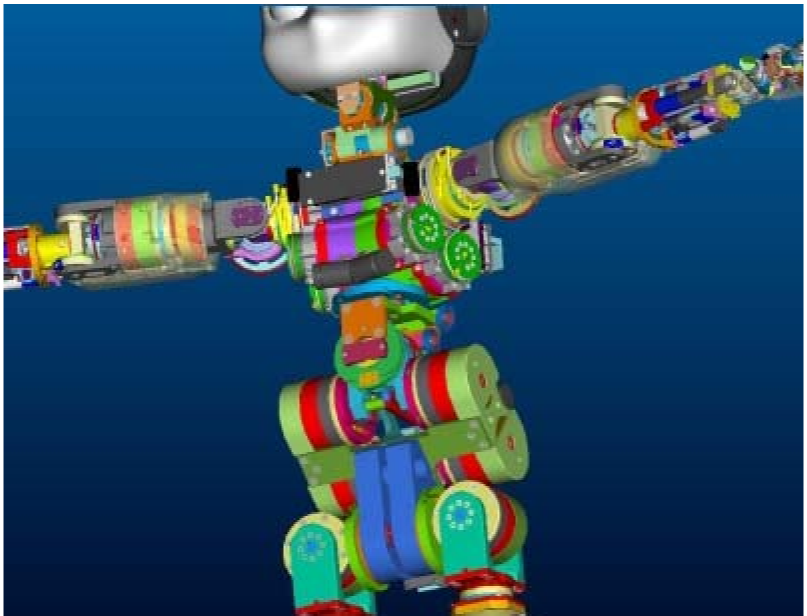
Units

All angles are given in DEGREES.

ICubForwardKinematics

From Wiki for RobotCub and Friends

The iCub forward kinematics are described with respect to a (root) reference frame which is positioned at the level of the waist in the center of the robot as represented in the following pic. (The x axis is in red. The y axis is in green. The z axis is in blue.).



The **origin of the root reference frame** is a point on the axis of rotation of the torso pitch. Among all points in this axis, the origin is located in the middle of the robot in between the two legs.

1. The **z** axis of the root reference frame is parallel to gravity but pointing upwards.
2. The **x** axis of the root reference frame points behind the robot.
3. The **y** axis of the root reference frame points laterally and is chosen according to the right hand rule.

Seven additional reference frames are defined with respect to this common root reference frame. The reference frames are located as shown in the CAD figure. The **x** axis is in **red**. The **y** axis is in **green**. The **z** axis is in **blue**.

1. **Left hand** reference frame (see the CAD picture Media:LeftHandCADRefFrame.jpg)
2. **Right hand** reference frame (see the CAD picture Media:RightHandCADRefFrame.jpg)
3. **Left foot** reference frame (see the CAD picture Media:LegsCADRefFrame.jpg)
4. **Right foot** reference frame (see the CAD picture Media:LegsCADRefFrame.jpg)
5. **Left eye** reference frame (see the CAD picture Media:HeadCADRefFrame.jpg)
6. **Right eye** reference frame (see the CAD picture Media:HeadCADRefFrame.jpg)
7. **Inertia sensor** reference frame (see the CAD picture Media:InertiaCADRefFrame.jpg)

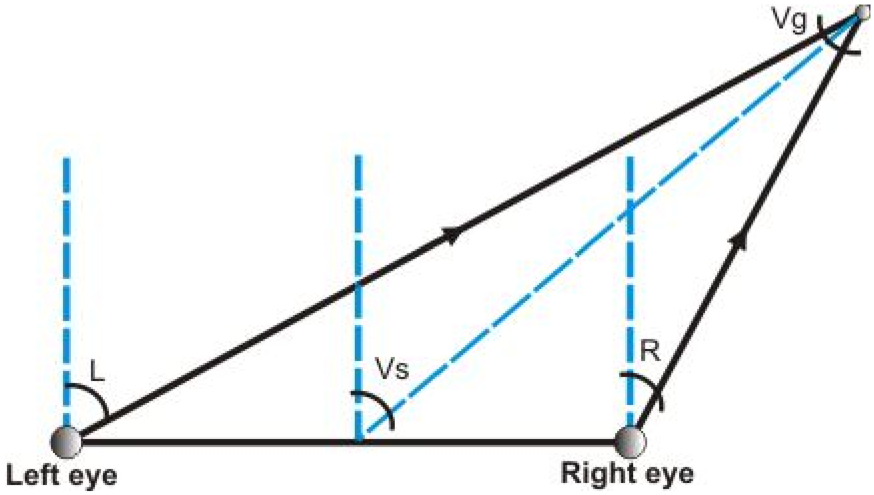
The roto-translation which converts a point in one of these reference frames to the root reference frame are given by the following SE(3) matrices:

End effector		Reference frame	SE(3) Matrix	Description	Matlab files
All	->	Root	-	Whole body (all reference frames)	Media: ICubFwdKinNew.zip
Left hand	->	Root	**T_RoLh**	ICubFowardKinematics_left	Media: ICubFwdKinNew.zip
Right hand	->	Root	**T_RoRh**	ICubFowardKinematics_right	Media: ICubFwdKinNew.zip
Left foot	->	Root	**T_RoLf**	iCubWaistLeftLegKinematics	Media: ICubFwdKinNew.zip
Right foot	->	Root	**T_RoRf**	iCubWaistRightLegKinematics	Media: ICubFwdKinNew.zip
Left eye	->	Root	**T_RoLe**	ICubHeadKinematics	Media:ICubFwdKinNew.zip
Right eye	->	Root	**T_RoRe**	ICubHeadKinematics	Media:ICubFwdKinNew.zip
Inertia sensor	->	Root	**T_RoIs**	iCubInertiaSensorKinematics	Media:ICubFwdKinNew.zip

Each of these matrices is constructed with two steps. The first consists in a rigid roto-translation from the points in the root reference frame to points in the 0*th* reference frame as defined by the Denavit-Hartenberg convention (http://www.cs.dartmouth.edu/~donaldclass/Bio/current/Papers/chap3-forward-kinematics.pdf) . The second step corresponds to the Denavit-Hartenberg description of the forward kinematic, i.e. the roto-translation from the 0th reference frame to the n*th* reference frame being n the number of degrees of freedom.

Vergence, Version and Disparity

From Wiki for RobotCub and Friends



VERGENCE ANGLE

Vg = L - R

where L is the angle of the left camera with respect to the homed optical axis (which is the cameras z-axis) as in the figure above; R is the angle of the right camera with respect to the homed optical axis.

VERSION ANGLE

The version angle is the angle between the axis orthogonal to the baseline and passing through the baseline's midpoint and a line connecting this midpoint and the vergence point. The version angle satisfies the following nonlinear relation:

tan(Vs) = (tan(L) + tan(R)) / 2;

However, the Firmware sends as version the following value:

Vs = (L + R) / 2,

which holds for small angles (where $\tan(x) \approx x$), so that even though there is no lack of information since L and R angles can be accurately retrieved (see hereafter), the version Vs has physical meaning only for small values of Vs, L and R.

CONVERTING [VERGENCE|VERSION] TO [DECOUPLED L|R]

Combining the above equations yields:

L = Vs + Vg/2;

R = Vs - Vg/2;

DISPARITY

Disparity is defined as:

$d = xl - xr$

where xl is the left image normalized coordinate and xr is the right image normalized coordinate.

Object closer to the cameras than the current point of fixation, will elicit a positive disparity value.

6. Software, Compiling YARP and iCub

PrepareLinux

From Wiki for RobotCub and Friends

These instructions explain how to set up your system to compile YARP and iCub. Here we assume you are using Debian Stable or Ubuntu Gutsy. It should not be different to setup other Linux, by just by installing similar packages.

Contents

- 1 Development environment
- 2 Libraries
  - 2.1 ACE
  - 2.2 GUIS
  - 2.3 Gnu Scientific Library
  - 2.4 OpenCV
  - 2.5 IPOPT
- 3 Simulator related packages

Development environment

Install the following packages:

```
cmake (at least version 2.4)
g++
```

If you don't know what cmake is and you are wondering why you need to install cmake please wait until Section 6.6 of the Manual. Of course you can jump there if you really can't wait.

Libraries

Install the package:

```
libncurses5-dev
```

ACE

To compile YARP you need ACE.

In Linux you have two options:

- Get precompiled versions of ace that are distributed with your Linux (please check here.

```
If you follow this procedure, all you need to do is to install the libace-dev package.
```

- Compile ace from sources, if you follow this procedure do not forget to set the environment variable ACE_ROOT to point to the directory where you have unpacked the sources. Compilation instructions are available here Installing

ACE.
- Which version of ace should I use? Follow this link to find out what is the most suitable version of ace depending on your system: Which version of ACE.

Environment variable you should have after this procedure: ACE_ROOT and LD_LIBRARY_PATH, only if you *do not* use the precompiled packages.

GUIS

GUIS are written using GTK/GTKMM and QT.

GTKMM:

```
libgtkmm-2.4-dev
libglademm-2.4-dev
```

QT:

```
libqt4-dev
qt4-dev-tools
libqt3-mt-dev
qt3-dev-tools
```

Environment variable you should have after this procedure: nothing new.

Gnu Scientific Library

YARP and some modules in iCub make use of the Gnu Scientific Library. In Debian it is easy to install gsl through the package:

```
libgsl0-dev
```

You can also download the library directly from http://www.gnu.org/software/gsl/.

Environment variable you should have after this procedure: nothing new.

OpenCV

Software in iCub makes extensive use of opencv. In Debian just install:

```
libcv-dev
```

Environment variable you should have after this procedure: nothing new.

IPOPT

Inverse kinematics modules need the IPOPT library. This is not a requirement but it could be a useful feature. See instructions here:

- Installing IPOPT

Environment variable you should have after this procedure: IPOPT_DIR

Simulator related packages

The simulator will also need ode and sdl. See the simulator instructions on how to do it, Simulator libraries.

PrepareWindows

From Wiki for RobotCub and Friends

These instructions explain how to set up your system to compile YARP and iCub. Here we assume you are using Windows Xp.

Important: in Windows cmake almost always use environment variables to check availability of the libraries and locate them on the disk. Instructions for installation of each single library explain what environment variable you need.

Contents

- 1 Development environment
- 2 Libraries
  - 2.1 ACE
  - 2.2 GUIs
  - 2.3 Gnu Scientific Library
  - 2.4 OpenCV
  - 2.5 IPOPT
- 3 Simulator related packages

Development environment

You need:

- cmake 2.6 (www.cmake.org)
- Microsoft Visual Studio 2005 or 2008

If you don't know what cmake is and you are wondering why you need it, please wait until Section 6.6 of the Manual. Of course you can jump there if you really can't wait.

Libraries

ACE

In windows you have to get the sources and compiled ace on your system.

- Which version of ace should I use? Follow this link to find out what is the most suitable version of ace depending on your system: Which version of ACE.
- Follow these instructions to compile ace: Compiling ACE.

Environment variable you should have after this procedure: ACE_ROOT

After this procedure you should have updated your system path to include: %ACE_ROOT%/lib

GUIs

GUIs are written using GTK+/GTKMM and QT.

GTK+ and GTKMM:

- GTK on Windows

Environment variable you should have after this procedure: GTK_BASEPATH, GTKMM_BASEPATH

QT:

- Qt on Windows

Environment variable you should have after this procedure: QTDIR

Gnu Scientific Library

YARP and some modules in iCub make use of the Gnu Scientific Library.

Follow instructions here:

- Installing GSL on Windows

Environment variable you should have after this procedure: GSL_DIR

OpenCV

Software in iCub makes extensive use of opencv. Follow instructions here:

- Installing OpenCV on Windows

Environment variable you should have after this procedure: OPENCV_DIR

After this procedure you should have updated your system path to include: %OPENCV_DIR%/bin

IPOPT

Inverse kinematics modules need the IPOPT library. This is not a requirement but it could be a useful feature. See instructions here:

- Installing IPOPT

Environment variable you should have after this procedure: IPOPT_DIR

Simulator related packages

The simulator will also need ode and sdl. See the simulator instructions on how to do it, Simulator libraries.

Getting Subversion

From Wiki for RobotCub and Friends

To download the software you *probably* (see below) need a Subversion client (note, we moved from CVS to Subversion in August 2009).

Getting Subversion

- Linux distributions come with an svn client already installed. In the worst case you just need to install one using the package manager (in Debian/Ubuntu *apt-get install subversion*).

- In Windows you are free to pick any client you like. We suggest tortoisesvn, available from http://tortoisesvn.net/.

If you are not familiar with svn we suggest at least you learn the basics. Some instructions are available from the sourceforge website:

```
https://sourceforge.net/apps/trac/sourceforge/wiki/Subversion
```

A lot of more details (including a quick introduction) can be found instead here:

```
http://svnbook.red-bean.com/
```

Important: the robotcub repository is now hosted by sourceforge. To commit changes to the repository you need a sourceforge account. Follow instructions at: http://www.sourceforge.net. Still in sourceforge, send an email to the robotcub project administrator to join the project.

Do I need Subversion?

Yes and no (but yes). You need Subversion to dowload and commit code to the repositories. At the moment you can download precompiled versions of YARP (we provide instructions for doing this in the appropriate sections of the manual). Precompiled versions of the RobotCub software are not available yet (although one day they will). So you definitely need Subversion to get the iCub code.

In any case using svn allows you to:

- get the most updated version of the code
- contribute to the iCub code by committing to the repository

Subversion client flags

From Wiki for RobotCub and Friends

In Subversion each file can have a set of properties. Properties are used by svn to handle files appropriately (for example svn tries to merge only text files but not binary/application files). Properties are stored in the repository and are usually set when a new file is added and committed. If you do not plan to commmit files you can safely assume properties are already correct in the repository and stop bothering reading this. **However if you are going to commit files to the repository you must configure your client correctly, see below**.

For cvs users. The default behavior of svn differs from cvs, concerning how text and binary files are treated and committed. This page gives more details on this; if you commit files to the repository (and you are not an experienced svn user) please read this carefully and/or make sure you understand how to deal with file properties with svn. *The following discussion can be quite technical. If you are a naive svn user, just jump to the instructions at the end of this page (How to configure your svn client).*

A Technical explanation

Individual committers can configure SVN locally with an "auto-props" section to automate how properties are assigned to new files they add/import. Once added/imported the properties stay the same; although they can be changed at any time, **we strongly recommend to set file properties correctly when files are added to the repository**.

Typically you have to **enable the auto-props with something like "enable-auto-props = yes"** (depending on the client program, see below for instructions). Then you might need to add rules on how to manage individual files (well, wildcards are accepted). Here a list that we find useful for YARP/iCub.

This list also reflects how files should be in the repository. Take this list as a generic guideline, each file can of course have particular properties depending on the situation.

Roughly speaking the meaning of the properties is:

- eol-style=native: for text files this tells svn to convert end of line characters to match the client systems (Windows or Unix). In this case svn will also try to merge differences when concurrent modifications have been made;

- svn:mime-type=application/*: this tells svn that the file is binary and should not be touched. No end of line conversion will be made. svn will not resolve conflicts, it will just make a copy of the locally modified file and download the one from the repository.

As a general rule we use *application/octet-stream* for generic applications, and more specific tags when available (application/x-msword, or postscript...), but it does not make much difference in this context.

List of Properties

```
*.txt = svn:eol-style=native
```

```
*.c = svn:eol-style=native
*.cc = svn:eol-style=native
*.cxx = svn:eol-style=native
*.hpp = svn:eol-style=native
*.cpp = svn:eol-style=native
*.h = svn:eol-style=native
*.hpp = svn:eol-style=native
*.inl = svn:eol-style=native
```

```
*.cmake = svn:eol-style=native
*.glade = svn:eol-style=native
*.gladep = svn:eol-style=native
*.ui = svn:eol-style=native
```

```
*.asv = svn:eol-style=native
*.dat = svn:mime-type=application/octet-stream
```

```
*.m = svn:eol-style=native
*.mdl = svn:eol-style=native
*.mat = svn:mime-type=application/octet-stream
*.mexw32 = svn:mime-type=application/octet-stream
```

```
*.wbt = svn:eol-style=native
```

```
*.bat = svn:eol-style=native
*.cmd = svn:eol-style=native
*.sh = svn:eol-style=native;svn:executable
*.py = svn:eol-style=native;svn:executable
*.pl = svn:eol-style=native;svn:executable
```

```
*.dsw = svn:eol-style=CRLF
*.sln = svn:eol-style=CRLF
*.dsp = svn:eol-style=CRLF
```

```
*.mcp = svn:mime-type=application/octet-stream
*.S = svn:mime-type=application/octet-stream
*.mcw = svn:mime-type=application/octet-stream
*.mptags = svn:eol-style=native
*.tagsrc = svn:eol-style=native
*.asm = svn:eol-style=native
*.hex = svn:eol-style=native
*.map = svn:eol-style=native
*.mcs = svn:eol-style=native
*.gld = svn:eol-style=native
*.cww = svn:eol-style=native
*.pjt = svn:eol-style=native
*.PJT = svn:eol-style=native
```

```
*.obj = svn.mime-type=application/octet-stream
```

```
*.dox = svn:eol-style=native
*.xml = svn:eol-style=native
*.template = svn:eol-style=native
*.ini = svn:eol-style=native
*.cfg = svn:eol-style=native
*.conf = svn:eol-style=native
*.howto = svn:eol-style=native
*.info = svn:eol-style=native
```

```
*.html = svn:eol-style=native
```

```
*.tex = svn:eol-style=native
*.cls = svn:eol-style=native
*.bib = svn:eol-style=native
```

```
*.png = svn:mime-type=image/png
*.jpg = svn:mime-type=image/jpeg
*.bmp = svn:mime-type=image/bmp
*.gif = svn:mime-type=image/gif
```

```
*.pdf = svn:mime-type=application/pdf
*.eps = svn:mime-type=application/postscript
```

```
*.doc = svn:mime-type=application/x-msword
*.dot = svn:mime-type=application/x-msword
*.xls = svn:mime-type=application/x-excel
*.ppt = svn.mime-type=application/x-mspowerpoint
```

```
*.bin = svn:mime-type=application/octet-stream
*.lib = svn:mime-type=application/octet-stream
*.a = svn:mime-type=application/octet-stream
*.exe = svn:mime-type=application/octet-stream
*.dll = svn:mime-type=application/octet-stream
```

```
*.gbr = svn:eol-style=native
*.BOT = svn:eol-style=native
*.ASB = svn:eol-style=native
*.SSB = svn:eol-style=native
*.AST = svn:eol-style=native
*.DRD = svn:eol-style=native
*.SST = svn:eol-style=native
*.tap = svn:eol-style=native
*.TOP = svn:eol-style=native
*.SMB = svn:eol-style=native
*.SMT = svn:eol-style=native

*.MAX = svn:mime-type=application/octet-stream
*.DSN = svn:mime-type=application/octet-stream
*.DBK = svn:mime-type=application/octet-stream
*.OPJ = svn:eol-style=native
*.opj = svn:eol-style=native
*.GTD = svn:eol-style=native
*.dxf = svn:eol-style=native
*.STEP = svn:eol-style=native
*.BOM = svn:eol-style=native
*.opj = svn:eol-style=native
*.1 = svn:mime-type=application/octet-stream
*.scl = svn:eol-style=native
```

```
*.sch = svn:mime-type=application/octet-stream
*.brd = svn:mime-type=application/octet-stream
*.pcb = svn:mime-type=application/octet-stream
```

```
Makefile = svn:eol-style=native
README = svn:eol-style=native
TODO = svn:eol-style=native
COPYING = svn:eol-style=native
ChangeLog = svn:eol-style=native
INSTALL = svn:eol-style=native
AUTHORS = svn:eol-style=native
```

How to configure your svn client

This is clearly client dependent. The goal is to change a config file, in Linux this is usually $HOME/.subversion/config.

For Windows we provide an example of how to change the config file of TortoiseSVN:

- launch Windows Explorer
- Go to File > TortoiseSVN > Settings
- Under "General" on the right panel, click "Edit" for "Subversion Configuration File:"
- The configuration file needs to be changed as follow:
- Uncomment the "enable-auto-props = yes" directive
- Uncomment the [auto-props] directive
- Copy the "List of Properties" from this page (above) to the end of the file, remove the existing list to avoid duplications and make sure you remove the space before the wild-card (*)
- Save

Check your system (optional)

From Wiki for RobotCub and Friends

Compile tests of the software are performed periodically. Results are posted on the web together with the configuration of the machine that performs the tests (packages, installed libraries and environment variables).

This configuration is the one that is most tested and less likely to have problems. You can use it as an example or reference to configure your machines.

See: http://eris.liralab.it/iCub/dox/html/compile_status.html

GettingYARP svn

From Wiki for RobotCub and Friends

Important: we now use Subversion. Instructions for getting the software from the repository have changed (see below).

Downloading YARP as a Source Package

Download YARP here: http://eris.liralab.it/yarp/specs/dox/download.html

Downloading YARP from the repository, Linux or Cygwin

You can get YARP with the command:

```
svn co https://yarp0.svn.sourceforge.net/svnroot/yarp0/trunk/yarp2
```

This will download a directory called yarp2 containing software.

Downloading YARP from from the repository, Microsoft Windows without Cygwin

You need an svn client. Among the many alternatives we suggest you get tortoise SVN. Binaries for windows are available here: http://tortoisesvn.net/.

Use file manager to browse to the location where you would like to download the code. Right click on an empty region of the window and select "SVN checkout" from the contextual menu.

Type the following string in the entry "URL of repository":

```
 https://yarp0.svn.sourceforge.net/svnroot/yarp0/trunk/yarp2
```

No password or username will be required.

Getting the iCub software

From Wiki for RobotCub and Friends

Contents

- 1 Getting the software from the snapshots
- 2 Getting the software using Subversion
  - 2.1 Getting the repository for Linux
  - 2.2 Getting the repository for Windows

The subversion repository contains the latest version of the code. From time to time we release a snapshot of the whole repository in a single tar file.

Getting the software from the snapshots

Periodic snapshots of the repository are available here: http://eris.liralab.it/iCub/downloads/src/

There are three types of files:

- iCub-src-x.y.z.tar.gz: the whole iCub repository version x.y.z
- iCub-dep-x.y.z.txt: a list of the libraries that are required to compile the code in snapshot x.y.z (including yarp version number).

Getting the software using Subversion

These are instructions to get the RobotCub software repository using svn.

Important since August 5th 2009, repositories have been moved to SourceForge (www.sourceforge.net).

These instructions explain how to download the software using svn. We assume you already know what svn is and how to use it.

```
Important note for CVS users. To get the software you no longer need to have a
user; however you do need one to commit changes to the repository. If you think
you need a user follow instructions in Section 6 of the manual ("Prepare your system,
get svn").
```

Getting the repository for Linux

You need an svn client. svn is normally distributed with Linux, use your package manager to install it (Debian/Ubuntu apt-get install subversion).

Pick the location where you would like to download the code.

At the console type:

```
svn co https://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCub
```

this will create in the local directory a new directory called iCub that contains the software.

Getting the repository for Windows

You need an svn client. Among the many alternatives we suggest you get tortoise SVN. Binaries for windows are available here: http://tortoisesvn.net/.

Use file manager to browse to the location where you would like to download the code. Right click on an empty region of the window and select "SVN checkout" from the contextual menu.

Type the following string in the entry "URL of repository":

```
https://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCub
```

No password or username will be required.

Environment

From Wiki for RobotCub and Friends

Depending on the operating system you are using, instructions in Section 6.1 "Prepare you system" have required you to set some environment variables. These depend on your system and your choices so we don't review them here. Just make sure you followed the instructions correctly.

YARP and iCub software require another couple of environment variables. This applies to all systems.

```
YARP_ROOT= point to where Yarp was unpacked (used by various applications)
YARP_DIR= typically points to YARP_ROOT (used by CMake)
YARP_CONF= where the yarpserver configuration file can be stored
ICUB_ROOT= point to where iCub code was unpacked
ICUB_DIR= points to ICUB_ROOT
```

Note for Windows users: be sure to use `C:/` (or whatever drive letter you need) and not `C:\`, e.g. `ICUB_ROOT=C:/iCub` and **not** `ICUB_ROOT=C:\iCub`. The \ version works most of the time but not always. The same applies for YARP.

New (since July 2009): if you have a robot, you also have to define:

```
 ICUB_ROBOTNAME= name of your robot (the directory in $ICUB_ROOT/app that stores your robot configuration fi
```

In Linux and Mac OS X you do this using the "export" command. In the case of Linux it is a good idea to place them in your .bashrc file (or equivalent). In Windows environment variables are in the System Properties tab in the Control Panel.

Append ICUB_DIR/bin and YARP_DIR/bin to your PATH

- Hint for Linux and Mac OS X:

```
export PATH=$PATH:$YARP_DIR/bin:$ICUB_DIR/bin
```

- Hint for Windows: check the current value of PATH in the control panel and extend it.

Important: YARP_ROOT and YARP_DIR have different meaning, although here they point to the same place. YARP_ROOT points to the location of the sources, YARP_DIR points to where you build your binaries. We here point them to the same place (cmake calls this in source build, in general they could be different).

Similar considerations apply to ICUB_ROOT and ICUB_DIR.

CMake icub

From Wiki for RobotCub and Friends

Contents

- 1 Why CMake?
- 2 Install CMake
- 3 CMake in Windows
- 4 CMake in UNIX
- 5 An Example

Why CMake?

We'd like you all to use the development environment you are used to, and not force you to switch to something else -- no Linux/g++/emacs vs Windows/DevStudio vs Mac/... fights please!

To achieve this without complete chaos, we ask you to install "CMake". CMake lets us describe our programs and libraries in a cross-platform way. CMake takes care of building the makefiles or workspaces needed by whatever development environment you like to work in.

- Read about CMake here: http://www.cmake.org/

Install CMake

- Install CMake from here: http://www.cmake.org/HTML/Download.html

- Tips for linux:
  - Debian Linux: apt-get install cmake (recommended)
  - Generic Linux: http://www.cmake.org/files/v2.4/cmake-2.4.6-Linux-i386.sh
  - SUSE Linux: add the GURU (http://en.opensuse.org/Additional_YaST_Package_Repositories#Guru) YAST repository and use YAST for installing CMake or download directly the rpm from GURU website access (http://linux01.gwdg.de/~pbleser/allpackages.php)
  - We've had reports that CMake has problems if you install it in one location and then try to run it via a symbolic link from another location. We suggest you use a short script rather than a symbolic link if you need to do something like this. For example, if the cmake binary is installed in /opt/cmake/bin/cmake and you wish to execute it as /usr/local/bin/cmake, then make a script at /usr/local/bin/cmake with the contents:

```
#!/bin/sh
exec /opt/cmake/bin/cmake $*
```

CMake in Windows

On Windows, the easiest way to use CMake is via its GUI. After installing, you should have an icon for CMake in your START menu. Click that, then fill in the path to your code, and the path you want CMake to build in (that can be the same if you want). Click "configure". Depending on the project, configuration may involve several steps -- you may have to answer new questions and click "configure" again. When the "OK" button becomes clickable, then CMake has enough information to set up your project. Click "OK" and you're done. Project files of the type you specified should exist in the build path you gave.

If you want to start over from the beginning with CMake, it is important to press the "delete cache" button to make it forget everything you've told it.

CMake in UNIX

On UNIX CMake can be used conveniently in two ways:

- From the command line : type "cmake ."
- Interactively: type "ccmake ."

If you are running CMake while in a directory different to where your code is, replace "." with the path to your code.

"ccmake" is very much like the Windows GUI, and you may need to iterate "configure" a few times before the option to "generate" appears.

"cmake" doesn't ask questions, and just uses defaults. You can pass it values on the command line:

```
cmake -DCMAKE_INSTALL_PREFIX:PATH=/usr .
```

The generated file "CMakeCache.txt" contains all settings stored by CMake. It can be useful to delete this if you want to start over completely.

An Example

Create a new directory, something like $HOME/cmake/example or C:\cmake\example.

Inside that directory, create a file called "CMakeLists.txt". In it place the following:

```
PROJECT(example)
ADD_EXECUTABLE(example main.cpp)
```

In the same directory, create a file called "main.cpp". In it place the following:

```
#include <stdio.h>
int main() {
  printf("CMake the world a better place!\n");
  return 0;
}
```

In UNIX, type "cmake ." in that directory, and then "make", and then "./example". Easy!

On Windows, run the CMake GUI, fill in the path to the example, click "configure", say what compiler you use, click "configure" again if needed, then click "ok". Then run your compiler, and finally the program. Easy!

Notice that the abstract description of our project above can be shared by developers on Windows, Linux, ...

- There's another example you can try here: http://www.cmake.org/HTML/Examples.html

- For more examples, and details, see: http://www.linuxjournal.com/article/6700

CMake problems

From Wiki for RobotCub and Friends

CMake is currently having lots of trouble if you have " "s

Gtk+ on Cygwin

pkg-config does not work well and cmake fails if GUI flag is turned on.

The emergency solution is editting yarp2/conf/FindGtkPlus.conf such as:

```
SET(GTKPLUS_LINK_FLAGS "-L/GTK/bin -lgtk-win32-2.0 -lglib-2.0 -lgdk-win32-2.0 -lgobject-2.0 -lgdk_pixbuf-2.G
SET(GTKPLUS_C_FLAGS "-I/GTK/include/atk-1.0 -I/GTK/include/pango-1.0 -I/GTK/include/gtk-2.0 -I/GTK/include/G
SET(GTKPLUS_INCLUDE_DIR "/GTK/include/atk-1.0 /GTK/include/pango-1.0 /GTK/include/gtk-2.0 /GTK/inlcude/glib-
IF (GTKPLUS_C_FLAGS)
  SET(GtkPlus_FOUND TRUE)
ELSE (GTKPLUS_C_FLAGS)
  SET(GtkPlus_FOUND FALSE)
ENDIF (GTKPLUS_C_FLAGS)
```

/GTK is GTK_BASEPATH.

CompileLinux

From Wiki for RobotCub and Friends

Important: we assume here that you have completed the previous steps in the manual (see Section 6.1 "Prepare Your System" in the manual's main page).

Before you compile the code you need to generate make files.

Here we assume you have completed the previous steps (preparing your system, getting the software, setting up your environment).

Contents

- 1 Compiling YARP
  - 1.1 Create the YARP Makefiles
  - 1.2 Compile
  - 1.3 Example -- is YARP available?
- 2 Compile the iCub software
  - 2.1 Generate makefiles
  - 2.2 Compile

Compiling YARP

Create the YARP Makefiles

Run (don't forget to set the environment variables first):

```
cd $YARP_ROOT
ccmake ./
```

Choose the following options:

- CMAKE_BUILD_TYPE, set to "Release" in case you'd like to optimize
- CREATE_GUIS, set to ON
- CREATE_LIB_MATH, set to ON

Important: CREATE_GUIS and CREATE_LIB_MATH require you have installed the libraries gtk and gsl (see PrepareLinux)

Create the makefiles by selecting configure several times and then generate.

Installation: CMake automatically creates an install rule for target/project. In the documentation we assume you install binaries in $YARP_ROOT/bin and $YARP_ROOT/lib. The compiler will build executables and libraries there, so you don't need to perform the installation. You can instruct CMake so that it generates make/project files that install to other places, for example $YARP_DIR/bin and $YARP_DIR/lib. You can do this by running cmake again and setting the variable:

- CMAKE_INSTALL_PREFIX to $YARP_DIR

When you do make install all binaries will be copied to $YARP_DIR/bin and $YARP_DIR/lib.

Of course you can customize the installation directory as you wish, however the remainder of the documentation assumes the above configuration.

- Depending on the hardware on your system you might want to compile additional device drivers. This is done for example on the pc104. Instruction for doing this are reported elsewhere.

Compile

Now we are ready to compile. This is easy.

Run:

```
cd $YARP_ROOT
make
```

Example -- is YARP available?

Now we're ready to run a simple Yarp code to test the installation so far. You might want to prepare a yarp.conf file in the conf directory similar to this one:

```
127.0.0.1 10000
```

which tells Yarp (the server) to start on the localhost and respond to port 10000. This allows Yarp applications to find the name server (see next chapter).

You can then try running the server. On a terminal window, type:

```
yarpserver &
```

and you should see:

yarp: Port /root active at tcp://127.0.0.1:10000 Name server can be browsed at http://127.0.0.1:10000/ yarp: Bootstrap server listening at mcast://224.2.1.1:10001

if you type on a web browser http://127.0.0.1:10000 you get information about the name server (registered ports, info, etc.).

For the time being we can just check functionality by running a simple example. On another terminal type:

```
yarp read /portread
```

on a third terminal:

```
yarp write /portwrite
```

and on yet another terminal:

```
yarp connect /portwrite /portread
```

you'll see the effect on the name server:

```
yarp: registration name /portwrite ip 127.0.0.1 port 10012 type tcp
yarp: registration name /portread ip 127.0.0.1 port 10002 type tcp
```

Now, anything typed on the yarp write will be sent and printed on the read side.

Compile the iCub software

Generate makefiles

First you need to generate make files. In $ICUB_ROOT:

```
ccmake ./
```

You don't need particular options. If you want to compile using optimization just set:

- CMAKE_BUILD_TYPE to "Release"

Other options are:

- CREATE_GUIS_GTK
- CREATE_GUIS_GTKMM
- CREATE_GUIS_QT

These options are recommended, because they enable compilation of some useful GUIs. Important: these options can be enabled only if you have installed the required libraries: gtk, gtkmm and qt (see PrepareLinux).

- Similarly to YARP, by default make will build executables and libraries in $ICUB_ROOT/bin and $ICUB_ROOT/lib. You can customize where "make install" will copy these files by setting: CMAKE_INSTALL_PREFIX to something you like.

If you need to compile devices that provide interface to the hardware you can follow this link Compilation on the pc104

Compile

Compile the code.

```
cd $ICUB_ROOT
make
```

- Disclaimers

CompileWindows

From Wiki for RobotCub and Friends

Important: we assume here that you have completed the previous steps in the manual (see Section 6.1 "Prepare Your System" in the manual's main page).

Working on:

```
Vista (the worst possible situation)
Visual Studio 8
```

For other compilers, e.g. Visual Studio express, additional documentation is required. Note: Visual Studio express requires the installation of the Windows Platform SDK. Visual Studio 8 comes pre-packed with the Platform SDK.

Contents

- 1 Compiling YARP
  - 1.1 Create the YARP Makefiles
  - 1.2 Compile
    - 1.2.1 Example
- 2 Options
- 3 Compile the iCub software
  - 3.1 Generate makefiles
  - 3.2 Compile

Compiling YARP

Create the YARP Makefiles

Run CMake. Point the source code directory ("Where is the source code" entry in the gui) to %YARP_ROOT% (where you put the YARP source files). Pick the directory in which you would like to generate project files ("Where to build the binaries" entry in the gui). We call this directory YARP_DIR. The most common situation to use YARP_DIR=YARP_ROOT, this is the assumption in this documentation.

Choose the following options:

- CMAKE_BUILD_TYPE, set to "Release" in case you'd like to optimize
- CREATE_GUIS, set to ON
- CREATE_LIB_MATH, set to ON

Important: CREATE_GUIS and CREATE_LIB_MATH require you have installed the libraries gtk and gsl (see PrepareWindows)

Create the makefiles by hitting configure several times and then ok.

Installation: CMake automatically creates an install rule for target/project. In the documentation we assume you install binaries in %YARP_ROOT%/bin and %YARP_ROOT%/lib. The compiler will build executables and libraries there, so you don't need to perform the installation. You can instruct CMake so that it generates make/project files that install to other places, for example %YARP_DIR%/bin and %YARP_DIR%/lib. You can do this by running cmake again and setting the variable:

- CMAKE_INSTALL_PREFIX to %YARP_DIR%

When you do make install all binaries will be copied to %YARP_DIR%/bin and %YARP_DIR%/lib.

Of course you can customize the installation directory as you wish, however the remainder of the documentation assumes the above configuration.

- Depending on the hardware on your system you might want to compile additional device drivers. This is done for example on the pc104. Instruction for doing this are reported elsewhere.

Compile

Now we are ready to compile. Open the YARP visual studio project file in %YARP_DIR% and compile it.

Example

Now we're ready to run a simple Yarp code to test the installation so far. You might want to prepare a yarp.conf file in the conf directory similar to this one:

```
127.0.0.1 10000
```

```
// start network description, don't forget to separate "Node="
// and names with space
[NETWORK_DESCRIPTION]
[END]
```

which tells Yarp (the server) to start on the localhost and respond to port 10000. This allows Yarp applications to find the name server (see next chapter).

You can then try running the server. On a terminal window, type:

```
yarpserver
```

and you should see:

yarp: Port /root active at tcp://127.0.0.1:10000 Name server can be browsed at http://127.0.0.1:10000/ yarp: Bootstrap server listening at mcast://224.2.1.1:10001

if you type on a web browser http://127.0.0.1:10000 you get information about the name server (registered ports, info, etc.).

For the time being we can just check functionality by running a simple example. On another terminal type:

```
yarp read /portread
```

on a third terminal:

```
yarp write /portwrite
```

and on yet another terminal:

```
yarp connect /portwrite /portread
```

you'll see the effect on the name server:

```
yarp: registration name /portwrite ip 127.0.0.1 port 10012 type tcp
yarp: registration name /portread ip 127.0.0.1 port 10002 type tcp
```

Now, anything typed on the yarp write will be sent and printed on the read side.

Options

If you need to compile devices that provide interface to the hardware you can follow this link Compilation on the pc104

Compile the iCub software

Generate makefiles

First you need to generate make files. Run cmake. Point **both** the source code and binary directories ("Where is the source code" and "Where you put the source files" entries in the gui) to %ICUB_ROOT%. Important: the iCub software does not allow to build binaries in a different directories.

You don't need particular options. If you want to compile using optimization just set:

- CMAKE_BUILD_TYPE to "Release"

Other options are:

- CREATE_GUIS_GTK
- CREATE_GUIS_GTKMM
- CREATE_GUIS_QT

These options are recommended, because they enable compilation of some useful GUIs. Important: these options can be enabled only if you have installed the required libraries: gtk, gtkmm and qt (see PrepareWindows).

- Similarly to YARP, by default make will build executables and libraries in %ICUB_ROOT%/bin and %ICUB_ROOT%/lib. You can customize where "make install" will copy these files by setting: CMAKE_INSTALL_PREFIX to something you like.

If you need to compile optional devices (for example devices that provide interface to the hardware) you can follow these links:

CompileDevices

CompileiCubDevices

iCubModulesList

Compile

Open the project files in %ICUB_ROOT%, compile and install the executables.

Compilation on the pc104

From Wiki for RobotCub and Friends

The pc104 runs Linux Debian (Live).

Compilation on the pc104 follows the same procedure we saw in Section 6 of the manual. The main difference is that in addition you need to enable certain flags in cmake that enable compilation of external modules. Some of these modules are generic and open source and are available in YARP; others are available in iCub because they are specific to the robot and in certain case require proprietary code (the API shipped with the hardware).

The Debian Live that comes with the robot should be configured correctly (in particular the environment of the iCub user). For robots shipped after August 2009 the following steps have been already done for you. If you need to upgrade the software on the pc104 (or you are upgrading from the CVS repository), you need to follow these instructions.

Before you go ahead make sure the Debian Live on the pc104 is configured to use SVN (image > 1.3). Section 9.1 gives more details on the pc104 Debian Live and instructions on how to update it.

Contents

- 1 Optimization
- 2 List of modules
- 3 Compilation of modules
  - 3.1 YARP modules
  - 3.2 iCub modules
  - 3.3 iCub 1.1 modules

Optimization

Do not forget to enable optimization, this will improve performances a lot. In cmake for both YARP and iCub set:

```
CMAKE_BUILD_TYPE: Release
```

List of modules

You need the following modules:

- serial and serialport: face expressions
- xsensmtx: inertial sensor
- canmotioncontrol: communication with the can bus
- pcan: plx can interface
- dragonfly2: cameras
- portaudio: microphones

and the calibrators:

- icubarmcalibrator
- icubheadcalibrator

- icublegscalibrator

For **iCub 1.1** (force sensors and hand position sensors) users: see at the end of this page.

Compilation of modules

Some modules are compiled with YARP others, specific to the robot, are compiled with iCub.

YARP modules

```
cd $YARP_DIR
ccmake $YARP_ROOT
```

Set:

```
CREATE_DEVICE_LIBRARY_MODULES:ON
```

Configure (hit c):

Now enable:

```
ENABLE_yarpmod_serial
ENABLE_yarpmod_serialport
ENAVLE_yarpmod_portaudio
```

Configure and generate makefiles.

Compile YARP:

```
make
```

To verify the procedure type:

```
yarpdev --list
```

among the others the list should contains also the new devices:

```
Device "serial", C++ class ServerInertial, is a network wrapper,
Device "serialport", C++ class SerialDeviceDriver, wrapped by "serial"
Device "portaudio", C++ class ...
```

iCub modules

The procedure in this case is similar. For licensing reasons not all the code is available in the repository. First you need to get the device drivers for the ecan and pcan modules. We provide precompiled binaries for the pc104 here:

http://eris.liralab.it/iCub/downloads/drivers/linux

Get the version of the precompiled drivers that matches your system, probably: http://eris.liralab.it/iCub/downloads/drivers/linux/pc104etchnhalf-bin-1.1.tgz

Download this in /usr/local/src/robot and unpack:

```
tar xvf pc104etchnhalf-bin-1.0.tgz
```

this will create a directory called *drivers* that contains the libraries you need.

Now you are ready to compile the new modules.

```
cd $ICUB_ROOT
ccmake ./
```

The following options are as usual:

```
CMAKE_BUILD_TYPE:Release
CREATE_GUIS_GTK:ON
CREATE_GUIS_GTKMM:ON
```

In addition enable compilation of modules, set:

```
USE_ICUB_MOD:ON
```

Hit c to configure. A long list of devices (in the form of ENABLE_icubmod_*) will appear.

You need to enable:

```
ENABLE_icubmod_pcan
ENABLE_icubmod_canmotioncontrol
```

```
ENABLE_icubmod_dragonfly2
ENABLE_icubmod_logpolarclient
ENABLE_icubmod_logpolargrabber
ENABLE_icubmod_xsensmtx
ENABLE_icubmod_icubarmcalibrator
ENABLE_icubmod_icubheadcalibrator
ENABLE_icubmod_icublegscalibrator
```

For iCub 1.1 (force sensors) users: see at the end of this page.

cmake should generate make files. Possible errors:

- pcan/ecan fails to detect API(s): check that you have unpacked plxCanApi/esdCanApi in /usr/local/src/robot /drivers. cmake uses the environment variables PLXCANAPI_DIR/ESDCANAPI_DIR to locate these libraries. If you the pc104 has a Debian Live image >= 1.4 these should be already set, otherwise you have to do this manually.

compile:

```
make
```

To verify the procedure type:

```
icubmoddev --list
```

among the others the list should contains also the new devices:

```
Device "dragonfly2", C++ class DragonflyDeviceDriver2, wrapped by "grabber"
Device "pcan", C++ class PlxCan, has no network wrapper
...
```

iCub 1.1 modules

In iCub 1.1 (equipped with force sensors) add:

- ecan
- icubarmcalibratorj8
- icubhandcalibrator

in cmake enable:

```
ENABLE_icubmod_ecan
ENABLE_icubmod_icubarmcalibratorj8
ENABLE_icubmod_icubhandcalibrator
```

7. Software, YARP

YARP Tutorial

From Wiki for RobotCub and Friends

Contents

- 1 Getting Started
- 2 Communicating
- 3 Image Processing
- 4 The Fake and Real Robot
- 5 Reference tutorials
- 6 Instructions for the Instructor

Getting Started

- Installing YARP: Getting_YARPed

- YARP Installation Check - make sure YARP is installed correctly on your computer.

- YARP Server Check - make sure all the computers you are working with can find the YARP "name server".

Communicating

- YARP Read and Write - If you have two computers sitting beside each other on a local network, you can try communicating between them using YARP. Maybe team up with a colleague for this.
- For more explanation on what is going on here, read Port tutorial 1 (http://eris.liralab.it/yarp/specs/dox/user/html/note_ports.html) .
- Try out some examples in $YARP_ROOT/example/os/
  - Tips on finding YARP on your computer.
- If you like, you can try playing a multi-user Game.

Image Processing

- Working with Image Streams - get started on processing images from the robot or a simulation.

- Then you can work on The Kibitzer, a project to process data coming from a robot head.

The Fake and Real Robot

(Wait for a teaching assistant to ask you to read this section.)

- Read information here on the Fakebot, a fake robot "head" to test on.

- The real robot camera is at:

```
/icub/cam/left
```

- (Check with teaching assistant if the robot is really running before trying to connect).
- Please use "mcast" (broadcast) when reading from the robot camera. This is more efficient when many people are viewing it. All you need to do is add "mcast" to your connection commands:

```
yarp connect /icub/cam/left /YOUR/PORT mcast
```

or from code:

```
Network::connect("/icub/cam/left","/YOUR/PORT","mcast");
```

- When you are ready to control the robot, and the teaching assistant says the robot is ready, read Output your target

Reference tutorials

- Port tutorial 1 (http://eris.liralab.it/yarp/specs/dox/user/html/note_ports.html)
- port tutorial 2 (http://eris.liralab.it/yarp/specs/dox/user/html/port_expert.html)
- Official YARP documentation (http://eris.liralab.it/yarp/specs/dox/user/html/index.html)
- Motor control
- Device tutorial (http://eris.liralab.it/yarp/specs/dox/user/html/note_devices.html)

Instructions for the Instructor

> The notes here are to assist whoever is running this tutorial. Students can ignore this, unless they are interested in what's running in the background.

You'll need to dedicate a computer to run some servers on. On that machine, do the following (instructions assume a Linux machine):

- Start a yarp server, if you don't have one running already in the lab.

```
yarp server
```

- Start a server for the online Game.

```
cd $YARP_ROOT/example/game/game_server
cmake .
make
./game_server
# game is available on port /game
```

- Start a "fakebot" fake robot server.

```
cd $YARP_ROOT/example/tutorial
cmake .
make
# edit fakebot.ini, remove /USERNAME from port names if present
./run_fakebot --file fakebot.ini --name fakebot
# fakebot is available on port /fakebot/camera, /fakebot/motor/rpc:i
# type help to "yarp rpc /fakebot/motor/rpc:i
```

- Start a "tracker" process - this isn't strictly necessary, but it gives a safe proxy for controlling the icub robot.

```
cd $ICUB_ROOT/src/tracker
cmake .
make
tracker --rel
# students can send summer-school style messages to /tracker/pos
```

- Make a view:

```
yarpview --name /admin/view --out /admin/clicker
```

- Connect things

```
yarp connect /fakebot/camera /tracker/img
yarp connect /tracker/img /admin/view
yarp connect /admin/clicker /tracker/pos
yarp connect /tracker/pos /fakebot/motor/rpc:i
```

Ideally, a physical robot will also be set up for the students to try things out on. There are no easy instructions on this, that depends on you. The tutorial assumes the iCub robot is available, with images on port:

```
/icub/cam/left
```

and that "summer school" format messages can be sent to control the head at:

```
/icub/target
```

(see Output your target)

8. Software, dependencies

Device drivers

From Wiki for RobotCub and Friends

In this section you find instructions to compile and install device drivers required to run the robot.

Code compilation on the pc104 is covered in Section 6.6.

You can install device drivers on your own machine by following instructions below.

Contents

- 1 Cameras
- 2 CAN bus
- 3 Inertial Sensor
- 4 Face expressions
- 5 Microphones

Cameras

Dragonfly cameras need the vendor's device driver in Windows or the firewire device drivers in Linux. See:

- Linux
- Windows

CAN bus

Proprietary device drivers for the esd can device:

- Esd CAN bus on Linux
- Esd CAN bus on Windows

- plx device. This is available only for Linux. Since this device is used only on the pc104 we point you to Section 6.6.

Inertial Sensor

No special library is required. Just add the xsensmtx in iCub (as described in Section 6.6).

Face expressions

Facial expressions are controlled using serial and serialport modules in YARP, see instructions in Section 6.6.

Microphones

Add portaudio to YARP as in Section 6.6.

List of Dependencies

From Wiki for RobotCub and Friends

Please try to respect this list of dependencies, it is important in order to develop code that can be integrated in the main build. If you need to update this list, please let us know. We cannot guarantee that your request will be satisfied, but we can update this list if there are good reasons. Please understand that we have to maintain this list as stable as possible and that new dependencies create overhead to everyone. To us as maintainers because we have to keep up to date the list of instructions for installing and compiling the libraries, to users because they have to install all the required dependencies to build the code.

Libraries

This is the list of dependencies for the iCub software. You should make sure your code compiles with the following libraries. If you develop using more recent libraries please make sure to check that you do use only features that are available in the releases listed below, otherwise it will not be possible to integrate your code.

- GTK/GTKMM Rel. 2.4
- QT3

- Gnu Scientific Library, GSL Rel. 1.8

- OpenCV 0.9

- Open Dynamics Engine: ODE Rel. 0.10
- Simple DirectMedia Layer: SDL Rel. 1.2
- Interior Point OPTimizer library: Ipopt Rel. 3.5.0

Check also configuration of compile test servers: http://eris.liralab.it/iCub/dox/html/compile_status.html.

Compilers and Tools

Make sure your code can be compiled with the following tools.

- Compilers
  - Linux: gcc up to 4.4.1
  - Windows: Visual Studio 2005/2008

- Tools, CMake:
  - Linux: version 2.4
  - Windows: version 2.6

9. Software, iCub

The Linux on the pc104

From Wiki for RobotCub and Friends

Contents

- 1 Organization
  - 1.1 Environment
- 2 Update the usb key on your pc104
  - 2.1 Download
  - 2.2 Burn a new image
  - 2.3 Testing the new image
- 3 Startup scripts update
- 4 Further customization

This page is about the Debian Linux that runs on the pc104, a Live Debian customization (see http://debian-live.alioth.debian.org).

From time to time images are updated. To determine what version of the image you are running you can have a look at the following files in the root of the filesystem:

- VERSION: contains the version of the image
- ChangeLog: list changes to the image

Organization

The Linux on the pc104 is a Debian etch with kernel 2.6.24 (etchnhalf).

In the "open call" configuration it is built as follows:

- static ip 10.0.0.2
- users: icub and root

During boot a script mounts the directory from 10.0.0.1:/exports/code-pc104 to /usr/local/src/robot. This directory contains other files used to finish the initialization, plus the code repositories.

```
/usr/local/src/robot/yarp2: yarp repository
/usr/local/src/robot/iCub: iCub repository
```

Environment

The environment of the user icub is configured as indicated in Section 6.1 (see ~/.bashrc). In addition since version 1.4 the following environment variables have been added:

```
 ESDCANAPI_DIR=/usr/local/src/robot/drivers/esdCanApi
 PLXCANAPI_DIR=/usr/local/src/robot/drivers/plxCanApi
```

These directories are used by cmake to localize the APIs required to compile some of the modules that run on the pc104 (can bus devices). See Section 6.6 for details.

Update the usb key on your pc104

Download

First you need to download the new image. We provide the standard images that come with the robot here: http://eris.liralab.it/iCub/downloads/pc104-images/

Images are as follows:

- image.1.3-OC.img for "open call" robots. Configured for CVS.
- image.1.4-OC.img for "open call" robots. Configured for SVN.

Burn a new image

This can be probably done in different ways. We provide instructions for Linux.

First you need to identify the device node that corresponds to your usb key. There are a couple of options:

Procedure 1. Type:

```
ls -la /dev/disk/by-id/usb-*
```

you will get something like:

```
lrwxrwxrwx 1 root root  9 2007-08-11 12:46
  /dev/disk/by-i/usb-Kingston_DataTraveler_II+_5B720CB323C1->../../sdc
```

here */dev/sdc* is the device node of your usb key.

Procedure 2. Alternatively you can try unplugging and plugging in the usb key and see what happens in /dev/sd*.

Important: make sure you identify the device correctly, otherwise you risk to wipe the content of your hard drive. To be sure you can try procedure 1 and 2 (above) a couple of time.

Now you can burn the new image by typing **as root**:

```
dd if=IMAGE_FILE of=DEVICE_NODE
```

where IMAGE_FILE is the name of the new image (e.g. image.1.4-OC.img) and DEVICE_NODE is, for example, /dev/sdc.

Important:

- make sure that you overwrite the whole device, not just the first partition of it that you normally mount, i.e. /dev/sdx instead of the more familiar /dev/sdx1.

- Don't forget to unmount the usb key before unplugging it.

More instructions are available here: http://wiki.debian.org/DebianLive/Howto/USB

Testing the new image

Plug the usb key in the usb port of the pc104, and turn it on. Wait some time (2 minutes or so) and connect to the pc104 with ssh:

```
ssh pc104
```

Before you do this, you need to fix or remove the known_hosts file:

```
rm /home/icub/.ssh/known_hosts
```

Startup scripts update

Besides the usual initialization sequence the pc104 loads and executes a list of scripts that are in $ICUB_ROOT/../pc104/hooks.

A copy of those scripts in also in the repository, you can update them from $ICUB_ROOT/pc104/startupscripts/opencall/hooks:

- cd $ICUB_ROOT
- svn update
- copy ./pc104/startupscripts/opencall/hooks/* ../pc104/hooks -r

- Customize S08_robotname.sh according to your setup.

Further customization

Unfortunately the current iCub Live is not writable. This means that **all** changes are lost when you reboot the pc104. Making a writable partition is possible, but it proved to be harder than expected. We are working on this. In the meanwhile there are a couple of options.

- If you need to perform some operations at startup you can add or modify scripts in the $ICUB_ROOT/../pc104/hooks directory. This directory is mounted from the remote server and is persistent. Scripts from the "hooks" directory are executed in alphabetical order from /etc/rc.local as root.

- You can rebuild your own version of the image (we can provide the configuration files/options if you like).

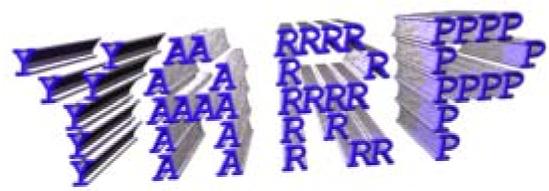
Deliverable 8.3

From Wiki for RobotCub and Friends

Contents

- 1 Software Architecture
- 2 Developing iCub Software with Yarp: A Novice User's View
- 3 Further information on Yarp
- 4 iCub specific material
- 5 RobotCub licenses

Software Architecture



The RobotCub software architecture is based on YARP, an open-source framework that supports distributed computation with an eye at robot control and efficiency. In short, we decided to adopt YARP for RobotCub while keeping the two project separated. Yarp is a set of libraries which can be embedded in many different systems and robots, while for the iCub we customize Yarp to handle its hardware and devices. iCub's code is client code with respect to YARP.

YARP provides a set of protocols and a C++ implementation for:

- Inter-process communication on a local network, in fact enabling parallel multi-processor computation.
- Standardization of the hardware interface through run-time dynamically loadable modules.
- Data types for images, vectors, buffers, etc.
- Various interfaces to commonly used open-source packages (e.g. openCV).

To learn more about the philosophy of Yarp you can see the paper:

```
G. Metta, P. Fitzpatrick, L. Natale. YARP: yet another robot platform.
In the International Journal on Advanced Robotics Systems, Special Issue
on Software Development and Integration in Robotics. March 2006.
 (pdf)
```

or a more recent submission:

```
P. Fitzpatrick, G. Metta, L. Natale. Towards Long-Lived Robot Genes. March 2007.
 (pdf)
```

A manual including a description of Yarp's standard protocols is available:

- Yarp reference manual (pdf) (http://eris.liralab.it/yarp/specs/dox/min/latex/refman.pdf)

And full documentation (online) from:

- Yarp documentation (http://eris.liralab.it/yarp/specs/dox/user/html/index.html)
- iCub software documentation (http://eris.liralab.it/iCub)

Developing iCub Software with Yarp: A Novice User's View

There are three ways you can view Yarp as an implementation platform for iCub software.

1. The first is to see it as a network of processes with which your code interfaces via ports using Yarp-compatible iCub control and data acquisition protocols. These protocols are specific to the iCub. Your code should also be written as a Yarp process, with a well-specified port-based interface protocol.
2. The second is a device view. Here, your code is more closely coupled with Yarp. Yarp is simply a class hierarchy and your iCub application code is directly linked with the Yarp objects, with control and data acquisition being achieved by method invocation.
3. Both of these views are *remote* views (in the sense that you can assume that all the iCub devices are set up and just need to be polled). There is a third view that is *local* and it is a counter-part of the second - device - view above. In this case, however, your iCub application software has much greater control but it also has much greater responsibility for configuration of the iCub devices and for bootstrapping them.

Most application developers will choose approach no. 1. Those who are particularly concerned about efficiency will choose no. 2, and those with very strong timing constraints will choose approach no. 3.

Using view no. 1, an application will typically comprise several Yarp processes. This means that to run your iCub application, you need to invoke each process and also instantiate the port-based communication between them. You *can* instantiate the communications between the Yarp iCub processes with in-line code embedded in your software but the Yarp philosophy is to decouple the process functionality from the specification of the inter-process (port-to-port) connections. This encourages modular software with reusable processes that can be used in a variety of configurations that are not dependent on the functionality of the process or embedded code. Thus, e.g., the processes might be invoked and the connections instantiated using a script.

We plan on implementing the iCub cognitive architecture as a set of Yarp processes. That is, we expect that each of the iCub phylogenetic abilities as well as the modules for their modulation, for prospection and anticipation, and for self-modification, will be implemented as distinct Yarp processes.

Further information on Yarp

For further information on how to obtain, compile, and use Yarp please see:

- SourceForge.net (http://www.sf.net) .
- This Wiki at RobotCub.
- The Yarp home page (http://yarp0.sf.net) .

iCub specific material

The RobotCub repository is described at:

- Notes on CVS: RobotCub repository
- iCub software documentation (http://eris.liralab.it/iCub)
- Details are also available from Deliverable 8.2 (http://www.robotcub.org/index.php/robotcub/more_information/deliverables/deliverable_8_2_pdf) and from Deliverable 7.3 (http://www.robotcub.org/index.php/robotcub/more_information/deliverables/deliverable_7_3_pdf)
- RobotCub CVS server (http://www.robotcub.org/iCub)
- VVV Summer School 2006

RobotCub licenses

iCub Cognitive Architecture

From Wiki for RobotCub and Friends

Contents

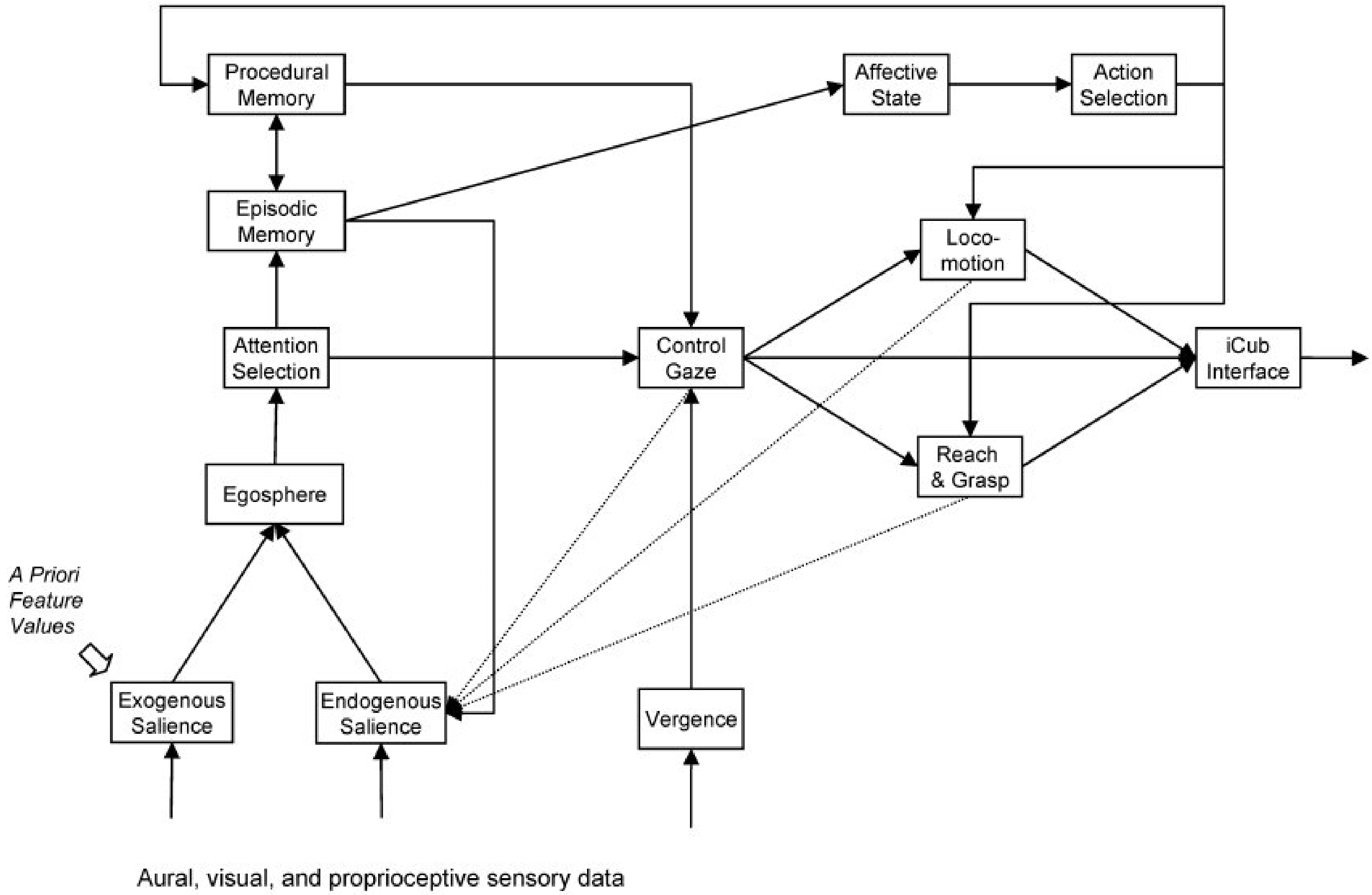
- 1 The Evolution of the Architecture
- 2 Differences from previous version
- 3 Notes
- 4 Links

The Evolution of the Architecture

The iCub cognitive architecture is the result of a detailed design process founded on the developmental psychology and neurophysiology of humans, capturing much of what is known about the neuroscience of action, perception, and cognition. This process and the final outcome is documented in Deliverable D2.1: A Roadmap for the Development of Cognitive Capabilities in Humanoid Robots (http://www.robotcub.org/index.php/robotcub /more_information/deliverables/deliverable_2_1_pdf) .

The architecture itself is realized as a set of YARP executables, typically connected by YARP ports. Early prototypes were developed at a RobotCub project meeting at the University of Hertfordshire in July 2007 as an exercise in consolidating the software development effort of the project partners. Several subsequent versions were produced at the RobotCub Summer School 2007 VVV '07. These prototypes were developed in parallel with the D2.1 Roadmap effort mentioned above. These two strands of design effort converged in the cognitive architecture shown below (Version 0.4). Previous versions can be accessed via the links at the end of the page. VVV '09 addressed the development of the architecture's (auto-associative) episodic and (hetero-associative) procedural memories.

The immediate purpose in developing the cognitive architecture is to create a core software infrastructure for the iCub so that it will be able to exhibit a set of target behaviours for an Empirical Investigations.



Differences from previous version

- Removed the `tracker` (should be handled by attention/salience sub-system)
- Removed the `face localization` (should be handled by attention/salience sub-system)
- Removed the `hand localization` (should be handled by attention/salience sub-system)
- Removed the `sound localization` (should be handled by salience module)
- Removed the `attention selection`
- Added `Exogenous Salience` and `Endogenous Salience`
- Added `Locomotion`
- Added `Matching`
- Added `Auto-associative episodic memory`
- Added `Hetero-associative procedural memory`
- Added `Affective state`
- Added `Action selection`

Notes

- Gaze implies 7 DoF: head and eyes
- Locomotion paradigm: “go where you are looking”
- Reaching paradigm: “reach where you are looking”
- Endogenous and exogenous salience implies salience based on internal and external events, respectively
- Gaze, reaching, and locomotion motor activities condition endogenous salience: i.e. motor states condition attention
- Sensory inputs condition exogenous salience: i.e. attention conditions motor states
- Episodic memory is memory of autobiographical events. Initially, this is purely visual and implemented as an auto-associative memory. Later it will be multimodal and will include sound as well as associated emotions. It will then have to be implemented as a hetero-associative network of unimodal auto-associative memories.
- Episodic memory storage is conditioned by poor matching and high salience
- Procedural memory is defined to mean perception-action event sequence
- Procedural memory recall:
  - Event A & Event D inputs recall sequence of intermediate events
  - Event A input recalls Event B (subsequent event)
- Affective state is a competitive network of three motives:

    - Distraction (exogenous salience prevalent)
    - Curiosity / Exploration (endogenous salience prevalent)
    - Social engagement (exogenous and endogenous salience balanced)
- Action selection is not a winner-take-all process: one or more actions are disinhibited
- The developmental drive is to construct a procedural memory that improves prediction

A more detailed description of the behaviour of each module and circuit in this architecture will be added in due course (both here on the iCub wiki and in Deliverable D2.1: A Roadmap for the Development of Cognitive Capabilities in Humanoid Robots (http://www.robotcub.org/index.php/robotcub/more_information/deliverables/deliverable_2_1_pdf)).

Links

- iCub cognitive architecture version 0.1
- iCub cognitive architecture version 0.2
- iCub cognitive architecture version 0.3
- iCub cognitive architecture version 0.4

The following links are to early versions of the iCub "software architecture", a design for an iCub application (i.e. a set of YARP modules) that approximated some of the elementary aspects of the iCub cognitive architecture which now supercedes them. These early versions have all now been deprecated, as has the title "software architecture" in this context. Software Architecture now refers, as it originally did, to the YARP (http://eris.liralab.it/wiki/Deliverable_8.3) system.

- iCub software architecture version 0.1
- iCub software architecture version 0.2
- iCub software architecture version 0.3
- iCub software architecture version 0.4

- See also the current draft iCub YARP module specifications
- iCub brain (http://eris.liralab.it/brain) - current source code documentation

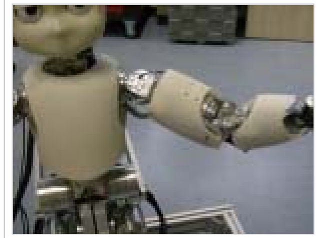
- [Episodic Memory Module (http://eris.liralab.it/iCub/dox/html/group__icub__episodicMemory.html)] and Episodic Memory Specification
- [Procedural Memory Module (http://eris.liralab.it/iCub/dox/html/group__icub__proceduralMemory.html)] and Procedural Memory Specification

- VVV09 Cognitive Architecture Group

Deliverable 2.2

From Wiki for RobotCub and Friends

Software Implementation of the iCub Cognitive Architecture (version 1.0)



This is the addendum to the Deliverable 2.2 (http://www.robotcub.org/index.php/robotcub/more_information/deliverables/deliverable_2_2_pdf) which is the first release of the iCub Cognitive Architecture. An placeholder document has been added to the RobotCub.org website with a pointer to this page. The demonstrations are provided here as videos.

The software implementation is a collection applications comprising YARP modules: each application realizes a given behaviour and runs independently on the iCub. The applications are available from the iCub applications (http://eris.liralab.it/iCub/dox/html/group__icub__applications.html) documentation. The modules are described also in the iCub modules (http://eris.liralab.it/iCub/dox/html/group__icub__module.html) documentation.

The final goal is to have an application which instantiates all the modules in the iCub Cognitive Architecture and which realizes the behaviours encapsulated in Empirical Investigations. At that point, the software implementation will be redesignated version 2.0.

Each application is described below, organized by workpackage.

Contents

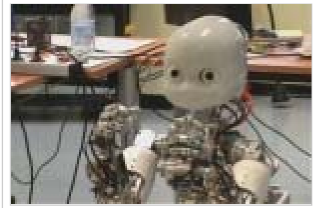
- 1 Important note
- 2 Generic (WP7 & WP8)
- 3 Attention system (WP3)
- 4 Body schema (WP3 and WP5)
- 5 Crawling (WP3)
- 6 Drumming (WP3)
- 7 Cartesian control (WP3)
- 8 Affordances (WP4 & WP5)
- 9 Interaction histories (WP6)

Important note

- Please note that often the browser won't display/embed your videos correctly because of coded and/or other player incompatibilities. In that case, we recommend downloading them to your computer and then playing them from there.

Generic (WP7 & WP8)

- These are older videos to show the functionality of the robot and they are typical mechanical stress-tests.

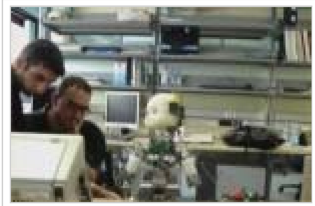

the iCub hands

- The main Doxygen documentation of the basic control modules is available as the iCub application called **demoy3** and can be browsed here (http://eris.liralab.it /iCub/dox/html/group__appdemos.html)

- Videos:
  - icub_yoga.wmv (http://eris.liralab.it/misc/icubvideos/icub_exercising.wmv)
  - icub_hands.wmv (http://eris.liralab.it/misc/icubvideos/iCub_Oct07.3.wmv)

- Paper:
  - G. Metta, G. Sandini, D. Vernon, L. Natale, F. Nori. **The iCub humanoid robot: an open platform for research in embodied cognition**. In PerMIS: Performance Metrics for Intelligent Systems Workshop. Aug 19-21, 2008, Washington DC - USA -PDF- (http://www.robotcub.org/misc/papers /08_Metta_Sandini_Vernon_etal.pdf)

Attention system (WP3)


the attention system

- The main Doxygen documentation of the attention system is available as the iCub application called **attention_distributed** and can be browsed here (http://eris.liralab.it/iCub/dox/html/group__icub__attention__distributed.html)

- Video: iCub-attention.wmv (http://eris.liralab.it/misc/icubvideos/iCub-attention-may14-2008.wmv)

- Paper:
  - Ruesch J., Lopes, M., Hornstein J., Santos-Victor J., Pfeifer, R. **Multimodal Saliency-Based Bottom-Up Attention - A Framework for the Humanoid Robot iCub**. International Conference on Robotics and Automation, Pasadena, CA, USA, May 19-23, 2008. pp. 962-967. -PDF- (http://www.robotcub.org/misc/papers/08_Ruesch_Lopes_Hornstein_Victor_Pfeifer.pdf)

Body schema (WP3 and WP5)


iCub reaching

- The main Doxygen documentation of the body schema is available as the iCub application called **lasaBodySchema** and can be browsed here (http://eris.liralab.it /iCub/dox/html/group__lasabodyschema.html)

- Videos:
  - bodyschema.avi (http://eris.liralab.it/misc/icubvideos/body_schema_sim.avi) , this video shows the learning procedure in simulation, or .wmv file (http://eris.liralab.it/misc/icubvideos/body_schema.wmv) .
  - fingerreach.avi (http://eris.liralab.it/misc/icubvideos/icub_finger_s.avi) or fingerreach.wmv (http://eris.liralab.it/misc/icubvideos/icub_finger_s.wmv) , learning to reach using a different effector (a different finger as the end-point).
  - gazing.avi (http://eris.liralab.it/misc/icubvideos/icub_gazing_s.avi) or gazing.wmv (http://eris.liralab.it /misc/icubvideos/icub_gazing.wmv) , learning to gaze appropriately (head-eye coordination).
  - reaching.avi (http://eris.liralab.it/misc/icubvideos/icub_reaching_s.avi) or reaching.wmv (http://eris.liralab.it /misc/icubvideos/iCub-reach-epfl.wmv) , learning to reach (whole body).

- Papers:
  - M. Hersch, E. Sauser and A. Billard. **Online learning of the body schema**. International Journal of Humanoid Robotics, (2008). -PDF- (http://www.robotcub.org/misc/papers/08_Hersch_Sauser_Billard.pdf)
  - M. Hersch, **Adaptive sensorimotor peripersonal space representation and motor learning for a humanoid robot**. PhD thesis (2009). link (http://library.epfl.ch/theses/?nr=4289)

Crawling (WP3)

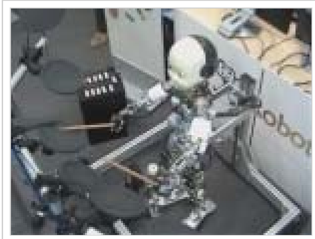
- The main Doxygen documentation of the crawling controller is available as the iCub application called **missing_application** and can be browsed [http:// here]


the iCub crawling

- Videos:
  - crawling.wmv (http://eris.liralab.it/misc/icubvideos/first_crawl.wmv) , a few steps with the crawling controller.
- Paper (and more):
  - -Deliverable 3.4- (http://ares.lira.dist.unige.it/ezpublish/index.php /robotcub_admin/content/download/1172/4095/file/DELIVERABLE_3_4.pdf)
  - A presentation on the controller structure: presentation.pdf
  - S. Degallier, L. Righetti, L. Natale, F. Nori, G. Metta and A. Ijspeert. **A modular bio-inspired architecture for movement generation for the infant-like robot iCub**. In Proceedings of the second IEEE RAS / EMBS International Conference on Biomedical Robotics an Biomechatronics (BioRob), 2008. -PDF of submitted paper-

Drumming (WP3)

- The main Doxygen documentation of the drumming controller is available as the iCub application called **drummingEPFL** and can be browsed here (http://eris.liralab.it/iCub/dox/html/group__icub__drummingEPFL.html)


the iCub drumming

- Videos, various videos of the robot drumming:
  - drumming1.wmv (http://eris.liralab.it/misc/icubvideos/icubDrumPart1.wmv)
  - drumming2.wmv (http://eris.liralab.it/misc/icubvideos/icubDrumPart2.wmv)
  - drumming3.wmv (http://eris.liralab.it/misc/icubvideos/icubDrumPart3.wmv)
  - drumming4.wmv (http://eris.liralab.it/misc/icubvideos/icubDrumPart4.wmv)
  - automatica08.wmv (http://eris.liralab.it/misc/icubvideos/automatica08-edited.wmv)
- Paper (and more):
  - -Deliverable 3.4- (http://ares.lira.dist.unige.it/ezpublish/index.php/robotcub_admin/content/download /1172/4095/file/DELIVERABLE_3_4.pdf)
  - A presentation on the controller structure: presentation.pdf
  - S. Degallier, L. Righetti, L. Natale, F. Nori, G. Metta and A. Ijspeert. **A modular bio-inspired architecture for movement generation for the infant-like robot iCub**. In Proceedings of the second IEEE RAS / EMBS International Conference on Biomedical Robotics an Biomechatronics (BioRob), 2008. -PDF of submitted paper-

Cartesian control (WP3)

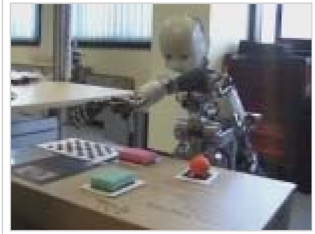
- The main Doxygen documentation of the cartesian controller is available as the iCub application called **armCartesianController** and can be browsed here (http://eris.liralab.it/iCub/dox/html/group__icub__armCartesianController.html) . The implementation includes also the trajectory generation implemented for the learning of the body schema (see above).


the Jacobian-based Cartesian controller

- Videos, Cartesian controller in action:
  - icub_guicontrolled.avi (http://eris.liralab.it/misc/icubvideos /icub_guicontrolled.avi) or icub_guicontroller.wmv (http://eris.liralab.it /misc/icubvideos/icub_guicontroller.wmv) , iCub arm receives commands from a GUI developed in MATLAB: the complete pose (position+orientation) is controlled.

- Paper:
  - Not yet!

Affordances (WP4 & WP5)


the iCub running the affordance modules

- The main Doxygen documentation of the affordance experiment is available as the iCub application called **missing application** and can be browsed here (http://eris.liralab.it/iCub/dox/html/)

- Video, an initial video of the affordance experiment on the iCub:
  - affordances.wmv (http://eris.liralab.it/misc/icubvideos/affordances.wmv)

- Paper (and more):
  - -Deliverable 4.1- (http://www.robotcub.org/index.php/robotcub/content/download/1039/3660/file/RC_IST_AB_Deliverable_D4.1.pdf)
  - Lopes, M., Melo, F., and Montesano, L. **Affordance-Based Imitation Learning in Robots**. IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, USA, October 2007.-PDF of submitted paper- (http://www.robotcub.org/misc/papers/07_Lopes_Melo_Montesano.pdf)

Interaction histories (WP6)


the Peekaboo game

- The main Doxygen documentation of the interaction histories experiment is available as the iCub application called **iha_manual** and can be browsed here (http://eris.liralab.it/iCub/dox/html/group__icub__iha__app.html)

- Video:
  - iha.wmv (http://eris.liralab.it/misc/icubvideos/iha.wmv) , full video of the experiment.

- Paper (and more):
  - -Deliverable 6.4- (http://www.robotcub.org/index.php/robotcub/content/download/1144/4009/file/d6.4.pdf)
  - Mirza, N. A., Nehaniv, C. L., Dautenhahn, K., AND te Boekhorst, R. 2005. **Using sensory-motor phase-plots to characterise robot-environment interactions**. In Proc. of 6th IEEE International Symposium on Computational Intelligence in Robotics and Automation. -PDF of submitted paper- (http://www.robotcub.org/misc/papers/05_Mirza_Nehaniv_Dautehahn_teBoekhorst.pdf)

Running applications

From Wiki for RobotCub and Friends

An application is a collection of modules that you need to run to instantiate a certain behavior on the robot. They go from simple applications that grab images on the disk to complex behaviors that control the robot attention or perform reaching.

Applications are described in terms of xml files. How you do this is explained in some details here:

- Managing applications

To prepare your system to run applications you need a python interpreter. Follow the instructions here:

- Prepare your system for running applications

Software interface: standard port names for hardware devices

From Wiki for RobotCub and Friends

Contents

- 1 Motor Control and Encoders
  - 1.1 Motor encoders
- 2 Sensors
  - 2.1 Cameras
  - 2.2 Microphones
  - 2.3 Inertial sensor
  - 2.4 Force/Torque Sensor
- 3 Other actuators and sensors
  - 3.1 Finger encoders
  - 3.2 Facial expressions
    - 3.2.1 Low level interface
    - 3.2.2 High level interface

Motor Control and Encoders

We have identified 6 robot parts: head, left_arm, right_arm, torso, left_leg and right_leg.

Each part has three ports for: rpc, straming input, streaming output.

All ports have are named using a prefix, followed by rpc:i, command:i, state:o respectively for rpc, streaming input and streaming output.

Prefixes for each part are:

```
/icub/head
/icub/left_arm
/icub/right_arm
/icub/torso
/icub/left_leg
/icub/right_leg
```

More informations about the robot joints and parts are reported here: ICub_joints

Motor encoders

Motor encoders are streamed on the port:

```
/icub/<part>/state:o
```

Data format: a vector containing all joints of the part (see ICub_joints), units are degrees.

Sensors

Cameras

Images from the two eyes are available in rgb format from the ports:

```
/icub/cam/left
/icub/cam/right
```

Data format: Yarp ImageOf<PixelRgb>, size depending on the framegrabber resolution.

Microphones

Data from the microphones is available on the port:

```
/icub/mics
```

Inertial sensor

Data from the inertial sensor is available on the port:

```
/icub/inertial
```

Data format: a vector of 12 values:

```
0:2 :euler angles roll, pitch yaw (degrees)
3:5 :calibrated acceleration along X,Y,Z (m/s^2)
6:8 :calibrated rate of turn (gyro), along X,Y,Z axes (rad/s)
9:11 :calibrated magnetic field X,Y,Z (arbitrary units)
```

More information about the sensor, data format and reference system are reported here: MTx specs.

Force/Torque Sensor

Six axis torque sensors on the arms (available only on iCub 1.1) are streamed on:

```
/icub/left_arm/analog:o
/icub/right_arm/analog:o
```

Data format: a vector of 6 floating point numbers.

```
0:2 :forces
3:5 :torques
```

Reference frames to be determined.

Other actuators and sensors

Finger encoders

On each hand finger positions are streamed out on the following ports (available only on iCub 1.1):

```
/icub/right_hand/analog:o
/icub/left_hand/analog:o
```

A vector of 15 floating point numbers, each corresponding to finger joints.

Facial expressions

Facial expressions are accessible through the following ports:

```
/icub/face/raw: low level interface
/icub/face/emotions: high level interface
```

Low level interface

On /icub/face/raw you have access to the low level interface. This port accepts rpc commands to control individual devices (eyelids, mounth ...).

We skip details here, more can be found in Expression control (http://robotcub.svn.sf.net/viewvc/robotcub/trunk /iCubPlatform/doc/manuals/RC_IST_110_D_000_07_EXPRESSIONS_CONTROL.pdf) .

High level interface

On /icub/face/emotions you have access through rpc to a higher level interface that allows controlling all devices together to produce facial expressions.

We report here a short summary:

rpc commands to /icub/face/emotions are

```
 set <subsystem> <emotion>
 get <subsystem>
```

<subsystem> can be: mou, eli, leb, reb, all

<emotion> can be: neu, tal, hap, sad, sur ,evi, ang, shy, cun

For example:

```
yarp rpc /icub/face/emotions
set mou hap
set eli hap
set all sad
```

More details are available here Expression control (http://robotcub.svn.sf.net/viewvc/robotcub/trunk/iCubPlatform /doc/manuals/RC_IST_110_D_000_07_EXPRESSIONS_CONTROL.pdf) .

Firmware

From Wiki for RobotCub and Friends

Contents

- 1 Bootloader update
  - 1.1 JTAG pinout in BLL and 4DC
- 2 Firmware update
- 3 Firmware versions
  - 3.1 Head
  - 3.2 Torso
  - 3.3 Left arm
  - 3.4 Right arm
  - 3.5 legs

Bootloader update

To update the bootloader you must have a programmer for Freescale DSP56F807 (USBTAP or a ParallelToJTAG programmer), CodeWarrior IDE that you can download from the http://www.freescale.com (the code of the bootloader is small then it is not required any license) and the CanLoader module. In order to see the bootloader version of the boards you must run the CanLoader module in gui mode and press the connect button within five seconds after switching on the boards. Information on where and what to buy from Freescale (Metrowerks CodeWarrior) can be found in here.

There is also a document describing the firmware versions available from here (http://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCubPlatform/doc/manuals/RC_DIST_100_D_12_01_firmware_versions.doc)

How to upload the bootloader step by step.

- Connect the JTAG to the connector P1 of the BLL or MC4 board. If you have to remove screws to do it, you have to put them in the same place after installing the bootloader. The pinout of the connector is the following

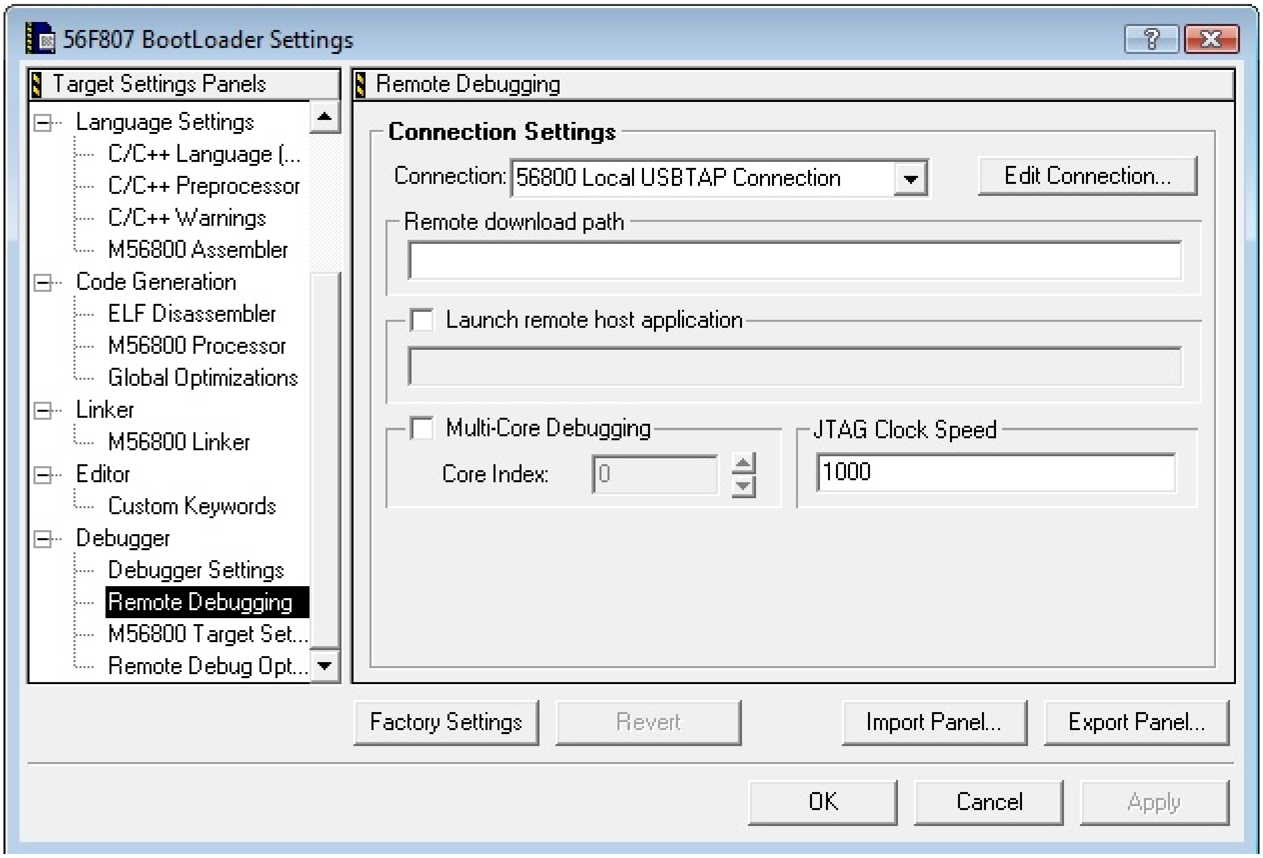
JTAG pinout in BLL and 4DC

Connector P1	Label
1	TDI
2	TDO
3	GND
4	TCK
5	~RESET
6	TMS
7	3.3V

8	~DE
9	~TRST
10	~RSTO

- Run CodeWarrior IDE and open the project $ICUB_ROOT$/src/firmware/loader56f807/RM-DownLoaderCAN.mcp

- Go to Edit -> 56807 BootLoader Setting and set the programmer you are using. In the case of the USBTAP this is the setting:



- Switch on the board

- Press the play button on the project tab and wait until the download is completed. It will take less then one minute.
- Now you must restart the board and download the right firmware using the CanLoader application in gui mode. **Note: You must download the firmware immediately after (from one to 5 seconds) the board is switched on** . So, you have to switch on the board and then press the connect button in the canLoader gui.

Firmware update

Note: the firmware upgrade is based on a command line version of the CanLoader module. Before you try to use this script please check that this module is compiled. You should also make sure that the hardware modules it needs (controlboard, canmotioncontrol, ecan/pcan) are compiled and liked correctly (this is usually the case on the robot pc104).

We here describe the procedure for updating the firmware (bugs should be reported on the mailing list robotcub-hackers@lists.sourceforge.net). Please notice that the firmware upgrade is delicate procedure which changes one of the core parts of the iCub robot software (and therefore **requires a corresponding update of the yarp and iCub modules**). If you decide to perform a firmware upgrade, **be aware that this might change the robot motor behaviour quite a lot**. Nevertheless, a firmware upgrade is always recommended because it usually correspond to improvements of the control board performances. Here are the steps for performing the firmware update:

- Connect to the PC104.
- Go the $YARP_ROOT directory and update the entire yarp module:

```
icub@pc104:YARP_ROOT$ svn update
```

- Recompile Yarp:

```
icub@pc104:YARP_ROOT$ make
```

- Go the $ICUB_ROOT directory and update the entire iCub module:

```
icub@pc104:ICUB_ROOT$ svn update.
```

- Recompile iCub:

```
icub@pc104:ICUB_ROOT$ make
```

- Go to the firmware directory:

```
icub@pc104:ICUB_ROOT$ cd $ICUB_ROOT/src/firmware/build
```

- Launch the firmware update script (this script assumes that you are using the CFW canbus board integrated in the PC104. If you are using the ESD usb canbus please change all occurences of *pcan* to *ecan* in the configuration file updateRobot.txt):

```
icub@pc104:ICUB_ROOT/src/firmware/build$ ./updateRobot.sh updateRobot.txt
```

Or alternatively you can update just a part of the robot:

```
icub@pc104:ICUB_ROOT/src/firmware/build$ ./updateRobot.sh updateLegs.txt
```

Firmware versions

We here give the information for associating the correct firmware versions to the different boards which control the iCub robot.

Head

boardLabel		canDeviceNum	boardId	firmware
0B0	->	0	1	4DC.1.11.out.S
0B1	->	0	3	4DC.1.15.out.S

Torso

boardLabel		canDeviceNum	boardId	firmware
0B3	->	0	5	2BLL.1.54.out.S

0B4	->	0	6	2BLL.1.52.out.S

Left arm

boardLabel		canDeviceNum	boardId	firmware
1B0	->	1	1	2BLL.1.50.out.S
1B1	->	1	2	2BLL.1.53.out.S
1B2	->	1	3	4DC.1.19.out.S
1B3	->	1	5	4DC.1.18.out.S
1B4	->	1	7	4DC.1.20.out.S

Right arm

boardLabel		canDeviceNum	boardId	firmware
2B0	->	2	1	2BLL.1.50.out.S
2B1	->	2	2	2BLL.1.53.out.S
2B2	->	2	3	4DC.1.19.out.S
2B3	->	2	5	4DC.1.18.out.S
2B4	->	2	7	4DC.1.20.out.S

legs

boardLabel		canDeviceNum	boardId	firmware
3B5	->	3	5	2BLL.1.51.out.S
3B6	->	3	6	2BLL.1.51.out.S
3B7	->	3	7	2BLL.1.51.out.S
3B8	->	3	8	2BLL.1.51.out.S
3B9	->	3	9	2BLL.1.51.out.S
3B10	->	3	10	2BLL.1.51.out.S

10. Standardization of methods

Organization of source code, binaries, scripts and config files

From Wiki for RobotCub and Friends

Author: Lorenzo Natale

Source files, binaries, scripts and config files are separated entities that can be placed in separate directories. This allows a better organization of your build and in turn simplifies development.

Contents

- 1 Sources
- 2 Binaries
- 3 Scripts and config/ini files
- 4 Robot name

Sources

The source tree is where you download the sources, either from CVS or from a zip/tar file. Sources can be shared between users or machines. Sometimes it is convenient to share sources between machines (or users) so that changes/updates are shared by everybody.

The environment variable ICUB_ROOT should point to the location of your sources.

Binaries

Your build directory is where you place: make/project files and files that are the result of the compilation (object files, for example). This also stores libs and executables that are produced when you compile. This directory is more difficult to share, as its content changes depending on i) the compile environment used and ii) the user's choices. If you have a cluster of similar machines (same compiler, same system libraries, compatible CPU) you can share the build directory and spare time. However, if you have machines with mixed architecture (32 versus 64 bits), or that have different Linux distributions, you need two separate build trees. Your build also contains CMake files used by other projects to use the binaries (CMake files).

ICUB_DIR should point to this directory. Typically you run:

```
cd $ICUB_DIR
cmake $ICUB_ROOT
```

Scripts and config/ini files

$ICUB_ROOT/app stores scripts and configuration files for running your applications.

Each application owns a directory in app. Inside this directory you should place a file that contains the documentation of the application and two directories:

- conf: contains all configuration files
- scripts: contains the xml script(s)

Robot name

Within the app directory there are some files that are private to certain robots (e.g. calibration files). They are stored in directories whose name matches that of the robot (e.g. iCubGenova01). These files should be well separated and are provided with the robot. We would like to keep them in the repository so that i) they are not lost and ii) we can update them remotely and propagate them to users.

Applications that need access to these parameters should be able to do so transparently.

Since July 2009 it was decided to delegate the localization of this robot specific directory to the ResourceFinder. To use this feature you need to set an environment variable called ICUB_ROBOTNAME and set it to your robot name (e.g. iCubGenova01):

```
ICUB_ROBOTNAME= name of your robot
```

for example:

```
ICUB_ROBOTNAME=iCubGenova01
```

The details of how the ResourceFinder works are provided elsewhere. In short the ResourceFinder will add app/$ICUB_ROBOTNAME/conf to the search path. Files will be searched in this directory after the context and before app/default/conf.

Summary of iCub Software Development Guidelines

From Wiki for RobotCub and Friends

Contents

- 1 Overview
- 2 The RFModule Class
- 3 Configuration
  - 3.1 Essential Command-line Parameters
    - 3.1.1 Configuration File
    - 3.1.2 Context
    - 3.1.3 Module Name and Port Names
    - 3.1.4 Robot Name
    - 3.1.5 Which Parameters Are Parsed Automatically?
  - 3.2 Configuration File Parameters
  - 3.3 Other Configuration Files
  - 3.4 An Example of how to Configure the Module
- 4 Graceful Shut-down
- 5 Thread-based Implementation
  - 5.1 Using Threads to Implement Your Algorithm
  - 5.2 An Example of how to use the Thread Class
  - 5.3 Creating, Starting, and Stopping the Thread
- 6 Run-time Interaction
  - 6.1 The respond() Method
  - 6.2 An Example of how to change module parameters at run-time
  - 6.3 Remote Connection
- 7 Documentation and Coding Guidelines
- 8 Application Description
- 9 Resources
- 10 The Complete myModule Example

Overview

Our goal in this document is to summarize all of the the different guidelines on standards for iCub software to make it easier for developers to find out what they need to do so that their code conforms to best practice (e.g. configuration management, parameter usage, port naming and renaming, use of RFModule helper class, use of threads, graceful shut-down, run-time interaction, documentation, coding standards, application description, etc). A full list of all the resources that were used in putting together this summary is provided at the end.

To help bring all this material together, we will develop a simple example module to highlight the key issues. The code for the complete module is also provided at the end.

If you want to jump in the deep end, start by reading

- Module Standards (http://eris.liralab.it/iCub/dox/html/module_standards.html) to find out the minimal responsibilities of your module with respect to handling parameters and ports, as well configuration and shut down. Then read the following tutorials:

- Resource finder overview

- How to organize the command line parameters of your modules (http://eris.liralab.it/iCub/dox/html/icub_resource_finder_basic.html)

- Organizing Parameters: Advanced Tutorial (http://eris.liralab.it/iCub/dox/html/icub_resource_finder_advanced.html)

- Using the module helper class to write a program (http://eris.liralab.it/iCub/dox/html/icub_tutorial_module.html)

The fourth of these tutorials is essential reading and introduces you to the RFModule (http://eris.liralab.it/yarpdoc/d9/d26/classyarp_1_1os_1_1RFModule.html) class, the mandatory starting point when developing iCub modules. Note that we are moving away from using Module (http://eris.liralab.it/yarpdoc/d1/d03/classyarp_1_1os_1_1Module.html) class as RFModule offers similar functionality to Module, but it adds support for the ResourceFinder class.

Alternatively, read through this page and then refer to the documents above for more detail and clarification, if necessary.

The `RFModule` Class

The starting point in writing a module for the iCub repository is to develop a sub-class of the yarp::os::RFModule (http://eris.liralab.it/yarpdoc/d9/d26/classyarp_1_1os_1_1RFModule.html) class.

First, we define a sub-class, or derived class, of the `yarp::os::RFModule` class. The module's variables - *and specifically the module's parameters and ports* - go in the private data members part and you need to override three methods:

- `bool configure();`
- `bool interruptModule();`
- `bool close();`

We will see later that there are three other methods which can be useful to override:

- `bool respond();`
- `double getPeriod();`
- `bool updateModule();`

In the following, we assume we are writing a module named `myModule`. This module will be implemented as a sub-class yarp::os::RFModule (http://eris.liralab.it/yarpdoc/d9/d26/classyarp_1_1os_1_1RFModule.html) called `MyModule` (capital M because we are going to create a sub-class).

```
#include <iostream>
#include <string>
#include <yarp/os/RFModule.h>
#include <yarp/os/Network.h>

using namespace std;
using namespace yarp::os;

class MyModule:public RFModule
{
   /* module parameters */

   /* class variables */

public:

   bool configure(yarp::os::ResourceFinder &rf); // configure all the module parameters and return true if s
   bool interruptModule();                       // interrupt, e.g., the ports
   bool close();                                 // close and shut down the module
   bool respond();
   double getPeriod();
   bool updateModule();
}
```

We will deal with the various issues of implementing the module under several headings, addressing configuration, doing some work, run-time interaction, graceful shut-down, standards, and application description.

Configuration

By configuration, we mean the ability to specify the way that a given module is used. There are two aspects to this:

1. how the module is presented (i.e. which particular interfaces are used: the names of ports used, the name of the configuration file, the path to the configuration file, the name of the module, and the name of the robot) and
2. the module parameters that govern its behaviour (e.g. thresholds, set-points, and data files)

We refer to these collectively as resources. Typically, the configuration file name, the configuration file path (called the context), the module name, and the robot name are specified as command-line parameters, while all the remaining resources, including the names of the ports, are typically specified in the configuration file.

What's important to realize, however, is that *all* resources are handled the same way using the ResourceFinder which not only greatly simplifies the process of finding the resources but also simplifies the process of parsing them. There is more detail on handling resources in Configuration and resource files.

It's worth noting that parameters that are specified in the configuration file can also be specified in the command-line if you wish. The reverse is also true, with some restrictions (e.g. it only makes sense to specify the configuration file and the configuration file path on the command-line). Finally, modules should be written so that default values are provided for all resources so that the module can be launched without any parameters. Again, the ResourceFinder makes it easy to arrange this.

Right now, what's important to grasp is that all these configuration issues are implemented by

- preparing the Resource Finder in the `main` function by setting the default configuration file and its path,
- overriding the `yarp::os::RFModule::configure()` method to parse all the parameters from the command-line and the configuration file.

The following sections explain the implementation details of each aspect of this configuration.

Essential Command-line Parameters

Configuration File

Configuration can be changed by changing configuration files. The configuration file which the module reads can be specified as a command line option.

```
--from myModule.ini
```

The module should set a default configuration file using `yarp::os::ResourceFinder::setDefaultConfigFile("myModule.ini")`. This should be done in the `main()` function before running the module.

This is overridden by the `--from` parameter.

The .ini file should usually be placed in the `$ICUB_ROOT/app/myModule/conf` sub-directory.

Context

The relative path from `$ICUB_ROOT/app/` to the directory containing the configuration file is specified from the command line by

```
--context  myModule/conf
```

The module should set a default context using `yarp::os::ResourceFinder::setDefaultContext("myModule/conf")`. This should be done in the `main()` function before running the module.

This is overridden by the `--context` parameter.

Module Name and Port Names

Warning: the naming convention for the --name, --robot, and port name arguments in key-value pairs has changed. The arguments of --name and --robot **do not** having a leading "/" prefix and port name arguments **always** have a leading "/" prefix ... exactly the opposite of what was considered acceptable practice in the past.

It should be possible to specify the names of any ports created by a module via configuration. There are two aspects to this: the stem of the port name and the port name itself.

A command-line option of

```
--name altName
```

sets the name of the module and will cause the module to use "/altName" as a stem for all port names provided the port names are generated using the `yarp::os::RFModule::getName()` method. Note that he leading "/" prefix has to be added explicitly to the module name to create the port name.

The module should set a default name (and, hence, a default stem) using `yarp::os::RFModule::setName("myModule")`.

This is overridden by the `--name` parameter but you must check for this parameter and call `setName()` accordingly, e.g.

```
string moduleName;

moduleName = rf.check("name",
             Value("myModule"),
             "module name (string)").asString();

setName(moduleName.c_str());  // do this before processing any port name parameters
```

The port names should be specified as parameters, typically as key-value pairs in the `.ini` configuration file, e.g.

```
myInputPort   /image:i
myOutputPort  /image:o
```

These key-value pairs can also be specified as command-line parameters, viz: `--myInputPort /image:i --myOutputPort /image:o`

The module should set a default port name using the ResourceFinder, e.g using `yarp::os::ResourceFinder::check();`

For example

```
string inputPortName  = "/";
       inputPortName += getName(
                        rf.check("myInputPort",
                        Value("/image:i"),
                        "Input image port (string)").asString()
                        );
```

will assign to `inputPortName` the value `/altName/image:i` if `--name altName` is specified as a parameter. Otherwise, it would assign `/myModule/image:i` On the other hand, it would assign `/myModule/altImage:i` if the key-value pair `myInputPort /altImage:i` was specified (either in the `.ini` file or as a command-line parameter) but not the `--name altName`

When providing the names of ports as parameter values (whether as a default value in `ResourceFinder::check`, as a value in the key-value list in the `.ini` configuration file, or as a command line argument), you always include the leading `/`.

All this code goes in the `configure()` method.

Robot Name

If you connect automatically to the robot, make sure the name of the ports to which you connect to can be changed from the command line. This will make it possible to switch from using the simulator (whose ports are prefixed with `icubSim`) to the real robot (whose ports are prefixed with `icub`).

Usually this is achieved with a `--robot` parameter. For example, to access the left camera on the simulator (`/iCubSim/cam/left`) use `--robot icubSim` and to access the left camera on the robot (`/icub/cam/left`) use `-–robot icub`

Which Parameters Are Parsed Automatically?

Parsing the `--from` and `--context` parameters is handled automatically by the `RFModule` but `--name` and `--robot` must be handled explicitly.

As noted above, you would handle the `--name` parameter by using `ResourcFinder::check()` to parse it and get the parameter value, then user `setName()` to set it. You should do this before proceeding to process any port name parameters, otherwise the wrong stem will be used when constructing the port names from the parameter values.

Typically, you would handle the `--robot` parameter by using ResourceFinder to parse the `--robot` or `--name` parameter to get the root of the port name and then construct the full port name by appending the specific part of the robot required. An example is provided below.

Configuration File Parameters

The configuration file, typically named `myModule.ini` and located in the `$ICUB_ROOT/app/myModule/conf` directory, contains a key-value list: a list of pairs of keywords (configuration parameter names) and values (configuration parameter values), e.g.

```
myInputPort  /altImage:i
myOutputPort /altImage:o
threshold 9
...
```

These parameters are parsed using the `ResourceFinder` (http://eris.liralab.it/yarpdoc/d9/ddf/classyarp_1_1os_1_1ResourceFinder.html) within an `RFModule` object (i.e. by methods inherited by your module such as `yarp::os::Searchable::check()` (http://eris.liralab.it/yarpdoc/d2/d0c/classyarp_1_1os_1_1Searchable.html#818029558a2d8772db43a5a3c8b61125)).

Typically, key-value pairs specify the parameters and their values that govern the behaviour of the module, as well as the names of the module ports, should you wish to rename them.

Other Configuration Files

Apart from processing the parameters in the configuration file `myModule.ini`, it's often necessary to access configuration data in other files. For example, you might want to read the intrinsic camera parameters from a camera calibration file. Let's assume this configuration file is called `icubEyes.ini` and we wish to extract the principal points of the left and right cameras. The coordinates of the principle points, and other intrinsic camera parameters, are generated by the camera calibration module (http://eris.liralab.it/iCub/dox/html/group__icub__camcalibconf.html) and are stored as a sequence of key-value pairs:

```
cx 157.858
cy 113.51
```

Matters are somewhat complicated by the fact that we need to read two sets of coordinates, one for the left camera and one for the right. *Both* sets have the same key associated with them so the left and right camera parameters, including the principal point coordinates, are typically listed under different *group* headings, viz.

```
[CAMERA_CALIBRATION_RIGHT]
...
cx 157.858
cy 113.51
...

[CAMERA_CALIBRATION_LEFT]
...
cx 174.222
cy 141.757
...
```

So, to read these two pairs of coordinates, we need to

- find the name of the file (e.g. `icubEyes.ini`)
- locate the file (i.e. get its full path)
- open the file and read its content
- find the `CAMERA_CALIBRATION_RIGHT` and `CAMERA_CALIBRATION_LEFT` groups
- read the respective `cx` and `cy` key values.

All of this is accomplished straightforwardly with the `ResourceFinder` (http://eris.liralab.it/yarpdoc/d9/ddf/classyarp_1_1os_1_1ResourceFinder.html) and `Property` (http://eris.liralab.it/yarpdoc/da/d1f/classyarp_1_1os_1_1Property.html) classes.

The first step is to get the name of the configuration file. This will typically be one of the key-value pairs in the module configuration file `myModule.ini`, e.g.

```
cameraConfig icubEyes.ini
```

so that it can be read in exactly the same way as the other parameters in the previous section, e.g. using `yarp::os::Searchable::check()` (http://eris.liralab.it/yarpdoc/d2/d0c/classyarp_1_1os_1_1Searchable.html#818029558a2d8772db43a5a3c8b61125) .

The full path can then be determined by the `yarp::os::ResourceFinder::findFile()` (http://eris.liralab.it/yarpdoc/d9/ddf/classyarp_1_1os_1_1ResourceFinder.html#355586da9ad41565a2a0daa36e7ec2e1) method.

The contents of this file can then be read into a `Property` (http://eris.liralab.it/yarpdoc/da/d1f/classyarp_1_1os_1_1Property.html) object using the `yarp:os:Property::fromConfigFile()` (http://eris.liralab.it/yarpdoc/da/d1f/classyarp_1_1os_1_1Property.html#06c34c056e399f1cad1ad74b3a147a76) method.

Locating the required group (e.g. `CAMERA_CALIBRATION_LEFT`) is accomplished with the `yarp:os:Property::findGroup()` (http://eris.liralab.it/yarpdoc/da/d1f/classyarp_1_1os_1_1Property.html#ed956fea82f3b54bc846946c1f836ccb) method.

This method returns a `Bottle` (http://eris.liralab.it/yarpdoc/d3/d3e/classyarp_1_1os_1_1Bottle.html) with the full key-value list under this group. This list can then be searched for the required key and value using the `yarp::os::Searchable::check()` (http://eris.liralab.it/yarpdoc/d2/d0c/classyarp_1_1os_1_1Searchable.html#818029558a2d8772db43a5a3c8b61125) method, as before.

Please refer to the `myModule` example for further details.

An Example of how to Configure the Module

The following simple module shows how to handle the foregoing configuration issues.

```
#include <yarp/os/all.h>
#include <yarp/os/RFModule.h>
#include <yarp/os/Network.h>
#include <yarp/os/Thread.h>
#include <yarp/sig/all.h>

using namespace std;
using namespace yarp::os;
using namespace yarp::sig;

class MyModule:public RFModule
{
   /* module parameters */

   string moduleName;
   string robotName;
   string robotPortName;
   string inputPortName;
   string outputPortName;
   string cameraConfigFilename;
   float  fxLeft,  fyLeft;          // focal length
   float  fxRight, fyRight;         // focal length
   float  cxLeft,  cyLeft;          // coordinates of the principal point
   float  cxRight, cyRight;         // coordinates of the principal point
   int thresholdValue;

   /* class variables */

   BufferedPort<ImageOf<PixelRgb> > imageIn;      //example input port
   BufferedPort<ImageOf<PixelRgb> > imageOut;     //example output port

public:

   bool configure(yarp::os::ResourceFinder &rf); // configure all the module parameters and return true if s
   bool interruptModule();                       // interrupt, e.g., the ports
   bool close();                                 // close and shut down the module
   bool respond(const Bottle& command, Bottle& reply)
   double getPeriod();
   bool updateModule();
};
```

```
/*
 * Configure method. Receive a previously initialized
 * resource finder object. Use it to configure your module.
 * If you are migrating from the old Module, this is the
 *  equivalent of the "open" method.
 */

bool MyModule::configure(yarp::os::ResourceFinder &rf)
{
   /* Process all parameters from both command-line and .ini file */

   /* get the module name which will form the stem of all module port names */

   moduleName            = rf.check("name",
                           Value("myModule"),
                           "module name (string)").asString();

   /*
    * before continuing, set the module name before getting any other parameters,
    * specifically the port names which are dependent on the module name
    */

   setName(moduleName.c_str());

   /* now, get the rest of the parameters */

   /*
    * get the robot name which will form the stem of the robot ports names
    * and append the specific part and device required
    */

   robotName             = rf.check("robot",
                           Value("icub"),
                           "Robot name (string)").asString();

   robotPortName         = "/" + robotName + "/head";

   /*
    * get the cameraConfig file and read the required parameter values cx, cy
    * in both the groups [CAMERA_CALIBRATION_LEFT] and [CAMERA_CALIBRATION_RIGHT]
    */

   cameraConfigFilename  = rf.check("cameraConfig",
                           Value("icubEyes.ini"),
                           "camera configuration filename (string)").asString();

   cameraConfigFilename = (rf.findFile(cameraConfigFilename.c_str())).c_str();

   Property cameraProperties;

   if (cameraProperties.fromConfigFile(cameraConfigFilename.c_str()) == false) {
      cout << "myModule: unable to read camera configuration file" << cameraConfigFilename;
      return 0;
   }
   else {
      cxLeft  = (float) cameraProperties.findGroup("CAMERA_CALIBRATION_LEFT").check("cx", Value(160.0), "cx
      cyLeft  = (float) cameraProperties.findGroup("CAMERA_CALIBRATION_LEFT").check("cy", Value(120.0), "cy
      cxRight = (float) cameraProperties.findGroup("CAMERA_CALIBRATION_RIGHT").check("cx", Value(160.0), "cx
      cyRight = (float) cameraProperties.findGroup("CAMERA_CALIBRATION_RIGHT").check("cy", Value(120.0), "cy
   }

   /* get the name of the input and output ports, automatically prefixing the module name by using getName()

   inputPortName         = "/";
   inputPortName        += getName(
                           rf.check("myInputPort",
                           Value("/image:i"),
                           "Input image port (string)").asString()
                           );

   outputPortName        = "/";
   outputPortName       += getName(
                           rf.check("myOutputPort",
                           Value("/image:o"),
                           "Output image port (string)").asString()
                           );

   /* get the threshold value */
```

```
int main(int argc, char * argv[])
{
   /* initialize yarp network */

   Network yarp;

   /* create your module */

   MyModule myModule;

   /* prepare and configure the resource finder */

   ResourceFinder rf;
   rf.setVerbose(true);
   rf.setDefaultConfigFile("myModule.ini"); //overridden by --from parameter
   rf.setDefaultContext("myModule/conf");   //overridden by --context parameter
   rf.configure("ICUB_ROOT", argc, argv);

   /* run the module: runModule() calls configure first and, if successful, it then runs */

   myModule.runModule(rf);

   return 0;
}
```

Graceful Shut-down

To achieve clean shutdown, two methods `yarp::os::RFModule::interruptModule()` and `yarp::os::RFModule::close()` should be overridden. The `interruptModule()` method will be called when it is desired that `updateModule()` finish up. When it has indeed finished, `close()` will be called. For example:

```
bool MyModule::interruptModule()
{
   imageIn.interrupt();
   imageOut.interrupt();
   return true;
}
```

```
bool MyModule::close()
{
   imageIn.close();
   imageOut.close();
   return true;
}
```

For `yarp::os::RFModule`, the method `yarp::os::RFModule::updateModule()` will be called from the main control thread until it returns false. After that a clean shutdown will be initiated. The period with which it is called is determined by the method `yarp::os::RFModule::getPeriod()`. Neither method need necessarily be overridden. The default methods provide the required functionality.

```
/* Called periodically every getPeriod() seconds */

bool MyModule::updateModule()
{
   return true;
}
```

```
double MyModule::getPeriod()
{
   /* module periodicity (seconds), called implicitly by myModule */

   return 0.1;
}
```

Note that the `updateModule()` method is not meant to run code that that implements the algorithm encapsulated in the module. Instead `updateModule()` is meant to be used as a periodic mechanism to check in on the operation of the thread that implements the module (e.g. gather interim statistics, change parameter settings, etc.). The `updateModule()` is called periodically by the RFModule object, with the period being determined by the `getPeriod()` method. Both `updateModule()` and `getPeriod()` can be overridden in your implementation of `myModule`.

Thread-based Implementation

Using Threads to Implement Your Algorithm

For the module to actually do anything, it should start or stop threads using the YARP Thread (http://eris.liralab.it/yarpdoc/d2/d2d/classyarp_1_1os_1_1Thread.html) and RateThread (http://eris.liralab.it/yarpdoc/d9/d9c/classyarp_1_1os_1_1RateThread.html) classes. Typically, these threads are started and stopped in the `configure` and `close` methods of the RFModule class. If you are writing a control loop or an algorithm that requires precise scheduling we strongly advise that you use the RateThread (http://eris.liralab.it/yarpdoc/d9/d9c/classyarp_1_1os_1_1RateThread.html) class.

Just as the starting point in writing a module for the iCub repository is to develop a sub-class of the yarp::os::RFModule (http://eris.liralab.it/yarpdoc/d9/d26/classyarp_1_1os_1_1RFModule.html) class, **the starting point for implementing the algorithm within that module is to develop a sub-class of either Thread (http://eris.liralab.it/yarpdoc/d2/d2d/classyarp_1_1os_1_1Thread.html) or RateThread (http://eris.liralab.it/yarpdoc/d9/d9c/classyarp_1_1os_1_1RateThread.html)** .

In the following, we will explain how to do it with Thread (http://eris.liralab.it/yarpdoc/d2/d2d/classyarp_1_1os_1_1Thread.html) ; it's straightforward to extend this to RateThread (http://eris.liralab.it/yarpdoc/d9/d9c/classyarp_1_1os_1_1RateThread.html) (effectively, you provide an argument with the `RateThread` instantiation specifying the period with which the thread should be spawned, the thread just runs once so that you don't have to check `isStopping()` to see if the thread should end).

Perhaps one of the best ways of thinking about this is to view it as a two levels of encapsulation, one with `RFModule`, and another with `Thread`; the former deals with the configuration of the module and the latter dealing with the execution of the algorithm. The only tricky part is that somehow these two objects have to communicate with one another.

You need to know three things:

1. The thread is instantiated and started in the `configure()` method.
2. The thread is stopped in the `close()` method.
3. When the thread is instantiated, you pass the module parameters to it as a set of arguments (for the constructor).

Let's begin with the definition of a thread `MyThread` (capital M because we are going to create a sub-class) and then turn our attention to how it is used by `MyModule`.

An Example of how to use the `Thread` Class

First, we define a sub-class, or derived class, of the `yarp::os::Thread` class. The algorithm's variables - *and specifically the thread's parameters and ports* - go in the private data members part and you need to override four methods:

1. `MyThread::MyThread(); // the constructor`
2. `bool threadInit(); // initialize variables and return true if successful`
3. `void run(); // do the work`
4. `void threadRelease(); // close and shut down the thread`

There are a number of important points to note.

First, the variables in the `myThread` class which represent the thread's parameters and port should be pointer types and the constructor parameters should initialize them. In turn, the arguments of the `myThread` object instantiation in the `configure()` should be the addresses of (pointers to) the module parameters and ports in the `myModule` object. In this way, the thread's parameter and port variables are just references to the original module parameters and ports that were initialized in the `configure` method of the `myModule` object.

Second, `threadInit()` returns `true` if the initialization was successful, otherwise it should return `false`. This is significant because if it returns `false` the thread will not subsequently be run.

Third, the `run()` method is where the algorithm is implemented. Typically, it will run continuously until some stopping condition is met. This stopping condition should include the return value of a call to the `yarp::os::Thread::isStopping()` method which flags whether or not the thread is to terminate. In turn, the value of `yarp::os::Thread::isStopping()` is determined by the `yarp::os::Thread::stop()` method which, as we will see, is called in `myModule.close()`

The following is an example declaration and definition of the `MyThread` class.

```
#include <yarp/os/Thread.h>

using namespace std;
using namespace yarp::os;

class MyThread : public Thread
{
private:

   /* class variables */

   int      x, y;
   PixelRgb rgbPixel;
   ImageOf<PixelRgb> *image;

   /* thread parameters: they are pointers so that they refer to the original variables in myModule */

   BufferedPort<ImageOf<PixelRgb>> *imagePortIn;
   BufferedPort<ImageOf<PixelRgb>> *imagePortOut;
   int *thresholdValue;

public:

   /* class methods */

   MyThread(BufferedPort<ImageOf<PixelRgb>> *imageIn,  BufferedPort<ImageOf<PixelRgb>> *imageOut, int *thres
   bool threadInit();
   void threadRelease();
   void run();
};
```

```
MyThread::MyThread(BufferedPort<ImageOf<PixelRgb>> *imageIn, BufferedPort<ImageOf<PixelRgb>> *imageOut, int
{
   imagePortIn    = imageIn;
   imagePortOut   = imageOut;
   thresholdValue = threshold;
}

bool MyThread::threadInit()
{
   /* initialize variables and create data-structures if needed */

   return true;
}

void MyThread::run(){

   /*
    * do some work ....
    * for example, convert the input image to a binary image using the threshold provided
    */

   unsigned char value;

   while (isStopping() != true) { // the thread continues to run until isStopping() returns true

      cout << "myThread: threshold value is " << *thresholdValue << endl;

      do {
         image = imagePortIn->read(true);
      } while (image == NULL);

      ImageOf<PixelRgb> &binary_image = imagePortOut->prepare();
      binary_image.resize(image->width(),image->height());

      for (x=0; x<image->width(); x++) {
         for (y=0; y<image->height(); y++) {

             rgbPixel = image->safePixel(x,y);

             if (((rgbPixel.r + rgbPixel.g + rgbPixel.b)/3) > *thresholdValue) {
                value = (unsigned char) 255;
             }
             else {
                value = (unsigned char) 0;
             }

             rgbPixel.r = value;
             rgbPixel.g = value;
             rgbPixel.b = value;

             binary_image(x,y) = rgbPixel;
          }
       }

       imagePortOut->write();
   }
}

void MyThread::threadRelease()
{
   /* for example, delete dynamically created data-structures */
}
```

Creating, Starting, and Stopping the Thread

As we said already, the thread is instantiated and started in the `configure()` method in `myModule`, the thread is stopped in the `close()` method, and when the thread is instantiated, you pass the pointers to the module parameters to it as a set of arguments. First, however, we add a new variable to the `MyModule` class.

```
/* pointer to a new thread to be created and started in configure() and stopped in close() */

MyThread *myThread;
```

The following code would then go in the `configure()` method.

```
/* create the thread and pass pointers to the module parameters */
myThread = new MyThread(&imageIn, &imageOut, &thresholdValue);
/* now start the thread to do the work */
myThread->start(); // this calls threadInit() and it if returns true, it then calls run()
```

The following code would go in the `close()` method.

```
/* stop the thread */
myThread->stop();
```

Run-time Interaction

The `respond()` Method

Often, it is very useful for a user or another module to send commands to control the behaviour of the module, e.g. interactively changing parameter values. The controlGaze2 (http://eris.liralab.it/iCub/dox/html/group__icub__controlGaze2.html) module is a good example of this type of usage (see also VVV09_Control_Gazers_Group).

We accomplish this functionality for the `yarp::os::RFModule` by overridding the `yarp::os::RFModule::respond()` method which can then be configured to receive messages from either a port (typically named `/myModule`) or the terminal. This is effected by the `yarp::os::RFModule::attach(port)` and `yarp::os::RFModule::attachTerminal()` methods, respectively. Attaching both the port and the terminal means that commands from both sources are then handled in the same way.

An Example of how to change module parameters at run-time

In the following example, we handle three commands:

- help
- quit
- set
  - set thr <n> ... set the threshold

  (where <n> is an integer number)

Apart from the way that the commands are parsed and the form of the reply, the key thing to note here is the fact that the value of `MyModule::thresholdValue` is updated. Since `myThread` references this variable, it too is updated and the updated value is used in the thread.

```
bool MyModule::respond(const Bottle& command, Bottle& reply)
{
  string helpMessage =  string(getName().c_str()) +
                        " commands are: \n" +
                        "quit \n" +
                        "set thr <n> ... set the threshold \n" +
                        "(where <n> is an integer number) \n";

  reply.clear();

  if (command.get(0).asString()=="quit") {
       reply.addString("quitting");
       return false;
   }
   else if (command.get(0).asString()=="help") {
      cout << helpMessage;
      reply.addString("ok");
   }
   else if (command.get(0).asString()=="set") {
      if (command.get(1).asString()=="thr") {
         thresholdValue = command.get(2).asInt(); // set parameter value
         reply.addString("ok");
      }
   }
   return true;
}
```

However, for any of this to work, we have to set up a port in the first place. We put port declaration in the private data member part of `MyModule` class

```
string handlerPortName;
Port handlerPort;      //a port to handle messages
```

and open it in the `configure()` method, viz.

```
/*
 * attach a port of the same name as the module (prefixed with a /) to the module
 * so that messages received from the port are redirected to the respond method
 */

handlerPortName =  "/";
handlerPortName += getName();         // use getName() rather than a literal

if (!handlerPort.open(handlerPortName.c_str())) {
   cout << getName() << ": Unable to open port " << handlerPortName << endl;
   return false;
}

attach(handlerPort);                  // attach to port

attachTerminal();                     // attach to terminal
```

Interrupt it in the `interrupt()` method, viz.

```
handlerPort.interrupt();
```

Close it in the `close()` method, viz.

```
handlerPort.close();
```

Remote Connection

Note that the `handlerport` can be used not only by other modules but also interactively by a user through the `yarp rpc` directive, viz.:

```
yarp rpc /myModule
```

This opens a connection from a terminal to the port and allows the user to then type in commands and receive replies from the `respond()` method.

Documentation and Coding Guidelines

RobotCub code follows some fairly strict documentation and coding standards defined in Section III of RobotCub Deliverable 8.2 (http://www.robotcub.org/index.php/robotcub/more_information/deliverables/deliverable_8_2_pdf) .

For convenience, here are the

- iCub File Organization Guidelines
- iCub Documentation Guidelines
- iCub Coding Guidelines

Please take the time to read through the three documents.

As we move towards the creation of a release version of the iCub software, we will begin to enforce a sub-set of these guidelines as mandatory standards. The current set of standards is set out in iCub Software Standards. **Ultimately, all modules to be included in the standard iCub release version will have to comply with these standards.**

The principal documentation for `myModule` is provided in the full example at the end of this page.

Application Description

iCub applications, i.e. collections of inter-connected YARP modules, are described in XML and launched using an automatically-generated GUI. Refer to Managing Applications for more details on how to write these application descriptions.

An application description containing an example invocation of the `myModule` with some command-line parameters is shown below.

```
<application>

<name>Test myModule</name>

<dependencies>
   <port>/icub/cam/left</port>
</dependencies>

<module>
   <name>myModule</name>
   <parameters>--threshold 128</parameters>
   <node>icub1</node>
   <tag>myModule</tag>
</module>

<module>
   <name>yarpview</name>
   <parameters>--name /rgbImage --x 000 --y 0 --synch</parameters>
   <node>icub1</node>
   <tag>left_image</tag>
</module>

<module>
   <name>yarpview</name>
   <parameters>--name /binaryImage --x 350 --y 0 --synch</parameters>
   <node>icub1</node>
   <tag>right_image</tag>
</module>

<connection>
  <from>/icub/cam/left</from>
  <to>/myModule/image:i</to>
  <protocol>tcp</protocol>
</connection>

<connection>
  <from>/icub/cam/left</from>
  <to>/rgbImage</to>
  <protocol>tcp</protocol>
</connection>

<connection>
  <from>/myModule/image:o</from>
  <to>/binaryImage</to>
  <protocol>tcp</protocol>
</connection>

</application>
```

To run the application, you simple need to run the XML application description shown in the previous section. To do this, however, you need to have a couple of new things installed:

- Python (this is used to interpret the iCub application description programs and launch iCub applications);
- the `icubapp` pseudo-command for launching the Python application manager. This is no more than a simple invocation of Python to run the application manager and interpret the XML application descripton, viz. `python $ICUB_ROOT\app\default\scripts\manager.py %1`.

Refer to Prepare your system for running applications for details of how to get these resources .

Do an update on your iCub repository to make sure you have the `icubapp` pseudo-command. Alternatively, you can launch the python application manager directly (see below).

Once you have done all this, you are *almost* ready to run your application. There's just one more thing to be aware of.

You need to start an instance of `yarprun --server` on the local machine (for a complete explanation see Cluster management). This `yarprun` is what the node in an XML application description gets mapped to. At present, the standard for creating these `yarprun`s is for the `yarprun` argument to be the name of the node identifier in the XML `<node></node>` field but prefixed by a / to make it explicit that the argument is a port.

So, if you have used, for example, `<node>icub1</node>` in your `<module>` description in the XML file, then you would do

```
PC> yarprun --server /icub1
```

In general, at present (this may change in the future), you need to do a

```
PC> yarprun --server /<mc_n>
```

for each <mc_n> node values specified in the xml file.

These `yarprun` commands are run on the machine to which that node is mapped. An XML `<node>` is a logical machine and the `yarprun` associates it with the physical machine on which it to be instantiated.

You can now launch an application. Simply navigate to the directory where the XML file resides (typically `$ICUB_ROOT/app/myModule/scripts`) and do

```
PC> icubapp myModule.xml
```

Alternatively, if you prefer, you can launch the Python application manager directly:

```
PC> manager.py myModule.xml
```

assuming that `$ICUB_ROOT\app\default\scripts\` is defined in you path and assuming `.py` files are associated with Python.

In either case, doing this will launch a GUI with which you can then "Run Modules" and "Connect" the ports by clicking on the appropriate buttons.

NB: turn off your firewall before launching the application.

Resources

- Module Standards (http://eris.liralab.it/iCub/dox/html/module_standards.html)
- Resource finder overview
- How to organize the command line parameters of your modules (http://eris.liralab.it/iCub/dox/html/icub_resource_finder_basic.html)
- Organizing Parameters: Advanced Tutorial (http://eris.liralab.it/iCub/dox/html/icub_resource_finder_advanced.html)
- Using the module helper class to write a program (http://eris.liralab.it/iCub/dox/html/icub_tutorial_module.html)

- RFModule Class Reference (http://eris.liralab.it/yarpdoc/d9/d26/classyarp_1_1os_1_1RFModule.html)
- Module Class Reference (http://eris.liralab.it/yarpdoc/d1/d03/classyarp_1_1os_1_1Module.html)
- Coding and Documentation Standards (http://eris.liralab.it/iCub/dox/html/coding_standards.html)
- exampleModule (http://eris.liralab.it/iCub/dox/html/group__icub__exampleModule.html)
- Cluster management
- exampleApplication (http://eris.liralab.it/iCub/dox/html/group__icub__exampleApplication.html)
- iCub tutorials (http://eris.liralab.it/iCub/dox/html/icub_tutorials.html)

The Complete `myModule` Example

The complete code for `myModule` is here.

Cluster

From Wiki for RobotCub and Friends

Important: this documentation might be obsolete. It describes a the cluster configuration that was used some time ago to run the iCub at University of Genova.

We are using a Blade server with 6 blades at the moment running Debian Linux and a server/interface machine running Windows 2003. We also have 5 dual-boot (XP/Linux)Shuttle PCs and two more rack mount machines, a bunch of screens and several laptops connected to the same network.

This document describes certain procedures for the maintenance of this system.

Most of the disk space is on the Windows 2003 machine that shares it using both NFS and Samba. The Windows 2003 machine also runs the Unix Services for Windows and the YP server, name and permission mapping, etc.

On each Linux machine we mount the home directories (or anything else need, e.g. Yarp) using NFS, while Windows machines we use Samba (i.e. the native Windows network). Linux machines use yp. The yp server is on the Windows 2003 machine where also the Windows domain is managed.

Contents

- 1 Configuration
- 2 Shared drives
- 3 Environment variables
- 4 Notes
- 5 SSH public key authentication
- 6 CYGWIN
- 7 Openssh on windows
- 8 Tips and tricks
  - 8.1 Making SSH read your environment variables on the target node
- 9 Name server
- 10 Missing things

Configuration

The cluster consists of a certain number N of machines with various operating systems (not necessarily uniform). One machine is directly attached to the robot hardware and this should be the case for compatibility with iCub (where all the hardware is interfaced through the on-board pc104 machine). Since Yarp doesn't provide security, the cluster network is physically separated from the outside world via a server/router. The server in our case here runs Windows 2003 but this is not a requirement. Separating the cluster from other networks also limits the potentially high bandwidth traffic to the robot subnetwork(s).

The blade computers are mutually connected via two Gbit/s switches. These are private networks responding to the 10.0.0.X and 10.0.1.X addresses. The server machine called MUSE in our implementation is also connected to both network switches and to the outside lab network. Several other machines are only connected to the first private network (10.0.0.X).

MUSE is a domain controller for the Windows network, it is also a NIS domain server for Linux (we use Debian). It is a

DNS for the machines in the private networks. MUSE maps users from Windows to Linux via NIS and also maps groups and other tables from Windows to Linux and vice versa. It exports directories using NFS for Linux and Samba for Windows. It runs also a NAT to map the private IP addresses into a pool of public ones.

Cygwin is used on all Windows machines to run a ssh server. This is used on MUSE also to allow access to the other machines through ssh tunneling (don't remember the port numbers at the moment) but also for scripting using bash irrespective of the operating system. Cygwin provides also the cvs client we currently use on Windows.

Shared drives

MUSE shares a **separate directory** for each operating system (e.g. Linux, Windows) that contains in turn Yarp, the iCub repository and other libraries if needed.

From Linux, typically mount the shared directory (see extract from mount):

```
muse.james.liralab.it:yarp on /usr/src/yarp type nfs (rw,intr,addr=130.251.4.3)
```

Which can be added to /etc/fstab:

```
muse.james.liralab.it:yarp      /usr/src/yarp   nfs     rw,intr
```

Which will then show up as (ls -la /usr/src/yarp):

```
drwx------ 2 4294967294 Domain Users   64 2006-10-28 15:56 .
drwxrwsr-x 9 root       src          4096 2006-08-23 17:20 ..
drwxr-xr-x 2 babybot    Domain Users   64 2006-10-28 16:04 iCub
drwxrwxr-x 2 babybot    Domain Users   64 2006-07-22 09:49 yarp
drwxrwxr-x 2 babybot    Domain Users   64 2006-10-28 16:25 yarp2
drwxr-xr-x 2 james      Domain Users   64 2006-10-29 00:08 yarp2-james
drwxr-xr-x 2 babybot    Domain Users   64 2006-10-27 10:54 yarp2-unstable
```

From Windows, typically mount the shared directory as drive Y (or anything else you like using the standard Windows "Map Network Drive".

Environment variables

These are the environment variables recommended (not all of them are required):

```
ICUB_ROOT
YARP_ROOT
YARP_BUILD
YARP_CONF
```

Important: if you are using CYGWIN check that the environment variable CYGWIN exist (for example se it to "smbntsec ntsec). The .bashrc script check $CYGWIN to determine it it is running on Linux or CYGWIN (see below).

For example in Linux you can do something like this in the .bashrc and perhaps make sure that the .profile calls a similar sequence of commands:

```
export ICUB_ROOT=/usr/src/yarp/iCub
export YARP_ROOT=/usr/src/yarp/yarp2
export YARP_BUILD=$YARP_ROOT
if [ -e /etc/debian_version ] ; then
  debtype=`cat /etc/debian_version | sed "s|.*/||"`
  if [ "k$debtype" = "kunstable" ]; then
    export YARP_BUILD=/usr/src/yarp/yarp2/build/$debtype
  fi
fi
export YARP_CONF=$YARP_ROOT
```

In Cygwin you can reuse the same .bashrc (highly recommended) conditioning on the operating system type:

```
if [ ! "k$CYGWIN" = "k" ]; then
  export ICUB_ROOT=//MUSE/yarp/iCub
  export YARP_ROOT=//MUSE/yarp/yarp2
  export CYGWIN="smbntsec ntsec"
  export YARP_BUILD=$YARP_ROOT
  export YARP_CONF=$YARP_ROOT
```

where the remote path //MUSE/yarp is used directly (this is safer than the mount Y when scripting). IMPORTANT: make sure the environment varible CYGWIN exist on Windows.

Note the *smbntsec* flag which is required to map Cygwin permission properly on Samba drives.

In Windows, you can add the same variable names using the appropriate dialog from the control panel. They might show like:

```
  ICUB_ROOT    Y:\iCub
  YARP_ROOT    Y:\yarp2
  YARP_BUILD   Y:\yarp2
  YARP_CONF    Y:\yarp2
```

Cygwin requires also the environment variable CYGWIN to be set. The best is to allow at least *smbntsec ntsec tty* which determine how permissions are also seen on mapped drives (useful if they are consistent).

In general, be gentle to others and set the variables only for your user and not in system-wide scripts, this applies to both Linux and Windows.

On Windows, add to the PATH the following directories:

```
Y:\ACE_wrappers\lib;Y:\yarp2\bin;Y:\iCub\bin
```

On linux do a similar operation depending on where you installed the executables:

```
$YARP_BUILD/bin:$ICUB_ROOT/bin
```

Notes

Getting so many problems is not typical, they are due to the fateful intersections of many different things, including the presence of mixed operating systems, network drives, cygwin, sshd, carriage return characters, and more.

Various notes on useful operations:

- Accessing a computer with the Windows remote desktop.
  - Applies to: **Windows XP**

        - You need to add the username to the local policy.
        - Go to the "Control panel", click on "System", go to the "Remote" tab and click to "Select Remote Users", then finally add the username to the list of users (possibly a domain user).
        - This operation has to be performed by an Administrator.

- Adding a user to NIS.
    - Applies to: **Windows 2003**, **Unix Services for Windows 3.5**
    - Don't forget to add the domain name into the Unix tab in the "Active Directory Users and Computers".
    - Add also the UID that will be shown on Unix/Linux machines and the other Unix parameters.
    - Then don't forget to update the name/permission mapping service (also from the Unix services administration application).
    - Make sure you "Reset the password" for the user in question so that the NIS (damn!) sets a new password that is compatible on both Windows and Linux; this doesn't happen automatically.

- Adding a user.
    - Applies to: **Windows**
    - Don't forget to run *mkpasswd* in a cygwin shell to update the passwd file (you must be Administrator). This step has to be performed on every Cygwin installation (on every Windows machine on the cluster).

```
mkpasswd -l -d > /etc/passwd
mkgroups -l -d > /etc/group
```

- Installing cygwin.
    - Applies to: **Windows**
    - Check the DOS/Windows text mode during installation and NOT the Unix mode for newlines
    - Install "open ssh" and at least one text editor (e.g. vim, it can be handy).

- Issues with the ssh keys.
    - Applies to: **Windows** and **Linux**
    - For scripting is convenient to have a key installed
    - This goes typically in $HOME/.ssh
    - The $HOME is a networked folder
    - Ssh requires that the keys are only readable by the owner
    - Thus, you have to make sure that the permissions are always correct no matter how you access them
    - For cygwin this requires for example the definition of the CYGWIN variable to be *smbntsec*
    - See below for the installation of the sshd on Windows.

- Weird issue with the user settings.
    - Applies to: **Windows 2003**
    - We need to investigate the problem
    - The "Active Directory Users and Computers" from the control panel doesn't show the Unix attributes that are required to set the UID and other Unix properties for the name/group mappings
    - Temporary solution is to open the Windows MMC from the "Microsoft Windows Services for UNIX" application, searching for "adding user nis" and click to "To add a user to an NIS domain". This will show a link to a version of the MMC that will show correctly the Unix attributes
    - In the hope this is not a persistent problem.

- Shell issue with CR/LF on Cygwin.
    - Applies to: **Windows**
    - bash 3.1.17(9) is strict on the endline and if Cygwin is installed with native endline support (i.e. Windows/DOS like) then the scripts would complain.
    - Solution: run the dos2unix utility on all the script files just after downloading from the CVS repository

SSH public key authentication

Bored to type the password every time you log in to one of your machines? You can set up ssh to use public key instead.

These instructions were taken almost verbatim from: http://cfm.gs.washington.edu/security/ssh/client-pkauth/

On the client machine:

```
client$ mkdir ~/.ssh
client$ chmod 700 ~/.ssh
client$ ssh-keygen -q -f ~/.ssh/id_rsa -t rsa
```

Enter an empty passphrase twice (yes, it is not safe... we don't care).

Make sure everything has the correct access rights:

```
chmod go-w ~/
chmod 755 ~/.ssh
chmod go-rwx ~/.ssh/*
```

Keys distribution. If the machines share the same users server and client are actually the same machine and you don't have to copy anything, anyway in general do:

```
client$ scp ~/.ssh/id_rsa.pub server.example.edu:
```

Log on into the server and type:

```
server$ mkdir ~/.ssh
server$ chmod 755 ~/.ssh
server$ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
server$ chmod 644 ~/.ssh/authorized_keys
server$ rm ~/id_rsa.pub
```

Now go back to the client and test ssh by doing:

```
ssh -o PreferredAuthentications=publickey server.example.edu
```

If everything is allright you should login to server without password. Otherwise it means that something is wrong. Most of the times this is due to bad configurations rights, check out this:

```
server$ chmod go-w ~/
server$ chmod 755 ~/.ssh
server$ chmod 644 ~/.ssh/authorized_keys
```

Or go to /var/log and have a look at the messages dumped by sshd.

Note on permissions: usually on linux it is recommanded to set permissions like:

700 for ~/.ssh

600 for ~/.ssh/authorized_keys

However this does not work on cygwin because the *sshd* service seems to have troubles accessing those files.

If you want to login to a machine from a different user, you can do:

```
ssh otheruser@machine.domain.edu -i ~/.ssh/id_rsa
```

Finally on Windows go to:

Administrative Tools --> Services --> CYGWIN sshd --> properties --> Allow service to interact with desktop

CYGWIN

A brief help on how to configure CYGWIN to run YARP/ICUB.

- Go to http://www.cygwin.com and download setup.exe. Default installation is fine just add openssh (although you might find it useful to have also a couple of packages like vi, nano or emacs).
- Open a shell and type:

```
mkpasswd -l -d > /etc/passwd
mkgroups -l -d > /etc/group
```

- On windows check if the environment variable CYGWIN is set to **smbnet ntsec**
- Install openssh, see next section.

Openssh on windows

Assuming you installed cygwin with openssh.

Open a cygwin window (by double clicking theg icon), a black screen pops open, type:

```
ssh-host-config
```

- When the script asks you about "privilege separation", answer **yes**
- When the script asks about "create local user sshd", answer **yes**
- When the script asks you about "install sshd as a service", answer **yes**
- When the script stops and asks you for "CYGWIN=" your answer is **ntsec tty**
- While you are still in the (black) cygwin screen, start the sshd service:

```
net start sshd
or
cygrunsrv  --start  sshd
```

The following steps are also required if you want to use login and run processes remotely:

- Stop the "Cygwin sshd" service (go to "Computer Managment", open "Services", right click on "Cygwin sshd").
- Select "Cygwin sshd" properties. Under the "Log On" tab, change the name of the account to the user (MY_USER) that runs the processes (e.g. james@james.liralab.it)
- Tweak the Local Policies (Control Panel -> Adminstrative Tools -> Local Security Settings -> Local Policies). Grant to MY_USER the following:
  - Adjust memory quotas for a process
  - Create a token object
  - Lon on as a service (this should be on already)
  - Replace a process level token
- Change the ownership of the files required to run the sshd daemon. On a cygwin bash type:

```
chwon MY_USER /var/log/sshd.log
chown -R MY_USER /var/empty
chwon MY_USER /etc/ssh*
```

Good luck, you need it.

[from http://pigtail.net/LRP/printsrv/cygwin-sshd.html]

[from http://ist.uwaterloo.ca/~kscully/CygwinSSHD_W2K3.html

Tips and tricks

Tricks you need sometimes to make things work on the cluster. This list will hopefully improve/grow with time.

Linux:

- increase the size of the udp packets:

```
(sudo) sysctl -w net.core.rmem_max=8388608
```

Add this to your /etc/init.d/bootmisc.sh

If you are a developer and you want to use cvs to checkout yarp2 with command line (linux or cygwin):

```
export CVS_RSH=ssh
cvs -z3 -d:ext:your_name@yarp0.cvs.sourceforge.net:/cvs root/yarp0 co -P yarp2
```

Making SSH read your environment variables on the target node

When you start processes remotely via SSH like it is done by the cluster manager ($ICUB_ROOT/app/default/icub-cluster.py) neither .bashrc nor .bash_profile might be read on the target node, so your environment variables that are set there are not available. This means that for example yarprun won't be found. In order to get ssh to read your environment variables when called in a non-interactive mode like 'ssh -f' add them to the file

/etc/environment (available for every user)

or alternatively to

~/.ssh/environment (available only for a particular user).

Name server

The name server runs on nike. On /usr/bin/yarpserver is a copy of yarp. A script in /etc/init.d yarpserver starts/stops the service at boot time.

Missing things

Cluster clock synchronization

How to write a module

From Wiki for RobotCub and Friends

Author: Lorenzo Natale. Work in progress. Trying to illustrate guidelines and requirements for writing a module, rationalization of Summary page currently in the main page of the manual.

Although you can contribute any working piece of software to the repository, we require that your module respects a minimum set of requirements. These are listed here:

- Module configuration: it should be possible to change the parameters used by the module from the command line. Paths to configuration files should not be machine or configuration dependent.
- Remotization: it should be possible to execute the module remotely, using yarprun. If the module accepts commands, there should be a yarp interface that allows to send these commands to the module (one or more yarp ports).
- Clean shutdown: make sure there is a way to shut down the module without killing it.

The software infrastructure we provide facilitate writing modules that have these characteristics.

We provide sections in the manual and software tutorials that explain how to achieve this behavior.

Module configuration

The class ResourceFinder in YARP simplifies this task.

- These tutorials explain all you need to know about the ResourceFinder and how you can use it to write your module:
  - How to organize the command line parameters of your modules (http://eris.liralab.it/iCub/dox/html/icub_resource_finder_basic.html)
  - Organizing Parameters: Advanced Tutorial (http://eris.liralab.it/iCub/dox/html/icub_resource_finder_advanced.html)
- A more detailed description of the ResourceFinder class and how it works is reported here: Resource finder overview.

Remotization and clean shutdown

The helper class RFModule in YARP allows to write a module that meets all these requirements. It also supports the use of the ResourceFinder class described above.

This tutorial shows how to use the RFModule:

- Using the module helper class to write a program (http://eris.liralab.it/iCub/dox/html/icub_tutorial_module.html)

Configuration and resource files

From Wiki for RobotCub and Friends

Authors: Lorenzo Natale and Paul Fitzpatrick

Contents

- 1 Introduction
- 2 ResourceFinder Overview
- 3 Resource Finder for localizing files
- 4 ResourceFinder API
- 5 Rules to locate the policy file
- 6 Order of preference when searching for resource files
- 7 Let's make an example
- 8 Test case examples

Introduction

The problem we are discussing here is how to organize files used by modules. These files can be configuration/ini files or resource files (for example data files or images used by a module). We will refer in general to them as *resource files*.

The problem originates from the fact that we want to decouple the location of the configuration files and that of the executable. When developing in fact we can assume we know the executable working directory (although when running using "yarp run" when cannot control the process working directory, at least not yet). When executable are "installed" (as in a binary distribution) executable and configuration files might end up being moved.

We would like to organize resource files in a directory structure. This structure can be in $ICUB_ROOT/app, but we organize things so that the whole tree can be easily relocated.

Resource files are accessed through a YARP object, the *ResourceFinder* (or RF). To locate a certain file the programmer asks the RF using a "key". The RF searches the file system to identify the requested file and, if successful, it returns a full path to it; the search is performed using the rules that are specified when the RF is first created or configured. Parameters are passed to the RF through the command line or a file. The RF is not only responsible for locating files but can also decode parameters passed to the module and make them available as a key-value list.

ResourceFinder Overview

The *ResouceFinder* should become the gateway to access parameters and resource files in a module. The RF is key-value list which contains parameters of a module. These parameters can be specified in the command line as:

```
mymodule --key1 value1 --key2 value2
```

or in an "initialization file" as:

```
mymodule --from file.ini
```

in which *file.ini* is:

```
key1 value1
key2 value2
...
```

The RF works as a searchable. The method *find(key)* (or *check*) returns a *value* starting from a *key*.

First you need to configure the RF:

```
rf.configure("ICUB_ROOT", argc, argv);
```

The first string specifies a "key" to locate the file which points to the "resource search path policy file" (or "policy file"). The policy file describes/specifies the policy used to search for resource files. For example it could specify that resource files are searched in $ICUB_ROOT/app. This opens up the possibility to modify the behavior of the RF later on, in case we decided we want to store resource files in different locations (for example an *app* directory inside the user's HOME). The policy file itself is searched stating from the "key", using certain rules, see below.

You can query the RF in this way:

```
ConstString value1=rf.find("key1").asString();
int value2=rf.find("key2").asInt();
```

to retrieve the corresponding values of the parameters specified either from the command line or from the initialization file (*file.ini*).

RF looks for initialization files following a certain policy. We skip now the details of how this policy works (see below). In short the RF will look in a directory called "initialization context", that is specified from the command line.

```
mymodule --from file.ini --context myModule
```

makes RF search in $ICUB_ROOT/app/myModule

If you like you can specify a default initialization file:

```
rf.setDefaultConfigFile("file.ini");
```

so that you don't have to repeat it every time you run your module:

```
mymodule --context myModule
```

you can specify a default context:

```
rf.setDefaultContext("myModule");
```

so that you can run your module simply as:

```
mymodule
```

If needed you can switch the context from which your module is initialized by running it as:

```
mymodule --context myModule2
```

assuming "file.ini" exists in $ICUB_ROOT/app/myModul2

Resource Finder for localizing files

The RF works also as file finder. The method *findFile(key)* picks the *value* corresponding to *key* and interprets it as a filename. The RF does its best to locate the file named *value* and if successful returns a string which contains the full path to the file. Localization of resource files follows the same rules described for the "initialization file" above.

ResourceFinder API

```
// Get the value of the specified key, then search for a file with that name.
// If the key has not been given a value, try searching for a file with the
// key's name.
ConstString findFile(const char *keyName);
```

```
// Set a default value of a key
bool setDefault(const char *keyName, const char *keyValue);
```

```
// Set a name which, in conjunction with the policy, defines a default directory
// to search for files.  Should be called before configure()
bool setDefaultContext(const char *contextName);
```

```
// Load the policy, and apply command line options.  Options can modify
// the policy, change key values, and change the search path
bool configure(const char *policyName, int argc, char *argv[]);
```

Rules to locate the policy file

Given:

```
rf.configure("KEY", argc, argv)
```

rf will:

- if the environment variable KEY exists, search for $KEY/KEY.ini
- if this fails, in Linux search in /etc/KEY.ini

Order of preference when searching for resource files

When searching for files the RF will:

- search if the file exists in the local directory (local here means with respect to the process working directory)
- check relative the directory where the default config file is (specified with --from), this is useful when the config file is specified with relative path (as in: --from ../../conf/file.ini)
- check relative to a context specified on the command line (--context)
- check relative to a context specified in the code (setDefaultContext)

According to the ICUB_ROOT policy it will also:

- new since July 2009: if the environment variable ICUB_ROBOTNAME exists, check robot specific directory (app/$ICUB_ROBOTNAME/conf)
- check default conf directory in app/default/conf

Let's make an example

Suppose you have a module that needs the following parameters to run: robot and part name plus a file.

You can run the module as:

```
mymodule --robot icub --part head --inifile moreparameters.ini
```

mymodule will use a RF to decode all these parameters and find *moreparameters.ini*.

First we need to instantiate and configure the RF.

Example code in *main.cpp*

```
...
ResourceFinder rf;
rf.configure("ICUB_ROOT", argc, argv);
...
```

this configures ResouceFinder using information in $ICUB_ROOT/ICUB_ROOT.ini.

We specify a default context:

```
rf.setDefaultContext("mymodule");
```

to be appended to the "app" directory for searching resources in $ICUB_ROOT/app/mymodule. A user will be able to overwrite the context at the command line with the '--context' parameter.

Passing argc and argv to the RF allows us to pass the parameters --robot, --part and --inifile, so that later we can query the RF to get the corresponding value:

```
const char *robotName=rf.find("robot").asString().c_str();
const char *partName=rf.find("part").asString().c_str();
...
```

When we need to locate (and open) *inifile* we use the *findFile* method:

```
const char *iniFile=rf.findFile("inifile");
if (iniFile==0)
  // print error
else
  // open file
```

Optionally we can write a short mymodule.ini file embeds all parameters to the module:

```
mymodule.ini:
robot icub
part  head
inifile moreparameters.ini
```

and run the module simply as:

```
myModule --from mymodule.ini
```

Test case examples

Note: this section contains examples that useful to discuss implementation details, it is not meant to be used for documentation.

Consider the *takeOverTheWorld* module.

The parameters of *takeOverTheWorld* are:

```
--robot : name of the robot
--weapon: weapon config file
--resource: a jpg image of the weapon in use
```

Example:

```
takeOverTheWorld --robot icub --weapon weapon.ini --resource /resources/sharks.jpg
```

These parameters can be placed in a file, e.g. takeOver.ini.

In *$ICUB_ROOT/app* there are two contexts:

```
takeOverWithSharks
takeOverWithSeaBass
```

which configure the module in slightly different ways.

The same structure is replicated locally in: In *$ICUB_ROOT/takeOverTheWorld/conf/example1* and *$ICUB_ROOT/takeOverTheWorld/conf/example2*

This is a list of example with the expected behavior.

Testing rule: use default context

```
 $ICUB_ROOT/src/takeOverTheWorld/takeOverTheWorld --from takeOver.ini
```

should resolve:

```
 $ICUB_ROOT/app/takeOverWithSharks/takeOver.ini
 $ICUB_ROOT/app/takeOverWithSharks/weapon.ini
 $ICUB_ROOT/app/takeOverWithSharks/resources/sharks.jpg
```

Currently: **works**, default context applies to takeOver.ini

Testing rule: use specific context, give priority to local files in this context

```
 $ICUB_ROOT/src/takeOverTheWorld/takeOverTheWorld --context takeOverWithSeaBass --from takeOver.ini
```

should resolve:

```
$ICUB_ROOT/app/takeOverWithSeaBass/takeOver.ini
$ICUB_ROOT/app/takeOverWithSeaBass/weapon.ini
$ICUB_ROOT/app/takeOverWithSeaBass/resources/seabass.jpg
```

Currently: **works**.

Testing rule: use specific from file (respect "tab" rule), give priority to local files

```
$ICUB_ROOT/src/takeOverTheWorld/takeOverTheWorld --from ./conf/example1/takeOver.ini
```

should resolve:

```
$ICUB_ROOT/src/takeOverTheWorld/conf/example1/takeOver.ini
$ICUB_ROOT/src/takeOverTheWorld/conf/example1/weapon.ini
$ICUB_ROOT/src/takeOverTheWorld/conf/example1/resources/sharks.jpg
```

Currently: **works**.

```
$ICUB_ROOT/src/takeOverTheWorld/takeOverTheWorld --from ./conf/example2/takeOver.ini
```

should resolve:

```
$ICUB_ROOT/src/takeOverTheWorld/conf/example2/takeOver.ini
$ICUB_ROOT/src/takeOverTheWorld/conf/example2/weapon.ini
$ICUB_ROOT/src/takeOverTheWorld/conf/example2/resources/seabass.jpg
```

Currently: **works**.

```
$ICUB_ROOT/src/takeOverTheWorld/takeOverTheWorld --from ./conf/example3/takeOver.ini
```

should resolve:

```
$ICUB_ROOT/src/takeOverTheWorld/conf/example3/takeOver.ini
$ICUB_ROOT/src/takeOverTheWorld/conf/example3/weapon.ini
and fail to locate $ICUB_ROOT/src/takeOverTheWorld/conf/example3/resources/sharks.jpg (file is missing)
```

Currently: **works**, but weird uses weapon.ini and sharks.jpg from default context.

Testing rule: use default file and default context

```
$ICUB_ROOT/src/takeOverTheWorld/takeOverTheWorld
```

should resolve:

```
$ICUB_ROOT/app/takeOverWithSharks/takeOver.ini
$ICUB_ROOT/app/takeOverWithSharks/weapon.ini
$ICUB_ROOT/app/takeOverWithSharks/resources/sharks.jpg
```

Currently: **works**.

Testing rule: use default file and specific context

```
$ICUB_ROOT/src/takeOverTheWorld/takeOverTheWorld --context takeOverWithSeaBass
```

should resolve:

```
$ICUB_ROOT/app/takeOverWithSeaBass/takeOver.ini
$ICUB_ROOT/app/takeOverWithSeaBass/weapon.ini
$ICUB_ROOT/app/takeOverWithSeaBass/resources/seabass.jpg
```

Currently: **works**.

12. Documentation

Module Documentation

This is a template you can use to document your module.

Replace "exampleModule" with the name of your module.

Look here to see the documentation produced by this code once it is parsed by Doxygen: exampleModule

```
/**
*
@ingroup icub_module
\defgroup icub_yourModule yourModule

Place here a short description of the module. This will
appear in the list of the modules.

\section intro_sec Description
This module is not implemented, it is only a template for writing
the documentation of modules.

Place here a description of your module. You might want to use a list as in:

The module does:
-       this
-       that
-       ...

You might find it convenient to include an image:
\image html EXAMPLE.jpg
\image latex EXAMPLE.eps "MODULENAME running on Linux" width=10cm

\section lib_sec Libraries
List here dependencies. Often these are just YARP libraries.

\section parameters_sec Parameters
Provide a comprehensive list of the parameters you can pass to the module. For example:

--file mymodule.ini: configuration file to use

\section portsa_sec Ports Accessed
This is important. List here ports accessed by the module. This is useful to build a list of dependencies between modules.

\section portsc_sec Ports Created
Provide the list of ports created by the module. Separate them in input and output ports, specify expected data format.

Example:

Output ports:
- /mymodule/head/out: streams out a yarp::sig::vector which contains the commanded velocity of the head, the size of the vector matches the number c
- /mymodule/right_arm/out: ...

Input ports:
- /mymodule/rpc:i: input ports to control the module, accept a yarp::os::Bottle which contains commands to start/stop/quit the module.
    -   [start]: start the module
    -   [stop]: stop the module (resume with start)
    -   [quit]: quit the module (exit)

\section in_files_sec Input Data Files
If your module expect data from a file, say so.

\section out_data_sec Output Data Files
If your module writes data to a file, say so.

\section conf_file_sec Configuration Files
If parameters to your module can be passed through a txt file, describe it here.

For example:
The module requires a description of the robot through the parameter
--file.

The file consists in a few sections:
\code
name        myModule
rate        20
\endcode

\e name determines the name of the module

\e rate specifies the rate (ms) of the thread

...

\section tested_os_sec Tested OS
Specify the operating systems on which the module was tested
Example:

Linux and Windows.

\section example_sec Example Instantiation of the Module
Provide here a typical example of use of your module.
Example:

myModule --from module.ini

\author your name

Copyright (C) 2008 RobotCub Consortium

CopyPolicy: Released under the terms of the GNU GPL v2.0.

This file can be edited at src/myModule/main.cpp.
**/
```

A few notes about the content of the Doxygen documentation:

- Please describe the command line options carefully. It is important to specify how to run the module.
- Please use Doxygen throughout the code. Public classes should be documented. Classes for internal usage (of the module) would be a plus.
- Please use namespaces (see existing **iCub** code): e.g. **iCub::contrib**.
- Your images (JPEG and EPS, see template) should be placed in iCub/src/doc.
- Please list all input and output ports with description of the data types. For complex protocols and/or message definition you can resort to additional pages (they can be also placed in /iCub/src/doc and referenced through \ref from the module documentation page).
- Please add links when appropriate by using \ref PAGENAME.
- Examples of existing pages can be found in /iCub/src/doc.

Application Documentation

This is a template you can use to document an application (a bunch of scripts in a directory in ICUB_ROOT/app, which instatiate and run a set of modules to produce a meaningful behavior of the robot).

Replace "exampleApplication" with the name of your application.

Look here to see the documentation produced by this code once it is parsed by Doxygen: exampleApplication

```
/**
*
@ingroup icub_applications
\defgroup icub_exampleApplication exampleApplication

Place here a short description of the application. This will appear in the list
of the applications.

\section intro_sec Description
This application does not exist for real, it is just a template to be used
as a guideline for writing good documentation.

Place here a description of the applciation. You might want to use a list as in:

The application does:
-       this
-       that
-       ...

\section dep_sec Dependencies
List here a list of applications that are assumed to be up and running. For example
your application could assume iCubInterface and the attention system are running.

Example:

This module assumes \ref icub_exampleModule "exampleModule" is already running.

\section modules_sec Instantiated Modules
List here the modules that are instantiated by this application. This is useful to
browse the documentation of other modules. Example:
- \ref icub_exampleModule "exampleModule"

\section config_sec Configuration Files
Provide a comprehensive list of the configuration files. Usually located in ./conf. You
do not have to necessarely explain what each file does, as this should be already explained
in the documentation of each module. Link each file with the relative module it configures so
that it is possible to look up the documentation.

\section example_sec How to run the application
List here xml script(s) that allows running the application.

\author your_name

Copyright (C) 2008 RobotCub Consortium

CopyPolicy: Released under the terms of the GNU GPL v2.0.


This file can be edited at \in app/exampleApplication/doc.dox
**/
```

This is a template you can use to document an application (a bunch of scripts in a directory in ICUB_ROOT/app, which instatiate and run a set of modules to produce a meaningful behavior of the robot).

Replace "exampleApplication" with the name of your application.

Look here to see the documentation produced by this code once it is parsed by Doxygen: exampleApplication

```
/**
*
@ingroup icub_applications
\defgroup icub_exampleApplication exampleApplication

Place here a short description of the application. This will appear in the list
of the applications.

\section intro_sec Description
This application does not exist for real, it is just a template to be used
as a guideline for writing good documentation.

Place here a description of the applciation. You might want to use a list as in:

The application does:
-       this
-       that
-       ...

\section dep_sec Dependencies
List here a list of applications that are assumed to be up and running. For example
your application could assume iCubInterface and the attention system are running.

Example:

This module assumes \ref icub_exampleModule "exampleModule" is already running.

\section modules_sec Instantiated Modules
List here the modules that are instantiated by this application. This is useful to
browse the documentation of other modules. Example:
- \ref icub_exampleModule "exampleModule"

\section config_sec Configuration Files
Provide a comprehensive list of the configuration files. Usually located in ./conf. You
do not have to necessarely explain what each file does, as this should be already explained
in the documentation of each module. Link each file with the relative module it configures so
that it is possible to look up the documentation.

\section example_sec How to run the application
List here xml script(s) that allows running the application.

\author your_name

Copyright (C) 2008 RobotCub Consortium

CopyPolicy: Released under the terms of the GNU GPL v2.0.


This file can be edited at \in app/exampleApplication/doc.dox
**/
```

Document location

From Wiki for RobotCub and Friends

This page contains a description of the locations of the documents that do not fit in the source code and/or mechanical design files and/or electronic design files categories.

iCub documents locations

Documents can be found on various servers connected to the iCub project. In particular:

- data sheets are stored on: http://eris.liralab.it/misc/datasheets
- videos and other large media material: http://eris.liralab.it/misc/videos
- manuals and other documents: click here (http://robotcub.svn.sourceforge.net/viewvc/robotcub/trunk/iCubPlatform /doc/manuals/) , this is part of our SVN repository on SourceForge for the **robotcub** project.
- deliverables and other RobotCub official documents: these are all available on the deliverables folder (http://www.robotcub.org/index.php/robotcub/more_information/deliverables)

How to add documentation guidelines

Each iCub component should be documented appropriately (within reasonable limits). In general, anything created in connection with the iCub has to comply with the documentation standards set forth in **Deliverable 1.2 (licensing strategy) (http://www.robotcub.org/index.php/robotcub/more_information/deliverables/deliverable_1_2_pdf__1)** and **Deliverable 8.2 (coding standards) (http://www.robotcub.org/index.php/robotcub/more_information/deliverables /deliverable_8_2_pdf)** . Of particular importance is to attach a proper license statement to the documents (see below). The manual structure and its chapters is only available from this Wiki.

The general idea is that:

- Each document has to contain the RobotCub preamble like for example in the above said deliverables. Templates are available from http://www.robotcub.org, plese ask a username and password if you do not have one to access the private section of the website.
- Each document has to clearly refer to the FDL license, mention the constant sections, etc. as explained in Deliverable 1.2. The copyright has to be explicitly assigned to the RobotCub Consortium (see addition to Deliverable 1.2). Authors and institutions supporting the preparation of the document can be mentioned in the license preamble under the "Authors" field.
- Documents should be added in source (e.g. MSWord doc files, set of gzipped LaTex files) and PDF format to the SVN repository under the folder iCubPlatform/doc/manuals. Please make sure the PDF is available.
- Videos and data sheets should be uploaded to the respective folders on "eris.liralab.it". If you do not have a password, then please send an enquiry to admin_(at)_robotcub.org (mailto:admin_(at)_robotcub.org) .
- A manual page should be added (or links to the files to existing pages) to the respective chapter of the manual (e.g. see chapter one of the manual). The manual page describes the general settings, lists other references and links to all files needed to clearly explain the component in question. The manual page is complementary to the documents and does not substitute them.

In case of doubt please email us admin_(at)_robotcub.org (mailto:admin_(at)_robotcub.org) .

Back to the table of contents

13. Committing changes

Committing changes to the software repository

From Wiki for RobotCub and Friends

Changes

Changes to the software should be committed by following certain procedures. With the exception of debugging and syntactic fixes, changes that affect libraries and code written by others (and thus have a "global" effect) should be agreed with the maintainers. This includes YARP and code in the iCub repository, and in particular anything that influences the usage and/or the compilation and/or the functionality of the code as a whole. The addition of independent modules in iCub does not need to be agreed (but see the specifications of the modules in the previous chapters). Please check your modules with the maintainers before adding them to the global build.

New files

Do not forget to add to the repository all files that are needed to compile your code. This include source files, cmake files and documentation files (text files are recommended). Don't commit project or make files. Never commit binaries, with some exceptions (device driver libraries or firmware).

Important: make sure you add copyright and license to files you commit. If you commit files for which you do not own the copyright, make sure they have a GPL or GPL compatible license.

Check List

We welcome new contributions, but please, before you commit new modules to the repository make sure you that:

1. The code respect the current dependencies, as explained in Chapter 8, Dependencies. In particular:
   - Library dependencies
   - Tools (CMake) and compilers
2. You followed the guidelines in Chapter 10, Standardization of Methods.
3. In particular make sure the code you commit is GPL or has a GPL compatible license, see Chapter 10 (Section Licensing)
4. You documented your modules and applications as explained in Chapter 12.

Committing changes to the hardware repository

From Wiki for RobotCub and Friends

Changes are committed by following certain procedures. In particular, the CAD commits are not very well supported by CVS. Changes should be agreed with the maintainers and pros/cons discussed before committing them. Debugging changes or documentation fixes do not need to follow this procedure unless for synchronizing the commit of binary files.

14. How to install the robot

HOW TO INSTALL ICUB

Rev.	Prepared by		Date	Approved		Date
0.0	EDL	Marco Maggiali	31/03/2009			
0.1	RBCS-EDL	Marco Maggiali	15/07/2009			
0.2	RBCS-EDL	Marco Maggiali	03/11/09			

1 Revision history

Rev.	Date	Revision description
0.0	**31/03/09**	**Preliminary emission**
0.1	**15/07/09**	**Changed icub_cluster.sh with the new phyton script icub_cluster.py**
0.2	**03/11/09**	**Added information about the power supplier**

2 Summary

1 Revision history..1
2 Summary..1
3 Installing icub..1
4 Switching off icub 5

3 Installing icub

1. Remove the robot icub from the box. Lift the robot between the legs and the upper arms in two persons (video_1).

2. Remove the envelope from the torso and fix the robot to the robot stand with the four screws (video_1).

3. Use two clamps to fix the robot with its stands on a table.

4. Put the two power supply close to the robot (1-2 meters) , take the power supply cables and connect them as specified in Extern Power Supply Cable document (video_2).

5. Do not connect the power supply cable to the robot.

6. Plug the power suppliers. Check the power supply voltage and current. The XANTREX XFR 60-46 DC must be set from 24 up to 42 volts with the OVP to 45V (to be set with a screw driver), and the current limit up to 20A. The XANTREX XFR 35-35 DC must be set to 13.4 volts and 15A, with the OVP to 14V (to be set with a screw driver). Look at the documentation for detailed instructions. (video_3)

7. Without connecting the power supply cable to the robot switch on the green switches and look at the power supply. Now the output of the power supplies should be from 24 to 48 and 13 volts.

8. Switch off the green switches. Connect the power supply cable to the robot, and the Ethernet cable to the hub.

9. Switch on the laptop and connect it to the hub with the Ethernet cable. The username is icub and the password is icub.

10. Now you can switch on the PC104 and wait about 60 seconds.

11. Click on the penguin icon in the deskbar. It is for an ssh connection with the PC104. We call this shell *icub@pc104*. If it does not ask you for a password it means that the boot of the pc104 is finished. If not wait few seconds more and repeat the step 11.

12. Press the red button (it is the fault button, if it is pressed there is no power to the motors).

13. Switch on the motor boards and wait 10 seconds.

14.

15. Open a shell in the laptop: icub@icubsrv and type:

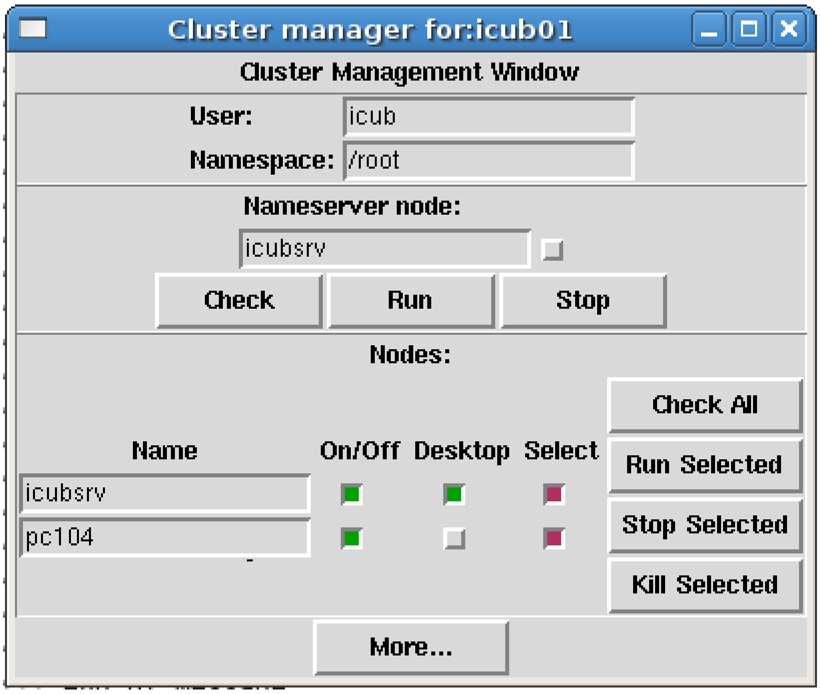
```
yarp server &
```

16. Open a shell icub@icubsrv and type:

```
cd /usr/local/src/robot/iCub/app/default/scripts
```

```
./icub_cluster.py
```

It opens a graphical interface:

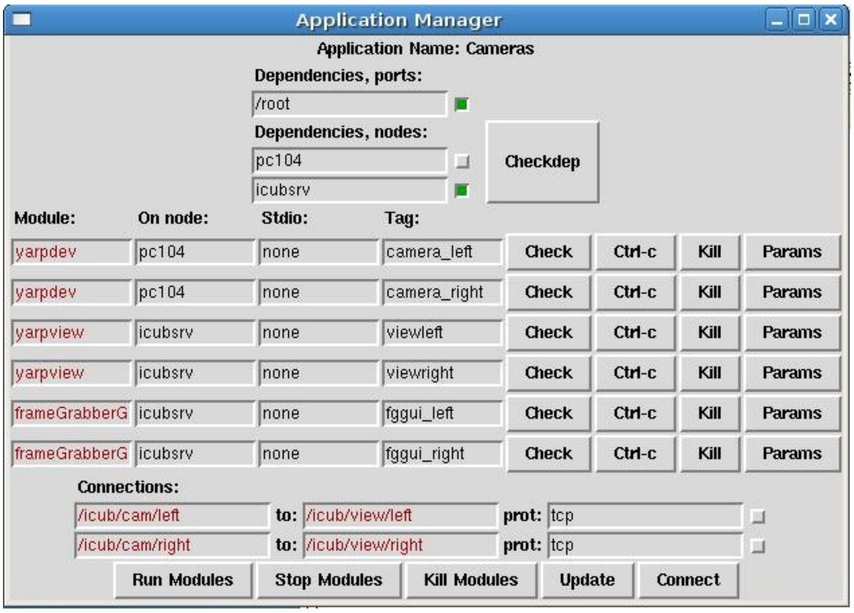


Press the button Run Selected and check if both the pc are runnig

17. Open a shell *icub@icubsrv* and type:

```
cd /usr/local/src/robot/iCub/app/default/scripts


./manager.py cameras.xml
```



18. Click on the penguin icon in the deskbar. It is for an ssh connection with the PC104.

19. Leave the red button pressed and run from the icub@pc104 shell:

```
iCubInterface --config  /usr/local/robot/icub/app/icub<cityXX>/conf/icubSafe.ini
```

20. Open an icub@icubsrv shell :

```
cd /usr/local/src/robot/iCub/app/ icub<cityXX>/conf/


robotMotorGui
```

21. Stand the robot in the home position (image_1).

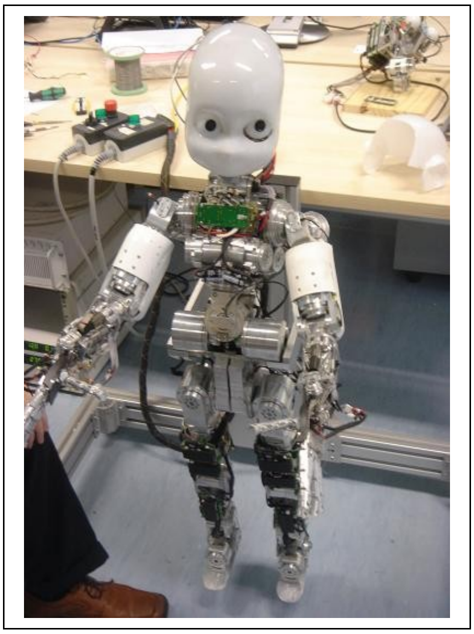


Fig 1 Home position. The robot must start the calibration in this position.

22. Look at the joint values. The value should be approximately:

    a. LeftLeg and RightLeg: 0 (± 15 degrees) for all the joints.

    b. LeftArm and RightArm: -30, +30, 0, +45, (± 15 degrees) for joints 0..3. Any number for the other joints.

    c. Torso: 0,0, 10 (± 15 degrees).

    d. Head: any number.

23. If step 22 is ok press 4 times crtl-c in iCubInterface.

24. Release the red button and run iCubInterface again with the same parameters (see steps 19). Now you will see the robot calibrating.

25. The robot calibration takes 20 seconds should.

4 Switching off icub

1. Press once *crtl-c* in iCubInterface. The robot goes to the park position and stops the power to the motors.

2. Press the red-button.

3. Open a shell *icub@icubpc104* and type:

```
sudo shutdown –h now
```

4. Wait 10 seconds and switch off the two power suppliers.

15. Unofficial documentation

Attention system for dummies

From Wiki for RobotCub and Friends

Attention! This how-to is written for linux users, but it may be useful for windows users also.

Contents

- 1 Startup the robot
- 2 Setting up the environment for running the attention system
- 3 Compiling the iCub repository
- 4 Setting up and Starting up the attention system
- 5 Using the attention system

Startup the robot

- Power up the robot. (laptop and powersupplies first, then when the robot-laptop has booted completely then turn the cpu and motor power switches on)
- Get the robot software interface running. You can run this script (in the robot-laptop) to automate even more the startup (you'll need to have a yarp server running before):

```
yarp clean
cd /usr/local/src/robot/iCub/app/default/scripts/
./icub-cluster.py &
./manager.py cameras.xml &
ssh -t pc104 iCubInterface --config /usr/local/src/robot/iCub/app/iCubMunich01/conf/icubSafe.ini iCubInterfa
killall icub-cluster.py
killall manager.py
killall yarpview
killall frameGrabberGui2
killall yarprun
ssh -t pc104 killall yarpdev
ssh -t pc104 killall yarprun
yarp clean
```

Don't close this console!
(If you want to turn off the robot software interface later, you have to press CTRL-C in this console, it will close all related software)
This will get the robot calibrated,up and running.
Disclaimer: This script is not perfect, if you find something you can improve then please modify this wiki entry.

- You still have to press the "Run Selected" button in the Cluster Manager window manually.
- You still need to activate the cameras: In the Application Manager window, press "Run" for the two yarpdev modules.

At this point the robot is ready to be used by the attention system.

Setting up the environment for running the attention system

You will probably need more than one computer to run the attention system. I was able to run the attention system with my laptop running only the graphical interface and a Intel(R) Core(TM)2 Quad CPU based computer running all the other modules.

Do the following steps in all the computers where you want to run modules.

- Get yarp compiled and installed. (let's suppose the yarp source directory is /home/user/local/src/yarp2)
- Get the iCub repository. (let's suppose the iCub base repository directory is /home/user/local/src/iCub)
- Set the following environmental variables:

```
export YARP_DIR=/home/user/local/src/yarp2
export ICUB_DIR=/home/user/local/src/iCub
export ICUB_ROOT=/home/user/local/src/iCub
export PATH=$PATH:/home/user/local/src/iCub/bin
```

Compiling the iCub repository

Do the following steps in all the computers where you want to run modules.

- I guess you will need some dependencies. I'm sure you'll need opencv to get the attention system compiled. Then install it!
- cd /home/user/local/src/iCub
- ccmake . (you'll probably need to activate CREATE_GUIS_GTK, CREATE_GUIS_GTKMM, CREATE_GUIS_QT, the rest worked for me. Be sure to check that there are no errors. Warnings are not a problem as I can tell)
- make

wait for some time ... (go and play around with your other robots for example :), or write a wiki entry for something that could be useful for somebody else)

Note: you need to compile the iCub repository for Qt (CREATE_GUIS_QT) but when you Configure Cmake (at least on Windows) you may get an error message saying iCubGui will not compile, typically because GLUT can't be found. Don't worry - you don't need iCubGui for attention - just carry on and Generate the solution file and build the repository as normal.

Setting up and Starting up the attention system

- Configure the application manager for the attention system
  - There is a configuration file template for the attention system in: /home/user/local/src/iCub /app/attentionDistributed/scripts/appConfig-visual.xml.template. Copy this file to appConfig-visual.xml in the same directory and edit it to your taste. Important things to change in the config file are the node names, if you have two computers lets name them (for Yarp purposes) "clientMachine" and "node1". Modify the config file so that the applicationGui module runs on the "clientMachine" and the other modules run on "node1" machine.

- Running yarprun
  - In your strong computer run: yarprun --server /node1
  - In your small computer run: yarprun --server /clientMachine

- Run the application manager
  - cd /home/user/local/src/iCub
  - ./app/attentionDistributed/scripts/appManager-visual.sh
  - Check that the dependency node names are the right ones (if not then just change it on the GUI)
  - Press CheckDep, all indicators should go green (this checks whether yarprun is running in the nodes)
  - Run all the modules: Press the "Run Modules" button.
  - Connect modules: Press the "Connect" button.

Now the system should be up and running.

Using the attention system

In the ApplicationGui (in the clientMachine computer) you can control the attention system.

It will initially not do anything. You have to:

- Go to salienceRight tab and press the button "Check all ports and connections"
- Inside of "Configuration Connection" check that the symbol ">>" is green when is not green then press it!
- Press "Connect All" button.
- Press "Initialize Interface"
- Now in "Salience Filter weights" you can select the weights you want for different things. Play around with it. In this moment the head should start moving looking towards your selected salience!

PrepareMacOSX

From Wiki for RobotCub and Friends

These instructions explain how to set up your Mac OS X system to compile YARP and iCub. Here we assume you are using Mac OS X 10.5 'Leopard'. It should not be too different to set up Mac OS X 10.4 'Panther' or versions newer than 10.5, just by installing similar packages.

CMake is available as a DMG installation image, while all the other libraries can be installed

- with the Fink package manager, or
- with the MacPorts package manager (formerly DarwinPorts), or
- manually.

Contents

- 1 Development environment
- 2 Libraries
  - 2.1 ACE
  - 2.2 GUIs
    - 2.2.1 Fink package names
    - 2.2.2 MacPorts package names
  - 2.3 Gnu Scientific Library
  - 2.4 OpenCV
  - 2.5 IPOPT
- 3 Simulator-related packages

Development environment

Install the following packages:

- CMake (at least version 2.4)
- g++

For CMake, there exists a precompiled installation package that you can get here: http://www.cmake.org/cmake/resources/software.html Alternatively, you can get it via Fink (`sudo fink install cmake`) or via MacPorts (`sudo port install cmake`).

In case of installing via the DMG image:
In order to use *cmake* or *ccmake* via command line, you need to add the directory which contains the binaries to your PATH variable in the *~/.profile* file, or export it each time manually. (In the author's case, this directory is */Applications/CMake\ 2.6-2.app/Contents/bin*).

g++ can be installed either via Fink/MacPorts or by installing Xcode (http://developer.apple.com/TOOLS/Xcode/) .

If you don't know what CMake is and you are wondering why you need to install CMake, please wait until Section 6.6 of the Manual. Of course you can jump there if you really can't wait.

Libraries

If you are using Fink, install the package

```
libncurses5-dev
```

If you are using MacPorts, install

```
ncurses
ncursesw
```

ACE

To compile YARP, you need ACE. The author is using ACE 5.6.3, it can be downloaded from the ace website. See also Installing ACE. Note by gsaponaro: ACE 5.7.0 also works fine.

Environment variable you should have after this procedure: ACE_ROOT

GUIs

GUIs are written using GTK/GTKMM and QT. Note that the author just installed everything that looked halfway relevant, so some of these packages might be redundant, but things definitively work with these packages.

Fink package names

GTKMM:

```
gtkmm2.4-dev
gtkmm2.4-gtk-dev
gtkmm2.4-shlibs
libglade2
libglade2-shlibs
libglademm2.4
libglademm2.4-shlibs
```

QT:

```
qt3
qt3-bin
qt3-shlibs
qt3-designer
qt3-designer-shlibs
qt3-linguist
```

MacPorts package names

```
glib2-devel
glibmm
gtk2
gtkmm
qt3
```

Environment variable you should have after this procedure: nothing new.

Gnu Scientific Library

YARP and some modules in iCub make use of the GNU Scientific Library (GSL). If you are using Fink, install packages

```
gsl
gsl-shlibs
```

If you are using MacPorts, install

```
gsl
```

Alternatively, you can download the library directly from http://www.gnu.org/software/gsl/.

Environment variable you should have after this procedure: nothing new.

OpenCV

Software in iCub makes extensive use of OpenCV. The author installed the version 1.1.0.

You can download OpenCV here (http://opencv.willowgarage.com/wiki/) .

Set the environment variable OPENCV_ROOT in your *~/.profile* file or export it manually to the location where you installed it (in my case */Users/frank/bin/opencv-1.1.0*)

IPOPT

The author is not using modules that require IPOPT. Any suggestions are welcome.

Simulator-related packages

The iCub Simulator will also need ODE and SDL. See the simulator instructions on how to do it here: Simulator libraries.

EnvironmentMacOSX

From Wiki for RobotCub and Friends

Depending on the operating system you are using, instructions in Section 6.1 "Prepare you system" have required you to set some environment variables. These depend on your system and your choices so we don't review them here. Just make sure you followed the instructions correctly.

YARP and iCub software require another couple of environment variables. This applies to all systems.

```
YARP_ROOT= point to where Yarp was unpacked (used by various applications)
YARP_DIR= typically points to YARP_ROOT (used by CMake)
YARP_CONF= where the yarpserver configuration file can be stored
ICUB_ROOT= point to where iCub code was unpacked
ICUB_DIR= points to ICUB_ROOT
```

New (since July 2009): if you have a robot, you also have to define:

```
ICUB_ROBOTNAME= name of your robot (the directory in $ICUB_ROOT/app that stores your robot configuration fi
```

In Linux and Mac OS X you do this using the "export" command. In the case of Linux it is a good idea to place them in your .bashrc file (or equivalent), in the case of Mac OS X you might want to place them in your .profile file (or equivalent). In Windows environment variables are in the System Properties tab in the Control Panel.

Append ICUB_DIR/bin and YARP_DIR/bin to your PATH

- Hint for Linux and Mac OS X:

```
export PATH=$PATH:$YARP_DIR/bin:$ICUB_DIR/bin
```

- Hint for Windows: check the current value of PATH in the control panel and extend it.

Important: YARP_ROOT and YARP_DIR have different meaning, although here they point to the same place. YARP_ROOT points to the location of the sources, YARP_DIR points to where you build your binaries. We here point them to the same place (cmake calls this in source build, in general they could be different).

Similar considerations apply to ICUB_ROOT and ICUB_DIR.

CompileMacOSX

From Wiki for RobotCub and Friends

Important: we assume here that you have completed the previous steps in the manual (see Section 6.1 "Prepare Your System" in the manual's main page).
Also **note that Mac OS X is officially not supported and everything said here should be taken as experimental**, which makes the building procedure slightly more time-consuming than under Linux or Windows but I hope that these hints will help.

Before you compile the code you need to generate make files.

Here we assume you have completed the previous steps (preparing your system, getting the software, setting up your environment).

Contents

- 1 Compiling YARP
  - 1.1 Create the YARP Makefiles
  - 1.2 Compile
  - 1.3 Example -- is YARP available?
- 2 Compile the iCub software
  - 2.1 Manual adjustments specific to Mac OS X
  - 2.2 Generate makefiles
  - 2.3 Compile

Compiling YARP

Create the YARP Makefiles

Run (don't forget to set the environment variables first):

```
cd $YARP_ROOT
ccmake ./
```

Choose the following options:

- CMAKE_BUILD_TYPE, set to "Release" in case you'd like to optimize
- CREATE_LIB_MATH, set to ON

In order to activate yarpview edit first $YARP_ROOT/conf/FindGtkPlus.cmake like described in Yarp on Mac (http://eris.liralab.it/wiki/YARP_on_Mac#Enabling_yarpview) and set

- CREATE_GUIS to ON

Important: CREATE_GUIS and CREATE_LIB_MATH require you have installed the libraries gtk and gsl (see PrepareMacOSX)

Create the makefiles by selecting configure several times and then generate.

Installation: CMake automatically creates an install rule for target/project. In the documentation we assume you install binaries in $YARP_ROOT/bin and $YARP_ROOT/lib. The compiler will build executables and libraries there, so you don't need to perform the installation. You can instruct CMake so that it generates make/project files that install to other places, for example $YARP_DIR/bin and $YARP_DIR/lib. You can do this by running cmake again and setting the variable:

- CMAKE_INSTALL_PREFIX to $YARP_DIR

When you do make install all binaries will be copied to $YARP_DIR/bin and $YARP_DIR/lib.

Of course you can customize the installation directory as you wish, however the remainder of the documentation assumes the above configuration.

- Depending on the hardware on your system you might want to compile additional device drivers. This is done for example on the pc104. Instruction for doing this are reported elsewhere.

Compile

Now we are ready to compile. This is easy.

Run:

```
cd $YARP_ROOT
make
```

Example -- is YARP available?

Now we're ready to run a simple Yarp code to test the installation so far. You might want to prepare a yarp.conf file in the conf directory similar to this one:

```
127.0.0.1 10000
```

which tells Yarp (the server) to start on the localhost and respond to port 10000. This allows Yarp applications to find the name server (see next chapter).

You can then try running the server. On a terminal window, type:

```
yarpserver &
```

and you should see:

yarp: Port /root active at tcp://127.0.0.1:10000 Name server can be browsed at http://127.0.0.1:10000/ yarp: Bootstrap server listening at mcast://224.2.1.1:10001

if you type on a web browser http://127.0.0.1:10000 you get information about the name server (registered ports, info, etc.).

For the time being we can just check functionality by running a simple example. On another terminal type:

```
yarp read /portread
```

on a third terminal:

```
yarp write /portwrite
```

and on yet another terminal:

```
yarp connect /portwrite /portread
```

you'll see the effect on the name server:

```
yarp: registration name /portwrite ip 127.0.0.1 port 10012 type tcp
yarp: registration name /portread ip 127.0.0.1 port 10002 type tcp
```

Now, anything typed on the yarp write will be sent and printed on the read side.

Compile the iCub software

Manual adjustments specific to Mac OS X

So far everything was pretty straight forward on the Mac. Now comes a bit of manual editing that is due to the fact that the icub software was not written for Mac OS X.

1. **If you modified *FindGtkPlus.cmake*** previously, undo your changes now
In order to make compile yarpview you had to outcomment a section in *$YARP_ROOT/conf/FindGtkPlus.cmake* like described in YARP on Mac (http://eris.liralab.it/wiki/YARP_on_Mac#Enabling_yarpview) . This very change will make the canloader module crash if you try to compile the icub repository as it is. So if you did outcomment the said section before in order to compile yarpview, do the opposite now and comment it again. This is a hack and not even an elegant one. So any suggestions how to circumvent this uncomment-comment are highly appreciated.

2. **Prerequisites for getting the simulator to compile** (thanks to Gianluca for pointing this out)
In order to compile the simulator (which will be compiled by default) you need to perform the following changes in *$ICUB_ROOT/src/iCubSimulation/CMakeLists.txt*:
Add below the line *LINK_LIBRARIES(${ODE_LIBRARIES})*

```
IF (APPLE)
  INCLUDE_DIRECTORIES(${OPENGL_INCLUDE_DIR})
  LINK_LIBRARIES(${OPENGL_LIBRARIES})
  ADD_DEFINITIONS(-DHAVE_APPLE_OPENGL_FRAMEWORK)
ENDIF(APPLE)
```

Add below the line *ADD_DEFINITIONS(-DEXPERIMENTAL_CONFIG_METHOD)*

```
IF(APPLE)
  ADD_EXECUTABLE(${PROJECTNAME} ${folder_source} ${folder_header}
  /Library/Frameworks/SDL.framework/Resources/SDLMain.m)
ELSE(APPLE)
  # original line without Apple-specific change
  ADD_EXECUTABLE(${PROJECTNAME} ${folder_source} ${folder_header})
ENDIF(APPLE)
```

3. **If you want to enable CREATE_GUIS_QT** (see also below), you have also to adjust the CMakeLists.txt files in the root directories of the following modules

(the root directory with regard to the *libYARP_QWidgets* module for example is *$ICUB_ROOT/src/gui/libYARP_QWidgets*):

- libYARP_QWidgets
- libICUB_QWidgets
- qViewerGui
- applicationGui
- iCubGui

in the following way:
Add in the corresponding *CMakeLists.txt* files of the mentioned modules below the lines *INCLUDE_DIRECTORIES* an additional one, for example like:

```
IF (APPLE)
   INCLUDE_DIRECTORIES(
   /usr/X11R6/include
   )
ENDIF (APPLE)
```

This is the directory where your GL/qt.h and GL/glu.h reside. (The author is not sure anymore if he did anything particular in order to have these files in this particular directory, they might come with the X11 installation which is a default with Mac OS X 10.5 and later versions but an optional install with Mac OS X 10.4).
Mac OS X comes with its own version of OpenGL and one typically seems to include *OPENGL/qt.h* instead of *GL/qt.h* which is the cause for the compiler errors. The headers of the Mac version of OpenGL reside typically in */System/Library/Frameworks/OpenGL.framework/Headers* were they are found automatically by cmake. So an alternative to the changes above would be for Mac users to change the headers of the mentioned modules to include *OPENGL/qt.h* and *OPENGL/qlu.h* instead of their *GL/..* counterparts.

A more elegant way to fix this would be to either use a cmake variable instead of an absolute path for example by defining a FindOPENGL.cmake or to introduce a OS switch in the preprocessor directives of the header files of the corresponding modules. Feel free to contribute.

Generate makefiles

Now you can generate make files (Technically speaking you could have generated them before already but the compilation based on those would have failed). In $ICUB_ROOT:

```
ccmake ./
```

You don't need particular options. If you want to compile using optimization just set:

- CMAKE_BUILD_TYPE to "Release"

Other options are:

- CREATE_GUIS_GTK
- CREATE_GUIS_GTKMM
- CREATE_GUIS_QT

These options are recommended, because they enable compilation of some useful GUIs. Important: these options can be enabled only if you have installed the required libraries: gtk, gtkmm and qt (see PrepareMacOSX).

- Similarly to YARP, by default make will build executables and libraries in $ICUB_ROOT/bin and

$ICUB_ROOT/lib. You can customize where "make install" will copy these files by setting: CMAKE_INSTALL_PREFIX to something you like.

If you need to compile devices that provide interface to the hardware you can follow this link Compilation on the pc104

Compile

Compile the code.

```
cd $ICUB_ROOT
make
```

Artoolkittracker mac

From Wiki for RobotCub and Friends

Background:

ARToolKit adapts its global settings like the pixel format to the operating system where it is compiled on. That means it is per se not configured to work with the image format used by the frame grabbers provided by yarp. The module in the icub repository was originally written for Windows and Linux and supports their image formats. Unfortunately when the yarp independent source code is compiled under Mac OS X, the default pixel format is set to ARGB. This works fine when used independently from the repository module but it will make said module crash as the yarp frame grabbers deliver a RGB pixel format. For the default pixel format under Windows, which is set to BGRA (ARToolKit viewpoint) there is a opencv conversion from the the pixel format delivered by the grabber (RGB) to the pixel format that is assumed by the ARToolKit libraries when compiled under Windows (BGRA). In the case of Linux it is even easier as most image/frame grabber frameworks like Video for Linux (V4L) or GStreamer produce the RBG pixel format so that no conversion is necessary. When compiled under Mac Os X ARToolKit sets the default pixel format to ARGB. Unluckily the OpenCV conversion command (*cvCvtColor(..)*) doesn't seem to be able to perform this particular conversion (RGB2ARGB). Therefore the easiest way to get the ARToolKit module to work under Mac OS X is to re-compile ARToolKit itself before compiling the module in the repository (which uses the libraries from your custom ARToolKit installation) and to 'tell it' explicitely that we are expecting our images to be of the type RGB and not ARGB despite beeing on a Mac Os machine.

Solution:

The easiest way to get the ARToolKit tracker module to run under Mac OS X is to recompile ARToolKit. Edit AFTER having called *./configure* the header file that sets the global constants like the default pixel format. Note that this edit most probably will cause problems with the standalone ARToolKit programs but it will enable you to use the ARToolKit tracker module from the repository without any changes. In

$ARTOOLKIT_DIR/include/AR/config.h

replace

```
#define AR_DEFAULT_PIXEL_FORMAT_ARGB
```

with

```
#define AR_DEFAULT_PIXEL_FORMAT_RGB
```

and call *make*.

Potential future solution:

Add an additonal OS switch for Mac Os (#ifdef __APPLE__) and write an image conversion method that does the RGB2ARGB conversion. Call this method from ARToolKitTracker::detectMarkers(IplImage *iplimg) in $ICUB_ROOT/src/artoolkittracker/ARToolKitTracker.cpp.

Cluster

From Wiki for RobotCub and Friends

Important: this documentation might be obsolete. It describes a the cluster configuration that was used some time ago to run the iCub at University of Genova.

We are using a Blade server with 6 blades at the moment running Debian Linux and a server/interface machine running Windows 2003. We also have 5 dual-boot (XP/Linux)Shuttle PCs and two more rack mount machines, a bunch of screens and several laptops connected to the same network.

This document describes certain procedures for the maintenance of this system.

Most of the disk space is on the Windows 2003 machine that shares it using both NFS and Samba. The Windows 2003 machine also runs the Unix Services for Windows and the YP server, name and permission mapping, etc.

On each Linux machine we mount the home directories (or anything else need, e.g. Yarp) using NFS, while Windows machines we use Samba (i.e. the native Windows network). Linux machines use yp. The yp server is on the Windows 2003 machine where also the Windows domain is managed.

Contents

- 1 Configuration
- 2 Shared drives
- 3 Environment variables
- 4 Notes
- 5 SSH public key authentication
- 6 CYGWIN
- 7 Openssh on windows
- 8 Tips and tricks
  - 8.1 Making SSH read your environment variables on the target node
- 9 Name server
- 10 Missing things

Configuration

The cluster consists of a certain number N of machines with various operating systems (not necessarily uniform). One machine is directly attached to the robot hardware and this should be the case for compatibility with iCub (where all the hardware is interfaced through the on-board pc104 machine). Since Yarp doesn't provide security, the cluster network is physically separated from the outside world via a server/router. The server in our case here runs Windows 2003 but this is not a requirement. Separating the cluster from other networks also limits the potentially high bandwidth traffic to the robot subnetwork(s).

The blade computers are mutually connected via two Gbit/s switches. These are private networks responding to the 10.0.0.X and 10.0.1.X addresses. The server machine called MUSE in our implementation is also connected to both network switches and to the outside lab network. Several other machines are only connected to the first private network (10.0.0.X).

MUSE is a domain controller for the Windows network, it is also a NIS domain server for Linux (we use Debian). It is a

DNS for the machines in the private networks. MUSE maps users from Windows to Linux via NIS and also maps groups and other tables from Windows to Linux and vice versa. It exports directories using NFS for Linux and Samba for Windows. It runs also a NAT to map the private IP addresses into a pool of public ones.

Cygwin is used on all Windows machines to run a ssh server. This is used on MUSE also to allow access to the other machines through ssh tunneling (don't remember the port numbers at the moment) but also for scripting using bash irrespective of the operating system. Cygwin provides also the cvs client we currently use on Windows.

Shared drives

MUSE shares a **separate directory** for each operating system (e.g. Linux, Windows) that contains in turn Yarp, the iCub repository and other libraries if needed.

From Linux, typically mount the shared directory (see extract from mount):

```
muse.james.liralab.it:yarp on /usr/src/yarp type nfs (rw,intr,addr=130.251.4.3)
```

Which can be added to /etc/fstab:

```
muse.james.liralab.it:yarp      /usr/src/yarp   nfs     rw,intr
```

Which will then show up as (ls -la /usr/src/yarp):

```
drwx------ 2 4294967294 Domain Users   64 2006-10-28 15:56 .
drwxrwsr-x 9 root       src          4096 2006-08-23 17:20 ..
drwxr-xr-x 2 babybot    Domain Users   64 2006-10-28 16:04 iCub
drwxrwxr-x 2 babybot    Domain Users   64 2006-07-22 09:49 yarp
drwxrwxr-x 2 babybot    Domain Users   64 2006-10-28 16:25 yarp2
drwxr-xr-x 2 james      Domain Users   64 2006-10-29 00:08 yarp2-james
drwxr-xr-x 2 babybot    Domain Users   64 2006-10-27 10:54 yarp2-unstable
```

From Windows, typically mount the shared directory as drive Y (or anything else you like using the standard Windows "Map Network Drive".

Environment variables

These are the environment variables recommended (not all of them are required):

```
ICUB_ROOT
YARP_ROOT
YARP_BUILD
YARP_CONF
```

Important: if you are using CYGWIN check that the environment variable CYGWIN exist (for example se it to "smbntsec ntsec). The .bashrc script check $CYGWIN to determine it it is running on Linux or CYGWIN (see below).

For example in Linux you can do something like this in the .bashrc and perhaps make sure that the .profile calls a similar sequence of commands:

```
export ICUB_ROOT=/usr/src/yarp/iCub
export YARP_ROOT=/usr/src/yarp/yarp2
export YARP_BUILD=$YARP_ROOT
if [ -e /etc/debian_version ] ; then
  debtype=`cat /etc/debian_version | sed "s|.*/||"`
  if [ "k$debtype" = "kunstable" ]; then
    export YARP_BUILD=/usr/src/yarp/yarp2/build/$debtype
  fi
fi
export YARP_CONF=$YARP_ROOT
```

In Cygwin you can reuse the same .bashrc (highly recommended) conditioning on the operating system type:

```
if [ ! "k$CYGWIN" = "k" ]; then
  export ICUB_ROOT=//MUSE/yarp/iCub
  export YARP_ROOT=//MUSE/yarp/yarp2
  export CYGWIN="smbntsec ntsec"
  export YARP_BUILD=$YARP_ROOT
  export YARP_CONF=$YARP_ROOT
```

where the remote path //MUSE/yarp is used directly (this is safer than the mount Y when scripting). IMPORTANT: make sure the environment varible CYGWIN exist on Windows.

Note the *smbntsec* flag which is required to map Cygwin permission properly on Samba drives.

In Windows, you can add the same variable names using the appropriate dialog from the control panel. They might show like:

```
  ICUB_ROOT    Y:\iCub
  YARP_ROOT    Y:\yarp2
  YARP_BUILD   Y:\yarp2
  YARP_CONF    Y:\yarp2
```

Cygwin requires also the environment variable CYGWIN to be set. The best is to allow at least *smbntsec ntsec tty* which determine how permissions are also seen on mapped drives (useful if they are consistent).

In general, be gentle to others and set the variables only for your user and not in system-wide scripts, this applies to both Linux and Windows.

On Windows, add to the PATH the following directories:

```
Y:\ACE_wrappers\lib;Y:\yarp2\bin;Y:\iCub\bin
```

On linux do a similar operation depending on where you installed the executables:

```
$YARP_BUILD/bin:$ICUB_ROOT/bin
```

Notes

Getting so many problems is not typical, they are due to the fateful intersections of many different things, including the presence of mixed operating systems, network drives, cygwin, sshd, carriage return characters, and more.

Various notes on useful operations:

- Accessing a computer with the Windows remote desktop.
  - Applies to: **Windows XP**

- You need to add the username to the local policy.
    - Go to the "Control panel", click on "System", go to the "Remote" tab and click to "Select Remote Users", then finally add the username to the list of users (possibly a domain user).
    - This operation has to be performed by an Administrator.

- Adding a user to NIS.
    - Applies to: **Windows 2003**, **Unix Services for Windows 3.5**
    - Don't forget to add the domain name into the Unix tab in the "Active Directory Users and Computers".
    - Add also the UID that will be shown on Unix/Linux machines and the other Unix parameters.
    - Then don't forget to update the name/permission mapping service (also from the Unix services administration application).
    - Make sure you "Reset the password" for the user in question so that the NIS (damn!) sets a new password that is compatible on both Windows and Linux; this doesn't happen automatically.

- Adding a user.
    - Applies to: **Windows**
    - Don't forget to run *mkpasswd* in a cygwin shell to update the passwd file (you must be Administrator). This step has to be performed on every Cygwin installation (on every Windows machine on the cluster).

```
mkpasswd -l -d > /etc/passwd
mkgroups -l -d > /etc/group
```

- Installing cygwin.
    - Applies to: **Windows**
    - Check the DOS/Windows text mode during installation and NOT the Unix mode for newlines
    - Install "open ssh" and at least one text editor (e.g. vim, it can be handy).

- Issues with the ssh keys.
    - Applies to: **Windows** and **Linux**
    - For scripting is convenient to have a key installed
    - This goes typically in $HOME/.ssh
    - The $HOME is a networked folder
    - Ssh requires that the keys are only readable by the owner
    - Thus, you have to make sure that the permissions are always correct no matter how you access them
    - For cygwin this requires for example the definition of the CYGWIN variable to be *smbntsec*
    - See below for the installation of the sshd on Windows.

- Weird issue with the user settings.
    - Applies to: **Windows 2003**
    - We need to investigate the problem
    - The "Active Directory Users and Computers" from the control panel doesn't show the Unix attributes that are required to set the UID and other Unix properties for the name/group mappings
    - Temporary solution is to open the Windows MMC from the "Microsoft Windows Services for UNIX" application, searching for "adding user nis" and click to "To add a user to an NIS domain". This will show a link to a version of the MMC that will show correctly the Unix attributes
    - In the hope this is not a persistent problem.

- Shell issue with CR/LF on Cygwin.
    - Applies to: **Windows**
    - bash 3.1.17(9) is strict on the endline and if Cygwin is installed with native endline support (i.e. Windows/DOS like) then the scripts would complain.
    - Solution: run the dos2unix utility on all the script files just after downloading from the CVS repository

SSH public key authentication

Bored to type the password every time you log in to one of your machines? You can set up ssh to use public key instead.

These instructions were taken almost verbatim from: http://cfm.gs.washington.edu/security/ssh/client-pkauth/

On the client machine:

```
client$ mkdir ~/.ssh
client$ chmod 700 ~/.ssh
client$ ssh-keygen -q -f ~/.ssh/id_rsa -t rsa
```

Enter an empty passphrase twice (yes, it is not safe... we don't care).

Make sure everything has the correct access rights:

```
chmod go-w ~/
chmod 755 ~/.ssh
chmod go-rwx ~/.ssh/*
```

Keys distribution. If the machines share the same users server and client are actually the same machine and you don't have to copy anything, anyway in general do:

```
client$ scp ~/.ssh/id_rsa.pub server.example.edu:
```

Log on into the server and type:

```
server$ mkdir ~/.ssh
server$ chmod 755 ~/.ssh
server$ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
server$ chmod 644 ~/.ssh/authorized_keys
server$ rm ~/id_rsa.pub
```

Now go back to the client and test ssh by doing:

```
ssh -o PreferredAuthentications=publickey server.example.edu
```

If everything is allright you should login to server without password. Otherwise it means that something is wrong. Most of the times this is due to bad configurations rights, check out this:

```
server$ chmod go-w ~/
server$ chmod 755 ~/.ssh
server$ chmod 644 ~/.ssh/authorized_keys
```

Or go to /var/log and have a look at the messages dumped by sshd.

Note on permissions: usually on linux it is recommanded to set permissions like:

700 for ~/.ssh

600 for ~/.ssh/authorized_keys

However this does not work on cygwin because the *sshd* service seems to have troubles accessing those files.

If you want to login to a machine from a different user, you can do:

```
ssh otheruser@machine.domain.edu -i ~/.ssh/id_rsa
```

Finally on Windows go to:

Administrative Tools --> Services --> CYGWIN sshd --> properties --> Allow service to interact with desktop

CYGWIN

A brief help on how to configure CYGWIN to run YARP/ICUB.

- Go to http://www.cygwin.com and download setup.exe. Default installation is fine just add openssh (although you might find it useful to have also a couple of packages like vi, nano or emacs).
- Open a shell and type:

```
mkpasswd -l -d > /etc/passwd
mkgroups -l -d > /etc/group
```

- On windows check if the environment variable CYGWIN is set to **smbnet ntsec**
- Install openssh, see next section.

Openssh on windows

Assuming you installed cygwin with openssh.

Open a cygwin window (by double clicking theg icon), a black screen pops open, type:

```
ssh-host-config
```

- When the script asks you about "privilege separation", answer **yes**
- When the script asks about "create local user sshd", answer **yes**
- When the script asks you about "install sshd as a service", answer **yes**
- When the script stops and asks you for "CYGWIN=" your answer is **ntsec tty**
- While you are still in the (black) cygwin screen, start the sshd service:

```
net start sshd
or
cygrunsrv  --start  sshd
```

The following steps are also required if you want to use login and run processes remotely:

- Stop the "Cygwin sshd" service (go to "Computer Managment", open "Services", right click on "Cygwin sshd").
- Select "Cygwin sshd" properties. Under the "Log On" tab, change the name of the account to the user (MY_USER) that runs the processes (e.g. james@james.liralab.it)
- Tweak the Local Policies (Control Panel -> Adminstrative Tools -> Local Security Settings -> Local Policies). Grant to MY_USER the following:
  - Adjust memory quotas for a process
  - Create a token object
  - Lon on as a service (this should be on already)
  - Replace a process level token
- Change the ownership of the files required to run the sshd daemon. On a cygwin bash type:

```
chwon MY_USER /var/log/sshd.log
chown -R MY_USER /var/empty
chwon MY_USER /etc/ssh*
```

Good luck, you need it.

[from http://pigtail.net/LRP/printsrv/cygwin-sshd.html]

[from http://ist.uwaterloo.ca/~kscully/CygwinSSHD_W2K3.html

Tips and tricks

Tricks you need sometimes to make things work on the cluster. This list will hopefully improve/grow with time.

Linux:

- increase the size of the udp packets:

```
(sudo) sysctl -w net.core.rmem_max=8388608
```

Add this to your /etc/init.d/bootmisc.sh

If you are a developer and you want to use cvs to checkout yarp2 with command line (linux or cygwin):

```
export CVS_RSH=ssh
cvs -z3 -d:ext:your_name@yarp0.cvs.sourceforge.net:/cvs root/yarp0 co -P yarp2
```

Making SSH read your environment variables on the target node

When you start processes remotely via SSH like it is done by the cluster manager ($ICUB_ROOT/app/default/icub-cluster.py) neither .bashrc nor .bash_profile might be read on the target node, so your environment variables that are set there are not available. This means that for example yarprun won't be found. In order to get ssh to read your environment variables when called in a non-interactive mode like 'ssh -f' add them to the file

/etc/environment (available for every user)

or alternatively to

~/.ssh/environment (available only for a particular user).

Name server

The name server runs on nike. On /usr/bin/yarpserver is a copy of yarp. A script in /etc/init.d yarpserver starts/stops the service at boot time.

Missing things

Cluster clock synchronization