Octopus-ReEL
User's Guide

Barkın İlhan
barkin@unrlabs.org
www.unrlabs.org/barkin/
Unified Neuroscience Research Labs

August 25, 2018

Chapter 1

Introduction

Octopus-ReEL, or "0ct0pu5" –as we name it in hacker homoglyphs for easy discrimination on the Net– is a **"holistic"** hard (Re)altime (E)ncephalography (L)aboratory. Originally it has been designed as an EEG source localization/imaging framework as part of a PhD thesis, however with the very holism in mind that the design will not ever be confined to any methodology *–i.e. only EEG–*, space *–i.e. size of equipment–* and time *–i.e. jitter–* constraints. This also brings up the need for the involvement of concepts in the technological frontier. The first issues that could potentially be foreseen could be MEG and MRI interfacing, wearable devices' development, and seamless cloud/HPC connectivity. As the needs will emerge, provided that the developers also have the opportunity, courage and time. As a summary, our motto is that "there shouldn't exist anything you can't do with octopus".

Octopus is a networked system having hard realtime parts involving realtime kernel modules, daemons communicating with them in userspace, and OpenGL-enhanced GUI applications for managing the whole system. It owes its strength basically to the underlying real-time host-based design and DAQ card driver base, thanks to the ingenius efforts of RTAI (https://www.rtai.org/) and COMEDI (https://www.comedi.org/) developers. As the lower-level side reaches to excessive hardware usage, the core kernel drivers' development is being done in C. The higher the functionality gets, choices relax to C++ with Qt support and Bash scripting. In the future, project is planned to evolve with incorporation of Erlang. Also some minimal Matlab code inside is planned to be converted to Python in the first place. We deem programming language choices very important in a project. As emphasized previously, technological frontiers have nothing to do with technological popularity, so one will potentially see some LISP code much likely than C#, Scala, etc. code –recalling some proprietary stuff– inside 0ct0pu5 codebase, even though it will cause a less crowded developer base.

Slow and proper evolution is better then it to be hasty, bogus, and full of cul-de-sacs.

The three types of computer systems on the typical 0ct0pu5 laboratory network are *servers*, *computational nodes*, and *workstations*. The *file server* has the file system and user accounts common to all computers handled via NFS and NIS, over which all the centralized action takes part. The acquisition and stimulus servers operate in real-time in-between, exploiting the synchrony provided by parallel port connections, and they provide data transferred to workstations (i.e. to be displayed, recorded, analyzed on-the-fly for neuro-feedback, etc.). Hence, the installation order is important and the servers are to be installed and configured first. In this document, we describe in detail, separate installation procedures for individual types of computers.

Since Octopus has been developed using Debian/GNU Linux OS, detailed recipes (and tips whenever necessary) for a proper Debian/GNU Linux installation will be given and all the material coming forth will assume this installation. The overall system given in *Figure 1* will be used as a general reference and nomenclature in the rest of this text.

Chapter 2

Brief History

The development started in 2005 and a fully functional version (which forms the master branch of this git repository) has been defended in September 2009 as a PhD thesis of the main developer (CITE). However, due to professional issues (making a postdoc in a semi-relevant area, etc.) it could not be published and maintained in the following years. It is only now that there's the time and a potentially proactive developer and userbase who have the needed motivation and momentum.

Chapter 3

Installation

3.1 Quick/automated Installation

Octopus is a sophisticated system. This sophisticated nature comes from the fact that some parts depend on hard real-time OS components and all of the system is by essence networked. Normally there is a policy to have dedicated services on dedicated machines (such as stimulation, data acquisition, computation, etc.)

3.2 Manual/detailed Installation

Some very detailed geek stuff...

Chapter 4

Debian/GNU Linux installation

-A minimal user-level literacy is assumed for unices and specifically for Linux is assumed, however due to system complexity we're doing the best we can, in order to save the user from all types of problematic issues. If you have any difficulties during installation, please report it to xxx@unrlabs.org and we'll update the scripting and documentation taking it into account.

scripts directory: -debian stable assumed -fast updates will occur during debian version changes.

1.install essential packages (in developer's guide describe in detail, where each package is used internally). Also, in 32-bit and 64-bit, several issues change. Indicate them where available. 2.download n extract kernel rtai comedi newlib 3.copy initial .config files to kernel and rtai 4.-minimal manual interaction (refer to RTAI doc and other projects s.a. RELACS 4b.setup finalization 5.compile kernel newlib rtai comedi 6.establish system-wide scripts 7.general diagnostics n validation come

In this chapter, a recipe-like installation procedure will be given for Debian/GNU Linux installation. Detailed ad-hoc comments will be given when necessary.

4.1 Basic installation

Debian can be installed using various media (for detailed documentation see official Debian GNU/Linux documentation and Wikis, http://www.debian.org, http://wiki.debian.org). Here, we'll describe installation using a minimal disk (i.e. the *netinstall* method).

* I recommend to have a check and have familiarity with the BIOS setups

of your computers you'd like to use as a part of OCTOPUS system.
Download netinst.iso image from debian site.
Write the image to a CD and boot your computer (you can use alternate media such as a USB-disk, but these will not be covered here).

4.2 Setting up a custom real-time kernel

The most stable kernel with a RTAI patch, tested so far with Octopus is 2.6.38.8.

4.2.1 Patching for RTAI

patch -p0 < /usr/src/rtai/base/arch/x86_64/xxx.patch

4.2.2 Kernel configuration

make mrproper
make menuconfig

4.2.3 Compiling and running

bla bla

Chapter 5

Selection of hardware

Octopus is designed in a standard neuroscience laboratory environment (i.e. having several old/obsolete computers, monitors, DAQ cards thrown away). In that sense, its minimum hardware specifications are modest. Needless to say, all computers should have Fast Ethernet supported by Linux. In-between cluster nodes, it is good to have Gigabit ethernet. For data acquisition, it mainly depends on your timing needs (i.e. sample rate), and the complexity of the individual stimuli for the stimulus servers. For the workstation, however, a more powerful computer with fast OpenGL graphics, and dual-head display should be preferred. So far, the only proprietary device taking part in Octopus system is the Polhemus FasTrak system, used for digitizing electrode positions for performing a realistic-head source localization. Octopus drives it very efficiently, resulting in very practical and reliable coordinate measurements.

As a rule of thumb, any EEG amplifier system giving analog outputs for individual channels can be used with Octopus. However, a custom amplifier scheme with 16-channel modules is supplied for electronics gurus, and other amplifier schematics (with higher CMRR, less cross-talk, etc.) will be added soon. OpenEEG based systems can also be used. Octopus also has the circuitry schematics for other devices such as the response pad, channel-matched audio amplifier, and several types of trigger cables.

Starting from Figure-1, a typical Octopus system will have these individual hardware components:

5.1 EEG amplifier

Most likely, it can be built and even customized for your needs from our white paper. A single module has 16 channels, and it is extendable to some 128

channels (even though supported 256 channels are overkill for most setups).

5.2 Auditory stimulator: amplifier and speakers

It can be built using the included schematics for the channel-matched Stereo audio amplifier with fine-tuned dB settings. Any good pair of speakers or audiological headphones (e.g. Etymotics ER-3A) can be used in conjunction.

5.3 Central server system

This should be a middle-to-high ranked computer with large HDD, as it is to serve as the central NIS/NFS/DNS server.

5.4 Acquisition server

Basically you won't need a very powerful computer (in the 1st generation Octopus we had a Pentium II with 128 MB RAM). However, you'll need single or multiple A/D cards supported by COMEDI project (the ideal is to have NI PCI-6071Es, as they were the ones used during development), and the second important thing is your motherboard. It should have perfectly working parallel and serial ports (one from each). This is especially important not to miss any triggers. USB ports are handy to have +5V supply when needed (e.g. response pad module needs one).

5.5 Stimulus server

Stimulus servers basically use a D/A card (again it should be supported by COMEDI project; the ideal is NI PCI-6711) to deliver auditory, or tactile, or any other type of stimulus and sends a relevant trigger to acquisition server(s) to register together with the continuously recorded EEG/electrophysiological etc. data. So again the power of the computer depends on what you'll do. For dynamic auditory stimulus synthesis, it should be capable of running a real-time loop 44100 times a second. If you'd like to deliver visual stimulus (isolated or in conjunction with auditory or tactile stimuli), it should have a supported display adapter connected to a CRT or LED monitor (again depending on the task). And finally it should have a perfectly working parallel port.

5.6 Position Digitizer

A Polhemus FasTrak system should be connected to the Operator workstation via serial port, if "realistic head" source localization features are desired to be used. Since this resides as an expensive and proprietary counterpart, there are ongoing efforts to replace it with a cheaper (most probably photogrammetric) method. Please e-mail me at ¡*barkin@unrlabs.org*¿ if you have any comments on this.

5.7 Beowulf cluster/compute nodes

If you plan to use HPC features for realistic head modeling from an MRI set and/or forward/inverse solutions, computers to serve as nodes of a Beowulf computational cluster (will be described separately). There are also plans to develop GPU-based code for head modeling part.

5.8 Operator's workstation

This is the only computer you directly interact with, so choosing a powerful one with two good monitors would be wise. It should have a serial port if you plan to use Polhemus FasTrak. All of the other servers and hardware components are controlled from the setting files and GUI software running on that computer.

Chapter 6

Tips for practical usage

bla bla

6.1 Acquisition servers

In most of the configurations there will be only a single acquisition server (i.e. that is acquiring multiple electrophysiological modalities), however there has been an open door left for multiple servers as *acq0,acq1,. . . .,acqn.*

6.2 Stimulus servers

Just like the acquisition servers, there can be multiple servers for stimulus delivery to subjects, i.e., stim0,stim1,. . . .,stimn.

6.3 Operator's workstation

Currently, there can be only a single operators workstation *(op0) , but we have plans to support multiple workstations soon.*

Chapter 7

FAQs

Q:What does Octopus stand for?
AIf we're to join the crowd, let's say its, errr. Open Cranium Tracking for Organized Programmable Unified Structures,... fine right?

Q:Are we to use Linux, is there plans to port to other OSs?
A:Octopus runs on Linux, definitively for servers because of the dependence upon RTAI. For workstations, however, it is a matter of preference, and other OSs having a Qt library (e.g. Mac OS X, Windows) can be used as well with proper configuration.

Chapter 8

Short-term Design and development needs (a.k.a.TODO)

Assuming properly installed Debian9 systems with sshd and admin.admin accts (uid=gid=1000) for leviathan, acqX, stmX and guiX:

-Script will be written which will accept IP and server type and automagically setup all the internal stuff to the maximum level as possible. Scripts will be launched from leviathan machine which will have a minimal manual setup, which will be described. Then:

i.e. install_server.py 10.0.11.10 0 leviathan (10.0.11.10 is leviathan already)

should install leviathan, whereas

install_server.py 10.0.11.30 0 acq 1

should install some acq1.octopus0 machine properly.

Note that there can be multiple acq and stim servers...