Signal Server – UserManual

Release 1.0

Christian Breitwieser – c.breitwieser@tugraz.at

August 01, 2011

Contents

1 Introduction

The TOBI signal server describes a programm using TiA (TOBI interface A) to acquire and distribute raw biosignals. Implementation was done using C++, static and shared library packages are available for Debian based systems (32bit and 64 bit), Microsoft Windows Xp, and Windows 7.

1.1 Supported hardware

- g.USBamp (Windows only)
- g.Mobilab
- g.BSamp
- BrainProducts Brainamp series (Windows only)
- generic joysticks
- software sine generator
- LifeTool IntegraMouse
- Generic mouses

1.2 Planned hardware support

- National Instruments DAQ cards
- Generic Keyboards
- NIRx NIRScout
- Neurosky MindSet
- Tunable EEG simulator
- Arduino embedded DAQ system

1.3 Planned extensions

- remote configuration
- client-based channel and sampling rate selection
- acquired data storage using .gdf files
- streaming of stored files

1.4 License

The TOBI signal server is licenced under the GPLv3 (http://www.gnu.org/licenses/gpl.html).

1.5 Contact

For further information please contact SignalServer@tobi-project.org (SignalServer@tobi-project.org).

2 Hardware Requirements

- CPU: at least 200 MHz

  (already tested on embedded systems, highly dependent on sampling rate and number of acquired channels)
- RAM: 32 MB
- Run the Signal Server with highest process priority

For networking usage:

- Ethernet min. 100 MBit (1 GBit recommended)

  Needed network connection bandwidth varying by the sampling rate and the number of channels acquired.

  Recomendation

  If the Signal Server is used via a network connection it is recommeded to use it behind a router. Otherwise traffic from other people could utilize a common network to its capacity and influence the transmission time of the Signal Servers data packets.

3 Installation Instruction

3.1 Hardware Drivers

To use data acquisition devices as g.tec's g.USBamp, the respective drivers, provided by the manufacturer have to be installed first.

Drivers provided by the manufacturers:

- g.USBamp (Windows only)
- BrainProducts Brainamp series (Windows only)

3.2 Debian/Ubuntu

Download (http://www.tobi-project.org/download) libtia.deb and signalserver.deb for your respective operating system and platform (32 or 64 bit). Install it with your preferred packet manager (e.g. synaptic). libtia.deb has to be installed first.

All required files are available at www.tobi-project.org/download (http://www.tobi-project.org/download).

3.3 Windows

Download (http://www.tobi-project.org/download) the Signal Server setup file and install it.

All required files are available at www.tobi-project.org/download (http://www.tobi-project.org/download).

4 Using the Signal Server

4.1 Generic Usage

The Signal Server is shipped with two demo configuration files (“server_config_comments.xml” and “server_config.xml”). Those files are showing exemplary configurations of the Signal Server. The “_commets.xml” file is equipped with additional comments to facilitate understanding the configuration of the Signal Server.

The Signal Server can be started by a quickstart where the file “server_config.xml” is automatically read (operating system dependent) or by passing an individual config file to the Signal Server.

Startup Commands:

Individual config files:

```
signalserver your_config_file.xml
```

or

```
signalserver -f your_config_file.xml
```

List possible hardware:

```
signalserver -l
```

Commands while the Signal Server is running:

q ... stop

r ... restart (problems with various DAQ devices – bugfix in progress)

l ... list possible hardware devices

h ... help

4.2 Debian/Ubuntu

With the first start of the Signal Server, a folder named “tobi_sigserver_cfg” is automatically created within the users home folder. Two files named “server_config_comments.xml” and “server_config.xml” are located inside this folder.

Starting the Signal Server without any parameters automatically reads and starts the “server_config.xml” inside the “tobi_sigserver_cfg” folder (quickstart).

4.3 Windows

Starting the Signal Server without any parameters (quickstart), the program automatically reads and starts the “server_config.xml” inside the folder where the Signal Server is installed (e.g. C:\Program Files\TOBI SignalServer\server_config.xml).

To use a different configuration, another .xml file can be handed to the Signal Server either using the commandline or just dragging the respective file on the Signal Server executable or a link to it.

5 The XML Configuration File – Basics

This is just a general description of the XML configuration file.

5.1 Subject Information

This tag has to be used to store meta information from the respective subject or patient participating in the measurement. (Self-explanatory tags are not described here, e.g. birthday)

Exactly one <subject> tag has to be inside the XML configuration file.

```
<subject>
  <id> asd52 </id>
  <first_name> Nobody </first_name>
  <surname> Nowhereman </surname>
  <sex> m </sex>
  <birthday> 31.12.1900 </birthday>
  <handedness> r </handedness>
  <medication> none </medication>

  <optional glasses="yes" smoker="no" />
</subject>
```

- ID

  The subjects identification code (e.g. ch17b)

- Optional

  The optional tag can be extended at will (e.g. technician = “tec1”)

5.2 Server Settings

Inside this tag the Signal Server except its attached hardware is configured.

Exactly one <server_settings> tag has to be inside the XML configuration file.

```
<server_settings>
  <ctl_port> 9000 </ctl_port>
  <udp_port> 9998 </udp_port>
  <udp_bc_addr> 127.0.0.255 </udp_bc_addr>
</server_settings>
```

- ctl_port

  A TCP port every client is connected to. This port has to be specified at the client (together with the IP address the server is running on; if the same machine is used, the IP address is 127.0.0.1) when connecting to the Signal Server.

UDP Data tranmission

The Signal Server supports data tranmission via UDP (reduced packet overhead, no guarantee that all data is tranmsitted to the client). If a client requests UDP transmission UDP packets are broadcasted into the whole subnet specified inside this tag.

Sending UDP packets to an address in an other subnet or two different ports is not supported yet.

- udp_port

  The target port.

- udp_bc_addr

  The broadcast address to transmit UDP packets to (e.g. 192.168.1. **255** – packets are sent to every computer with an IP 192.168.1.XXX).

5.3 The Hardware sections

The Signal Server supports data acquision from multiple devices at the same time. For this reason every device has its own hardware tag. Thus more than one hardware section is allowed inside the XML configuration file.

```
<hardware name="sinegenerator" version="1.0" serial="">
  <mode> master </mode>
  <device_settings>
    <sampling_rate> 512 </sampling_rate>
    <measurement_channels nr="1" names="eeg" type="eeg" />
    <blocksize> 8 </blocksize>
  </device_settings>

 <channel_settings>
    <selection>
      <ch nr="01" name="C3" type="eeg" />
      <ch nr="02" name="Cz" type="eeg" />
      <ch nr="03" name="Hand" type="emg" />
    </selection>
  </channel_settings>
</hardware>
```

Hardware

```
<hardware name="sinegenerator" version="1.0" serial="">
```

- Name

  This attribute defines the respective hardware device to acquire data from.

- Version

  Not used yet. (May be removed in the future.)

- Serial

  Serial number of the device if available. Processed at particular devices (e.g. g.USBamp).

Mode

Possible values are:

- Master
- Slave
- Aperiodic

  e.g. buttons; data is only delivered if a value is altered

Inside the XML configuration file **exactly one device** has to be defined as master, all others have to be slave or aperiodic devices. The master device **must** have the highest “virtual” sampling rate compared to possible slave devices.

Virtual sampling rate = sampling rate / blocksize (e.g. fs = 512 Hz, bs = 8 ... v_fs = 64 Hz)

Data is acquired from the master in blocking mode. Every time data is available at the master, the latest data is acquired from all slaves and aperiodic devices. There is **no software synchronization** of the acquired data!

Not all devices support master, slave, and aperiodic mode.

Device Settings

In this tag settings affecting the whole data acquisition device and not only particular channels are specified.

The simplest data acquisition device has at least a sampling rate, a blocksize and a certain number of channels.

- Sampling Rate

  The sampling rate data is acquired with.

- Blocksize

  The number of samples grouped together before transmission.

- Measurement Channels

  (This setting can be used for quick configuration; all channels get the same name and the same signal type. For individual channel names and signal types use the channel setting section.)

  – nr

    The number of channels to acquire, starting at channel 1.

  – names

    The name for **all** channels.

  – type

    The signal type for **all** channels.

Samples are grouped into blocks from the same channel if a blocksize >1 is used (e.g. block size = 2: ch1s1 ch1s2; ch2s1 ch2s2; ...) and transmitted inside the same data packet. Altering the blocksize does not affect the sampling rate itself, but the rate data packets are transmitted over the network.

Using a sampling rate of 1000 Hz and a blocksize of 10, samples are still acquired with 1000 Hz, but data packets are sent with only 100 Hz, whereby every packet stores 10 samples for all acquired channels.

As incoming packets are used for timing control at the client, a bigger blocksize introduces a certain jitter, as the client has to wait for a new packet storing more than one sample and processes all samples immediately afterwards.

Some hardware devices have to be used with a blocksize >1 to avoid data acquisition errors!

Channel Settings

The channel settings tag can be used to customize individual channels and override settings done in the device settings section.

- Selection

  Select only specific channels for recording. The sum of all channels here can be different from the settings done in measurement_channels in device_settings. Settings here override the prior channel selection!

  ```
  <ch nr="01" name="Cz" type="eeg" />
  <ch nr="02" name="C3" type="eeg" />
  ```

  – nr

    Number of a channel to acquire data from.

  – name

    The name for the respective channel (e.g. Cz).

  – type

    The signal type for the respective channel (e.g. eeg).

6 Specific Hardware Section Configuration with the XML Config File

Before using some hardware with the Signal Server, please read the manufacturers user manual!

6.1 Sine Generator

The Sine Generator is a hardware emulator creating a 1 Hz sine with an amplitude of 1. The phase is increased every 4 channels.

Only the number of channels, signal types, signal names, the sampling rate and the blocksize are customizeable.

```
<hardware name="sinegenerator" version="1.0" serial="">
  <mode> master </mode>
  <device_settings>
    <sampling_rate> 512 </sampling_rate>
    <measurement_channels nr="1" names="eeg" type="eeg" />
    <blocksize> 8 </blocksize>
  </device_settings>

 <channel_settings>
    <selection>
      <ch nr="01" name="C3" type="eeg" />
      <ch nr="02" name="Cz" type="eeg" />
      <ch nr="03" name="Hand" type="emg" />
    </selection>
  </channel_settings>
</hardware>
```

6.2 EEG Simulator

The EEG Simulator is a hardware emulator generating normal distributed random noise with variable amplitude and offset. It is furthermore possible to add a sine to the simulated EEG signal (e.g. to test a classifier).

An external control program to modify the EEG simulators parameters is currently in development (a network protocol will be used for this purpose and the “port” tag defines the used port; it is not evaluated yet).

```
<hardware name="eegsim" version="1.0" serial="">
  <mode> master </mode>
  <device_settings>
    <sampling_rate> 500 </sampling_rate>
    <measurement_channels nr="48" names="eeg_name" type="eeg" />
    <blocksize> 1 </blocksize>
    <port> 9123 </port>

    <eeg_config  scaling="1" offset="1" />
    <sine_config frequ="2" amplitude="1"  phase="0" />
  </device_settings>

   <channel_settings>
    <selection>
      <ch nr="01" name="eeg" type="eeg" />
      <ch nr="02" name="eeg" type="eeg" />
      <ch nr="03" name="eeg" type="eeg" />
    </selection>

    <eeg_config>
      <ch nr="01" scaling="1" offset="-500" />
      <ch nr="02" scaling="1" offset="0" />
      <ch nr="03" scaling="1" offset="0" />
    </eeg_config>

    <sine_config>-->
      <!--WARNING: device settings are not overwritten here!
          sines are added to the signal; multiple entries per channel possible-->
      <ch nr="01" frequ="2" amplitude="100"  phase="0" />
      <ch nr="01" frequ="4" amplitude="1000"  phase="0" />
      <ch nr="02" frequ="3" amplitude="100"  phase="0" />
      <ch nr="03" frequ="6" amplitude="100"  phase="0" />
    </sine_config>

  </channel_settings>
</hardware>
```

6.3 g.USBamp

The g.USBamp is a mulipurpose biosignal DAQ device produced by g.tec (Guger Technologies, Graz, Austria). Up to now only the Windows API is included into the Signal Server. The Linux API is planned to be implemented soon.

Important

Using a too small blocksize for data acquisition might result in a loss of data. The following values are recomendations from g.tec:

Sampling Rate [Hz]	Block Size
32	1
64	2
128	4
256	8
512	16
600	32
1200	64
2400	128
4800	256
9600	512
19200	512
38400	512

According to g.tec, the minimum buffersize can also be determined by following equation:

block_size >= sampling_rate * 0.06

Data acquisition has already been succesfully performed with a sampling rate of 512 Hz and a blocksize of 4.

Notice: You can use blocksize smaller than the recommeded values on your own risk!

(If doing so, it is recommeded to perform extensive tests before.)

Notice: g.USBamp running as slave without external sync is not supported yet! (4.2.2011)

Possible messages, warnings and known issues:

- Received not enough data:

  Starting the signal server with a connected g.UABamp, sometimes this message occurs for the first sample(s). This could happen, because the amplifier returns a not completely filled buffer, especially at startup. If this message occurs during runtime, often the signal servers process priority is too low or the used blocksize is too small. Missing values are filled up with "0".

- Timeout:

If timeouts occur, please restart the g.USBamp and check the sync-cables, if multiple amps are used. (Due to a already fixed bug, this happened frequently if the signal server was stopped and started again. Starting and stopping the signal server a second time solved the problem.)

- Unable to set filter settings:

  Setting wrong filter settings or also a wrong sampling rate (e.g. 500 Hz) produces errors setting the hardware filter. Please re-check your settings if the desired filter is really supported.

Description of g.USBamp specific configuration tags:

The g.USBamp provides adjustable online filtering and other features. The following sections explains which configuration settings can be done.

```
<hardware name="g.usbamp" version="" serial="UA-2008.06.42">
  <mode> master </mode>
  <device_settings>
    <sampling_rate> 512 </sampling_rate>
    <measurement_channels nr="1" names="eeg" type="eeg" />
    <blocksize> 8 </blocksize>

    <filter type="chebyshev" order="8" f_low="0.5" f_high="100"/>
    <notch f_center="50"/>

    <shortcut> off </shortcut>

    <usbamp_master> yes </usbamp_master>

    <common_ground>
      <gnd block="a" value="1" />
      <gnd block="b" value="1" />
      <gnd block="c" value="1" />
      <gnd block="d" value="1" />
    </common_ground>

    <common_reference>
      <cr block="a" value="1" />
      <cr block="b" value="1" />
      <cr block="c" value="1" />
      <cr block="d" value="1" />
    </common_reference>
  </device_settings>

  <channel_settings>
    <selection>
      <ch nr="01" name="C3" type="eeg" />
      <ch nr="05" name="Cz" type="eeg" />
      <ch nr="06" name="ecg" type="ecg" />
      <ch nr="16" name="eyes" type="eog" />
    </selection>

    <filter>
      <ch nr="06" type="chebyshev" order="8" f_low="0.5" f_high="30"/>
      <ch nr="16" type="chebyshev" order="8" f_low="0.5" f_high="60"/>
    </filter>

    <notch>
      <ch nr="05" f_center="50"/>
      <ch nr="16" f_center="60"/>
    </notch>

    <bipolar>
      <ch nr="1" with="05" />
    </bipolar>

  </channel_settings>
</hardware>
```

Every g.USBamp is equipped with a unique serial number. The respective device used for acquisition is specified via its serial in the “serial” tag (here: UA-2008.06.42).

```
<hardware name="g.usbamp" version="" serial="UA-2008.06.42">
```

Device Settings

The g.USBamp has the possibilty to use different built in filters for pre-signal processing. Possible filter settings are listed in a supplementary file called “g.USBamp_filter_settings.txt” or can also be listed with the program “list_usbamp_filter_settings.exe”. (g.USBamp driver version 3.10.0 – only chebyshev filters are suported yet by the amplifier)

```
<filter type="chebyshev" order="8" f_low="0.5" f_high="100"/>
```

The g.USBamp has the possibilty to use a hardware notch filter at 50 or 60 Hz.

```
<notch f_center="50"/>
```

A TTL high impulse on the SC input socket can be used to disconnect all electrode input sockets from the input amplifiers and to connect the inputs to ground potential. (copied from the g.USBamp manual)

Turning this setting on or off enables or disables the SC socket to react on incomming TTL signals.

```
<shortcut> off </shortcut>
```

It is possible to use multiple g.USBamps at the same time. Those ampliefiers have to be connected via an external sync cable (see g.USBamp manual). One of all linked amplifiers has to provide the sync signal and act as the master device for the other USBamps (this master has nothing to do with the master defined in every hardware tag). For the amplifiers getting their sync signal via the “SYNC IN” socket, the <usbamp_master> tag has to be set to “no”.

```
<usbamp_master> yes </usbamp_master>
```

The g.USBamp has 4 channel groups and every group has its own ground connector. It is possible to connect those grounds to a common ground via this tag by setting “value” to 1 or 0.

```
<common_ground>
  <gnd block="a" value="1" />
  <gnd block="b" value="1" />
  <gnd block="c" value="1" />
  <gnd block="d" value="1" />
</common_ground>
```

The g.USBamp has 4 channel groups and every group has its own reference connector. It is possible to connect those references to a common reference via this tag by setting “value” to 1 or 0.

```
<common_reference>
  <cr block="a" value="1" />
  <cr block="b" value="1" />
  <cr block="c" value="1" />
  <cr block="d" value="1" />
</common_reference>
```

Channel Settings

The g.USBamp provides the possibility to set a filter for every channel individually. This setting only overrides the global setting for the respective channel. All others remain with the global configuration.

```
<filter>
  <ch nr="06" type="chebyshev" order="8" f_low="0.5" f_high="30"/>
</filter>
```

The g.USBamp provides the possibility to set a notch filter for every channel individually. This setting only overrides the global setting for the respective channel. All others remain with the global configuration.

```
<notch>
  <ch nr="05" f_center="50"/>
</notch>
```

The g.USBamp provides the possibility to acquire a bipolar channel combination. The channel given with the attribute “with” is subtracted from the channel given by “nr”. The Signal Server still delivers both channels. An option to supress the channel defined by “with” is planned.

```
<bipolar>
  <ch nr="1" with="05" />
</bipolar>
```

6.4 g.Mobilab

The g.Mobilab and the g.Mobilab+ are portable mulipurpose biosignal DAQ systems produced by g.tec (Guger Technologies, Graz, Austria). Both are available in two configurations, either able to acquire just EEG or also other biosgnals as EOG and ECG. The g.Mobilab+ is can be connected via Bluetooth emulating a serial port.

Up to now only the g.Mobilab has been tested, but the g.Mobilab+ should work as well as the API is the same.

```
<hardware name="g.mobilab" version="1.0" serial="">
   <mode> master </mode>
   <device_settings>
     <serial_port> /dev/ttyS0 </serial_port>
     <mobilab_type> eeg </mobilab_type>
     <measurement_channels nr="1" names="eeg" type="eeg" />
     <blocksize> 1 </blocksize>
   </device_settings>

   <channel_settings>
     <selection>
       <ch nr="01" name="eeg" type="eeg" />
       <ch nr="02" name="eeg" type="eeg" />
     </selection>
   </channel_settings>
 </hardware>
```

6.5 g.BSBamp

The g.BSBamp is a mulipurpose biosignal DAQ device produced by g.tec (Guger Technologies, Graz, Austria). Different variants are available. Device suitable for EEG, ECG, and EOG data acquisition or devices able to acquire just one of those signal types. The g.BSamp is acquired using National Instruments (Austin, TX, USA) DAQ cards.

```
<hardware name="g.bsamp" version="2004" serial="BS-2004.08.02">
 <mode> master </mode>
 <device_settings>
   <sampling_rate> 512 </sampling_rate>
   <measurement_channels nr="1" names="eeg" type="eeg" />
   <blocksize> 1 </blocksize>
   <filter type="eeg" f_high="100" notch="on" f_low="2" sense="0.1" />
   <filter type="eog" f_high="100" notch="on" f_low="2" sense="1" />
   <filter type="emg" f_high="100" notch="on" f_low="2" sense="5" />
   <filter type="ecg" f_high="100" notch="on" f_low="2" sense="5" />
   <notch f_center="50"/>
 </device_settings>

 <channel_settings>
   <selection>
     <ch nr="01" name="eeg" type="eeg" />
     <ch nr="02" name="eeg" type="eeg" />
   </selection>

   <filter>
     <ch nr="2"  type="eeg" f_high="30"  notch="off" f_low="0.01" sense="0.05"/>
     <ch nr="15" type="ecg" f_high="100" notch="off" f_low="0.01" sense="5"/>
   </filter>
 </channel_settings>
</hardware>
```

6.6 BrainAmp Series

BrainAmps are EEG acquisiton system produced by Brain Products (Gilching, Germany). All amplifiers from the BrainAmp Series are supported.

```
<hardware name="brainamp" version="1.0" serial="">
  <mode> master </mode>
  <device_settings>
    <sampling_rate> 500 </sampling_rate>
    <measurement_channels nr="2" names="eeg" type="eeg" />
    <blocksize> 5 </blocksize>

    <use_low_impedance>  no </use_low_impedance>
    <trigger_hold_value> 0 </trigger_hold_value>

    <lowpass_250Hz> off </lowpass_250Hz>
    <dc_coupling> off </dc_coupling>
    <resolution>  100nV </resolution>

    <calibration_mode on="no" signal="sine" freq="10" />
  </device_settings>

  <channel_settings>
    <selection>
      <ch nr="01" name="eeg" type="eeg" />
      <ch nr="02" name="eog" type="eog" />
      <ch nr="03" name="emg" type="emg" />
      <ch nr="04" name="emg"  type="emg" />
    </selection>

    <lowpass_250Hz>
      <ch nr="1" value="on"/>
      <ch nr="16" value="off"/>
    </lowpass_250Hz>
    <dc_coupling>
      <ch nr="1" value="on"/>
      <ch nr="16" value="off"/>
    </dc_coupling>

    <resolution>
      <ch nr="1" value="100nV"/>
      <ch nr="16" value="152muV"/>
    </resolution>

  </channel_settings>
</hardware>
```

6.7 Generic Joysticks

It is possible to acquire data from attached joysticks with the Signal Server using the SDL library (simple direct media layer) in Windows and Linux. Up to now only aperiodic mode is supported. It is not possible to do any configuration for the joystick. The number of bottons, axes, and balls is automatically determined by the Signal Server.

```
<hardware name="joystick" version="1.0" serial="">
  <mode> aperiodic </mode>
  <device_settings>

  </device_settings>
</hardware>
```

6.8 IntegraMouse + Generic Mouses

It is possible to acquire data from attached mouses with the Signal Server under Windows and Linux using libusb. The mouse is detached from the operating system, thus the respective mouse does not control the mouse cursor. The data is directly fed into the Signal Server.

The IntegraMouse (LifeTool, Linz, Austria) is also supported via this configuration.

Only aperiodic mode is supported yet.

Linux: To configure the mouse device, the VendorID and the ProductID of the device which define it uniquly are needed. On Linux one can find them by the command `lsusb -v`. Both IDs are written in the first line of the device-block. Further the right usb-port needs to be stated. One can find it also there, it is named `bEndpointAddress` and is listed in the `Interface Descriptor` of the device-block. All numbers must be added decimally.

Windows: When using a mouse device on a Windows system, there is an additional tool needed. Therefore the `devcon`-tool included in the Windows Driver Kit (WinDDK (http://msdn.microsoft.com/en-us/windows/hardware/gg487428)) has to be installed first. Further `libusb-win32` (libusb-win32 (http://sourceforge.net/apps/trac/libusb-win32)) must be installed on the system. It is used to decouple the used mouse device from the opterating system. The full path to this tool must be stated. When configuring the TOBI SignalServer for a mouse device, one need to generate a `mouse.inf` file in the directory `bin/libusb`. Therefore the `INF Wizard` (included in libusb-win32) can be used. The right configuration for VendorID, ProductID and Usb-Port (`bEndpointAddress` in the `Interface Descriptor` of the device-block) can be found using the `Test (Win) Program` (also included in libusb-win32). All numbers must be added decimally.

The following example of a configuration includes the data for a Mouse. The usb-port might vary, depending on the used port.

Important

Entered values have to be in **decimal** format! Therefore a hexadecimal representation given by `lsusb -v` has to be converted to its decimal equivalent.

```
<hardware name="mouse" version="1.0" serial="">
  <mode> aperiodic </mode>
  <device_settings>
    <vendorid> 1351 </vendorid>
    <productid> 4136 </productid>
    <usb_port> 130 </usb_port>

    <devcon_path> C:\WinDDK\7600.16385.1\tools\devcon\i386\devcon.exe </devcon_path>

  </device_settings>
</hardware>
```