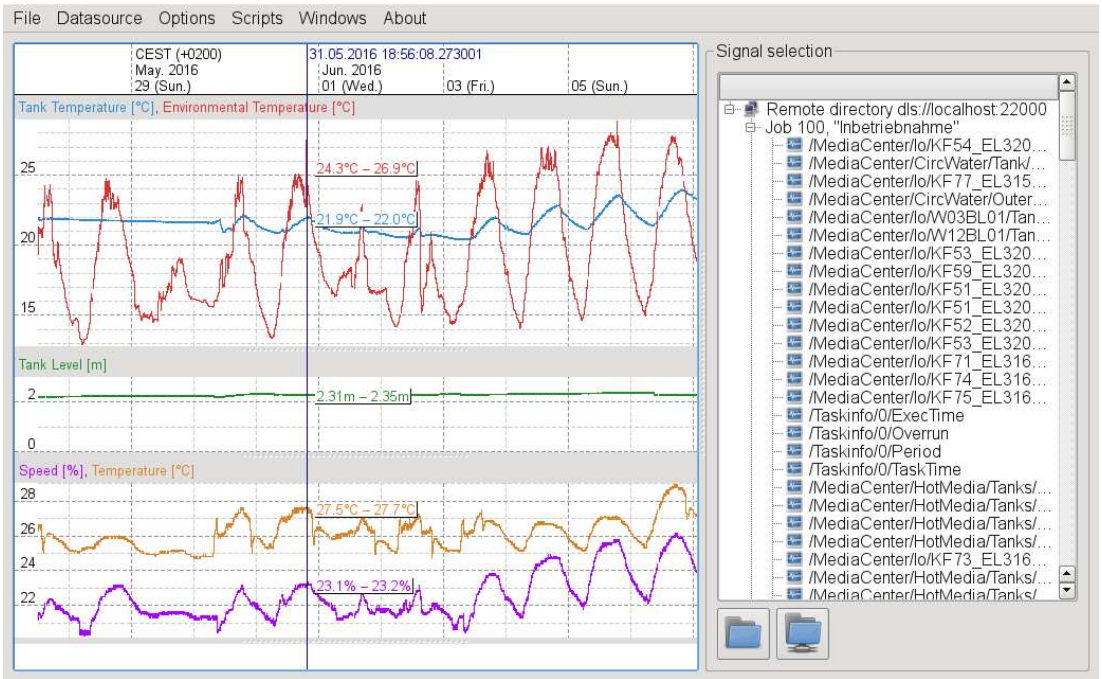
# Tutorial
## for EtherLab
# TestManager & DLS

## CTRL.00.00.009.REP

**Sébastien BLANCHET**

August 06, 2021




Institut Radio Astronomie Millimétrique
300 rue de la piscine
38406 Saint Martin d'Hères - France

# Contents

# Chapter 1

# Introduction

## 1.1 Abstract

EtherLab[1] is a set of open-source software developed by IgH to build real-time control applications. This tutorial explains how to install two EtherLab components: TestManager (realtime visualization software) and DLS (Data Logging Service). It provides also an easy overview before reading the official documentation.

## 1.2 Architecture overview

An overview of the architecture can be seen in Figure 1.1.

- There are 3 computers: the controller, the viewer and the recorder.
- On the controller, the control application exports its channels in a shared memory (SHM), then an other process, running the PdServer library publishes the channels on the network through the MSR protocol (TCP port 2345 by default). When the channels are exported on the network any client that speaks the MSR protocol can be used to display or record the channels.
- On the recorder, the program `dlsd` (Data Logging Service Daemon) records the selected channels on the filesystem (FS). The program `dls_ctl` manages the configuration files for `dlsd` through the filesystem. For example, it can configure the list of channels to record. `dls_view` is a simple viewer that can read directly the data files written by `dlsd`.

[1] `http://etherlab.org/en/components.php`

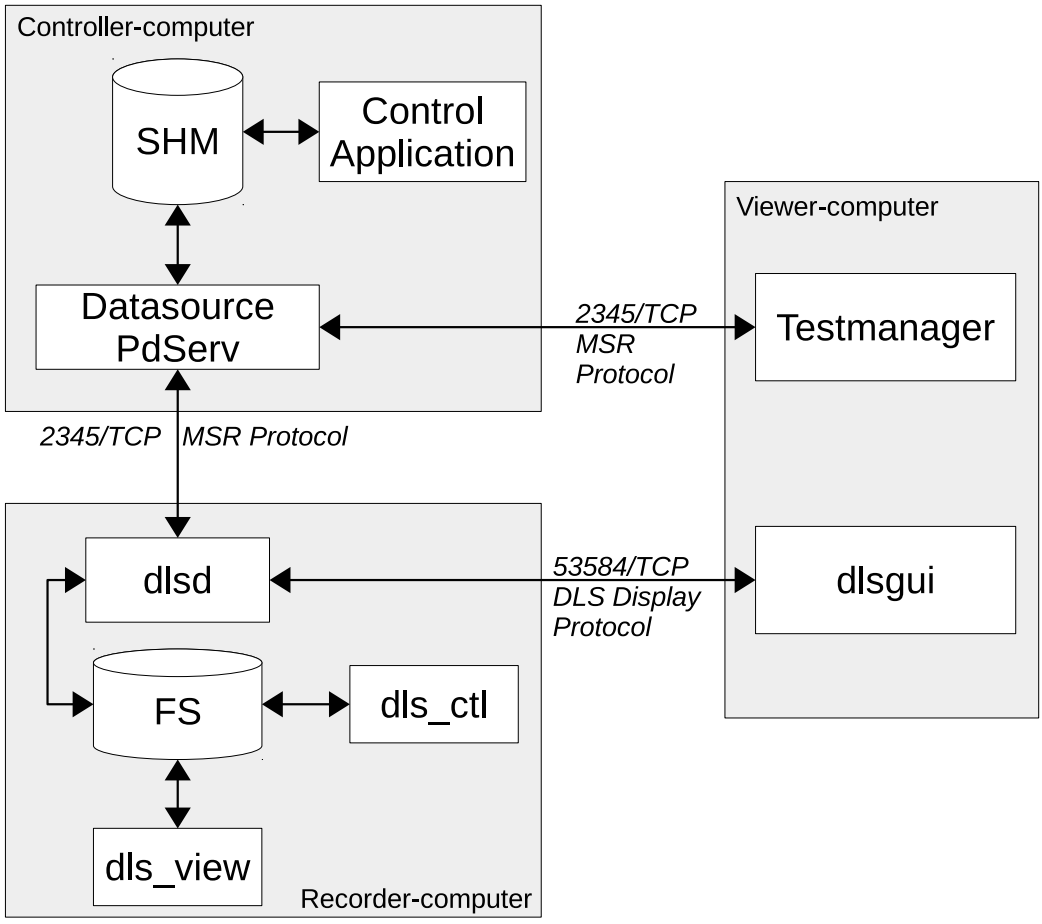



Figure 1.1: Architecture overview

- On the viewer, the program `testmanager` connects directly to the PdServer to display the channels in real-time. On the other hand `dlsgui` is an advanced data viewer for DLS that connects `dlsd` through a specific network protocol (TCP port 5384 by default) to plot recorded data.

# Chapter 2

# Prepare the virtual machine

This chapter explains how to prepare a single virtual machine running Ubuntu 20.04 LTS amd64 to run TestManager, DLS and a test server. It means that a single virtual machine will replace the 3 computers described in Figure 1.1, because it makes an easier tutorial.

## 2.1 Creation of the virtual machine

Create a virtual machine with VirtualBox, with the the following settings:

- vCPU: 1
- RAM: 4 GB
- vDisk: 20 GB
- Guest OS: Ubuntu 64 bits

Download `ubuntu-20.04-desktop-amd64.iso` from `https://www.ubuntu.com` and install it in your new virtual machine.

Select a minimum installation with only a web browser and basic utilities to save disk space.

**Notes**

- For a very first installation, it is recommended to use exactly the same distribution than this guide to avoid any distribution issues.
- For simplicity, all the components will be installed on the same host, but obviously the server and client can run on different machines.

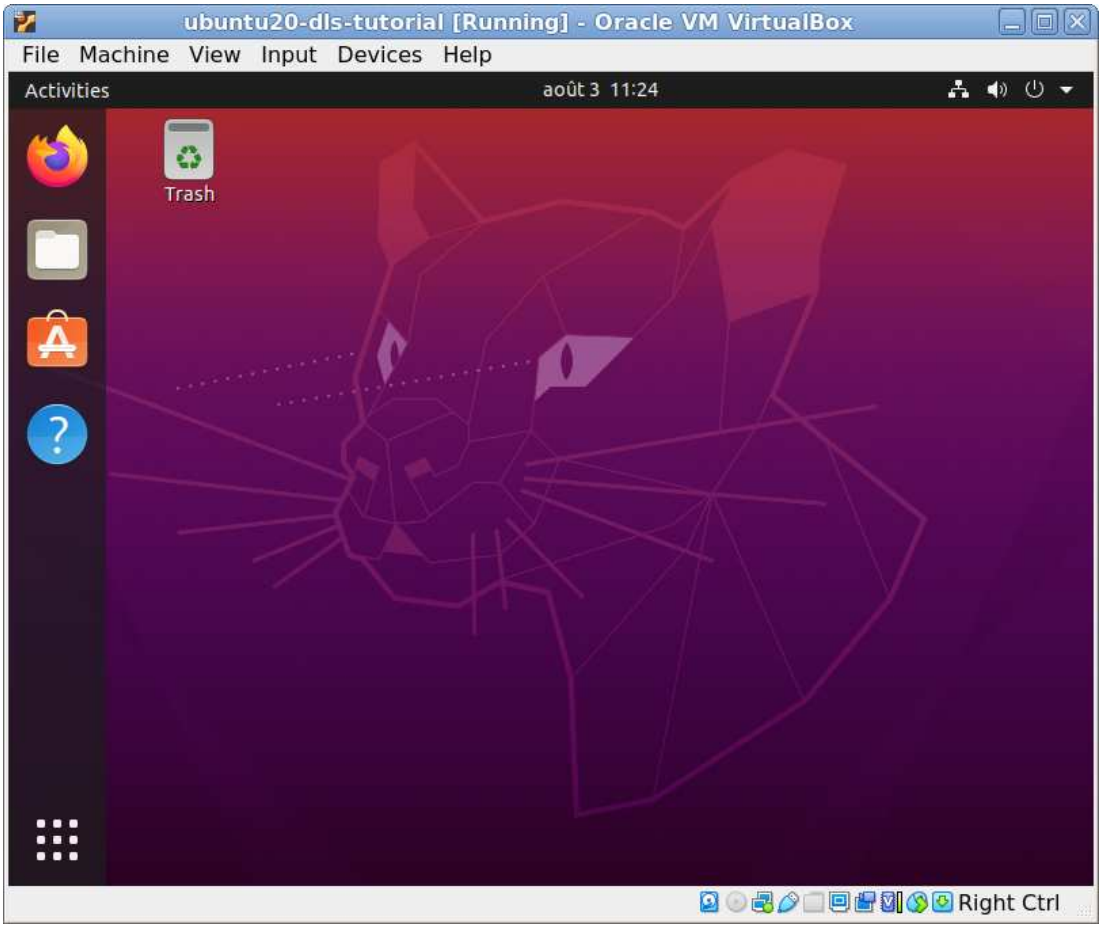



Figure 2.1: Ubuntu 20.04 LTS amd64 in VirtualBox

# Chapter 3

# PdServ

PdServ is the library that implements the MSR protocol on the server side.

## 3.1 Install prerequesites

The following prerequesites must be installed to compile PdServ and its test server.
Open a terminal and type (the ending backslash means that the command continues on the next line).

```
sudo apt install git cmake g++ pkg-config     \
                 libdb-dev libsasl2-dev libyaml-dev \
                 liblog4cplus-dev libcommoncpp2-dev
```

## 3.2 Get PdServ source code

Download the source code with mercurial[1], then switch to the revision that has been tested in this tutorial (rev 528).

```
cd /tmp
git clone https://gitlab.com/etherlab.org/pdserv.git
cd pdserv
git checkout https://gitlab.com/etherlab.org/pdserv.git
```

## 3.3 Compile PdServ source code

PdServ needs `cmake` to create the Makefile.
Do not forget the dot just after `cmake`

[1]The Mercurial main command tool is named `hg` because the chemical symbol for mercury is *Hg*. Mercurial is a source control management tool very similar to `git`.

```
cd /tmp/pdserv
cmake . -DCMAKE_INSTALL_PREFIX=/opt/etherlab
make
```

## 3.4 Install PdServ library

The library will be installed in `/opt/etherlab`
Become root with `sudo -i` then run the installation commands

```
sudo -i
cd /tmp/pdserv
make install
echo /opt/etherlab/lib > /etc/ld.so.conf.d/etherlab.conf
/sbin/ldconfig
ln -sf /opt/etherlab/lib/pkgconfig/libpdserv.pc /usr/share/pkgconfig/
```

## 3.5 Compile a test server

PdServ comes with a test server, but it needs to be patched to have more visible signals in the viewer.
Edit the file `/tmp/pdserv-code/test/test1.cpp` with `gedit`,

```
gedit /tmp/pdserv/test/test1.cpp
```

Go to line 165 to add at the beginning of the while loop.

```
        s1.c = (s1.c + 1) % 20;
```

Check the modification with mercurial

```
cd /tmp/pdserv
git diff

diff --git a/test/test1.cpp b/test/test1.cpp
index db5cb2d..20c2f68 100644
--- a/test/test1.cpp
+++ b/test/test1.cpp
@@ -170,6 +170,7 @@ int main(int argc, const char *argv[])

     int i;
     while (1) {
+        s1.c = (s1.c + 1) % 20;
         //sleep(1);
         usleep(100000);
         clock_gettime(CLOCK_REALTIME, &time);
```

Compile the test server

```
g++ -o test1 test1.cpp -I/opt/etherlab/include -L/opt/etherlab/lib -lpdserv
```

Run the program

```
./test1
```

Keep this program running until the tutorial end to have a test server for TestManager and DLS.

# Chapter 4

# TestManager

TestManager is a graphical application to visualize channels exported by PdServer. The official documentation is available in [IGH10].

## 4.1 Install prerequesites

### Run a windows application on Linux

TestManager is a Windows application, that needs `wine` to run on Linux. Wine exists in two flavors: `wine32` for 32-bit applications and `wine64` for 64-bit ones. As a 32-bit application, Testmanager requires **wine32** to run on Linux. It does not work with wine64!

### Install wine32

`wine32` is available only for the `i386` architecture which is disabled by default on Ubuntu 64-bit. Therefore the `i386` architecture must be enabled to install `wine32`.

```
sudo -i
dpkg --add-architecture i386
apt update
apt install wine32
```

## 4.2 Install TestManager

TestManager is a single-executable application, that it very easy to setup. Download the file:

```
wget http://etherlab.org/download/testmanager/msr_test_manager_3_6_4.exe
```

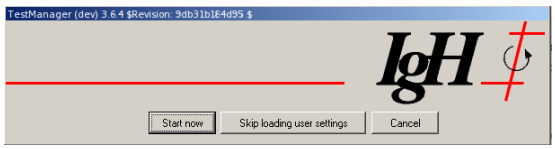
The first starting may be very long, because `wine` is initialising a full windows environment. Fortunately, the next starting will be faster.

## 4.3 Configure TestManager

- Run TestManager with `wine`

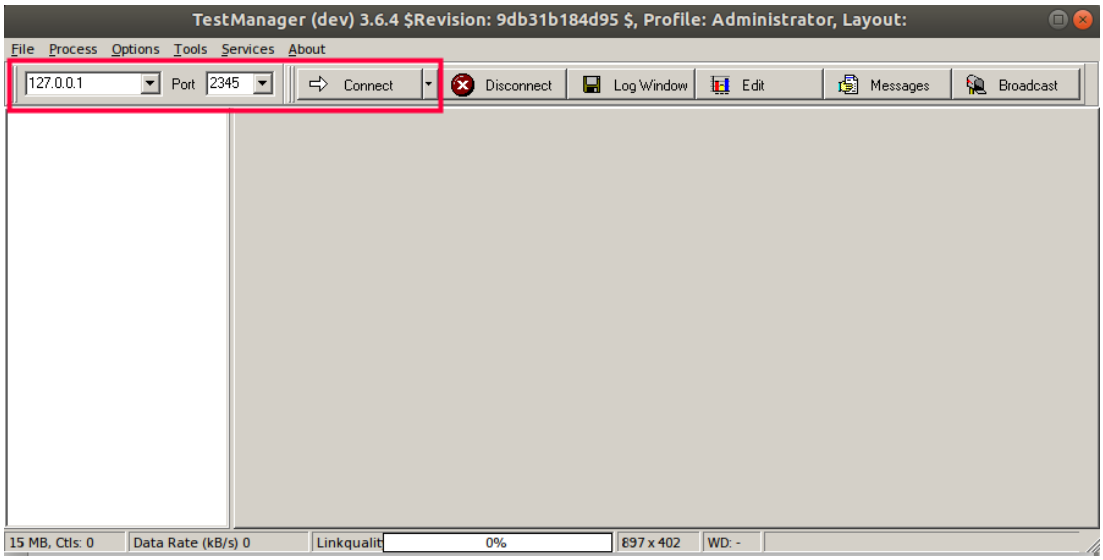
  ```
  wine msr_test_manager_3_6_4.exe
  ```

- The splash screen appears

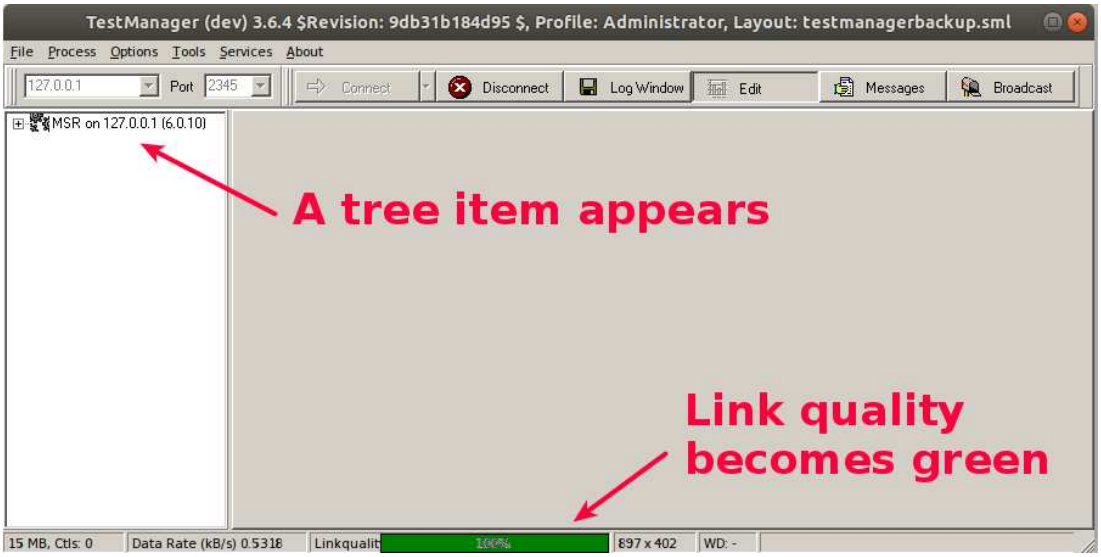



  It can be closed by clicking on the *Start Now* button, or by waiting few seconds until the red progression bar reaches IgH logo at the right side.

- The main windows appears. Fill the host (127.0.0.1) and port (2345) input boxes, then click *Connect*.

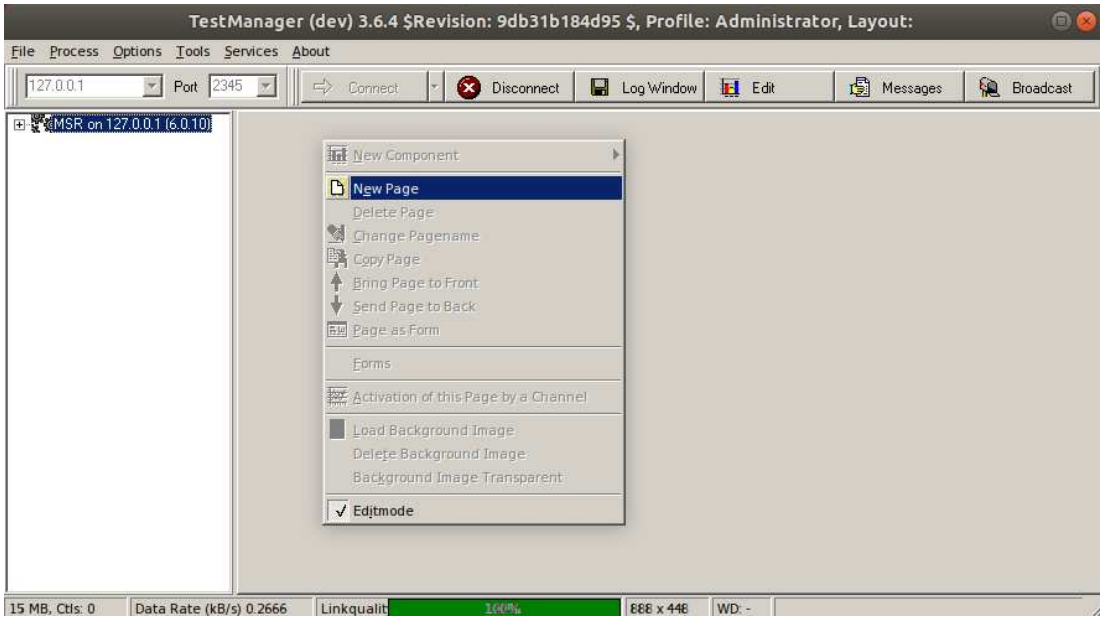



- If the connection is successful, a tree item appears on the left side, and the link quality bar becomes green




If nothing appears, check that the test server[1] is still running and that the host and port are correctly typed in the input boxes.

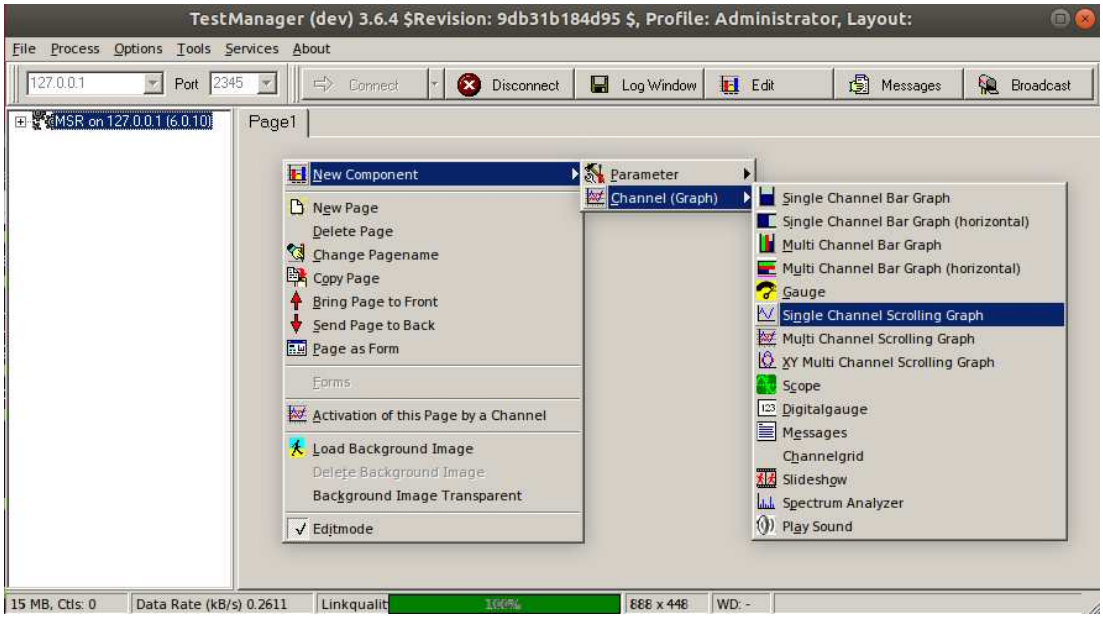
- Create a widget to display the signal. Right-click on the right panel, and select *New Page*.



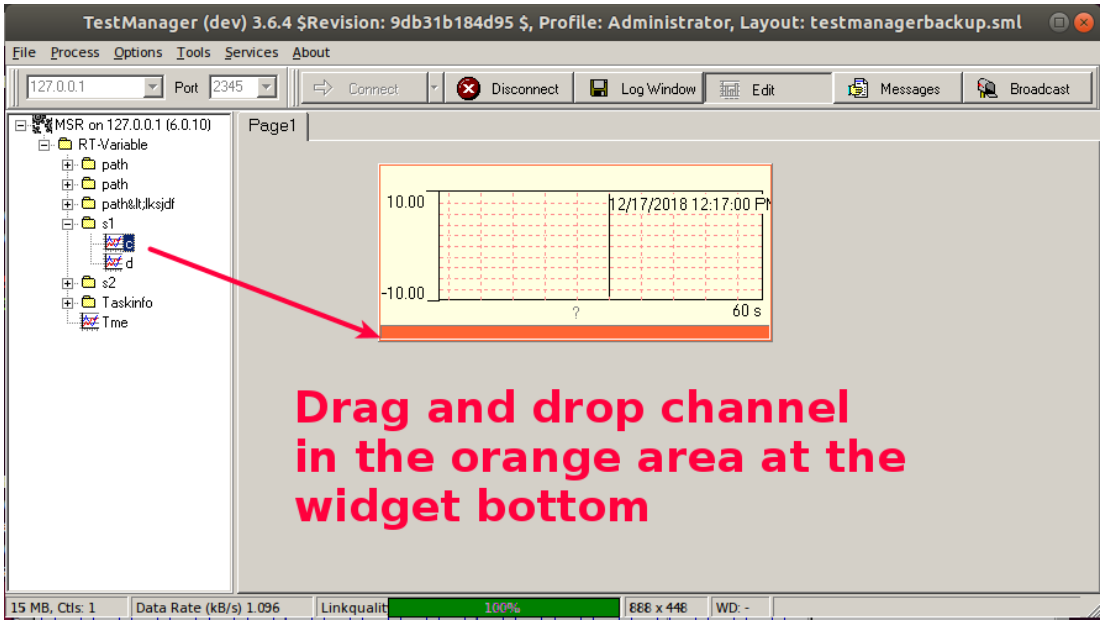

- Enter a page name, for example *Page1*

[1] see section 3.5

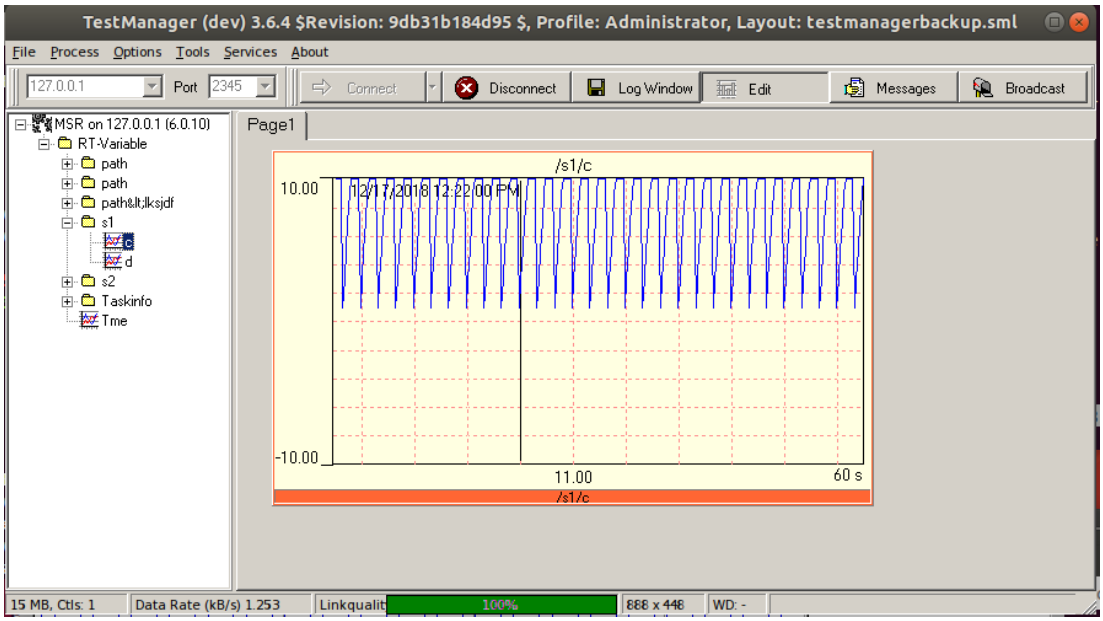

- Right-click on *Page1*, and select *New Component | Channel (Graph) | Single Channell Scrolling Graph*



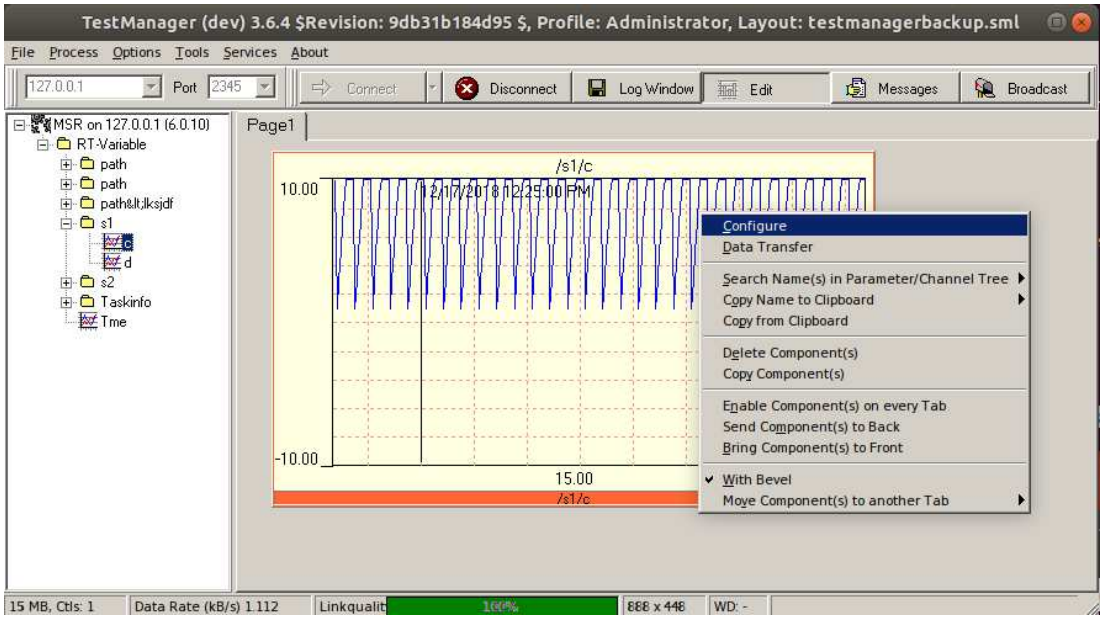

- Expand the tree in the left panel to find the *RT-Variable/s1/c.* channel. Drag-and-drop it to the orange area at the bottom of the channel widget.




- The channel is plotted in real-time. Drag the corner of the widget to increase the size to see better the channel.

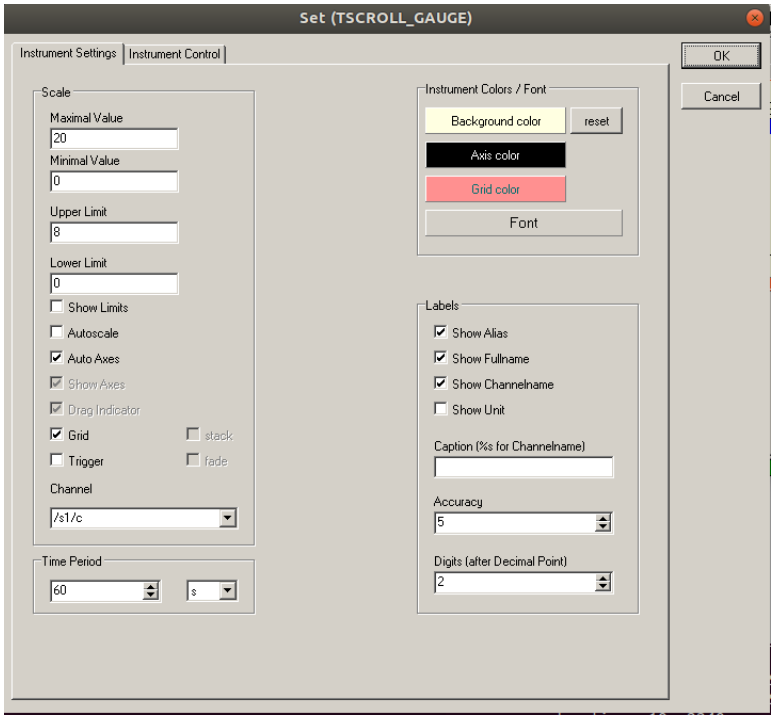



- To edit the widget properties, right-click on it, then select *Configure*

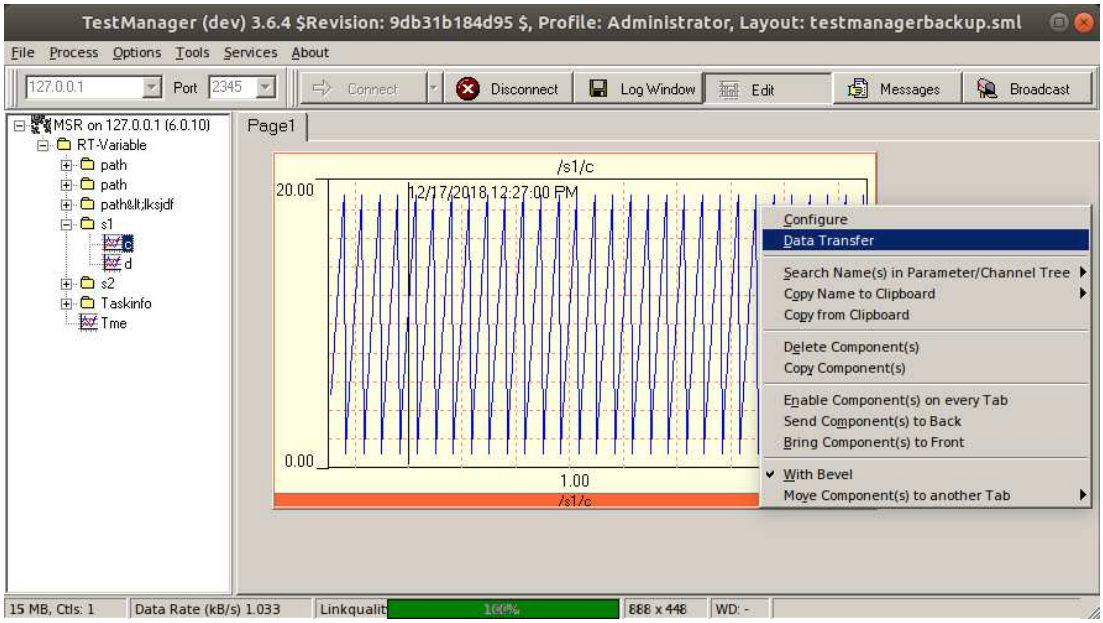



The property window appears. Change the following values:

- Maximal Value = 20
- Minimum Value = 0

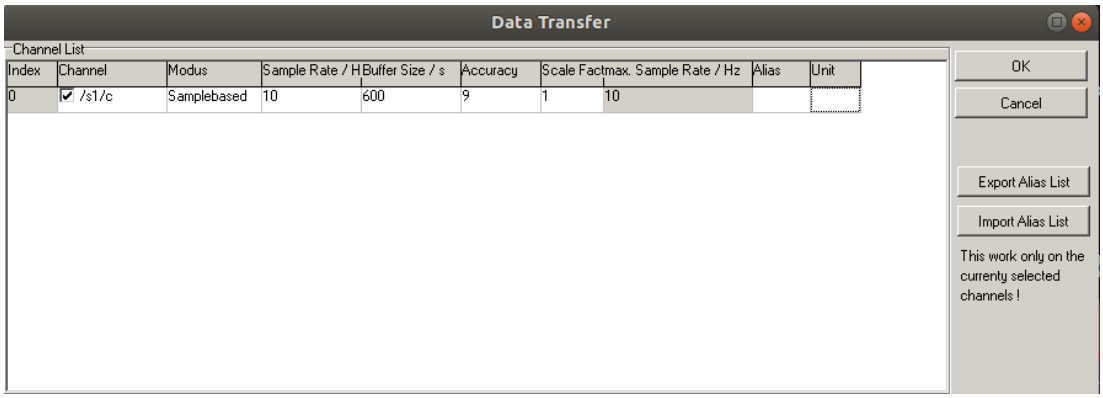


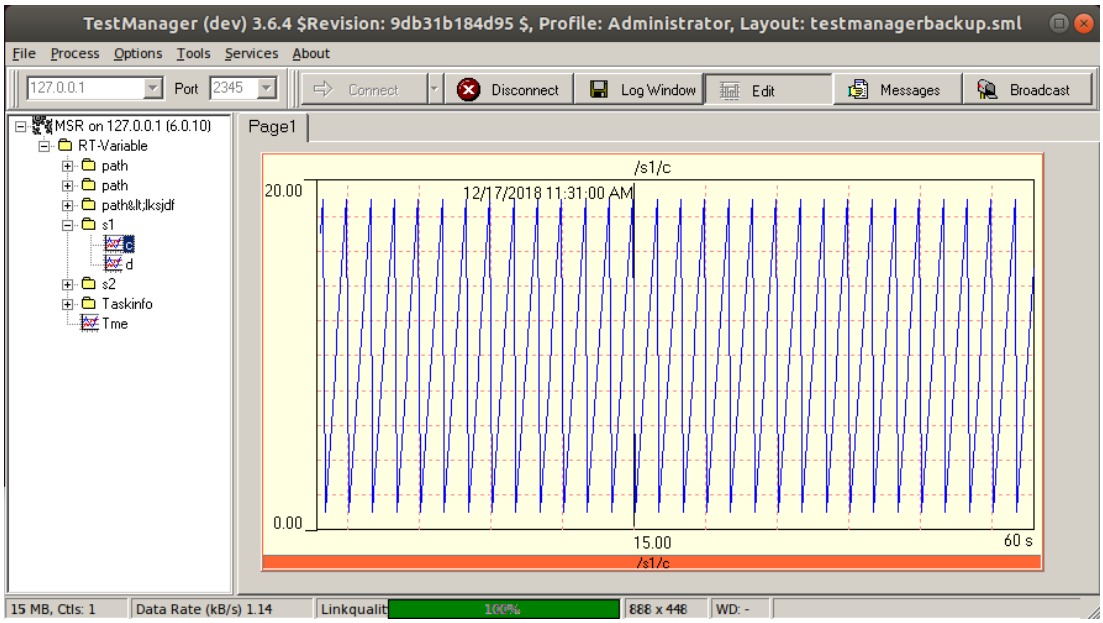
- The default sample rate is low (2 Hz), to increase it, right-click on the widget, then select *Data Transfer*




The Data Transfer window appears. To have a smoother curve, increase both *Sample Rate* and *Buffer Size/s*.

- Sample Rate = 10
- Buffer Size /s = 600






## 4.4 Go further

To go further with Testmanager, you need to read the official documentation which is available in [IGH10].

# Chapter 5

# DLS

Data Logging Service (DLS) is a data logging system for EtherLab. It connects to PdServ and stores the time-series to disk. The official documentation is available in [POS12].

## 5.1 Install prerequesites

The following prerequesites must be installed to compile DLS.

**Binary packages**

```
sudo apt install autoconf libtool libexpat1-dev    \
                 libxml2-dev libfltk1.3-dev        \
                 libfftw3-dev liburiparser-dev     \
                 protobuf-compiler libprotobuf-dev \
                 libpcre3-dev
```

**PdCom library**

PdCom is the library to speak with PdServ. It must be downloaded and compiled from the sources.
Download and compile PdCom library

```
cd /tmp
git clone https://gitlab.com/etherlab.org/pdcom.git
cd pdcom
git checkout 3.0.9
./bootstrap.sh
./configure --prefix=/opt/etherlab
make
```

Install the PdCom library

```
sudo make install
```

## 5.2 Get DLS source code

Download the source code with mercurial, then switch to the revision that has been tested in this tutorial (rev 370553789444).

```
cd /tmp
git clone https://gitlab.com/etherlab.org/dls.git
cd dls
git checkout 370553789444fc6c19ae22091db00067791370ef
```

## 5.3 Compile DLS source code

DLS uses a classic autoconf script to generate the Makefile.

```
cd /tmp/dls
./bootstrap.sh
./configure --prefix=/opt/etherlab
make
```

## 5.4 Install DLS programs

The programs are installed in `/opt/etherlab`

```
sudo make -C /tmp/dls install
sudo ldconfig
```

### Create system account

Create a system account `dls`

```
sudo useradd -s /usr/sbin/nologin -r dls
```

### Create data directories

Create a directory (`/home/dls/data`) to store recorded data.

```
sudo install -d -o dls -g dls /home/dls/data
```

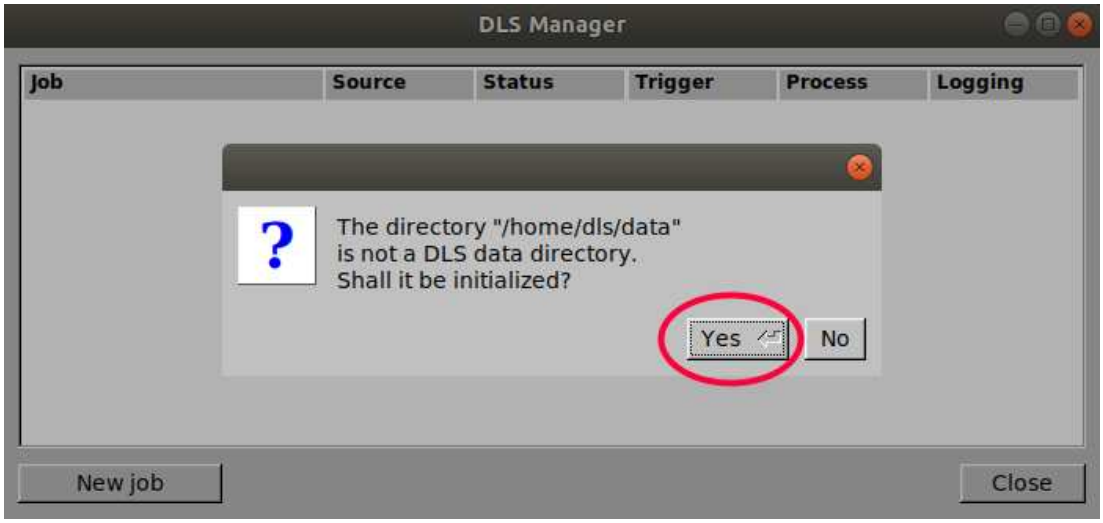
## 5.5 Manage acquisition with dls_ctl

The acquisition are managed with `dls_ctl`.

```
sudo -i
export PATH=/opt/etherlab/bin:$PATH
dls_ctl -d /home/dls/data
```

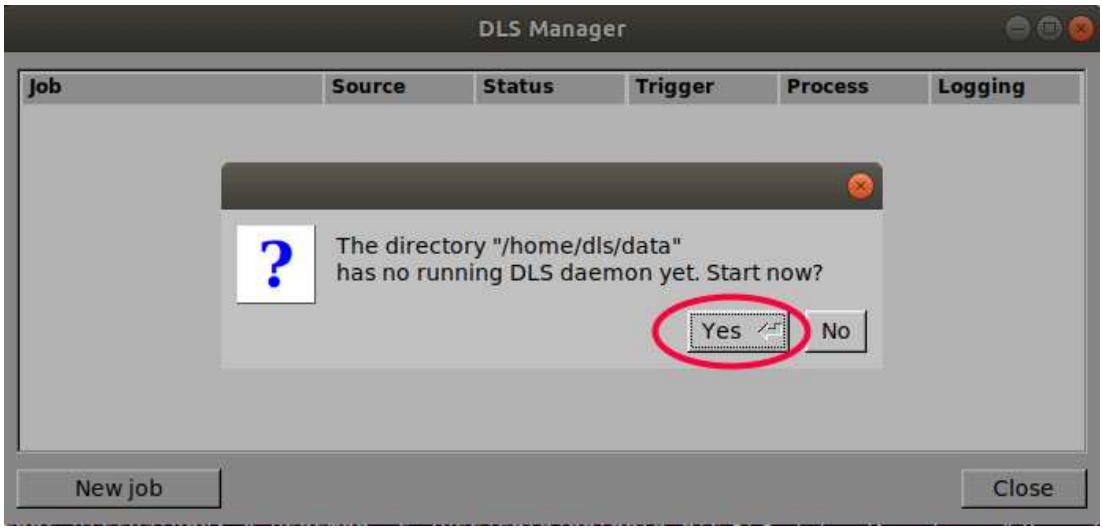
### Initialize the data directory

The main window appears. The program asks if the data directory shall be initialized. Answer YES.




### Start the acquisition daemon

The program asks if the acquisition daemon shall be started. Answer YES.

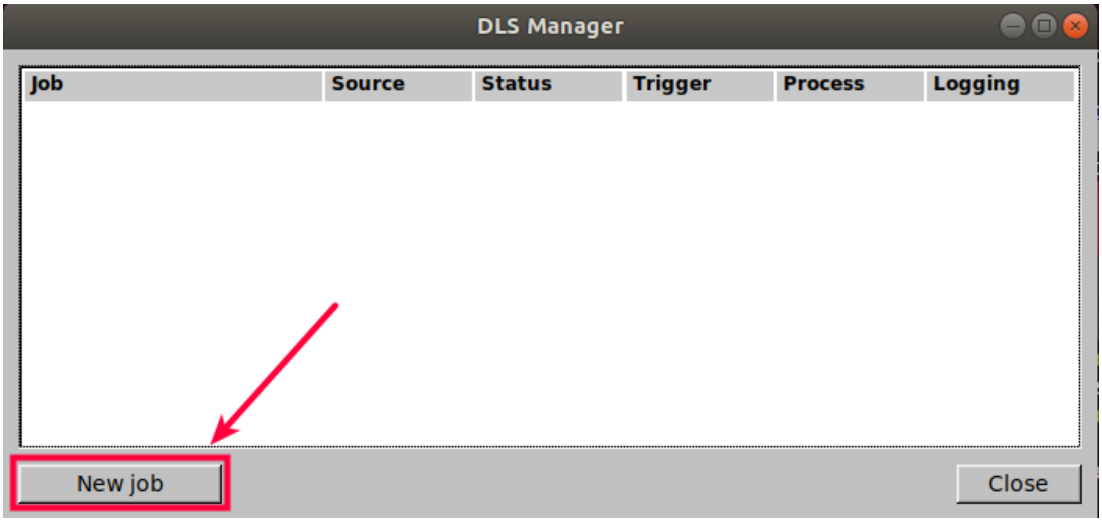


In the console, a message shows that the dls daemon is now running.

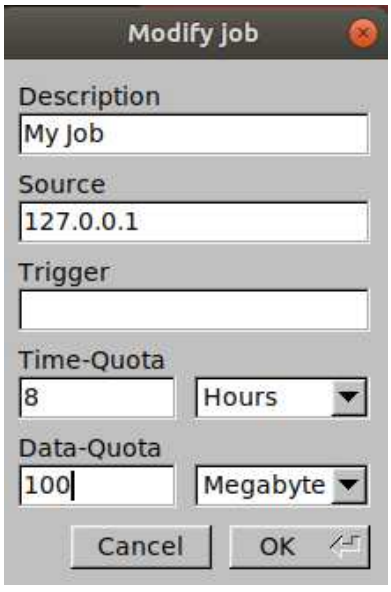
```
dls_ctl 1.4.0-rc2 revision 51dc3443178a
DLS data directory "/home/dls/data"
dlsd 1.4.0-rc2 revision 51dc3443178a
Using dls directory "/home/dls/data"
DLS running with PID 26307 [daemon]
```

### Configure acquisition

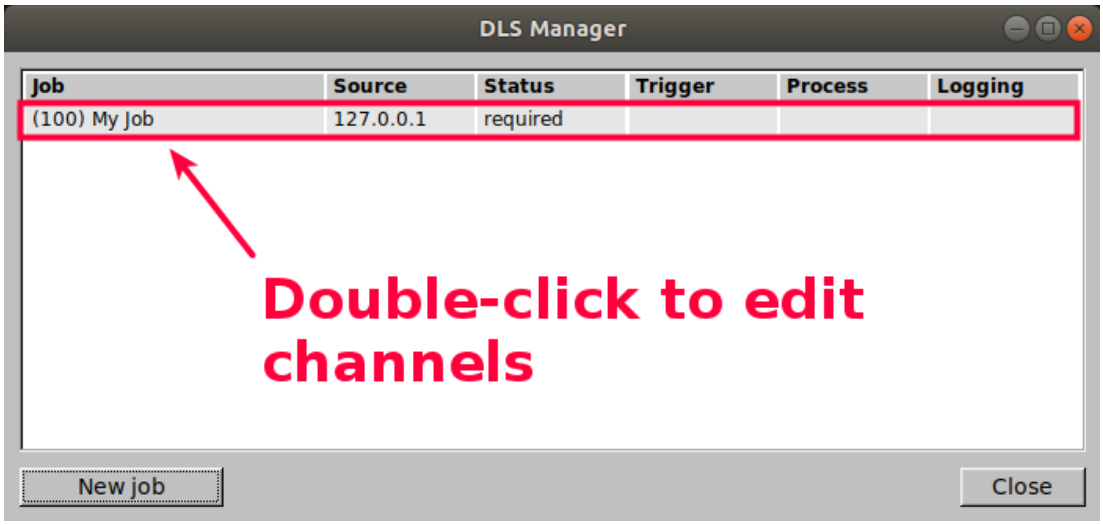
Click *New job* to create a new acquisition job



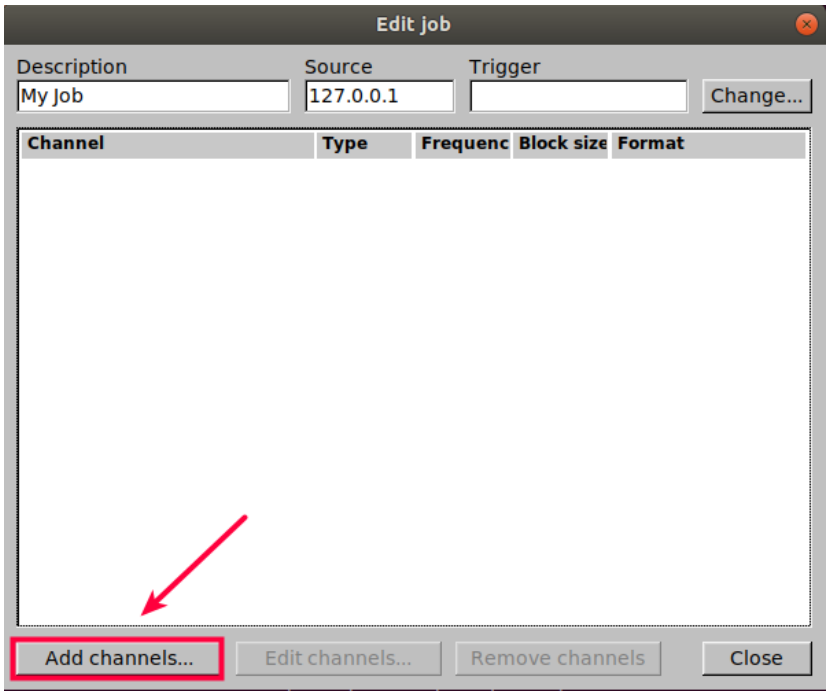

Fill the form with the job properties. Warning: unlike the others, the field *Source* cannot be modify later. therefore it must be enter correctly at the first time.



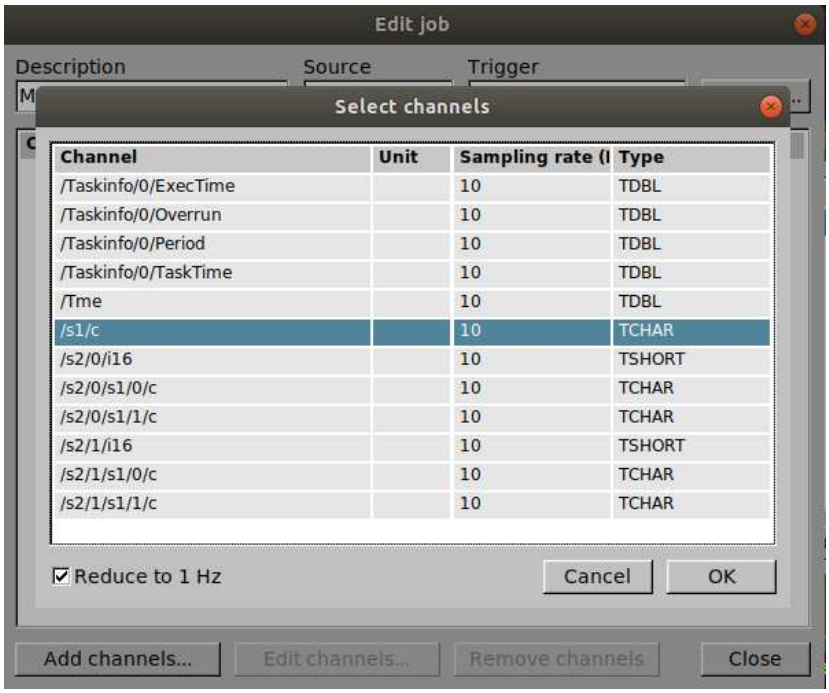
The new job appears in the list, double-click on it to edit the channels.



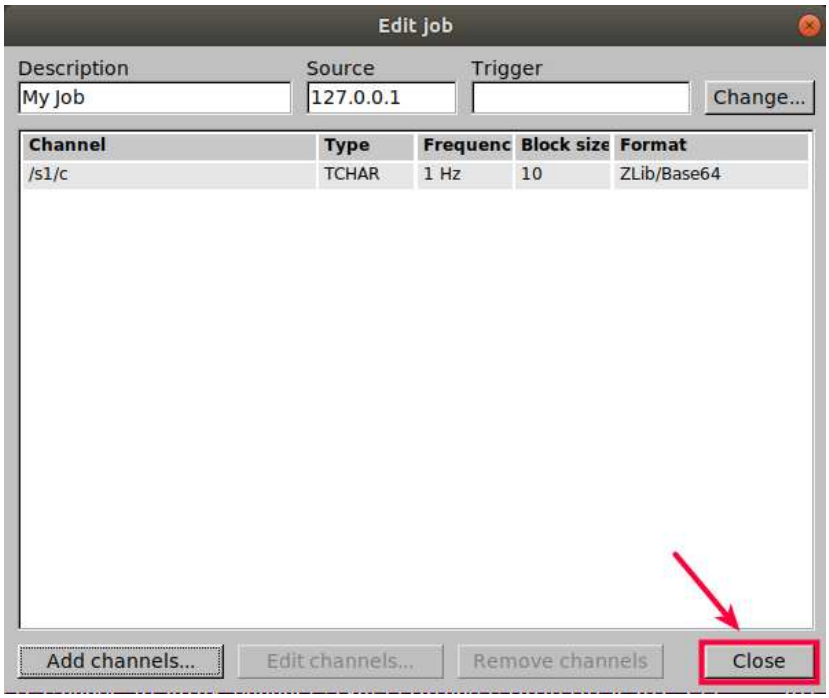

Click on *Add channels...*



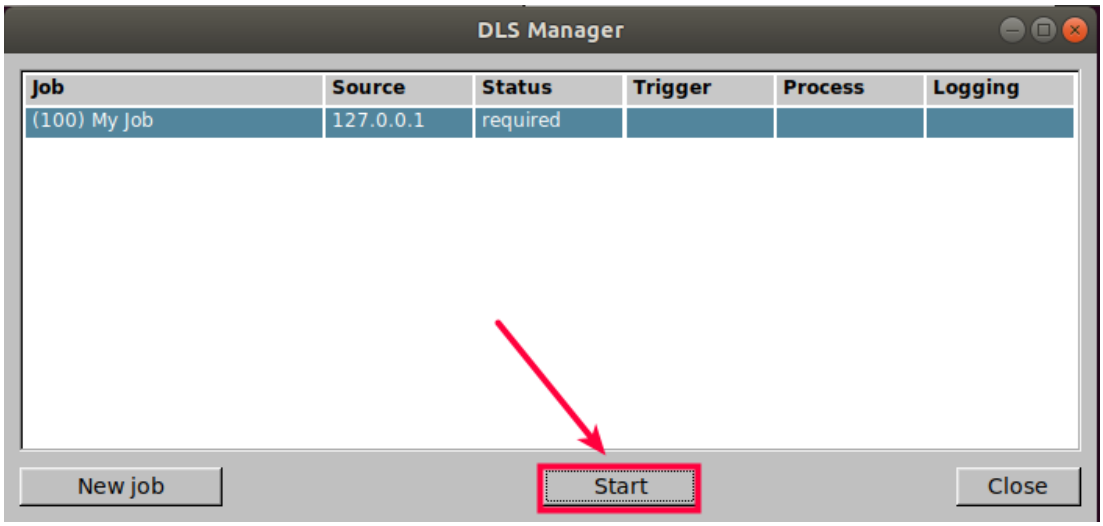

Select channel */s1/c*



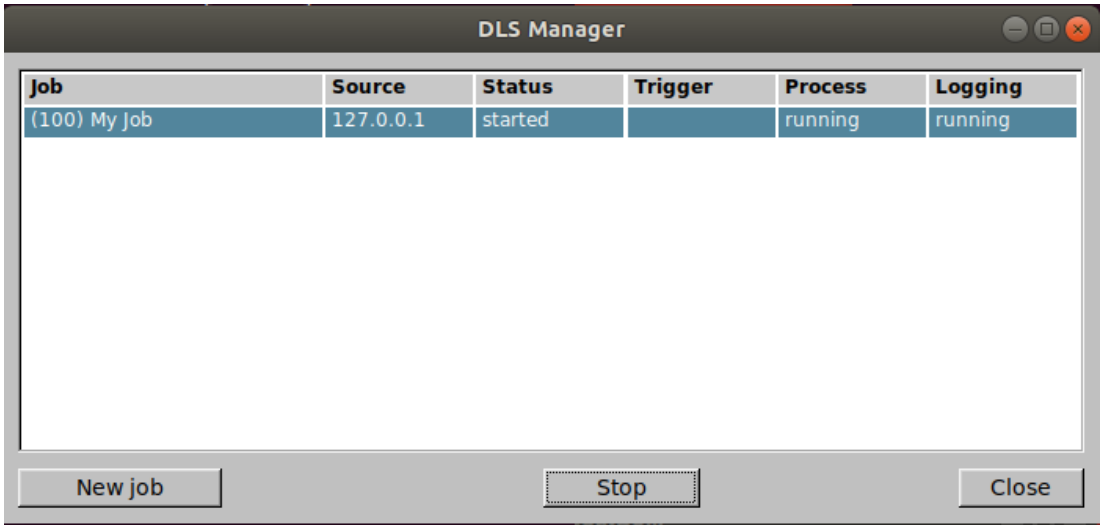

Close the job window.




Click on the button *Start*




The aquisition is running.




## 5.6 Plot channels with dls_view

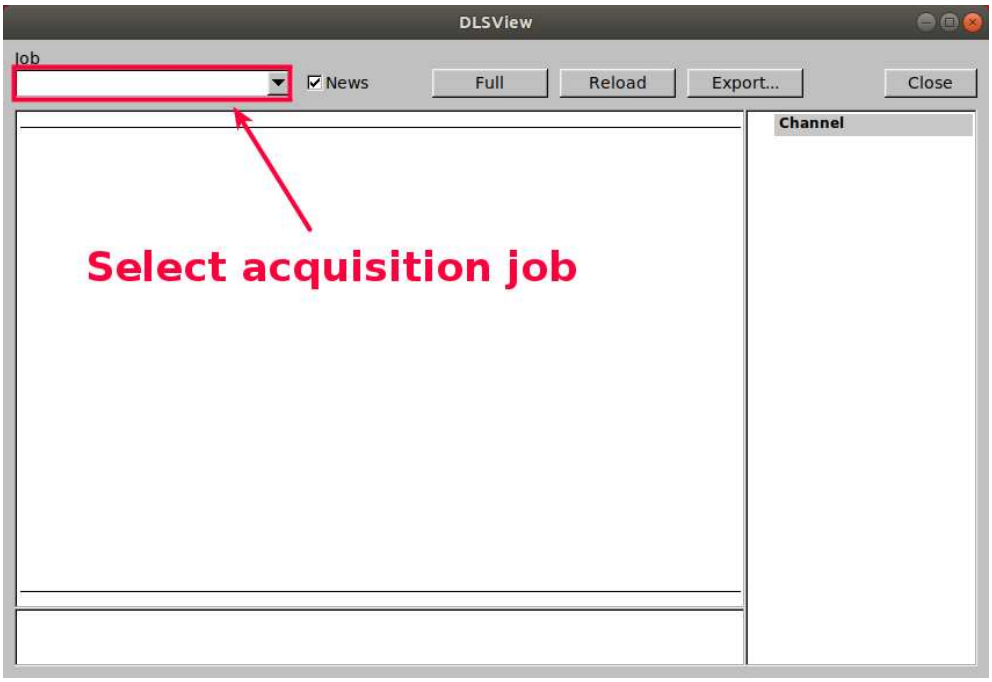
`dls_view` is the legacy viewer for DLS. The newer viewer (`dlsgui`) is explained in section 6.

### Run dls_view

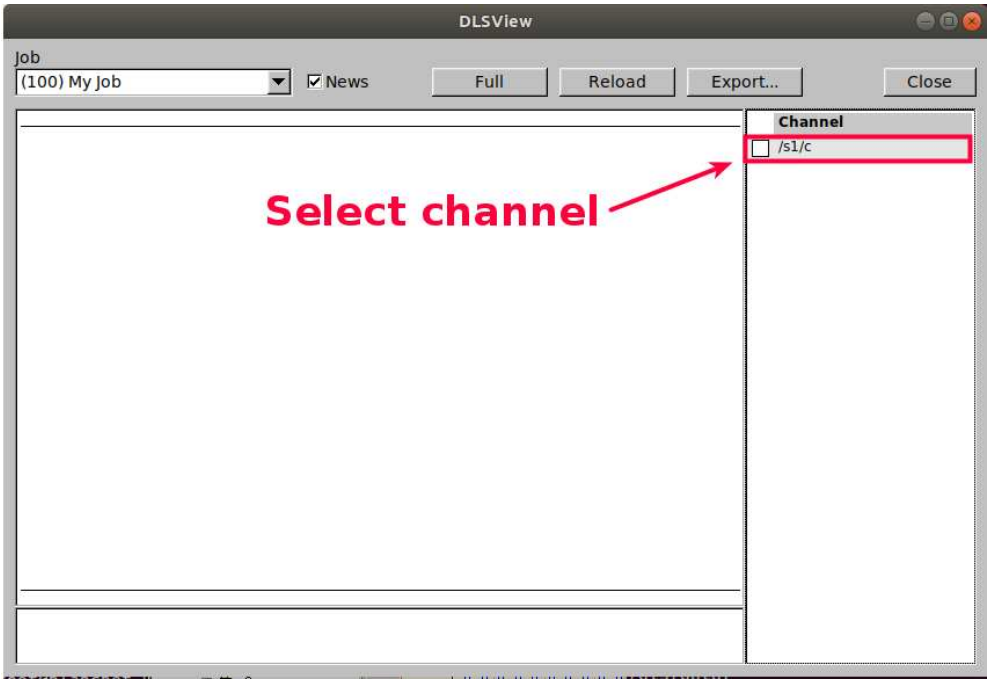
Unlike `dlsgui`, `dls_view` needs filesystem access to the recorded data.

```
sudo -i
export PATH=/opt/etherlab/bin:$PATH
dls_view -d /home/dls/data
```

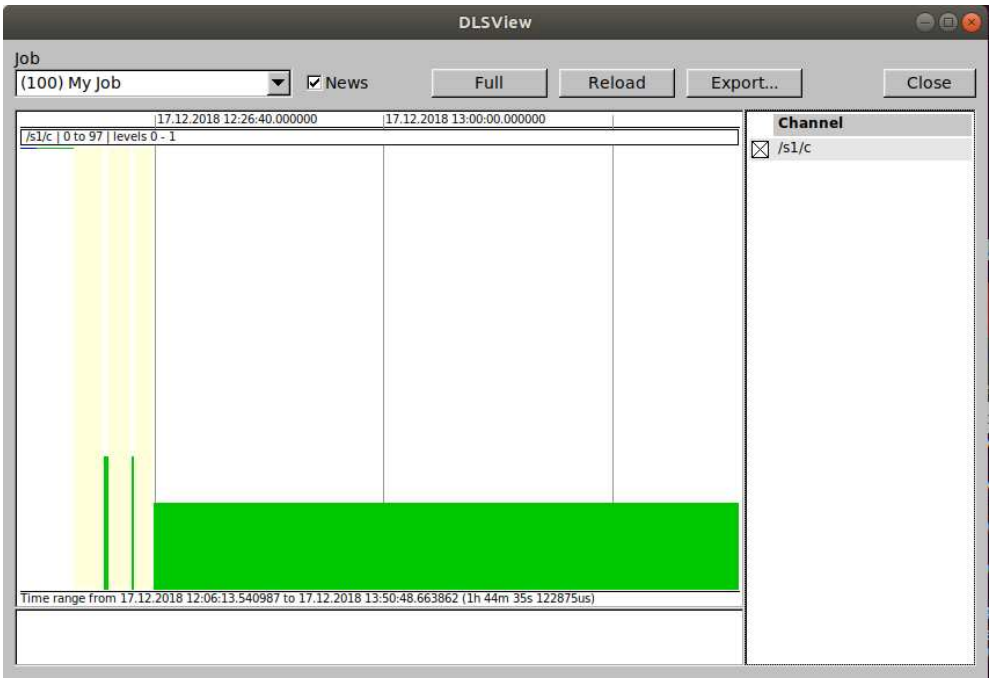
Select the acquisition job to display



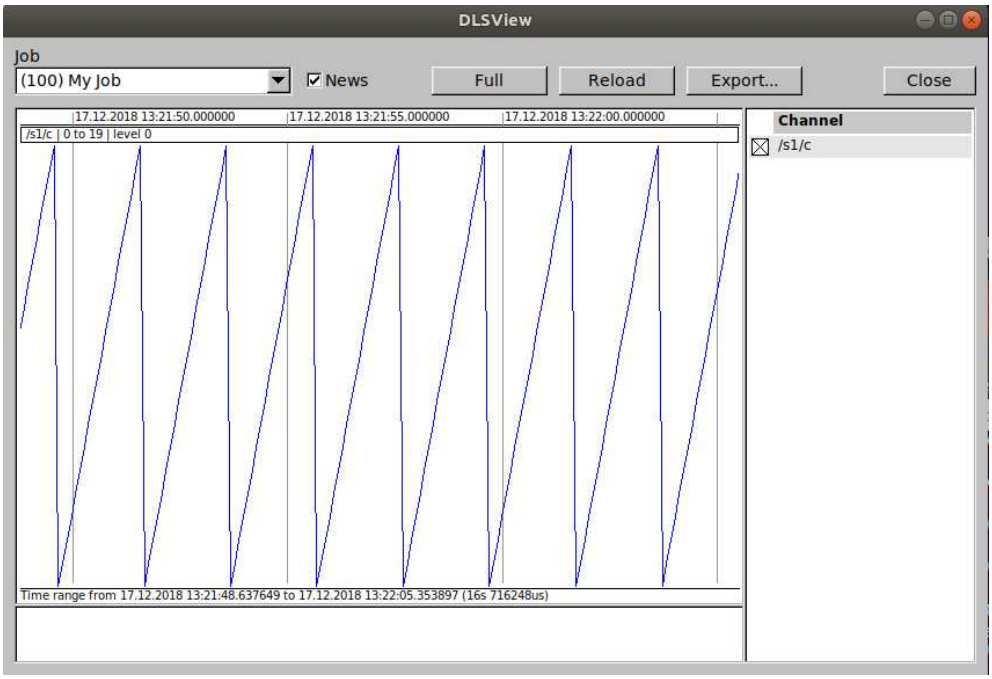

Select the channel to display




The plot appears as a green rectangle because but the lines are overmuch condensed.




Draw a rectangle in the viewing area to zoom in, click on the viewing area to zoom out. See [POS12] for the complete description of the user interface.



# Chapter 6

# Dlsgui

This chapter explains the installation of `dlsgui` which is the newer viewer for DLS.

## 6.1 Install prerequesites

The following prerequesites must be installed to compile DLS.

**Binary packages**

```
sudo apt install  qt5-qmake qt5-default libqt5svg5-dev qttools5-dev
```

Compile widgets

```
cd /tmp/dls/widgets
qmake
make
```

Compile dlsgui

```
cd /tmp/dls/gui
qmake
make
```

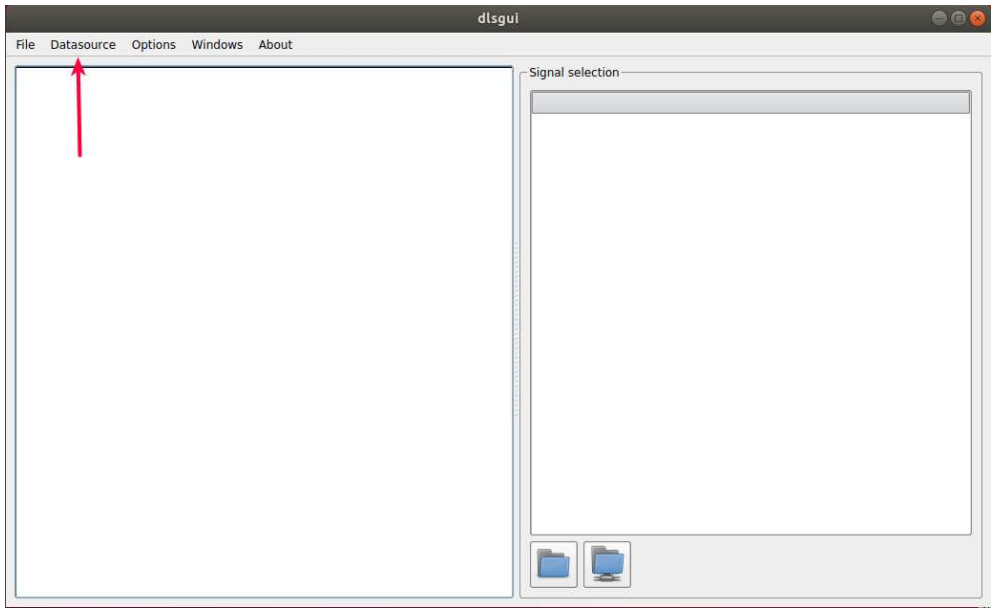
Install the program

```
sudo -i
cd /tmp/dls
install -m755 -oroot -groot gui/dlsgui /opt/etherlab/bin/
install -m644 -oroot -groot widgets/libDlsWidgets.so  /opt/etherlab/lib/
echo /opt/etherlab/lib > /etc/ld.so.conf.d/etherlab.conf
make -c /tmp/dls/gui install
ldconfig
```

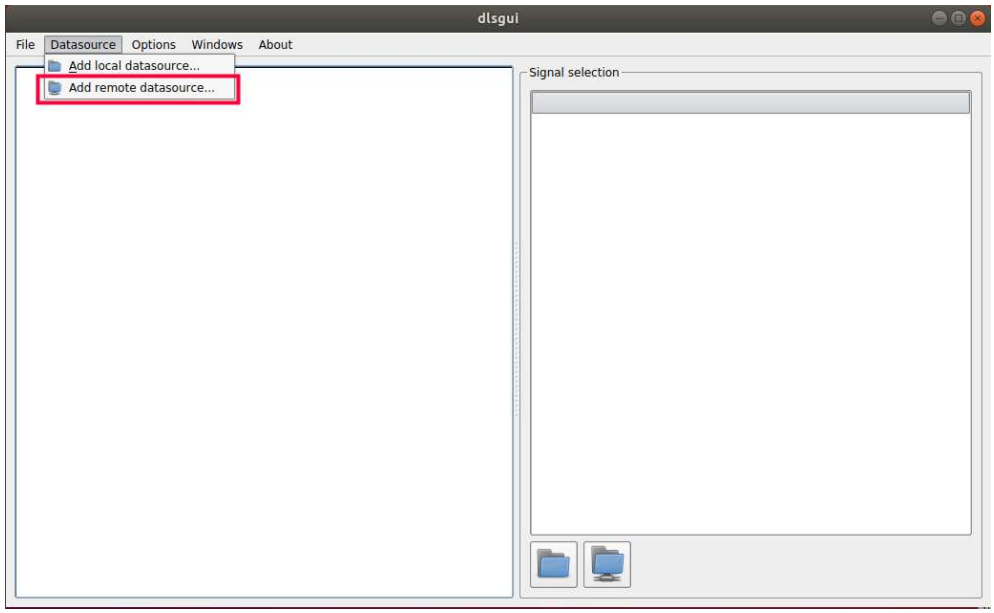
## 6.2 Run dlsgui

`/opt/etherlab/bin/dlsgui`

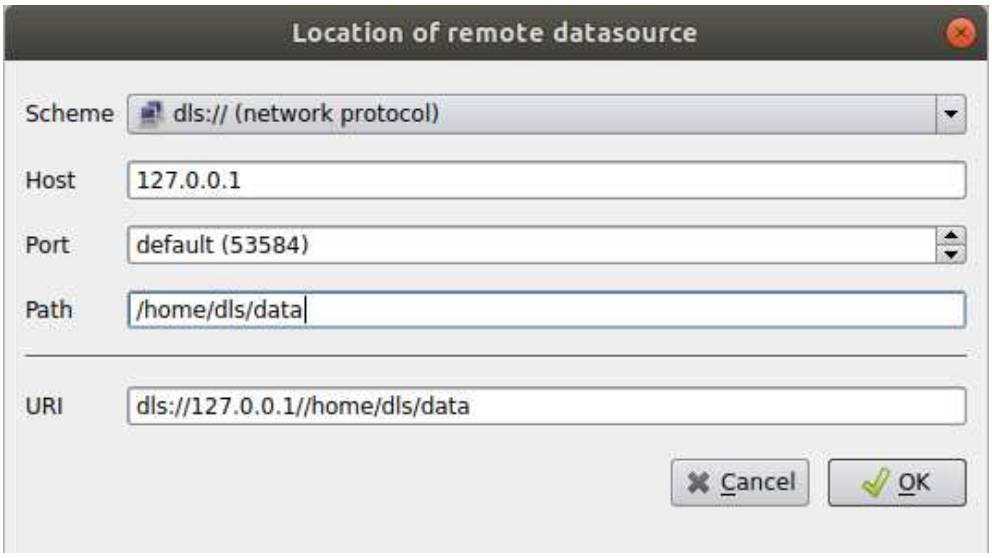
Open the datasource menu



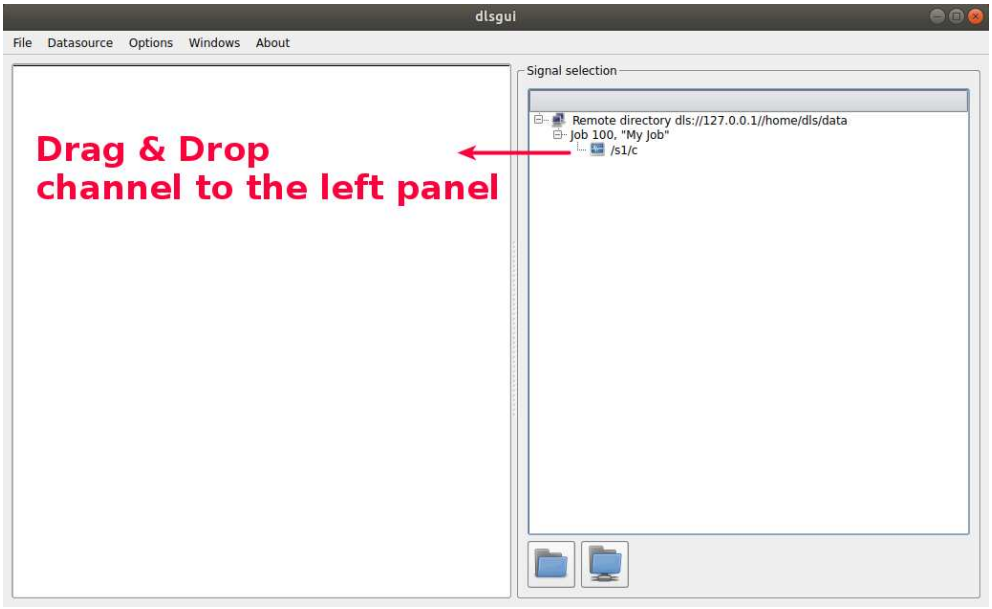

Then *Add remote datasource...*



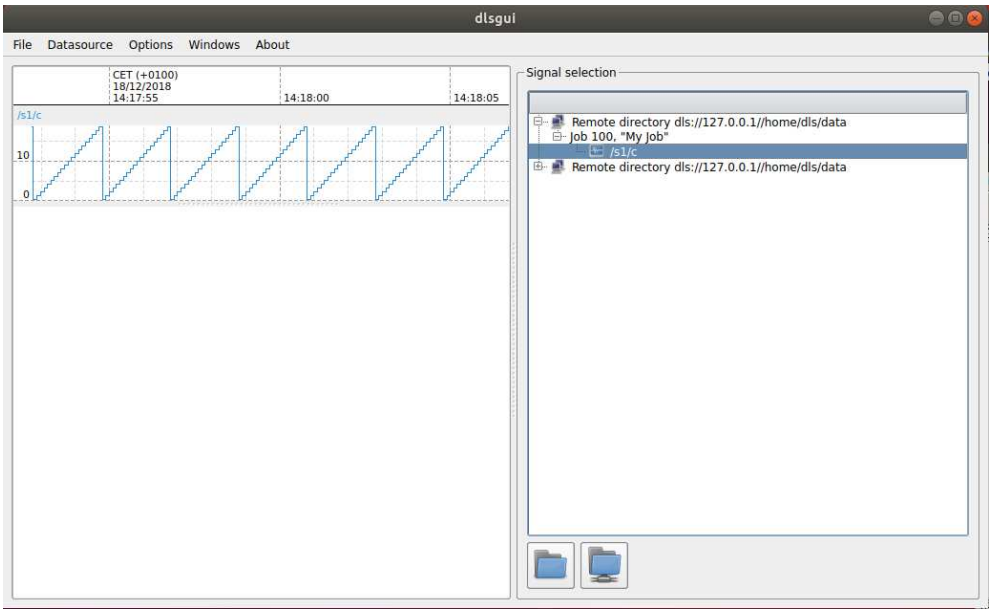

Fill the form




Expand the tree in the right panel to find the channel to display, then drag-and-drop it to the left panel.




Zoom and navigate with keyboard and mouse inside the left panel to see the information.


dlsgui
File
Datasource
Options
Windows
About
CET (+0100)
18/12/2018
14:17:55
14:18:00
14:18:05
/s1/c
10
0
Signal selection
Remote directory dls://127.0.0.1//home/dls/data
Job 100, "My Job"
/s1/c
Remote directory dls://127.0.0.1//home/dls/data

# Chapter 7

# Conclusion

After completing this tutorial, you can understand better how TestManager and DLS work. If you want to go futher with these programs, it is time to read their official documentations. See the bibliography to get the links.

# Bibliography

[ETH18] EtherLab website http://etherlab.org

[IGH10] IgH, *IgH Software Testmanager*, version 3.4 – September 2010

[POS12] Florian POSE, *Data Logging Service (DLS)*, version 1.2, – February 1, 2012