SURF Project Control System Documentation

## Overview:

The different devices in the SURF setup are each operated in different manners. Some are controlled through functions accessible from shared libraries, .so/.dll files (Linux/Windows respectively), and some are controlled via RS-232 communication protocol. Tango is a layer we can put on top of all of these different communication protocols, in order to homogenize the way the user controls them. A sketch of the control system layers are shown in Fig 1.

Figure 1: The different layers in the interaction with the equipment. Using Tango let’s the user interact with devices in the same way regardless of the underlying communication protocols.

**Hardware**

**Driver**

A:

native serial

port driver

B:

Manufactors

.so library)

**Wrapper layer**

A:

serial cmnds

B:

.so wrapper

C/C++->Py

**Tango**

**Device**

**Class**

**GUI**

**Hardware**

**Driver**

A:

native serial

port driver

B:

Manufactors

.so library)

**Wrapper layer**

A:

serial cmnds

B:

.so wrapper

C/C++->Py

**Tango**

**Device**

**Class**

**GUI**

* **Hardware**

The SURF experiment is composed of a number Spectrometers, pumps, light sources and shutters, which we will need to send and receive information from. The devices are detailed below.

* **Driver**

The drivers fall in two broad categories. Some devices are operated through RS-232 serial port communication and rely on the users computer to have the necessary drivers and software for interacting with them. These devices can for example be controlled via python using the pySerial library.

The other category consists of devices which are controlled through .dll (Windows) or .so (Linux). These types of drivers are often not updated very often and may require deprecated software. In my experience getting these drivers working can be a very time consuming task.

* **Wrapper**

The wrapper layer makes the functions or commands provided by the driver accessible in a pythonic way. For serial port communication this can be done with pySerial and for .so libraries this can for example be done with the python library Ctypes.

From the old SURF project we have Ctypes wrappers for both Nidaqmx and Omnidriver. I have written pySerial wrappers for the XLP6000 pumps and for the Cornerstone130 Monochromator.

* **Tango layer**

Tango devices classes allow us to control the devices using tango guis and tango syntax. This makes it easy to control the equipment as well as to develop applications for it. Each device has a number of attributes, states and commands. Figure 2 and 3 show some examples of how tango devices are controlled.

There are different ways to get Tango. Manual installation is tedious and will require you to install several dependencies and to configure MariaDB/MySQL correctly. Hopefully KITS can help you with an easy installation tool.

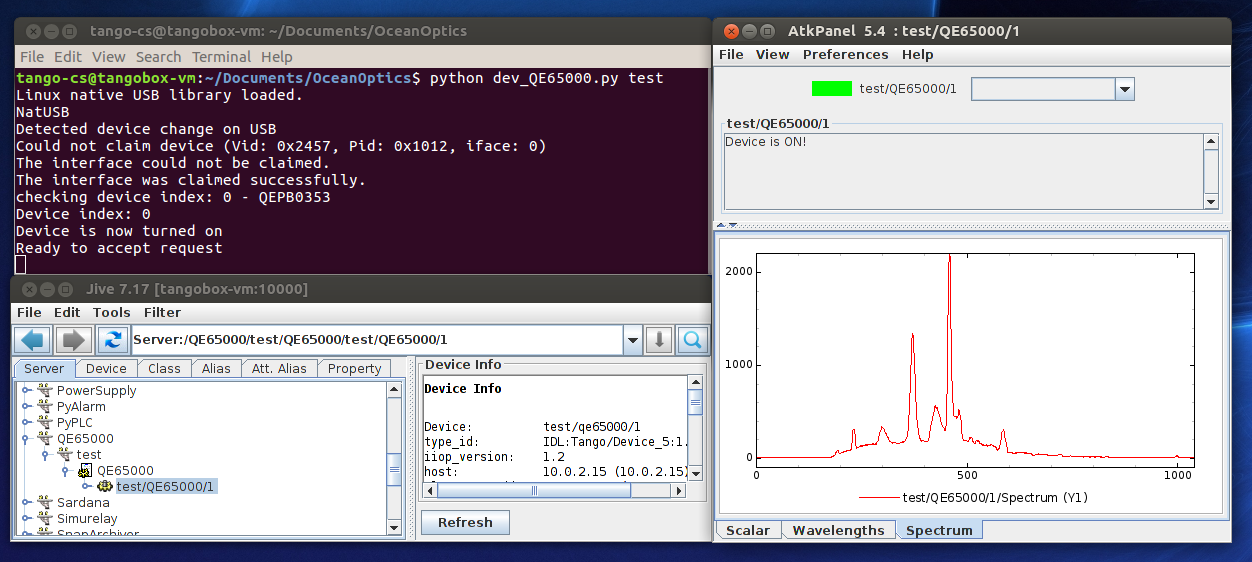
* **GUI**

Once the Tango devices exist a GUI needs to be built for use in experiments. A priority here is that it should be easy to use even for non experts. The NOMAD software from SOLEIL can be an inspiration for this task.

Once drivers are installed, wrappers and tango device written, then we can control the equipment via a standard GUI or directly from a terminal. First the devices must be imported to the database and then the device server must be run. For an example see:

https://pytango.readthedocs.io/en/stable/quicktour.html

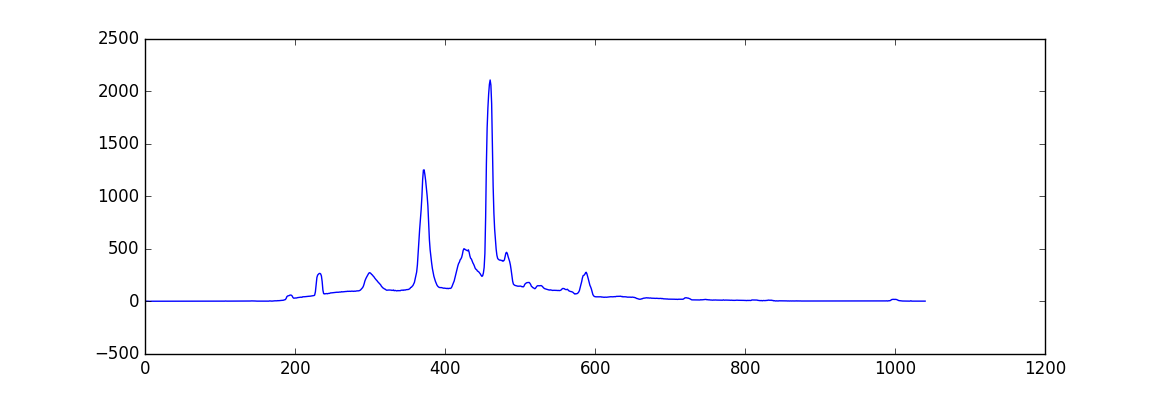
Fig 2 shows how to run the UV-spectrometers device server as well as how to open the devices ATK panel through jive.

  
Figure 2: Controlling a spectrometer through Jive.

The USB spectrometers can be used via the ATK panels opened by Jive, however the Pumps and the monochromator rely on much slower RS-232 communication, which leads to timeout errors when opening them as ATK panels. This can be solved via threading, but I did not have time to do that. However, all the devices work fine when accessed through a script or an Ipython console.

The spectrum shown in Fig 3 was acquired from the same Ocean Optics QE65000 spectrometer as used in Fig 2. The following commands were issued from an Ipython console:

|  |
| --- |
| In [1]: import tango  In [2]: FL = tango.DeviceProxy('test/QE65000/1')  In [3]: import matplotlib.pyplot as plt  In [4]: FL.darkcorrection = True  In [5]: FL.integrationtime=5000  In [6]: FL.scanstoaverage=5  In [7]: plt.plot(FL.spectrum);plt.show()  In [8]: |

  
 Figure 3: Spectrum from QE65000 spectrometer.

## Current Status of the Project

The Status Table below shows the different devices in the SURF setup, their communication interfaces, how far the project was when I left it, and what has yet to be done. The end goal is to have a user friendly and intuitive graphical user interface built upon tango devices.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Status table** | | | | |
| **Physical devices** | **Communication protocol** | **Intermediate layer. fx C→py** | **Tango device classes** | **GUI** |
| Spectrometer:  HR4000 UV, | “.so” libs Omnidriver | Python wrapper  from old SURF | Tango Class  For both spectrometers | Graphical user interface  In the style of nomad |
| Spectrometer:  QE65000 Fl |
| Monochromator | RS-232 protocol  int terminal | Python script:  using pyserial library | Tango Class  Monochromator |
| IRaman Setup | BW SDK,  (Windows ) | Python wrapper | Tango Class |
| NI-daq | “.so” libs  Ni-DAQmx | Python wrapper  from old SURF | Tango Class  NI-daq |
| Tecan  Pumps | RS-232 protocol  int terminal | Python script:  using pyserial library | Tango Class  Tecan Pump |
| Temperature controller | RS-232 | Python script:  using pyserial library | Tango Class  Temp controller |

### Ocean Optics Spectrometers:

The Ocean Optics spectrometers HR4000 and QE65000 require Omnidriver, which can be downloaded from Ocean Optics website. We have a password for the combined OmniDriver+SPAM software package.

* Password: Etm4QBn7gd

Note that this password does not work for the separate Omnidriver installer. You must download the combined OmniDriver and Spam installer even though we will only use OmniDriver. After installing it you must go to Home/OmnidriverSpam/utilities (or equivalent path) and run the utility script. It will instruct you on if and how to fix missing symbolic links. Then you need to fix the environmental variables. I added the following to my .bashrc file on a Ubuntu Tangobox machine:

|  |
| --- |
| export OMNIDRIVER\_HOME="/home/tango-cs/OmniDriverSPAM"  export OOI\_HOME="/home/tango-cs/OmniDriverSPAM/OOI\_HOME"  export PATH="/home/tango-cs/OmniDriverSPAM/OOI\_HOME:$PATH"  export JAVA\_HOME="${OMNIDRIVER\_HOME}/\_jvm"  export JDK\_HOME="/usr/lib/jvm/java-8-openjdk-amd64"  export JDK\_INCLUDE\_FILE\_ROOT="/usr/lib/jvm/java-8-openjdk-amd64"  export JVM\_ROOT="${OMNIDRIVER\_HOME}/\_jvm"  export LD\_LIBRARY\_PATH="$OOI\_HOME:${JAVA\_HOME}/lib/amd64/server" |

But you can also just define these variables in your IDE, if you don’t want them to be globally defined. Note that .bashrc is not used when you run as sudo, so if you try to run the spectrometer server as sudo you will get errors due to missing environmental variables.

The driver functions are accessed through the shared libraries omnidriver.so and common.so found in OmniDriverSPAM/OOI\_HOME. On Ubuntu I had to compile them, together with jvm.so into a new shared library, libw.so, to fix problems with missing references.

There exists a tango class from SOLEIL for this device, but since it is Windows only I instead wrote a new tango class using a wrapper, written by Sylvio Haas, for the old SURF project.

Note: Occasionally the tango device can’t connect to the spectrometer. It is then necessary to unplug and plug back in the USB connection to the spectrometer. From the documentation of the old SURF project I can see that this was also necessary back then.

### Monochromator Cornerstone 130:

There is an existing cpp project from SOLEIL. It comes with a makefile which needs to be thoroughly adapted to your systems architecture if used. It also depends on some tango extension, which you will need to install to use the device: yat, yat4tango.

As an alternative I have made a python library for communicating with the monochromator using pyserial. To connect to the spectrometer you will need to run python as sudo, to get access to the port.

An alternative approach is to use the RS-232 communication protocol that is described in the Cornerstone 130 user manual. For example in python via PySerial library.

In order to connect to the device you will need to run the python script or Ipython console as sudo. If you are still unable to connect to the device, then try rebooting the machine with the USB connection still plugged in, and try to connect again.

A problem is that the default device polling is too fast for the monochromators slow serial communication. This can cause “Timeout errors”. If you set the timeout property to 15000 ms in your script or Ipython session then it should be solved. From Ipython:

|  |
| --- |
| In [1]: import tango  In [2]: M = tango.DeviceProxy('test/Cornerstone/1')  In [3]: M.get\_timeout\_millis()  Out[3]: 3000  In [4]: M.set\_timeout\_millis(15000)  In [5]: M.get\_timeout\_millis()  Out[5]: 12000 |

I did not find out how to change the timeout limit for the ATK panels, and changing polling time did not help there. So you will still get timeout errors when accessing the monochromator through an ATK panel.

Note that in order to have access to the USB port you will need to run the server as sudo.

### IRAMAN:

Although I did not spend a lot of time on the Raman part of the SURF setup, I did learn that the SDKs for the spectrometer and laser are Windows only.

### Tecan XLP6000 Pumps:

The XLP6000 pumps are controlled via an RS-232 communication protocol as described in the XLP 6000 operating manual. I wrote a Python wrapper for them using PySerial.

The pumps have the same problems with slow communication and in scripts and Ipython it can be solved as shown above for the monochromator.

Note that in order to have access to the USB port you will need to run the server as sudo.

### NiDAQ USB-6501:

The driver for the daq, Nidaqmx, is only supported for Red Hat Linux, Centos, Mandriva, and Windows, So working in the Ubuntu tangobox. It can probably be made to work for Ubuntu, but since MAXIV uses CentOS, it makes more sense to migrate everything to a CentOS machine.

Sylvio Haas wrote a python wrapper for this device in ctypes, which can be found on the old surf windows laptop. Since NidaqMX Linux version has not been updated since 2011 this wrapper will still work.

A small problem with the Nidaqmx is that it requires kernel 2.6. So you will need to install an extra (not a replacement) kernel.

### Temperature Controller:

I did not study this component thoroughly, but it is controlled via RS-232 serial interface like the monochromator and the pumps, so writing a pySerial wrapper will be a good approach.

## Proposed work plan for future development:

* *Familiarize yourself with lab setup, the old SURF software as well as the existing SURF tango devices.*
* *Migrate to CentOS machine with Tango + PyTango installations.*
* *Install Omnidriver and Nidaqmx.*
* *Build a tango device class for NI USB-6501 using Sylvios driver wrapper.*
* *Make a demo script for performing an experiment with pumps, UV and fluorescence spectrometers and monochromator and perform measurements.*
* *Develop Tango device class for Temp controller*
* *Develop Tango device class for IRAMAN. This may have to be windows based, so ask KITS for advice on how best to integrate it with the rest of the setup.*
* *Develop a graphical user interface*