Python Based Software To Measure Wavelength At Optically Pumped Polarized Ion Source

BROOKHAVEN NATIONAL LABORATORY

Abstract: *Often diagnostic tools are packaged with proprietary software and it is challenging to integrate with native environment. The HighFinesse Angstrom Wavemeter used at OPPIS experiment for laser wavelength measurement is controlled using commercial software not supported by RHIC style controls. This paper will describe the integration of such a complex system and use of python for cross platform data acquisition*

System In Use:

• The RHIC OPPIS had been upgraded for Run-2013 to higher intensity and polarization.

• The FABS source produces a high intensity and brightness primary proton beam. Part of this beam is converted to electron-spin polarized H-atoms by electron pick-up in an optically pumped Rb vapor cell.

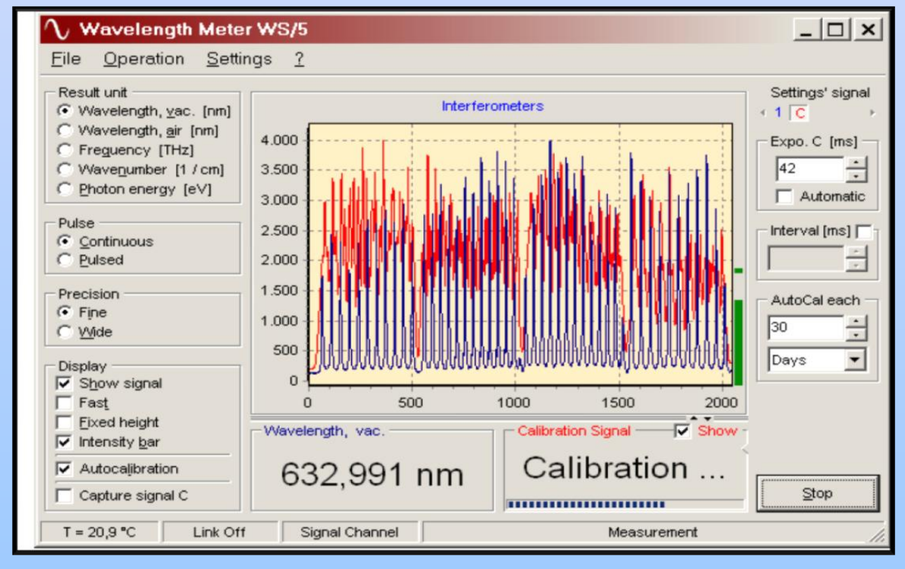
• The beam polarization strongly depends on the power, the frequency, and the line width of the pumping laser. Monitoring and control of these parameters are critical to prevent polarization loss.

• An Angstrom WS/6 200 wavelength meter is used to control the laser.

This meter is designed to measure wavelength of continuous wave and pulsed laser with high accuracy and measurement rate.

Problem Statement:

The measurement results, settings and all controllable parameters are accessible for user's program via calls of wlmData.dll routine. The RHIC control infrastructure is Linux based and does not support interfacing with windows style dynamic linked libraries (DLL). It is required to write a special program that interfaces the two development environments seamlessly to make use of existing control applications such as loggers.



Proprietary software provided along with wavelength meter for measurement and control.

Solution:

Use of a Foreign Function Interface (FFI) language to make calls to the DLL and publish data to RHIC's controls framework. Many alternatives are available such as Perl, Ruby, python, or cross platform compilers.

Implementation:

• Using Python's Ctypes module.

• Ctypes Supports all complex features needed to wrap DLL calls written in C/C++ including ability to simulate C structures, unions, bit-fields and event loops.

• Ctypes can also gracefully handles callback functions (a non trivial task for FFI)

• Development environment is a standard python IDE such as IDLE which is very lightweight contrary to cross platform compiler as CYGWIN.

Possible improvements

• The header file provided by the manufacturer had to be rewritten as a python module for error codes and macros parsing. An interface layer can be implemented to directly use the C code..

• Replace current DLL with 64 bit version provided with the source code for possible better performance and compatibility with Ctypes.

WLM.py - C:\Users\pkankiya\Downloads\HgSetup-master\HgSetup-master\WLM.py

File Edit Format Run Options Windows Help

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 15 09:52:56 2015

@author: Prerana Kankiya
"""

from ctypes import *
from WLMconstants import *
import time

class WLMError(Exception):
    def __init__(self, message):
        Exception(self, message)

class WavelengthMeter:
    def __init__(self):
        self.dll=windll.wlmData

        #self.dll.Instantiate(cInstCheckForWLM, 0,0,0)

        # start wlm if not running, using a timeout of 10 seconds and extended return information
        res = self.dll.ControlWLMEx(cCtrlWLMHide+cCtrlWLMStartSilent+cCtrlWLMWait, 0, 0, 10000, 1)
        print hex(res)
        if res>=flErrUnknownError:
            print 'Unknown Error'
            res-=flErrUnknownError
        if res>=flErrTemperatureError:
            print 'Temperature Error - unable to
            res-=flTemperatureError
        if res>=flErrUnknownSN:
            print 'Unknown Serial Number'
            res-=flErrUnknownSN
        if res>=flErrWrongSN:
            print 'Wrong Serial Number'
            res-=flErrWrongSN
        if res>=flErrUnknownDeviceError:
            print 'Unknown Device Error'
            res-=flErrUnknownDeviceError
        if res>=flErrUSBError:
            print 'USB Error'
            res-=flErrUSBError
        if res>=flErrDriverError:
            print 'Driver Error'
            res-=flErrDriverError
        if res>=flErrDeviceNotFound:
            print 'Device Not Found'
            res-=flErrDeviceNotFound
        if res>=flServerStarted:
            print 'High Finesse Wavelength Meter
```

Screen 1 shows Code written in python to import wlmData.dll. It exports wavelength meter as a python object.
Screen 2. shows use of object created with ctypes to read wavelength of beam on demand.

rdesktop - 130.199.85.200

testWLM1.py - C:\Users\pkankiya\Downloads\HgSetup-master\HgSetup-master\testWLM1.py (2.7.9)

File Edit Format Run Options Windows Help

```
Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
>>> X = WavelengthMeter()
OK
0x1
High Finesse Wavelength Meter started successfully
>>> getWL

Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    getWL
NameError: name 'getWL' is not defined
>>> X.getWL
<bound method WavelengthMeter.getWL of <__main__.WavelengthMeter instance at 0x029B4990>>
>>> X.getWL()
795.0769143487829
>>>
```

pkankiya@bnl.gov

2015 ICALEPCS melbourne • australia