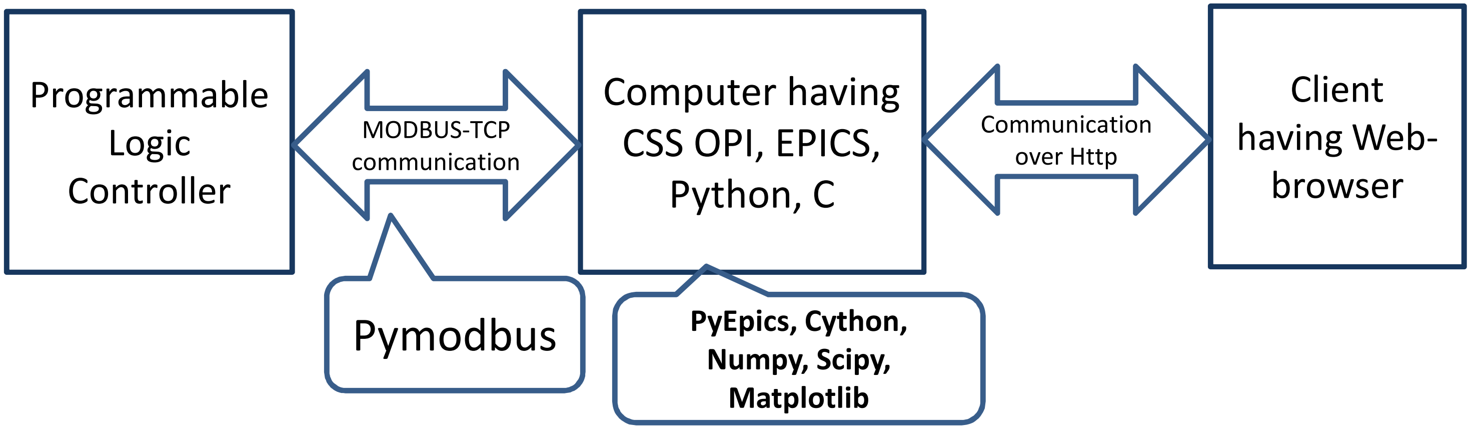
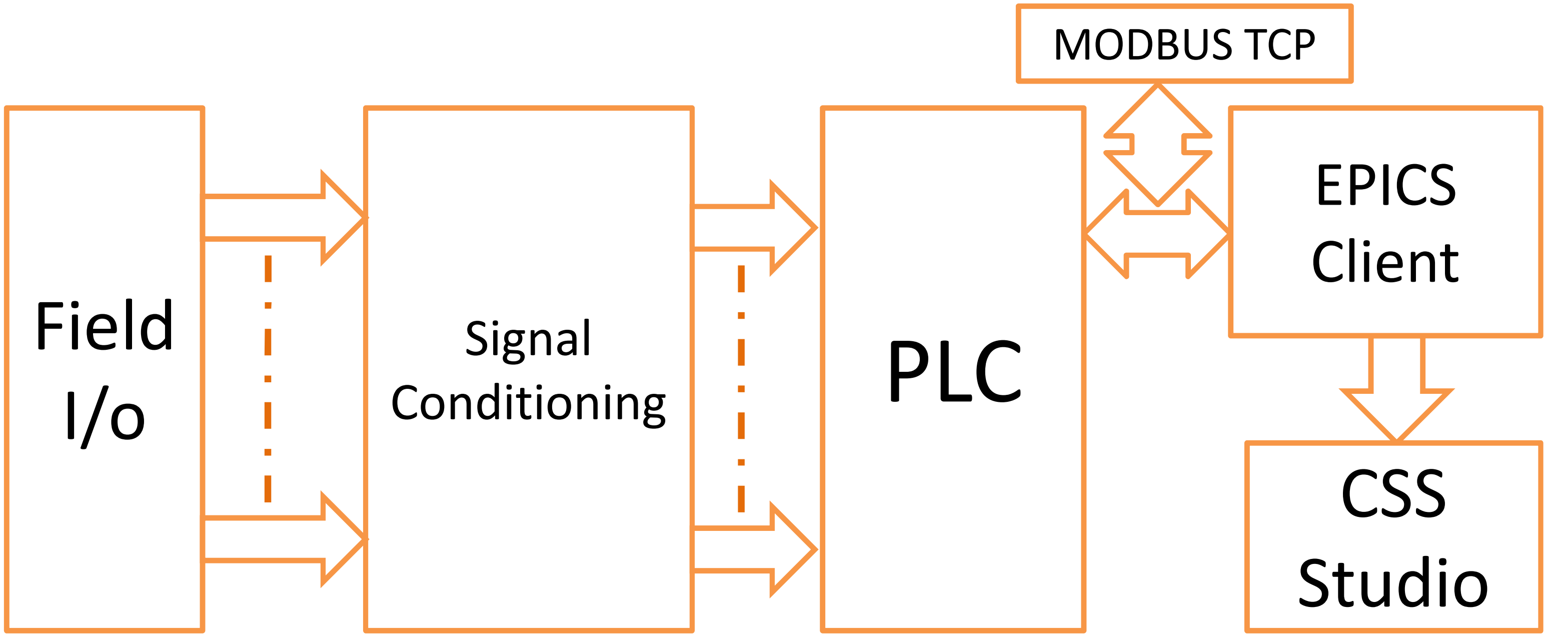
DAQ application using open source tools for Plasma heating experiment

by Rameshkumar Joshi

Institute for Plasma Research, India

Integrated DAQ architecture


Programmable Logic Controller
MODBUS-TCP communication
Computer having CSS OPI, EPICS, Python, C
Communication over Http
Client having Web-browser
Pymodbus
PyEpics, Cython, Numpy, Scipy, Matplotlib



Basic Requirement

- There are multiple DAQ systems are running simultaneously in the Institute. They must exchange data during experiment. Experimental Physics and Industrial Control Systems (EPICS) is provide the infrastructure by which each DAQ can exchange data as per requirement.

- User Interface has been designed using Java based Control System Studio (CSS) which can glue with the EPICS network variable to plot as well as display values.

- The system is designed and implemented the real time programmable logic controllers (PLC) data to monitoring and control using EPICS.

- The Interface between PLC and computer is based on Ethernet. Using MODBUS/TCP communication data have been exchanged between PLC and Computer. For MODBUS communication Pymodbus library has been used as implementation platform.

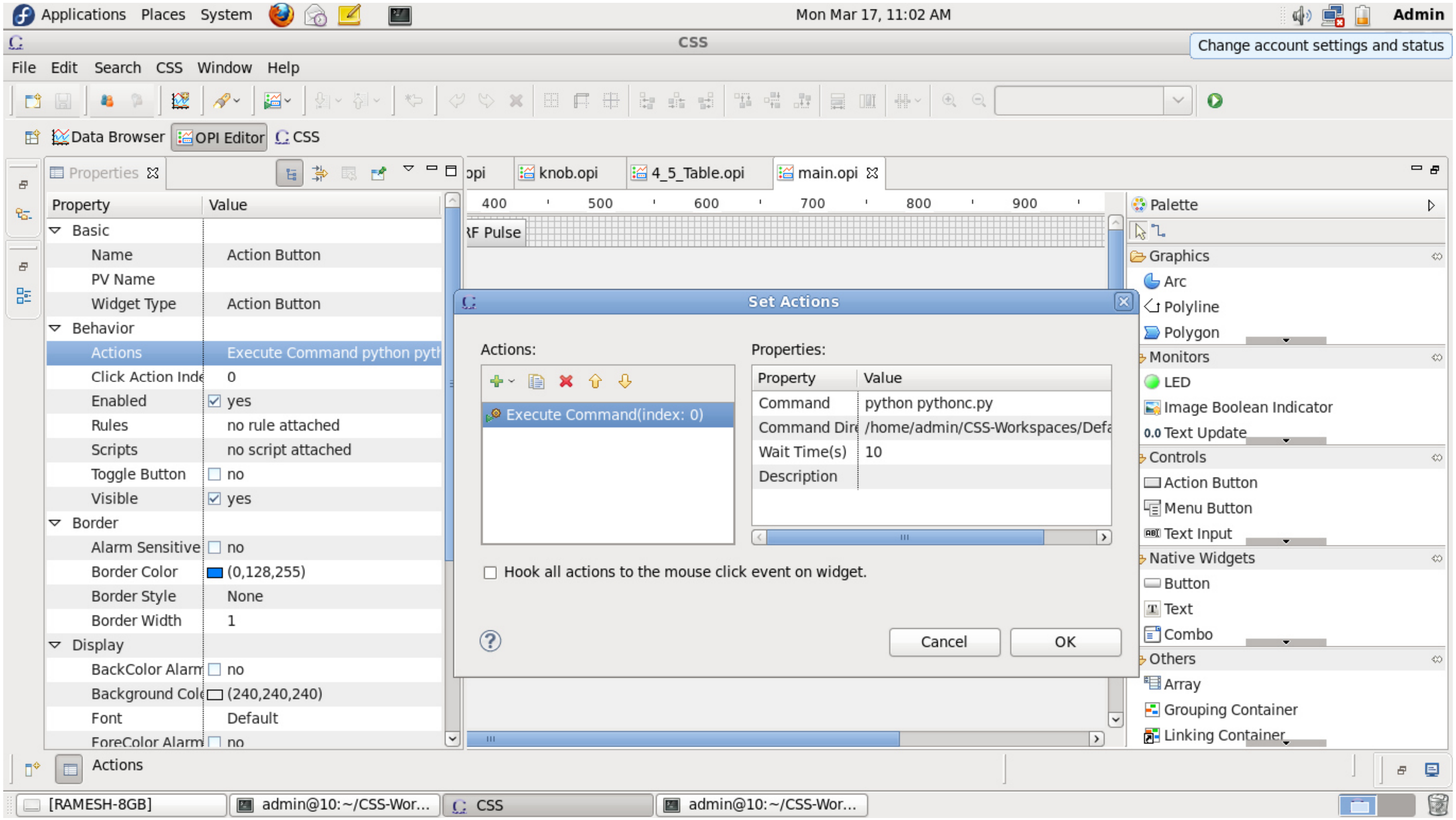
- Logical control has been designed using Python. The calibration and algorithm implementation is designed using C. The C functionality can be called using cython interface with Python.

- Calibration of the instrumentation data has been implemented by curve fitting using Numpy, Scipy and Matplotlib.

The infrastructure layer: EPICS

- The infrastructure layer is implemented with EPICS (Experimental Physics and Industrial Control System)
- EPICS is
  - an open-source control system toolkit
  - used in hundreds of large and small experimental physics projects world-wide: light sources, high energy physics, fusion (KSTAR, NSTX), telescopes
  - maintained and further developed by a world-wide community of users (including ITER)
- The same infrastructure for the CODAC servers and for the plant system controllers to ensure a uniform standard interface.

Python action script on action button click event handling



Property	Value
Basic	
Name	Action Button
PV Name	
Widget Type	Action Button
Behavior	
Actions	Execute Command python pyth
Click Action Inde	0
Enabled	☑ yes
Rules	no rule attached
Scripts	no script attached
Toggle Button	☐ no
Visible	☑ yes
Border	
Alarm Sensitive	☐ no
Border Color	(0,128,255)
Border Style	None
Border Width	1
Display	
BackColor Alarm	☐ no
Background Col	(240,240,240)
Font	Default
ForeColor Alarm	☐ no

Set Actions

Actions:

Execute Command(index: 0)

Properties:

Property	Value
Command	python pythonc.py
Command Dire	/home/admin/CSS-Workspaces/Defa
Wait Time(s)	10
Description	

☐ Hook all actions to the mouse click event on widget.

Cancel OK

Used tools and libraries

Name	File	Download
Base	baseR3.14.12.2.tar.gz	http://www.aps.anl.gov/epics/base/R3-14/12.php
Extensions	extensionsTop_20070703.tar.gz	http://www.aps.anl.gov/epics/extensions/configure/index.php
PYTHON	Python-2.7.6.tar.xz	http://www.python.org/getit/
EDM	edm-1-12-71.tgz	http://ics-web.sns.ornl.gov/edm/
CSS	CSS-3.0	http://cs-studio.sourceforge.net/

PV Access from a Shell

EPICS Base provides several utilities for reading and modifying Process Variables from command shell. Here *caget* and *caput* are demonstrated by accessing our softIoc.

```
$ caget calc:a calc:b calc:sum
calc:a                         0
calc:b                         0
calc:sum                       0
$ caput calc:a 2

Old : calc:a                         0

New : calc:a                         2
$ caget calc:a calc:b calc:sum
calc:a                         2
calc:b                         0
calc:sum                       2
$ caput calc:b 3

Old : calc:b                         0

New : calc:b                         3
$ caget calc:a calc:b calc:sum
calc:a                         2
calc:b                         3
calc:sum                       5
```

Database Creation:

```
record(ai,"ICRH:KW2_PLATE_VOLTAGE")
{
field(DTYP,"Soft Channel")
field(VAL,0)  field(UDF,1)  field(FLNK,"ICRH:KW2_PLATE_VOLTAGE")
}
```

```
import ctypes
import time
from epics import caget, caput
tmy_test_lib = ctypes.cdll.LoadLibrary('/home/admin/CSS-Workspaces/Default/icrh/libds.so')
print "now initialization is %d" % tmy_test_lib.DsInit()
print "now connection is %d" % tmy_test_lib.DsConnect()

while True:
tmy_test_lib.DsRead()
                    f3 = tmy_test_lib.DsReadAnalogIp(2)*20/4095
                    ---------
                    caput("ICRH:KW2_PLATE_VOLTAGE",f3)
                    ---------
                    time.sleep(.100)
```

C implementation

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>
#include <tcl.h>
#include "nrutil.h“

#define MYPORT                          4461
#define NO_OF_SAMPLES   2
#define         ACQ_LIMIT       100000

int  DsReadAnalogIp(int chno)
{
            int             status ;
            int             value ;
            int             sine_bit = 1 ;
            unsigned short int b ;
            b = disp_packet.data[chno] & 0x00ff ;
            disp_packet.data[chno] = disp_packet.data[chno] & 0xff00 ;
            disp_packet.data[chno] = disp_packet.data[chno] >> 8  ;
            b = b << 8 ;
            disp_packet.data[chno] = disp_packet.data[chno] + b ;

            /* convert into double */
            if ( disp_packet.data[chno] > 0x7ff0 )
            {
                            disp_packet.data[chno] = 0xfff0 - disp_packet.data[chno] ;
                            sine_bit = -1 ;
            }               return value;
}
```

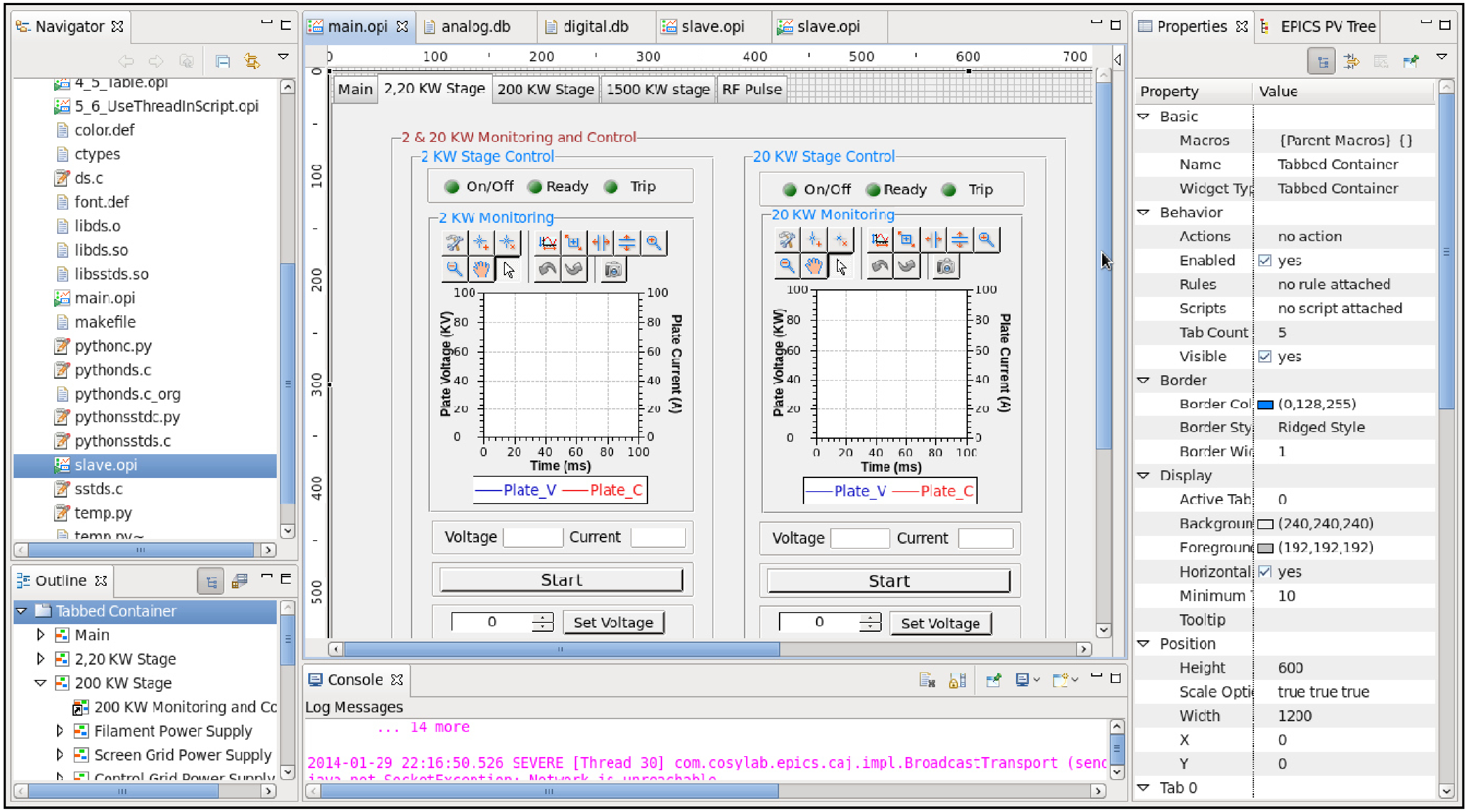
Numpy and Scipy and Matplotlib

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
from scipy.interpolate import interp1d
from scipy import interpolate

with open('kw20_plate_c.calib', 'r') as f2:
                        lines = f2.readlines()
                        data = [line.split()for line in lines]
                        data2 = np.asfarray(data)
                        x1 = data2[:,0]
                        y1 = data2[:,1]
print len(x1)
new_length=len(x1)
new_x = np.linspace(x1.min(), x1.max(), new_length)
#new_y = sp.interpolate.interp1d(x, y, kind='cubic')(new_x)
#new_y = interp1d(x1, y1, kind='cubic')(new_x)

print x1.min()
print x1.max()
print new_x
#print new_y
print interp1d(x1, y1, kind='cubic')(0.88)
print interp1d(x1, y1, kind='cubic')(2.88)
print interp1d(x1, y1, kind='cubic')(4.88)
new_y = interp1d(x1, y1, kind='cubic')(new_x)
plt.plot(x1, y1,'r')
plt.plot(new_x, new_y,'g')
plt.show()
```

OPI user interface screen for DAC software



Property	Value
Basic	
Macros	{Parent Macros} {}
Name	Tabbed Container
Widget Typ	Tabbed Container
Behavior	
Actions	no action
Enabled	☑ yes
Rules	no rule attached
Scripts	no script attached
Tab Count	5
Visible	☑ yes
Border	
Border Col	(0,128,255)
Border Sty	Ridged Style
Border Wi	1
Display	
Active Tab	0
Backgroun	(240,240,240)
Foregroun	(192,192,192)
Horizontal	☑ yes
Minimum	10
Tooltip	
Position	
Height	600
Scale Opti	true true true
Width	1200
X	0
Y	0
Tab 0	

```
        ... 14 more

2014-01-29 22:16:50.526 SEVERE [Thread 30] com.cosylab.epics.caj.impl.BroadcastTransport (senc
java.net.SocketException: Network is unreachable
```

1.5 MW monitoring and control screen for DAC software

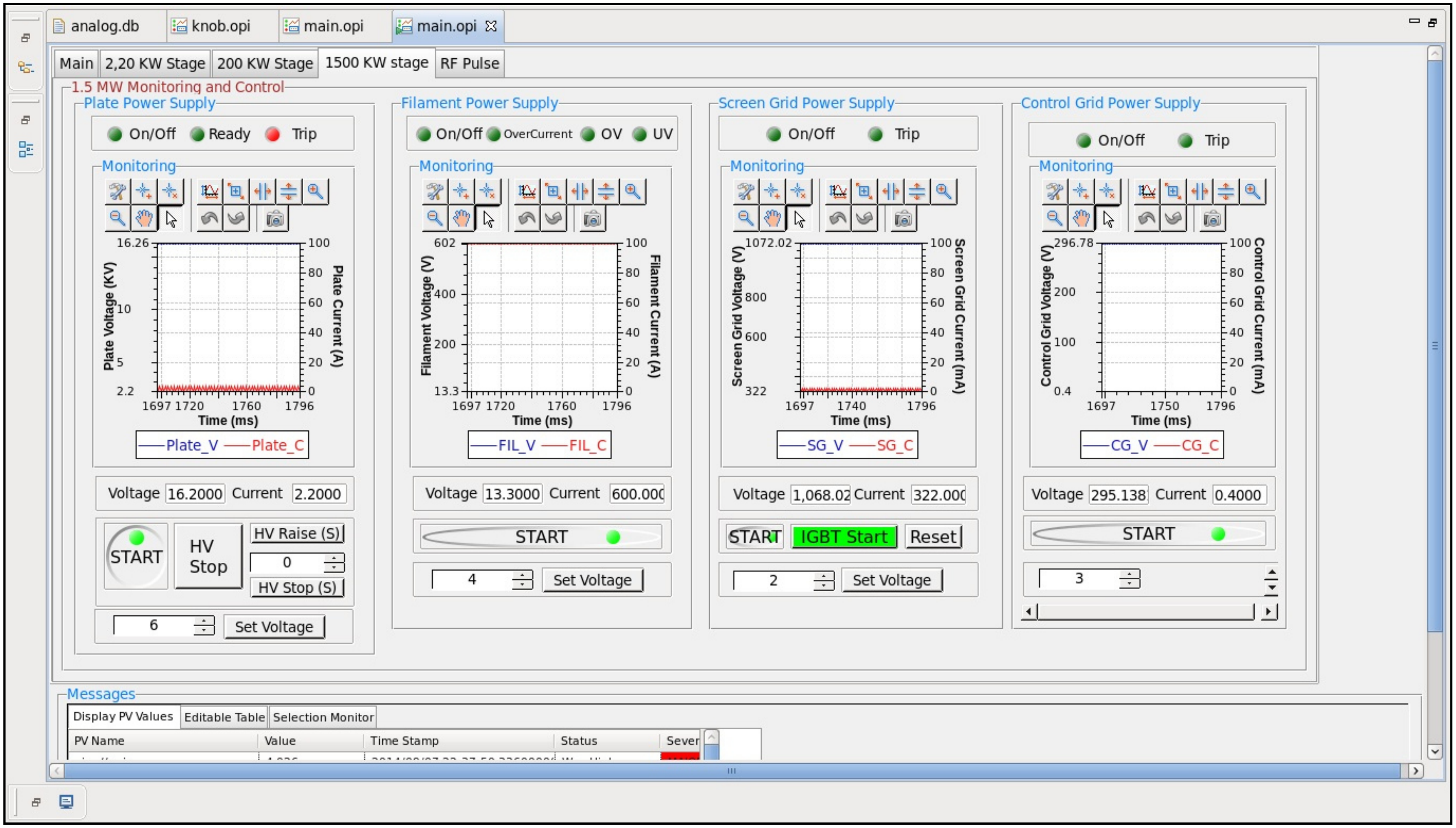


Plate Power Supply: Voltage 16.2000 Current 2.2000

HV Raise (S): 0

Set Voltage: 6

Filament Power Supply: Voltage 13.3000 Current 600.000

Set Voltage: 4

Screen Grid Power Supply: Voltage 1,068.02 Current 322.000

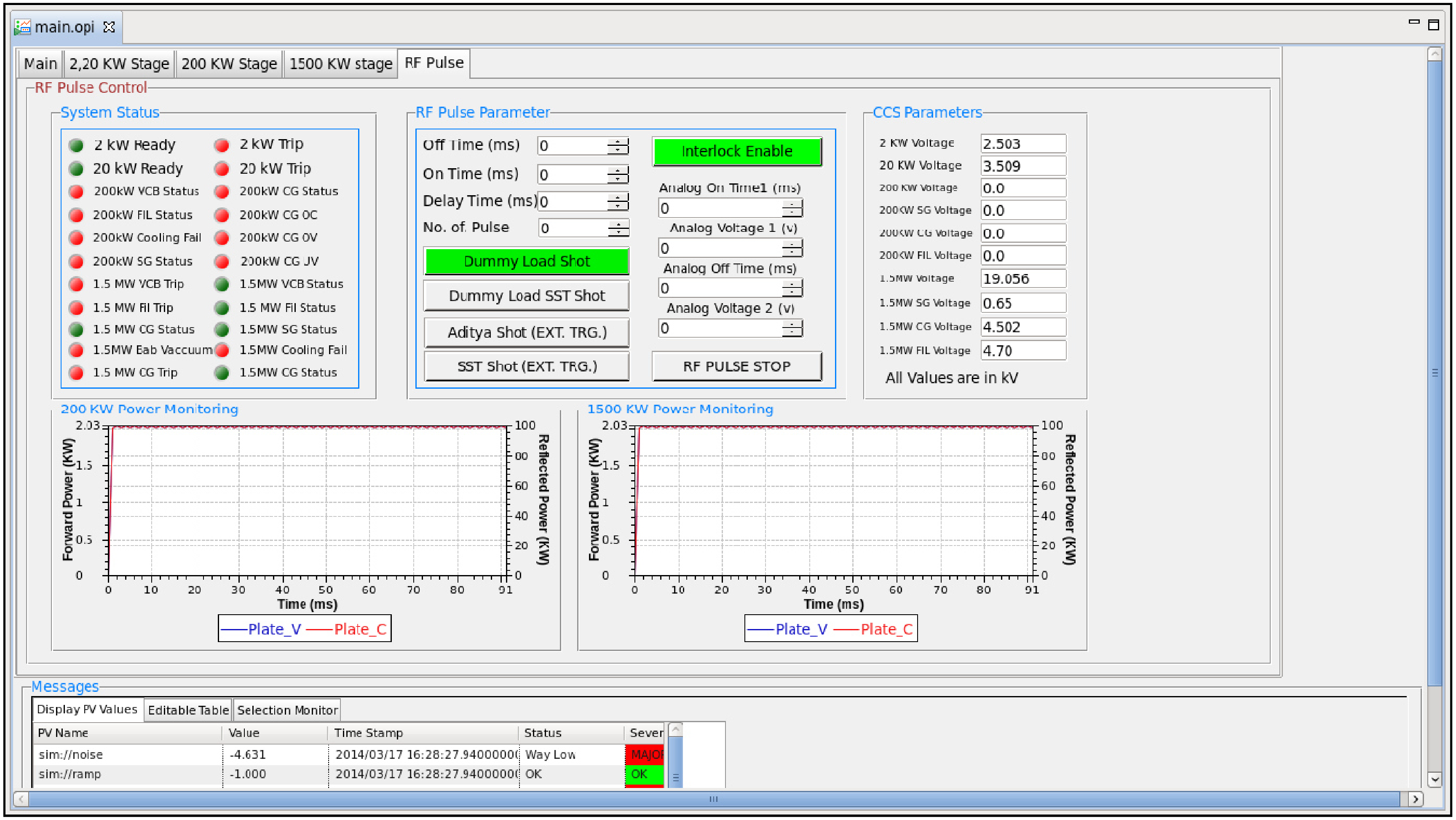
Set Voltage: 2

Control Grid Power Supply: Voltage 295.138 Current 0.4000

Set Voltage: 3

PV Name	Value	Time Stamp	Status	Sever

Experimental shot panel for DAC software



System Status

● 2 kW Ready	● 2 kW Trip
● 20 kW Ready	● 20 kW Trip
● 200kW VCB Status	● 200kW CG Status
● 200kW FIL Status	● 200kW CG OC
● 200kW Cooling Fail	● 200kW CG OV
● 200kW SG Status	● 200kW CG UV
● 1.5 MW VCB Trip	● 1.5MW VCB Status
● 1.5 MW Fil Trip	● 1.5 MW Fil Status
● 1.5 MW CG Status	● 1.5MW SG Status
● 1.5MW Eab Vaccuum	● 1.5MW Cooling Fail
● 1.5 MW CG Trip	● 1.5MW CG Status

RF Pulse Parameter

Off Time (ms)	0
On Time (ms)	0
Delay Time (ms)	0
No. of. Pulse	0
Analog On Time1 (ms)	0
Analog Voltage 1 (v)	0
Analog Off Time (ms)	0
Analog Voltage 2 (v)	0

CCS Parameters

2 KW Voltage	2.503
20 KW Voltage	3.509
200 KW Voltage	0.0
200KW SG Voltage	0.0
200KW CG Voltage	0.0
200KW FIL Voltage	0.0
1.5MW Voltage	19.056
1.5MW SG Voltage	0.65
1.5MW CG Voltage	4.502
1.5MW FIL Voltage	4.70

All Values are in kV

PV Name	Value	Time Stamp	Status	Sever
sim://noise	-4.631	2014/03/17 16:28:27.9400000	Way Low	MAJOR
sim://ramp	-1.000	2014/03/17 16:28:27.9400000	OK	OK

Summary

- ❑ EPICS and Extended modules have been compiled with Linux 32 bit machine as host computer.
- ❑ 1000 process variables have been created with unique PV name which have been exported with softIOC command provided by EPICS base.
- ❑ PyEPICS and python script program has been used to run recursively at specified time delay for data monitoring and acquisition.
- ❑ Cython package gives ability to make C library as shared object which functionality can be called using Python program.
- ❑ Wireshark and system monitor utilities have been used to check system parameters and validation with benchmarking.
- ❑ CSS OPI (user interface) can be run from any client computer which gets parameter and display data on periodic time scale.

References

1. http://www.internet-of-things-research.eu/pdf/IERC_Cluster_Book_2012_WEB.pdf
2. Piero Fraternali, Gustavo Rossi, Fernando Sánchez-Figueroa, "Rich Internet Applications", IEEE Internet Computing, vol.14, no. 3, pp. 9-12, May/June 2010, doi:10.1109/MIC.2010.76
3. Delta AH-500 PLC module reference:
4. http://www.ferret.com.au/ODIN/PDF/Showcases/105517.pdf
5. http://www.aps.anl.gov/epics/
6. Pymodbus Python Package, https://pypi.python.org/pypi/pymodbus.
7. Pyepics Python Package, cars9.uchicago.edu/software/python/pyepics3/
8. Ramesh Joshi et al. Conceptual design of EPICS based implementation for ICRH DAC system, Conference preceding of SocProS 2013, IIT Roorkee
9. http://www.alexandra.dk/uk/services/publications/documents/iot_comic_book.pdf
10. http://en.wikipedia.org/wiki/Smart_city#Examples_of_use
11. http://sydney.edu.au/health-sciences/research/healthinformatics/projects/internet-of-things.pdf
12. http://www.theguardian.com/local-governmentnetwork/2011/aug/18/internet-of-things-local government
13. http://en.openei.org/apps/FRED/web/#layer
14. http://www.ourplanetaryskin.org
15. X. Chen, K. Kasemir, “BOY, a modern graphical operator interface editor and runtime”. Proceedings of 2011 Particle Accelerator Conference, New York, NY, USA.
16. http://eclipse.org/rap/
17. http://tools.ietf.org/html/rfc1122#section-1.1.3
18. http://mqtt.org/
19. http://xmpp.org/
20. http://m2m.eclipse.org/
21. http://www.iot-a.eu/public
22. http://en.wikipedia.org/wiki/Open-source_hardware
23. http://semanticweb.org/wiki/Main_Page
24. http://stephendnicholas.com/archives/1217
25. http://www.ohwr.org/projects/cernohl/wiki

Thanks