A GENERAL DEVICE DRIVER SIMULATOR TO HELP COMPARE REAL TIME CONTROL SYSTEMS

M.Mohan, European Gravitational Observatory, Cascina (Pisa) , Italy.

Abstract

***S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition systems (**SCADA**) such as *Epics*, *Tango* and *Tine* usually provide small example device driver programs for testing or to help users get started, however they differ between systems making it hard to compare the SCADA. To address this, a small simulator driver was created which emulates signals and errors similar to those received from a hardware device. The simulator driver can return from one to four signals: a ramp signal, a large alarm ramp signal, an error signal and a timeout. The different signals or errors are selected using the associated software device number. The simulator driver performs similar functions to *Epic's* clockApp [1], *Tango's* TangoTest and the *Tine's* sinegenerator but the signals are independent of the SCADA. A command line application, an *Epics* server (IOC), a *Tango* device server, and a *Tine* server (FEC) were created and linked with the simulator driver. In each case the software device numbers were equated to a dummy device. Using the servers it was possible to compare how each SCADA behaved against the same repeatable signals. In addition to comparing and testing the SCADA the finished servers proved useful as templates for real hardware device drivers.

INTRODUCTION

The current vacuum system at EGO [2] is controlled by os9 crates and uses custom software. The software is used to monitor values such as temperature, pressure and displacement. This control system is now being updated for Advanced Virgo. The os9 crates will be replaced by PLCs and a new SCADA will replace the custom software. The SCADA chosen may also be used for other systems at EGO.

Integration of new devices into a SCADA requires the hardware be setup correctly and the protocol (such as modbus) be implemented correctly. Rather than setup a PLC and test different conditions a small library was written to simulate signals. The simulator library generates simple scalar signals and errors similar to those which could be generated by a PLC.

Epics, Tango and Tine SCADA

Epics, tango and tine SCADA were chosen for comparison as they are all stable mature systems which are actively supported and developed. They are all open source projects which run on multiple platforms. The preferred platform at EGO is linux and both epics and tango have been recently packaged for Debian linux using the aptitude [3] package manager which eases installation and updating.

Epics is the most popular and widely supported SCADA followed by tango and then tine. Larger support implies stability [4] and a larger supply of pre-written device servers whereas smaller projects tend to be more flexible and use more advanced techniques.

The architectures of the SCADA investigated are similar and a generalised SCADA view is illustrated in Figure 1. Information from devices such as PLCs D1, D2, D3, and D4 is packaged and put on "software bus" by a Device Server. Different tools under the control of an operator are used to monitor and control the devices. Simulator signals were used in place of signals from D1, D2, D3 and D4.

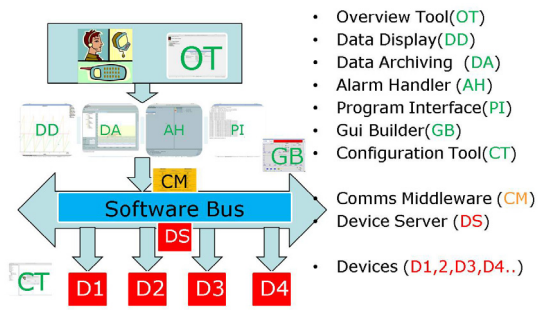


Figure 1: SCADA requirements.

SIMULATOR DEVICE SIGNALS

Four generated signals which are named according to the type of signal returned are shown in Table 1.

Table 1: Simulator Device Names

Device Number	Device Name
1	test/simulator/ramp
2	test/simulator/rampx2
3 (default)	test/simulator/error
4	test/simulator/timeout

The test signal is selected using the Device Number in the same way a modbus value is read using its modbus address. The signals are…

1. ramp: A ramp which rises from 0 to 59.999999 every minute.
2. rampx2: A large ramp which rises from 0 to 119.999998 every minute. Used to test alarms.

3. error: An error signal set at 60. Used to test how the SCADA reacts if the PLC returns a fault (-1).
4. timeout: A timeout signal. Used to emulate when there is no response from a PLC.

The signals are shown in figures 2, 3 and 4 using epics, tango and tine data displays.

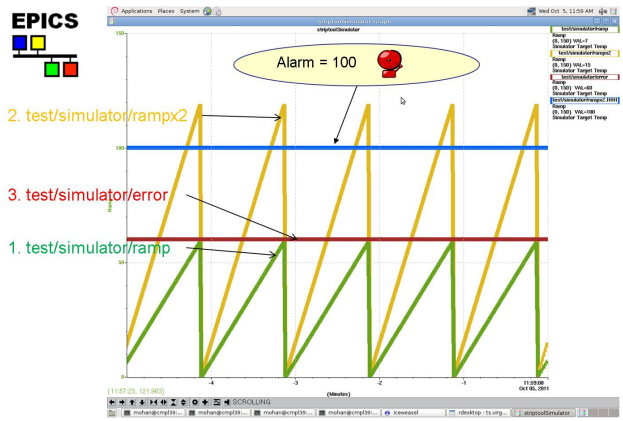


Figure 2: Simulator Signals using epics striptool.

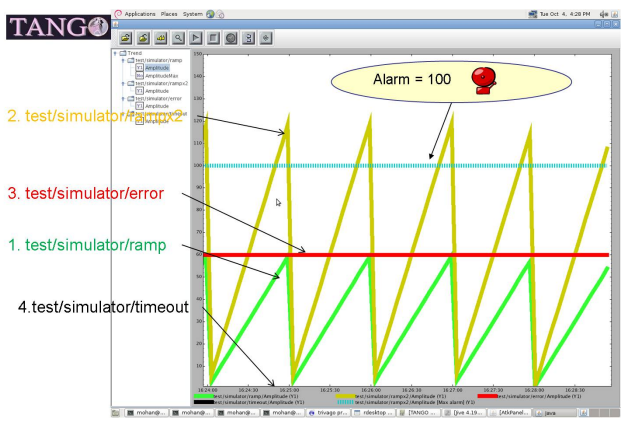


Figure 3: Simulator Signals using tango atkmoni.

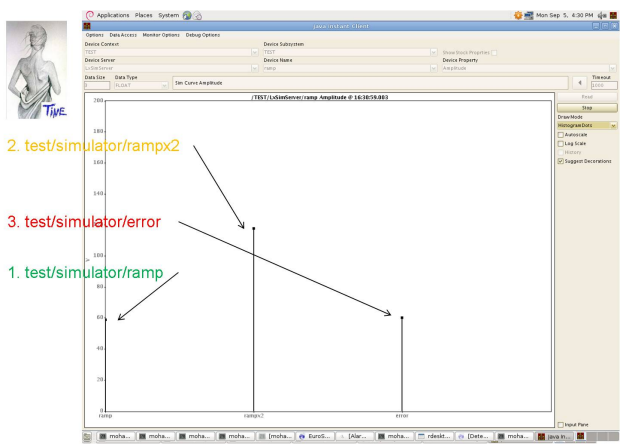


Figure 4: Simulator Signals using tine JClient. "Histogram view". There are several view types in Tine.

How Simulator Signals are Generated

A small c program was used to generate the four signals shown in Figure 2, 3 and 4. The function to generate the signals is shown in Figure 5.

```
…
int GetAmplitude(int DeviceNumber, double *Amplitude)
{
      struct timeval  tv; struct timezone tz;
      struct tm      *tm;
      gettimeofday(&tv, &tz); tm = localtime(&tv.tv_sec);

      // *Amplitude= Seconds time of system clock
      *Amplitude=((double)((1000000*tm-
>tm_sec)+tv.tv_usec)/1000000); //  0-59.999999

      int ErrorValue =0; // default 0=ok

      switch (DeviceNumber)
      {
            case 1: // 0-59.999999 = ramp
                  break;
            case 2: // 0-119.999998 = rampx2
                  *Amplitude=*Amplitude*2;
                  break;
            case 4: // =timeout
                  sleep(99999999);
                  break;
            default: // =error
                  *Amplitude=60;
                  ErrorValue=-1; //  -1 = Error
                  break;
      }
      return ErrorValue; // return 0 or -1
}
…
```

Figure 5: C function to generate Simulator signals.

A library called "libSimulator" was made using the code in Figure 5 and linked with SCADA device servers.

CREATING DEVICE SERVERS

What is a Device?

The concept of the device is at the heart of epics, tango and tine. A device can be considered a container of hardware data and an analogy for this is a shipping container because the devices all look the same although the contents are different. Like devices shipping containers also have unique names [3].

The "device container" in Figure 6 is labelled with static information such as the name of the device, units of measurement and alarm limits. The device is packaged by the device server which updates dynamic information in the device. When the device server is running the device is updated with dynamic content such as Amplitude, state and timestamp. This "device container" is then put on its

respective software bus (epics, tango or tine) where it is identified by its unique name.

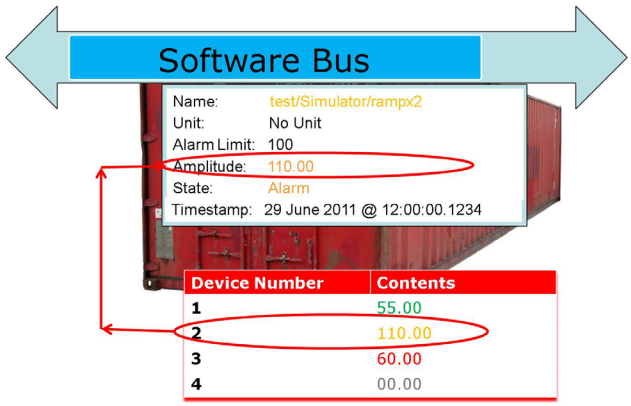


Figure 6: Example: Device test/simulator/rampx2 is updated periodically and results put on software bus.

Integrating Devices in Device Servers

The source code for the software device servers were generated using the template tools makeBaseApp, pogo and srvwizard which are supplied by epics, tango and tine respectively.

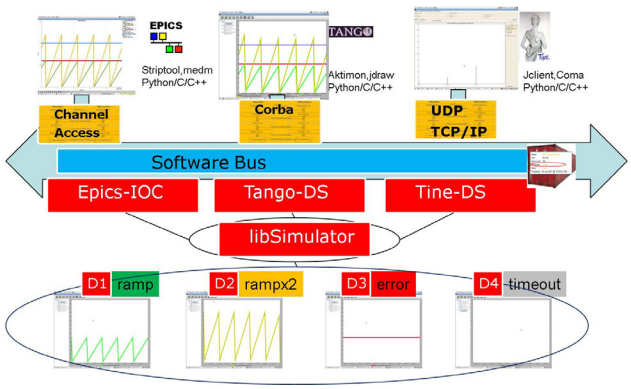


Figure 7: libSimulator test setup.

The device servers were linked to the libSimulator library Figure 7 and device signals generated by calling the function GetAmplitude Figure 5. The code added to device servers to read the devices is similar to the following.

C Command line

```
if(GetAmplitude(DeviceNumber,&Amplitude)==-1)
printf("Error\n");
```

Epics

```
if(GetAmplitude(DeviceNumber,&Amplitude)==-1)
recGblSetSevr(pai,STATE_ALARM,MAJOR_ALARM);
```

Tango

```
if(GetAmplitude(DeviceNumber,&Amplitude)==-1)
set_state(Tango::FAULT);
```

Tine

```
if(GetAmplitude(DeviceNumber,&Amplitude)==-1)
SetAlarm(SIMEQM_TAG,devnr,512,NULL);
```

The codes above read the Amplitude and sets an alarm if an ErrorValue (-1) is returned. This means the SCADA always set an error alarm for device 3 (test/simulator/error).

The default code generated for epics and tine hung on the timeout from device 4 (test/simulator/timeout) because the device server blocks waiting for a reply. Tango however by default sets an alarm state "timeout" and continued to read the other devices if no reply came within 3000 ms. There are several methods to overcome the timeout in epics and tine such as defining each device as thread or using epics async but this involves modifying the default generated code.

For real devices pre-written device servers can be downloaded from the SCADA websites and some hardware devices even come with inbuilt device servers such as the F3RP61 PLC from yokogawa ref [6] which includes an inbuilt epics IOC.

CONFIGURING DEVICES

SCADA configuration tools are used to input static information for devices such as names and alarm thresholds. For the simulator the Device number 1, 2, 3 or 4 must be associated with a unique device name. Epics naming is not compulsory and tango and tine use similar formats so it was possible to use the same Simulator signals names in all SCADA see Table1.

Epics

Configuration information for epics devices are stored in ASCII ".db" files and modified using a text editor. This device information is called a record. In the example in Figure 8 the field HIHI is used to set max alarm threshold to 100.

```
record(ai, "test/simulator/rampx2")
{
      field(DESC, "Signal ramp 0 to 119.999998 per
minute")
      field(INP,"2")
      field(HIHI, "100.0")
..
}
..
```

Figure 8: Epics configuration using .db file

Tango

Configuration information for tango devices are stored in a Mysql database and configured using the tango tool

jive. In the example in Figure 9 Max alarm for "test/simulator/rampx2" is set to 100 for the Amplitude attribute.

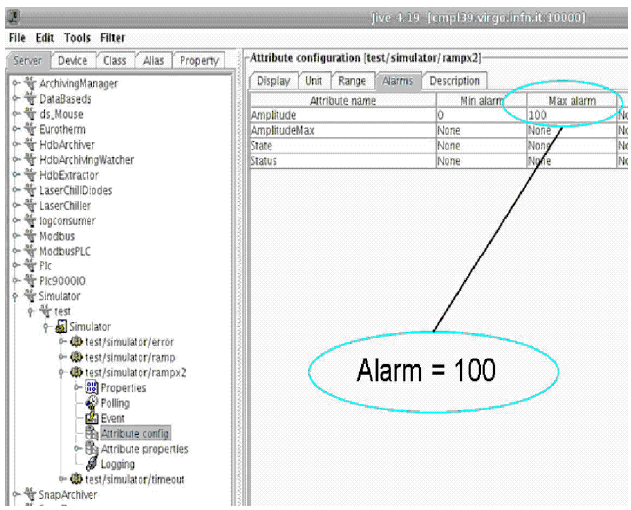
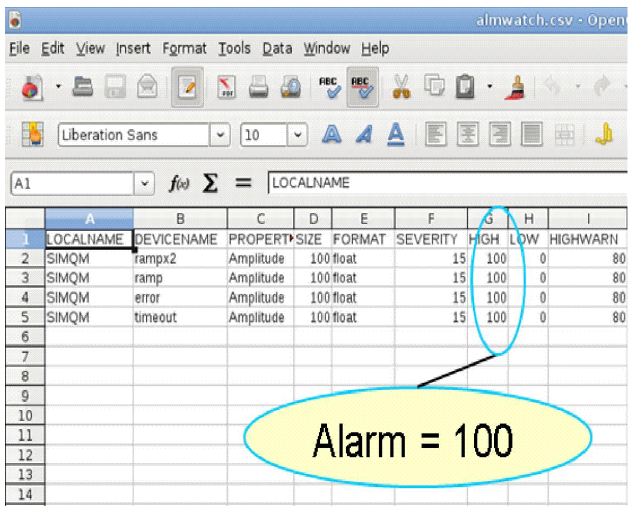


Figure 9: Tango configuration using jive.

Tine

Configuration information for tango devices are stored in a csv files and configured using spreadsheet tools or a text editor. Tine also provides the alternative options of storing configuration information in xml files or in the code.

In the example in Figure 10 High alarm for "test/simulator/rampx2" devices is set to 100.



	A	B	C	D	E	F	G	H	I
1	LOCALNAME	DEVICENAME	PROPERT	SIZE	FORMAT	SEVERITY	HIGH	LOW	HIGHWARN
2	SIMQM	rampx2	Amplitude	100	float	15	100	0	80
3	SIMQM	ramp	Amplitude	100	float	15	100	0	80
4	SIMQM	error	Amplitude	100	float	15	100	0	80
5	SIMQM	timeout	Amplitude	100	float	15	100	0	80
6									
7									
8									
9									
10									
11									
12									
13									
14									

Figure 10: Tine configuration using an excel tool.

In the above configuration cases setting the alarm to 100 successfully caused the SCADA's Alarm handlers (or state handler for tango) to trigger a warning alarm when the property Amplitude exceeded 100 which happened for device rampx2 every 50 seconds.

SCADA TOOLS

After device information had been packaged and configured correctly they are put on the respective "software bus" and made available for the epics, tango or tine tools.

Some tools to view data online in epics, tango and tine are shown in figures 2, 3 and 4. These data display tools are called striptool, atkmoni and JClient respectively.

There are many other monitoring and control tools for epics, tango and tine which perform functions such as Data Display, Alarm Handling, Data Archiving and gui Building (Figure 1). The tools have many similarities and in fact the gui builder tool jddd [7] can be used to build guis for epics, tango and tine. In all cases a unique device name is used to identify the device. Additional tool comparisons can be found in reference [8].

CONCLUSION

Epics, tango and tine are all able to fulfil the requirements for the vacuum control at EGO and more. The learning curve for writing and configuring device servers is high but once mastered devices can be integrated quite quickly. There are many pre-written device servers available especially for the more popular SCADA. These pre-written device servers can be freely downloaded although they must still be configured and tested.

The Simulator is a simple aid to generate repeatable signals independently of the SCADA or hardware. It was useful for learning to write simple device servers for epics, tango and tine and for comparing SCADA tools using known signals.

All the SCADA provide useful tools for monitoring and controlling devices Figure 1. The tools are similar and some tools such as jddd [7] can even be used with all epics, tango and tine.

REFERENCES

[1] F.Furukawa,"Very Simple Example of EPICS Device Suport", http://www-linac.kek.jp/epics/second
[2] http://www.ego-gw.it/public/about/whatIs.aspx
[3] http://www.isbu-info.org/all_about_shipping_containers.html
[4] Raymond, Eric S.,"The Cathedral and the Bazaar", October1999, http://www.catb.org/~esr/writings/cathedral-bazaar/
[5] http://en.wikipedia.org/wiki/Intermodal_container
[6] A. Uchiyama… Proceedings of PCaPAC08, Ljubljana, Slovenia, Development of embedded epics on F3RP61-2L.
[7] http://jddd.desy.de/
[8] M.Mohan, VIR-0371A-11.pptx, "Advanced Virgo Vacuum control Epics, Tango and Tine SCADA comparison", https://tds.ego-gw.it/ql/?c=8426