Managing detectors (and detector systems) with the Experimental Physics and Industrial Control System (EPICS)

S. Ivanov
S. Ivanov
Sofia University
FIRST SUMMER SCHOOL ON SPACE RESEARCH,
TECHNOLOGY AND APPLICATIONS, 2021

Detector Control Software Philosophy

The wrong way (most of the time):

Whip up your own code: device drivers, processing, communication protocols, GUI

Spend inordinate amount of time debugging instead of doing something useful

Get a third of the features you need working half the time (if you are lucky)

Do it again from the start for your next project

The right way:

Use a standard system (hopefully feature-rich, free and open source)

Read the docs, figure out how it works, map your needs onto the package features

Maybe write your drivers

Use it for all your projects

EPICS – Experimental Physics and Industrial Control System

An excellent choice for automation and control

Compact – the current stable release is less than 20MB

Versatile – can be applied to run accelerator, or your grandmother pills alarm system

Complete – from low-level drivers to GUI to data collection

Secure – if you understand and apply the security model properly

Easy – especially compared to using the code you wrote 3 weeks ago

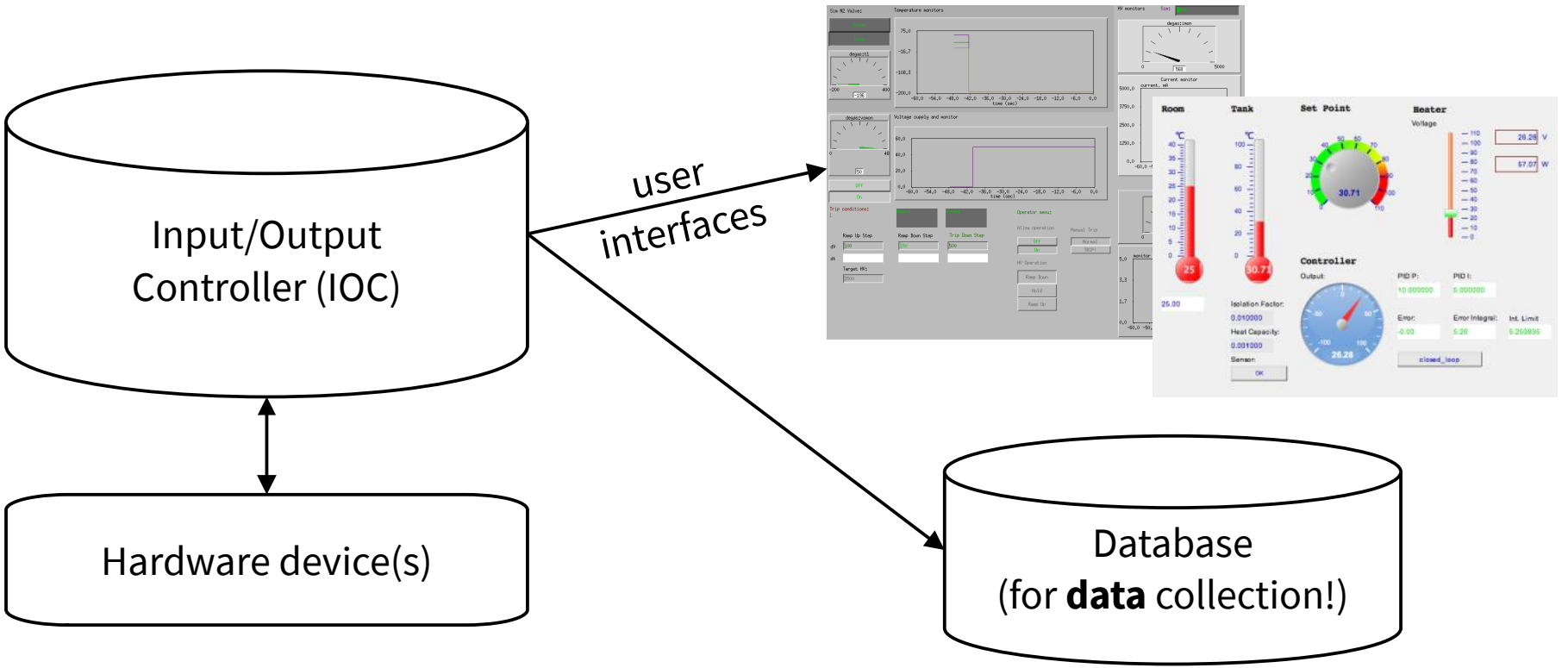
Efficient – runs with very small load on most existing processor architectures

beaglebone, raspberry pie, orange pi, the nvidia AI SoC modules, your nearby CAMAC or Caen crate

Portable – runs on every OS you are likely to use (and some that are long forgotten now)

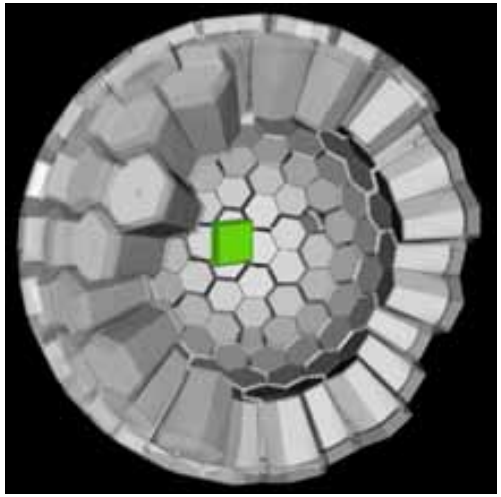
windows, all unix flavors (aix, hpux, linux, darwin, *bsd, vax), ios, android and probably minix (but check that yourself)

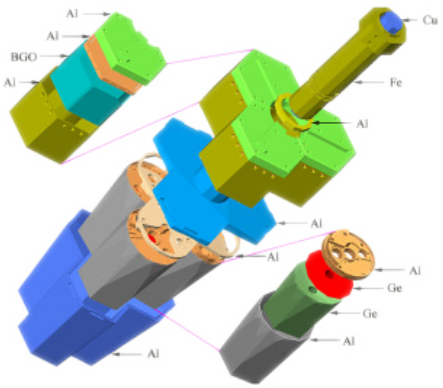
EPICS Architecture

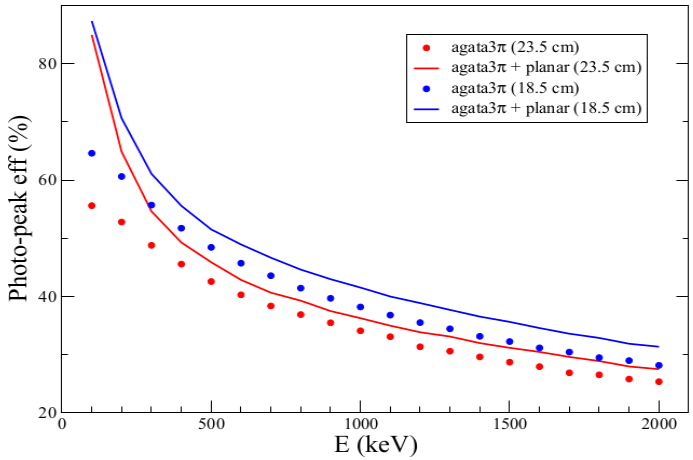


Example: controlling a real gamma ray detector

DEGAS gamma ray detector – a gamma ray detector for the DESPEC experiment at FAIR. Read the full story here: DOI 10.1016/j.nima.2017.04.019







Slow control components

To control a device, you need a detailed specification of its parameters...

Control targets:

- Three platinum PT100 sensors read out with high precision ADCs, using I^2C protocol
- High voltage control (0 to 6kV), set with a high precision DAC, using I^2C protocol
- Monitors for the high voltage and the electric current through the detector
- Monitors for the detector power supply
- Trip alarms (that can power down the device in an emergency, like coolant loss or power failure)

Also required:

- Configuration management (save/restore settings)
- Network management interface
- Graphical User Interface

Actual Device Management

IOC – epics running on beaglebone AI

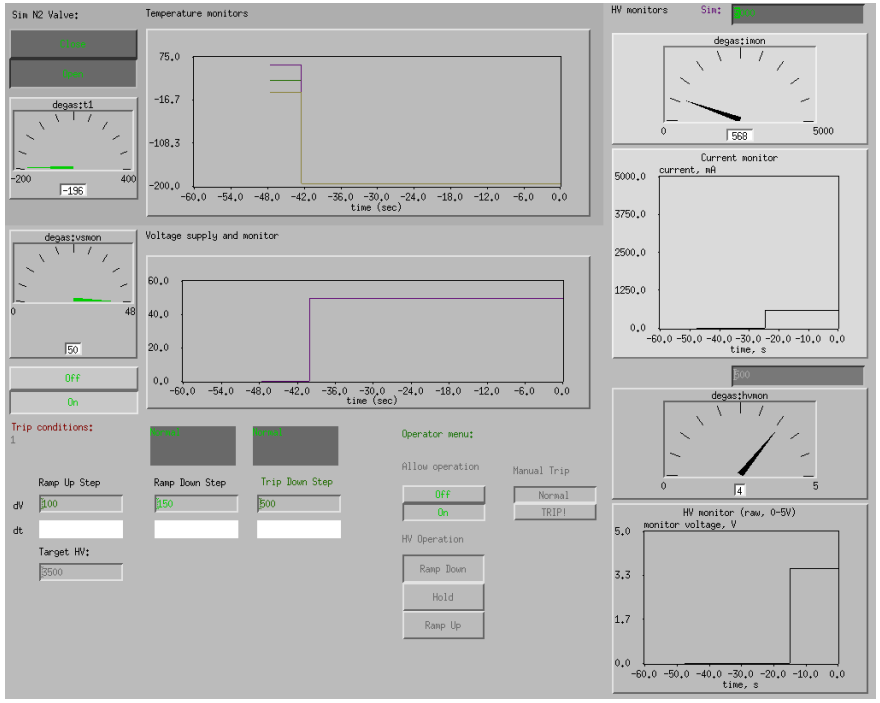
Controls the hardware, much like you control your temperature sensors with your Arduino (except that we designed and built all the hardware as well)

Collects and sends status and operation parameters

State machine handles not only normal operation, but also alarms, emergencies and manages configuration

GUI – medm, running on a workstation

Allows an operator to control a whole bunch of detectors



Ciao, bambino, sorry!