Status of PANDA DCS Activities in Magurele Part II

Alexandu Mario BRAGADIREANU, Dorel PIETREANU, Matei-Eugen VASILE

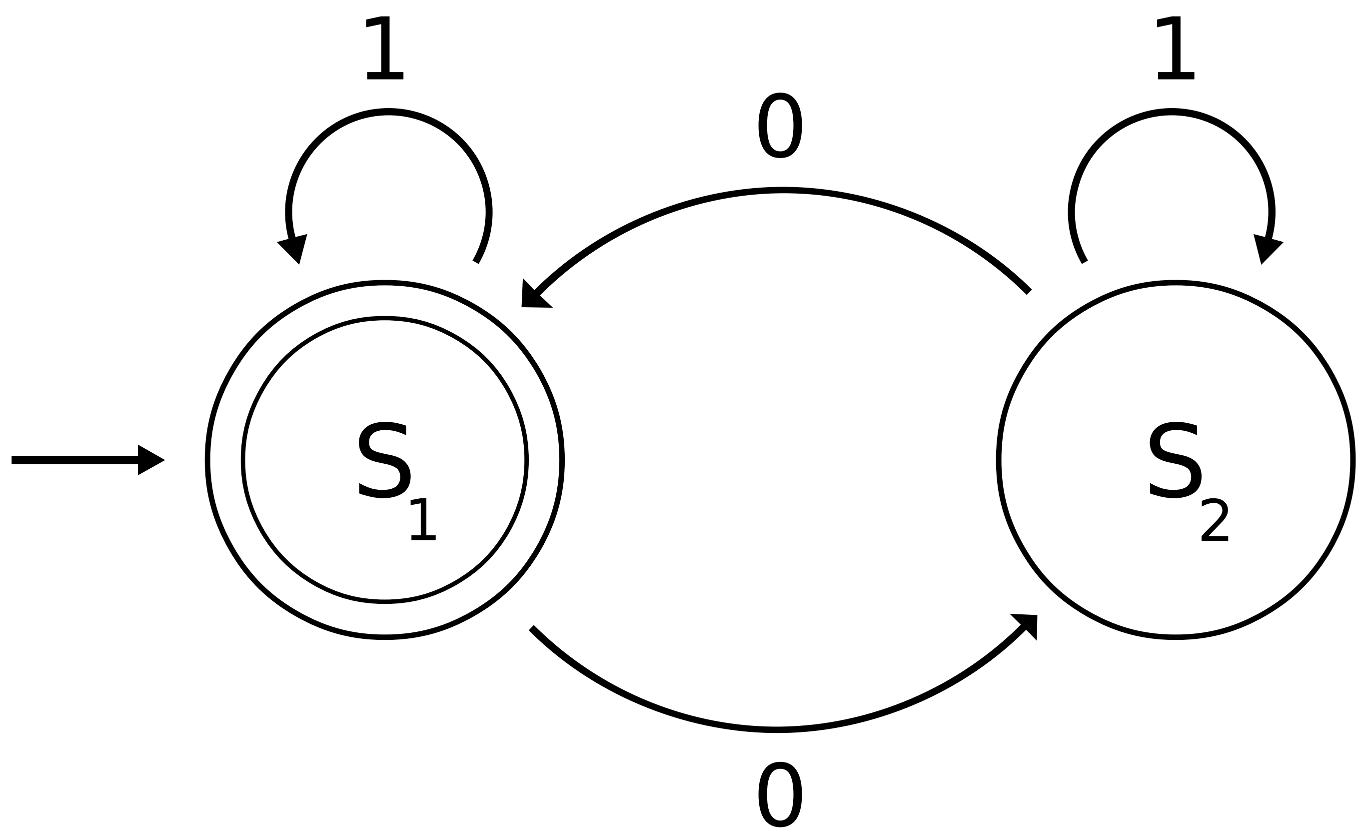
National Institute for Physics and Nuclear Engineering – Horia Hulubei

Finite State Machines

Finite State Machines

- In control engineering, a **discrete event dynamic system** is a discrete state dynamic system whose state evolution depends entirely on the occurrence of asynchronous discrete events.
- A **finite state machine** is a discrete event system that can be formally represented by a 5-tuple (Q,Σ,δ,q0,F), where:
  - Q is the finite set of states of the FSM
  - Σ is the finite set of symbols that make up the alphabet of the FSM
  - δ is the transition function of the FSM: $\delta: Q \times \Sigma \rightarrow Q$
  - q0 is the initial state of the FSM
  - F is s a set of states of Q (i.e. $F \subseteq Q$) called **accept states**

Finite State Machines

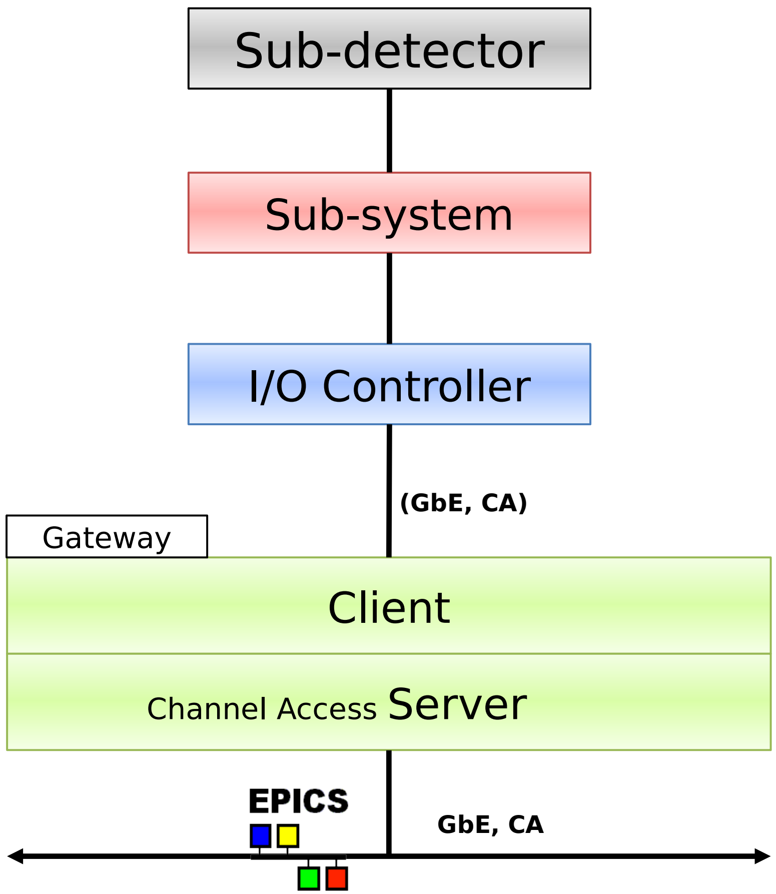


Finite State Machines - Example

- One example of using finite state machines in physics research is the **EPICS State Notation Language & Sequencer** (http://www-csr.bessy.de/control/SoftDist/sequencer/)
- The **State Notation Language** is a domain specific programming language "designed for programming finite state machines in such a way that it is easy for the program to interact with EPICS process variables (PVs)"
- The **Sequencer** is a set of tools, libraries and applications that can be used to create distributed real-time control systems and which is based on the State Notation Language

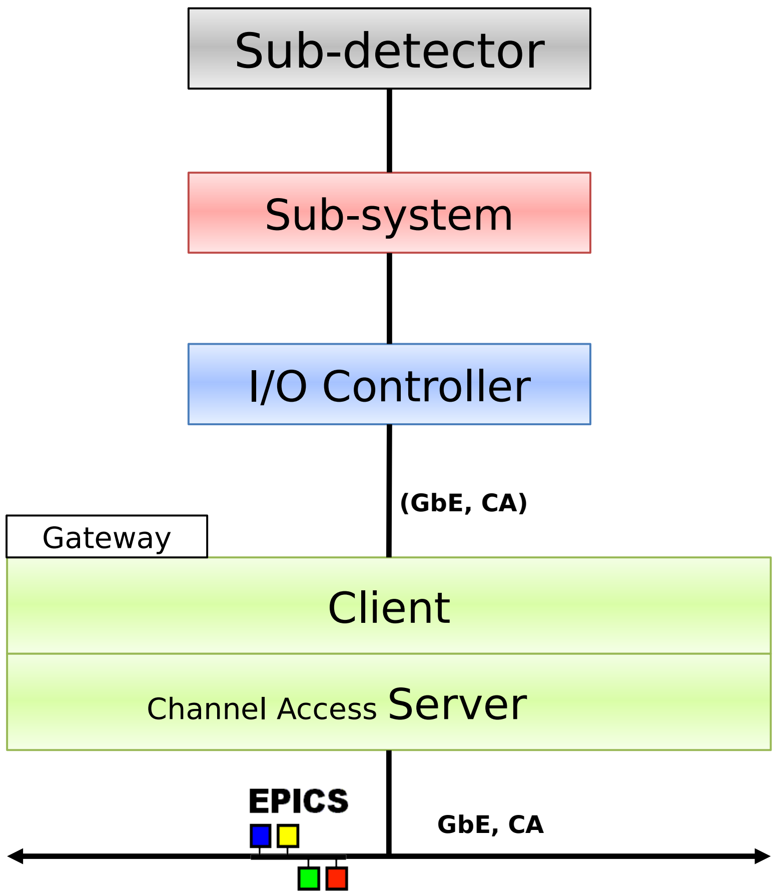
DCS Developments

DCS Subdetector System Architecture



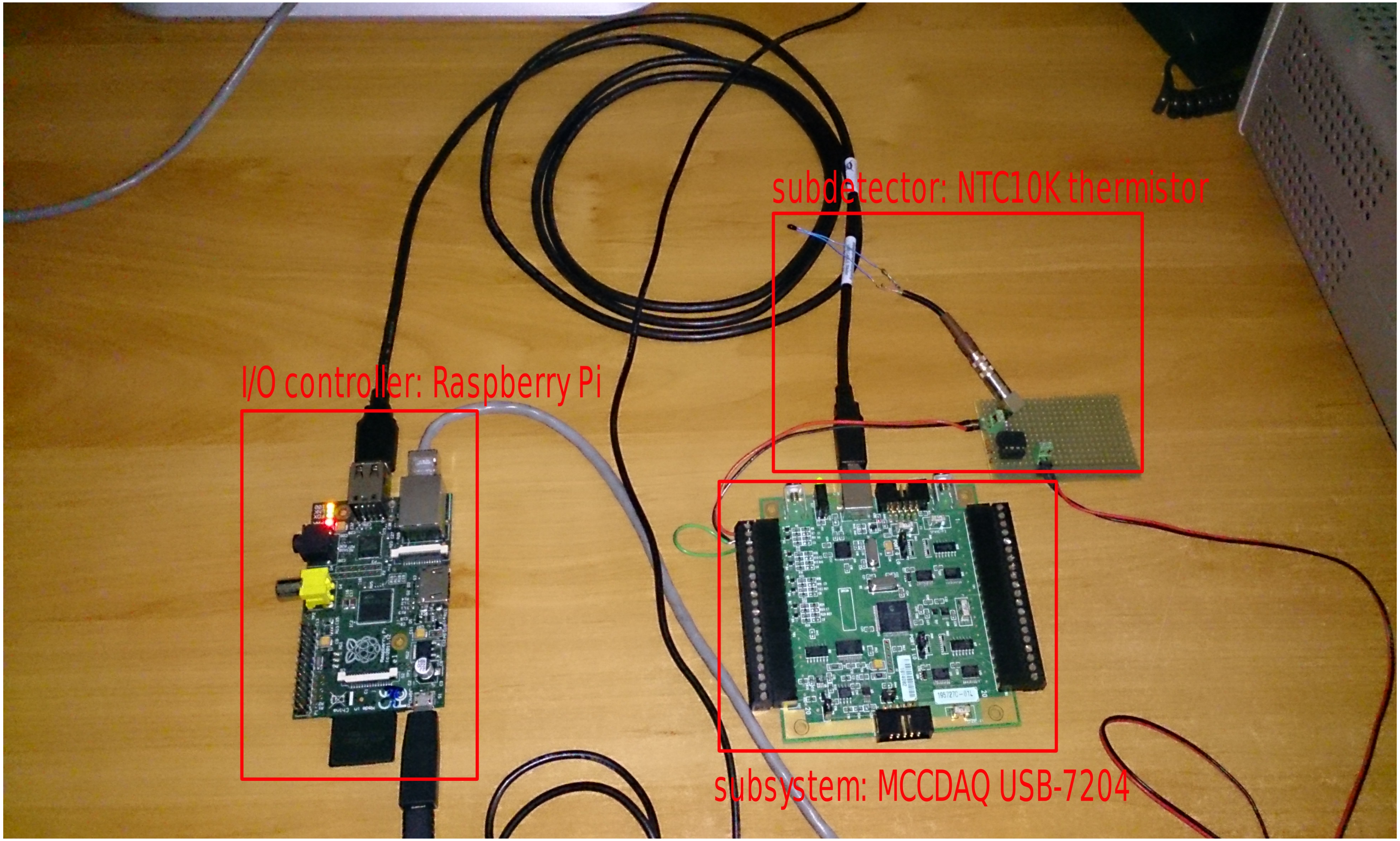
- A DCS partition is made up of:
  - Subdetector (served by all the subsystems of a DCS partition)
  - Subsystems
  - I/O Controllers
  - Gateway (shared by all the I/O controllers of a DCS partition)

DCS Subdetector System Architecture



- In order to test the feasibility of this architecture, a development setup was created:
  - subdetector: a thermistor
  - subsystem: a data acquisition board
  - I/O controller: communication software and an *EPICS soft IOC* running on a single-board computer
  - gateway: *EPICS PV Gateway* extension running on a regular PC

DCS Subdetector System Architecture



DCS Subdetector System Architecture

- Development setup:
  - subdetector: *NTC10K* thermistor and its circuit board, connected to one of the analog inputs of the:
  - subsystem: *MCCDAQ USB-7204* data acquisition board, connected via USB to the:
  - I/O controller: *Raspberry Pi*, *ARMv6*-based single-board computer, running *Linux* (tested with multiple distributions: *Raspbian* and *Arch Linux*), a *libusb*-based communication server developed by IFIN-HH for interfacing with the data acquisition board, and an *EPICS soft IOC* that is accessible, over *Ethernet*, via the:
  - gateway: regular PC, running *Linux* and the *EPICS PV Gateway extension*

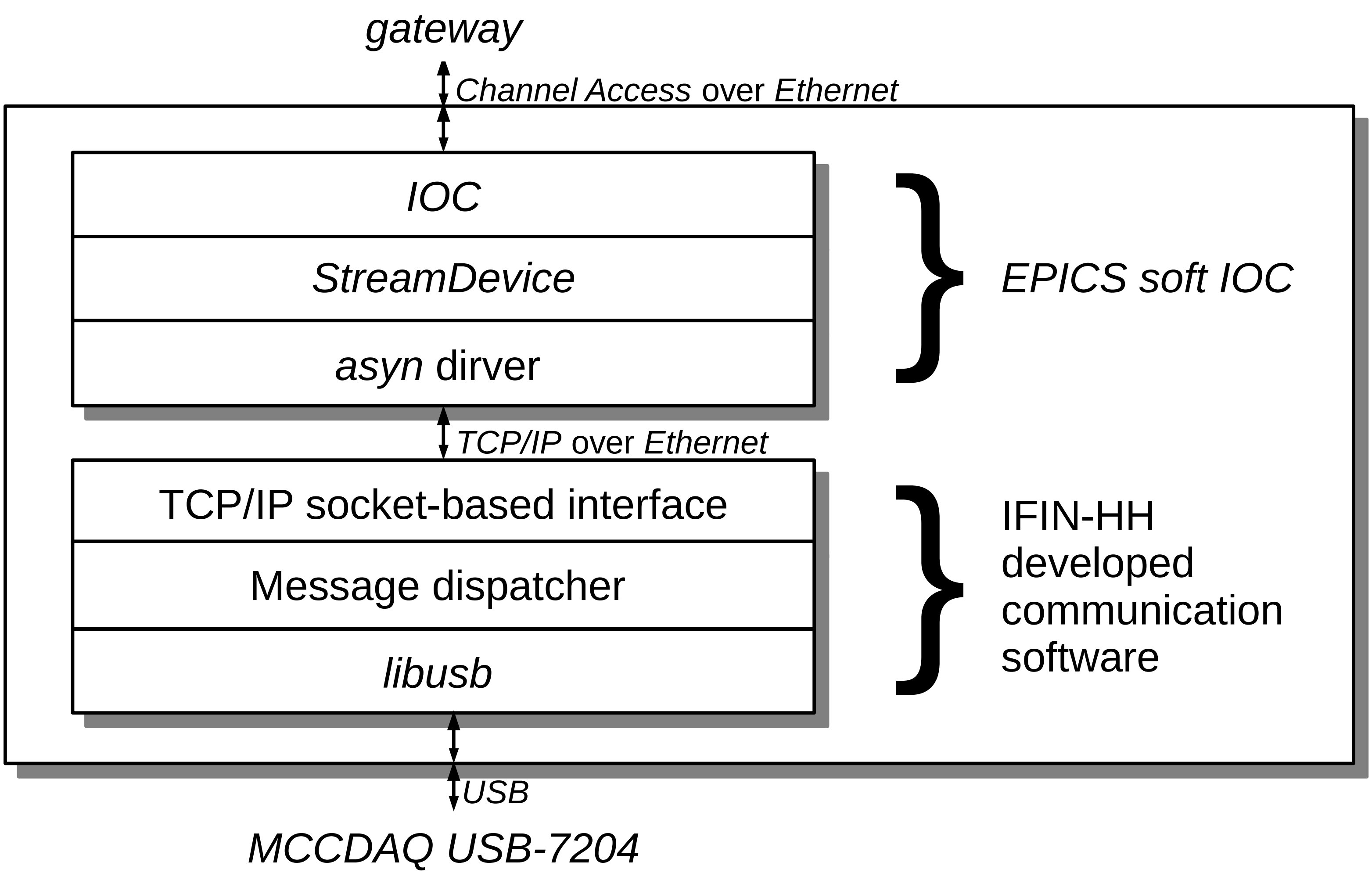
DCS Subdetector System: I/O controller

- The I/O controller runs two pieces of software:
  - The IFIN-HH developed communication software to interface with the *MCCDAQ USB-7204* data acquisition board
  - The *EPICS soft IOC* that uses the communication software to interface the *USB-7204*'s with the outside world
- Given that the *USB-7204* provides a string-based interface over USB, the communication software leverages this by using the *libsusb* library to communicate with the *USB-7204*
- Given the asynchronous nature of the *USB-7204*'s string-based interface, the *EPICS asyn* driver was a good fit for the interface between the *USB-7204* and the *EPICS soft IOC*

DCS Subdetector System: I/O controller

- However, the *asyn* driver alone wouldn't have been the best choice. In order to have a more flexible interface between the two I/O controller components, *StreamDevice* over the *asyn* driver was chosen
- The communication software communicates with the *soft IOC* over a standard TCP/IP socket. This setup has multiple advantages:
  - Allows the easy use of *StreamDevice*, which makes getting the *soft IOC* to work with the communication software much easier
  - The communication software is not intrinsically dependent on *EPICS*. It could be used with any higher level interface that can be made to communicate over TCP/IP sockets

DCS Subdetector System: I/O controller



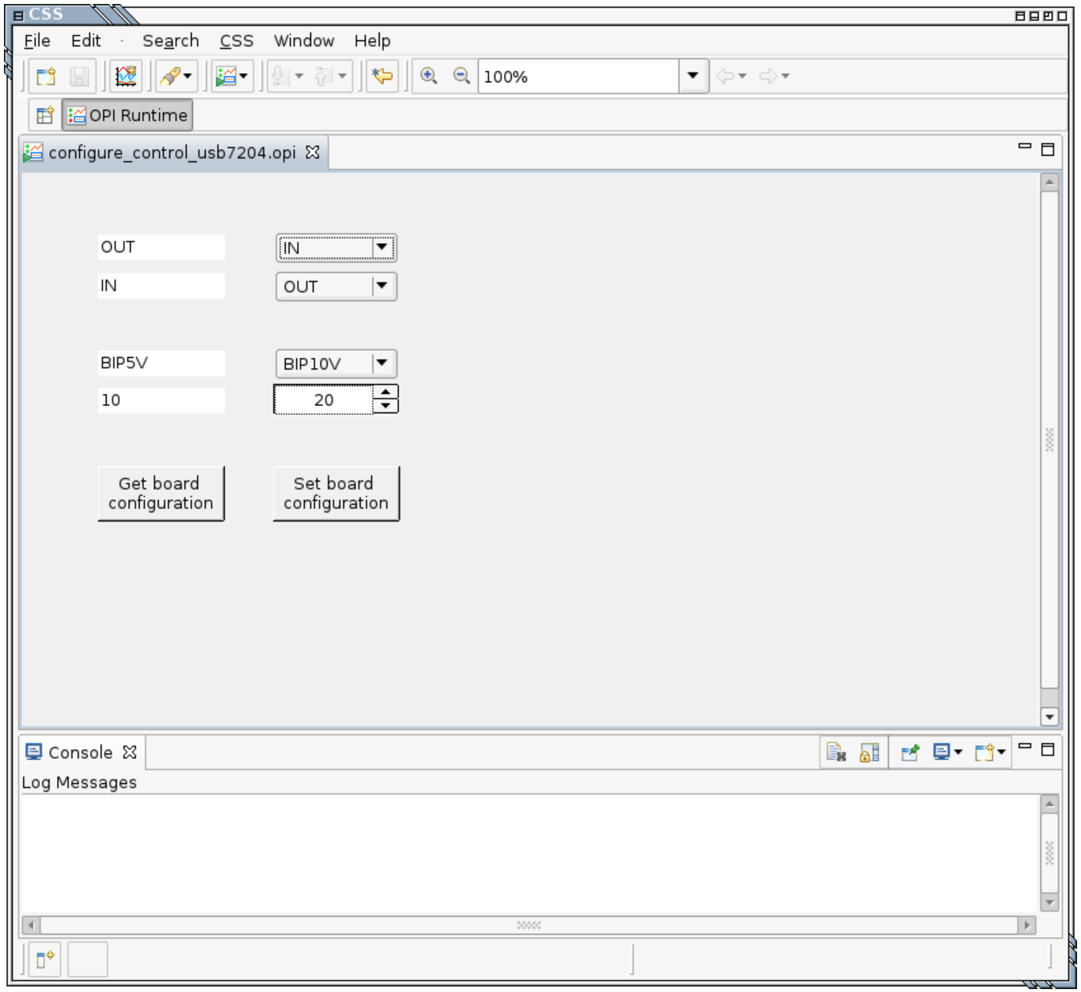
DCS Subdetector System: Gateway

- The gateway can be any kind of computer that can run the *EPICS PV Gateway* extension and has two network interfaces: one on the internal network, the one to which the I/O controller is connected as well, and one on the external network, the one to which the *EPICS* clients from the supervisory layer are connected
- The *EPICS PV Gateway* extension works as a server software that, for the I/O controller acts as an *EPICS* client and for the client devices in the supervisory layer acts as an *EPICS* server
- The *EPICS PV Gateway* can:
  - Control which process variables (PVs) are available on the supervisory layer, thus filtering access by PV
  - Control who is allowed to access which PVs, thus filtering access by end user
  - Provide PV aliases for the PVs published by the IOCs behind the gateway

DCS Supervisory Layer: CSS EPICS Client

- The subdetector system presented up to this point can be controlled, from the Supervisory Layer, by an *EPICS* client
- For this purpose, *CSS* (*Controls System Studio*) was used to develop an operator interface that can control the *MCCDAQ USB-7204* based subsystem:
  - The operator interface is built using the *BOY* (*Best OPI Yet*) *CSS* plugin
  - It has two components:
    - A configuration interface
    - A control/monitoring interface

DCS Supervisory Layer: CSS Configuration Interface



DCS Supervisory Layer: CSS Monitoring Interface