Running IOCs from ci-scripts

Torsten Bögershausen

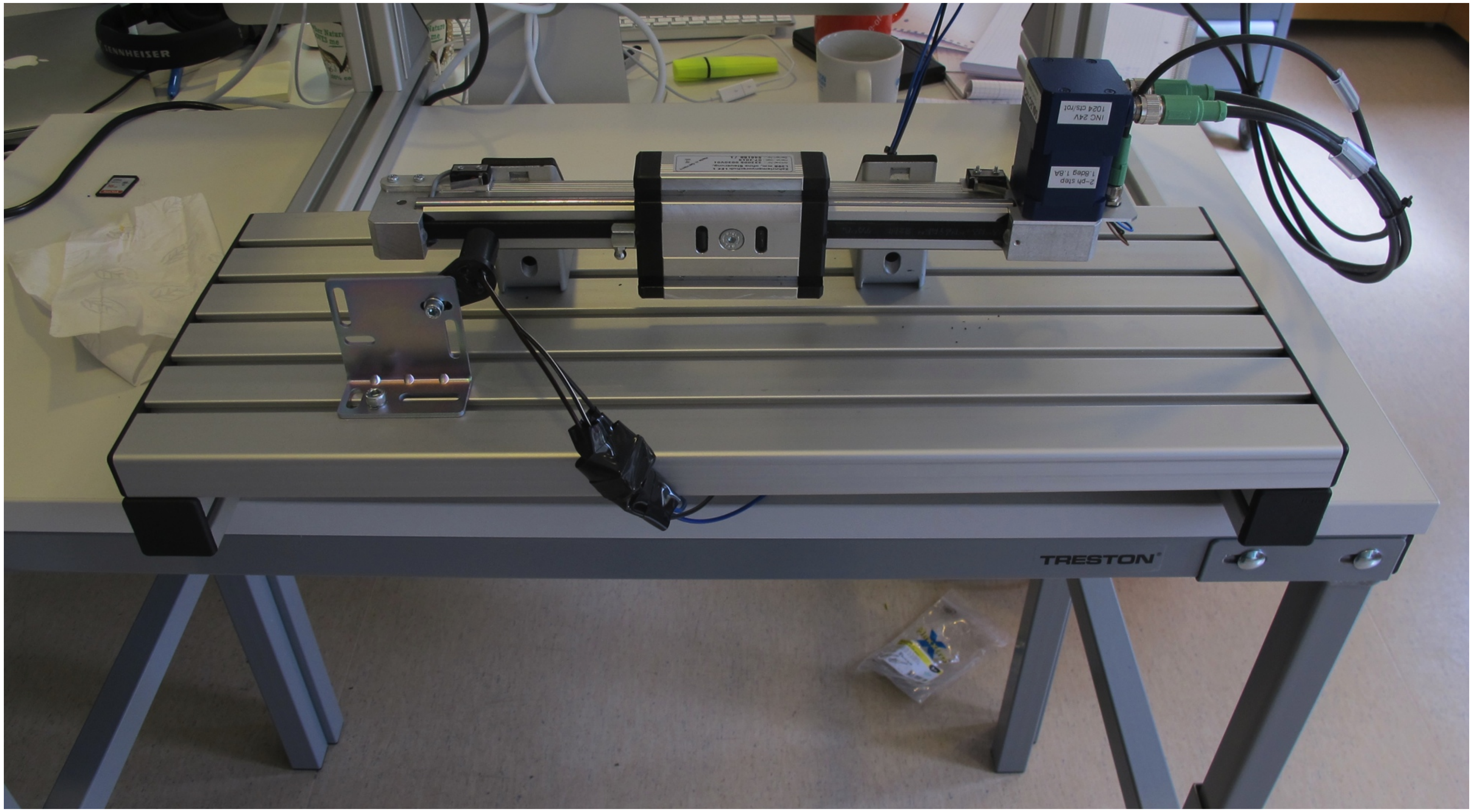
SW Engineer

www.europeanspallationsource.se

Running IOCs from ci-scripts

- After using the ci-scripts to build EPICS
  I wanted more:
  - - start a “real” IOC
    - connect to a (simulated) motion controller
    - move a (simulated) axis
    - run different test cases
  - This talk summarizes some experiences
     and is an invitation to join the test train

Single Axis



Terminal Window 1

- Simulator:
  - Speaks the same language as the
    real motion controller.
  - written in C, compiled by `make`
- How to start the simulator
  ./run-ethercatmc-simulator.sh

Terminal Window 2

- EPICS IOC:
  - compiled EPICS base, modules
  - assemble st.cmd
  - run st.cmd
  - collect log file

Terminal Window 3

- Run test cases:
  - python based: pytest
- Python modules:
  - Need pytest, numpy, pyepics, p4p
  - passed/failed
  - log file

Terminal: simulator

```
$ ./run-ethercatmc-simulator.sh
[snip]
listening on port 5000
listening on port 48898
```

Terminal: EPICS IOC

```
$ ./run-ethercatmc-ioc.sh simulator
[snip]
Starting iocInit
[snip]

iocRun: All initialization complete


epics>
```

Terminal: test cases

```
$ ./run-ethercatmc-tests.sh ca://IOC:m1
[snip]
collected 2 items
========== 2 passed in 18.97s ==========
```

Running remote

- Running remote tests needs:
  - Another collection of shell script files
  - starts the simulator, the IOC, the test execution
- Running remote tests gives:
  - A result (passed/failed)
  - All debug outputs one big logfile
  - Collect debug prints of those 3 tasks

Summary

- I love automated testing
- ci does a health check of the SW stack
- Running local tests check health state of the commissioned hardware + SW stack
- Links

```
https://github.com/epics-base/ci-scripts

https://github.com/EuropeanSpallationSource/
m-epics-ethercatmc/blob/master/.travis.yml

https://docs.pytest.org/en/stable/contents.htm
```

Test covarage

- EPICS base, asyn, motor
- Simulator
  - Functions inside motor like backlash logic
- Real hardware
  - Acceleration, Velocity, Homing
  - Soft limits (limit switch not activated)
  - Movements (no error, position reached)
  - Direction (limit switch is activated)

Small headaches

- Hard do debug failures
- Get the preconditions right
  - motor is homed
  - far away from limit switch
- Get the timeouts right