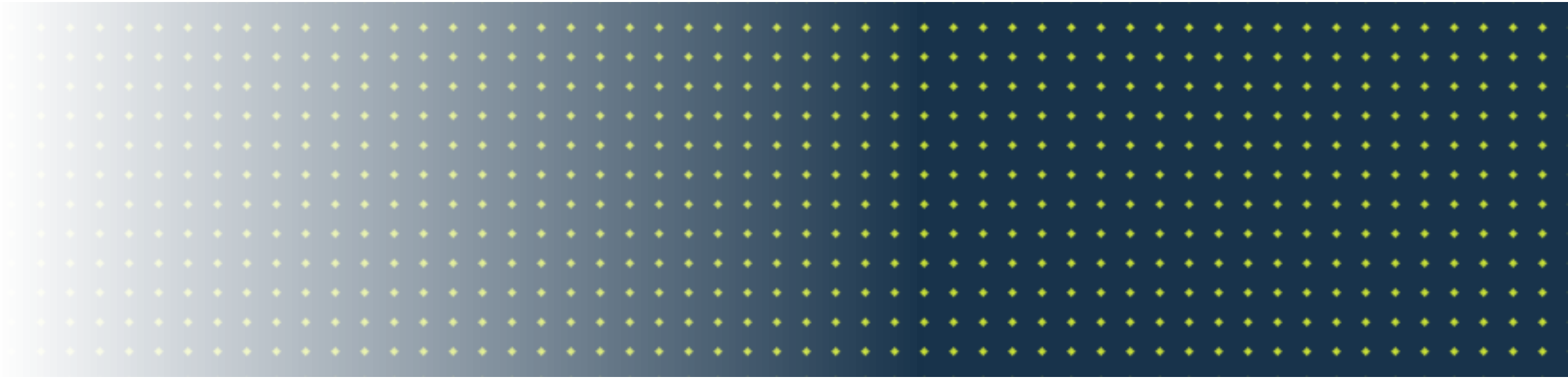
EPICS SUMMER SCHOOL 2026



EPICS IN SMALL LABORATORIES

Why the IOC — and C/C++ — still beat a Python application layer

Heinz Junkes · Fritz-Haber-Institut der Max-Planck-Gesellschaft · 2026-08-17

EPICS

FRITZ-HABER-INSTITUT
MAX-PLANCK-GESELLSCHAFT

AGENDA

TALK OVERVIEW

Six chapters, from motivation to a live IOC

EPICS

WHY THIS TALK

- Small labs and single-beamline setups face the **same control problems** as big facilities — with a fraction of the staff
- EPICS is often seen as "too heavy" for a 3-rack experiment — this talk argues the opposite
- Focus: the **IOC** (Input/Output Controller) as the right abstraction, and **C/C++** as the right language for it — not a Python service bolted on top

CHAPTER 1

WHAT SMALL LABS ACTUALLY NEED

Reliability, longevity, and one person to maintain it

THE SMALL-LAB REALITY

TYPICAL CONSTRAINTS

- 1 – 2 people covering controls, part-time
- Hardware kept running 10–20 years
- No dedicated software/IT team
- Budget favours reuse over custom builds

WHAT THAT DEMANDS

- A **stable, boring** core (few independent components)
- Vendor- and OS-agnostic drivers
- Something the **next student** can still read
- Long-term community support, not one maintainer

DECADES OF DEVICE SUPPORT, ALREADY WRITTEN

- Most small-lab hardware isn't exotic: vacuum pumps, pressure gauges, temperature controllers, power supplies, motion controllers — the same commercial instruments used at large facilities
- Large EPICS installations have been writing device support for exactly this class of hardware since the 1990s
- EPICS is open source with an active collaboration behind it — that device support is a `git clone`, not a project
- Setting up a small-lab experiment increasingly means **integration**, not **invention**

SERIAL IS EVERYWHERE — STREAMDEVICE MAKES IT IMMEDIATE

- Most of these instruments still talk the way lab instruments have talked for 30 years: RS232, RS485, or a raw socket over Ethernet
- `asyn` + `StreamDevice` turn "talks serial" into **a text protocol file** — no C driver, no compiler, no rebuilding the IOC for a new instrument
- A `.proto` file describes the conversation; a `.db` file wires it to PVs — that's the entire "driver"
- The same pattern covers RS232, RS485, and TCP/IP sockets alike — StreamDevice doesn't care which transport carries the bytes

FROM MANUAL TO PV IN TEN LINES

```
# maxigauge.proto — the entire "driver"
getSensorValue {
    out "PR$1\r";
    in  ACK CR LF;
    out ENQ;
    in  "%s" CR LF;
}

# maxigauge.db — wired straight to the protocol
record(stringin, "...:STM:getter") {
    field(DTYP, "stream")
    field(INP,  "@maxigauge.proto getSensorValue(6) $(PORT)")
    field(SCAN, "1 second")
}
```

Ten lines, no compiler — a working PV for a real UHV pressure gauge.

CASE STUDY: HP-SPM — A UHV SURFACE-SCIENCE LAB

- A real FHI experiment: ultra-high-vacuum scanning-probe microscopy (UHV-SPM) — chemistry/physics surface science, not a big-facility beamline
- One IOC, run by the same 1–2 people who run the experiment, talking to **24 commercial instruments**
- Every device lives in its own `devices/<Name>/` folder: `.db`, `.proto`, manual PDF, README — nothing else required to add one more
- Full source: **`gitlab.fhi.mpg.de/fhi-automation/hp-spm`**

ONE IOC, TWO DOZEN OFF-THE-SHELF INSTRUMENTS

CATEGORY	EXAMPLES	INTERFACE
Vacuum pumps	TurboVac 350i, Agilent ion pump & TSP, Edwards nXDS15i	Serial/USB, Ethernet via MOXA
Pressure gauges	Pfeiffer MaxiGauge TPG256-A	Ethernet via MOXA (was RS232)
Temperature control	CryoVac TIC500, TC09	TCP/Telnet
Power supplies	Delta Elektronika, FUG, Korad	Ethernet (SCPI), GPIB-over-Ethernet
Surface analysis	SPECS ErLEED 3000D, SRS RGA200, sputter gun	RS232, TCP/IP, UDP
Motion & imaging	Stepper controllers, GigE cameras	Serial, GigE Vision

Almost every row is a `.proto` file and a `.db` file — not a line of new C.

LEYBOLD TURBOVAC 350I

- Turbomolecular pump, DN100 CF, ~290 l/s — spins at tens of thousands of rpm to pump the chamber from rough vacuum toward UHV
- Connects over USB-serial (`/dev/ttyACM0`, 19200 baud) — an ordinary serial link from StreamDevice's point of view
- `TurboVac.proto` + `TurboVac.db`: no vendor SDK, no binary blob, just the manual's command set

PFEIFFER MAXIGAUGE TPG256-A

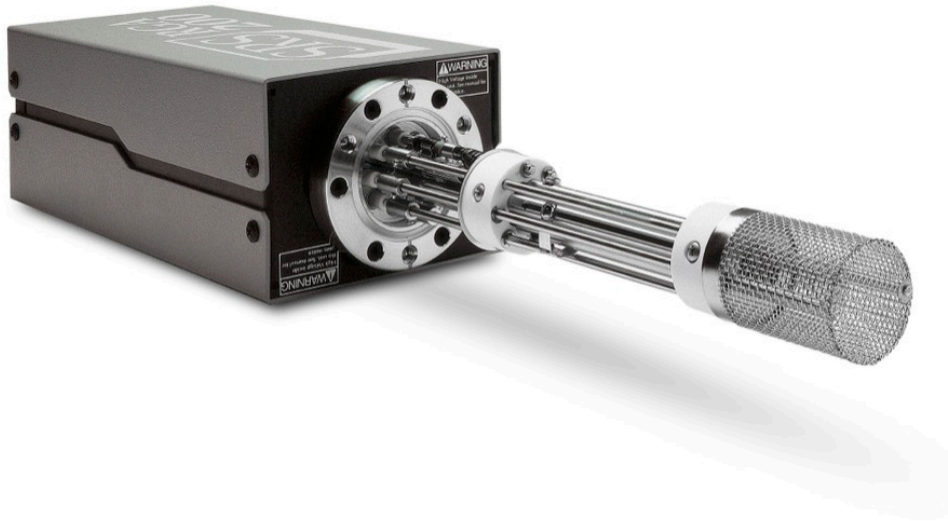
- Six-channel pressure controller — the "eyes" of the vacuum system, from rough vacuum through UHV
- Originally RS232 on `/dev/ttyS5`; now reached over the network through a MOXA serial-to-Ethernet converter — the StreamDevice protocol didn't change at all
- The `maxigauge.proto` shown earlier in this chapter **is** this device's driver

SPECS ERLEED 3000D

- Low Energy Electron Diffraction controller — drives electron-beam optics so you can see, live, whether a surface is atomically clean and ordered
- A genuinely specialized surface-science instrument — not something a facility IT department maintains
- RS232, 9600 baud, ~50 GET/SET parameters — written once by the community, reused here unchanged

SRS RGA200

- Quadrupole residual gas analyzer, 1–200 amu — a mass spectrometer for the vacuum itself: which gas species are present, and how much of each
- Talks straight TCP/IP — no serial adapter needed at all, same StreamDevice pattern regardless of transport
- Full spectrum scans, per-mass partial pressure, login handshake — all already handled



None of this hardware is exotic, and none of this device support is new.
A small lab building an EPICS IOC starts decades deep — not from zero.

CHAPTER 2

INSIDE THE IOC: A REAL-TIME SYSTEM

One process, many threads, and a promise about time

EPICS

THE IOC IS ITS OWN LITTLE OPERATING SYSTEM

- **Process Database** gives a uniform, self-describing model for every signal — motor, temperature, valve, detector — regardless of vendor
- Channel Access / PV Access decouples the *control logic* from the *GUI / logger / archiver* — swap any client without touching the IOC
- A single compiled binary (**one thing to start, one thing to log, one thing to restart**) — much easier for a one-person lab than a stack of scripts and services
- Underneath that, it's a **multithreaded real-time program**: scan threads, callback threads, a CA/PVA server, driver threads — each with a *priority that means something*
- That last part only works if the **host OS actually honors it** — which is the rest of this chapter

EPICS HAS OPINIONS ABOUT PRIORITY

`epicsThread.h` defines the scale (0 – 99) and names the anchor points:

PRIORITY	SYMBOLIC CONSTANT	TYPICAL THREAD
99	`epicsThreadPriorityMax`	reserved
90	`epicsThreadPriorityHigh`	`cbHigh`
70 → 60	`epicsThreadPriorityScanHigh` → `ScanLow`	`scan-0.1` … `scan-10`
50	`epicsThreadPriorityMedium`	`cbMedium`
40 / 20	`CAServerHigh` / `CAServerLow`	`caserver`, CA beacon
10	`epicsThreadPriorityLow`	`cbLow` , diagnostics
0	`epicsThreadPriorityMin`	`iocsh`

This isn't arbitrary — it encodes which work must never wait behind which other work.

RATE MONOTONIC SCHEDULING, HIDING IN PLAIN SIGHT

- Classic real-time scheduling result (Liu & Layland, 1973): for periodic tasks, **assign priority inversely to period** — shortest period gets the highest priority — and the whole set is provably schedulable below a known CPU-utilization bound
- Look at EPICS's own scan lists: `scan-10` (period 10 s) sits at the **bottom** of the scan priority range; `scan-0.1` (period 100 ms, 10 Hz) sits at the **top**
- That is Rate Monotonic Scheduling, applied by default, every time a `.db` file sets a `SCAN` field

Every EPICS developer who ever chose a SCAN rate has been doing RMS — whether they called it that or not.

```c
/* modules/libcom/src/osi/os/posix/osdThread.c — EPICS 7 */
static int epicsThreadGetPosixPriority(epicsThreadId pthreadInfo) {
    int maxPrio = sched_get_priority_max(SCHED_FIFO);  /* 99 */
    int minPrio = sched_get_priority_min(SCHED_FIFO);  /* 1  */
    double scale = (double)(maxPrio - minPrio) / 99.0;

    return (int)(minPrio
        + ((epicsThreadOSD *)pthreadInfo)->osiPriority * scale + 0.5);
}
```

> *`PTHREAD_EXPLICIT_SCHED` must be set when the thread is created — otherwise it silently inherits whatever scheduling policy the parent thread had.*

This priority number is only half the story — a separate `pthread_attr_setinheritsched(&attr, PTHREAD_EXPLICIT_SCHED)` call at thread-creation time (not shown above) is what makes the OS actually use it.

A priority scheme is a promise.
RTEMS keeps that promise by construction — on a general-purpose OS, you have to ask for it, and check that you got it.

RTEMS: THE IOC'S NATIVE HABITAT

- Single address space, single process — no user/kernel split, no other users, no competing processes on the CPU
- Task priorities are **real hardware-level priorities** the instant a thread is created — no capabilities, no PAM, no systemd unit required
- No demand paging → no page faults on the real-time path
- Brings its own network stack and driver model as standard: the IOC *is* the embedded system, not a process negotiating for a share of one

EVIDENCE: A REAL RTEMS IOC

```
tIocSh> epicsThreadShowAll
           NAME       EPICS ID   PTHREAD ID   OSIPRI  OSSPRI  STATE
         _main_     0x8099c5e0            0       0       0      OK
         errlog     0x809a6d88    184614915      10     229      OK
        BankEvT     0x817dd1a0    184614924      92      22      OK
         cbHigh     0x818b6eb8    184614928      71      75      OK
          PyroT     0x81ff8590    184614933      91      24      OK
        scan-10     0x8217dd80    184614935      60     103      OK
       scan-0.1     0x827818c0    184614941      66      88      OK
     CAS-beacon     0x82897930    184614944      17     211      OK
OSD priority range min: 1 max 254, memory not locked
```

Live capture — BeagleBone Black / RTEMS7 pyrometer-array IOC. Every EPICS priority lands on its own distinct OS priority.

SAME IOC, ORDINARY UNIX, NO SPECIAL RIGHTS

```
epics> epicsThreadShowAll
           NAME       EPICS ID   PTHREAD ID   OSIPRI  OSSPRI  STATE
         _main_    0x100f3c680            0        0       0      OK
         errlog    0x100f4a520   6168129536       10      31      OK
         cbHigh    0x9f1164240   6183825408       71      31      OK
        scan-10    0x9f1164840   6194556928       60      31      OK
       scan-0.1    0x9f1164a80   6207434752       66      31      OK
      CAC-event    0x9f1164f00   6213496832       51      31      OK
```

`OSSPRI` is 31 for every thread shown. The database still enforces record-processing order — the OS no longer does. `cbHigh` and `CAC-event` now run at exactly the same OS priority.

SAME IOC, UNIX, WITH `RTPRIO` + `MLOCKALL`

```
epics> epicsThreadShowAll
           NAME       EPICS ID   LWP ID   OSIPRI  OSSPRI  STATE
         _main_ 0x55e8fd70d0e0      711      0       0       OK
         errlog 0x55e8fd734f00      715     10      10       OK
         cbHigh 0x55e8fd8d0020      730     71      70       OK
        scan-10 0x55e8fd958e60      734     60      59       OK
       scan-0.1 0x55e8fd959ca0      740     66      65       OK
      CAC-event 0x7f6db402f900      881     52      51       OK
OSD priority range min: 1 max 99, memory locked
```

Grant `CAP_SYS_NICE` / `rtprio` and call `mlockall()`, and Linux keeps the same promise RTEMS keeps for free.

WHAT `RTPRIO` + `MLOCKALL` ACTUALLY BUY YOU

WITHOUT THEM

- `SCHED_OTHER` only — every thread competes as an equal
- Page faults can land on the real-time path
- Priority inversion is possible
- `cbHigh` can be starved behind `cbLow`
- Latency jitter is effectively unbounded

WITH THEM

- Real `SCHED_FIFO` priorities, preserved end-to-end
- Pages locked → no page-fault jitter
- `PTHREAD_PRIO_INHERIT` protects mutex sections
- Priority order matches what the database assumes
- Latency becomes bounded and predictable

Mars Pathfinder, 1997: a priority inversion in a VxWorks mutex tripped the watchdog and reset the lander.

EPICS's `epicsMutex` enables `PTHREAD_PRIO_INHERIT` by default — but that only helps if the OS underneath honors priority at all.

SETTING IT UP IN PRACTICE

APPROACH	WHERE IT FITS
`/etc/security/limits.conf` + PAM	interactive dev login, quick to try
`systemd` unit: `LimitRTPRIO=99` , `LimitMEMLOCK=infinity`	production IOC service — recommended
`chrt` + a per-thread script keyed on `/proc/PID/task/*/comm` *	one-off diagnosis, existing running IOC

All three end at the same place: EPICS's own priority scheme, actually enforced.

* `chrt -p <PID>` only sets the main thread — the script reads each pthread's name from `/proc/<PID>/task/<TID>/comm` and calls `chrt -f -p <prio> <TID>` on it individually.

TRY IT YOURSELF: REAL DEVICE SUPPORT ON RTEMS

- `PyroT` and `BankEvT` from the RTEMS capture earlier in this chapter aren't hypothetical — they're the acquisition and event-banking threads of a real pyrometer-array driver
- Full source: **`gitlab.fhi.mpg.de/rtems_epics/bbb_rtems7_epics/beaglepyroioc`**
- BeagleBone Black + RTEMS 7 + EPICS — a complete, small-lab-scale example of writing device support against a real hard real-time IOC

CHAPTER 3

IOC / C/C++ VS. A PYTHON APPLICATION LAYER

Same job, different guarantees

EPICS

WHERE THE TWO APPROACHES ACTUALLY DIFFER

EPICS IOC IN C/C++

- Compiled, statically checked before it ever runs
- Deterministic timing, real scan/callback threads
- Runs headless on minimal hardware (incl. old VME/embedded targets)
- One process supervises the hardware for years unattended
- ABI-stable driver/record interfaces across decades of EPICS releases

PYTHON APPLICATION ON TOP

- Errors surface at runtime, often mid-experiment
- GIL and interpreter overhead fight soft real-time needs
- Drags in an interpreter + package/venv management per box
- Convenient for **prototyping and orchestration** — less so as the thing that owns the hardware
- Great as a *client* of the IOC, not a replacement for it

THE HONEST TRADE-OFF

- Python **is** the better tool for scripting an experiment sequence, data analysis, or a quick GUI — this talk is not "Python is bad"
- The claim is narrower: **the hardware-facing control layer** benefits from a compiled, long-lived IOC — Python belongs *above* that layer, talking CA/PVA, not *instead of* it
- Small labs in particular pay for every extra independent component in person-hours — the IOC removes a whole category of "who restarts the Python service after the power cut?" problems

```c
/* Illustrative device support skeleton — replace with a real example */
static long init_record(aiRecord *prec) {
    /* one-time setup: open device, cache addressing info */
    return 0;
}

static long read_ai(aiRecord *prec) {
    /* deterministic, synchronous read — no interpreter, no GC pause */
    prec->val = read_hardware_channel(prec->inp.value.instio.string);
    return 0;
}
```

Example only — swap in the driver snippet you actually want to walk through.

This isn't just for reading gauges — the same deterministic, interpreter-free record processing scales up to genuinely sophisticated, closed-loop control tasks. See the appendix for a real FHI example.

CHAPTER 4

GETTING A SMALL LAB RUNNING

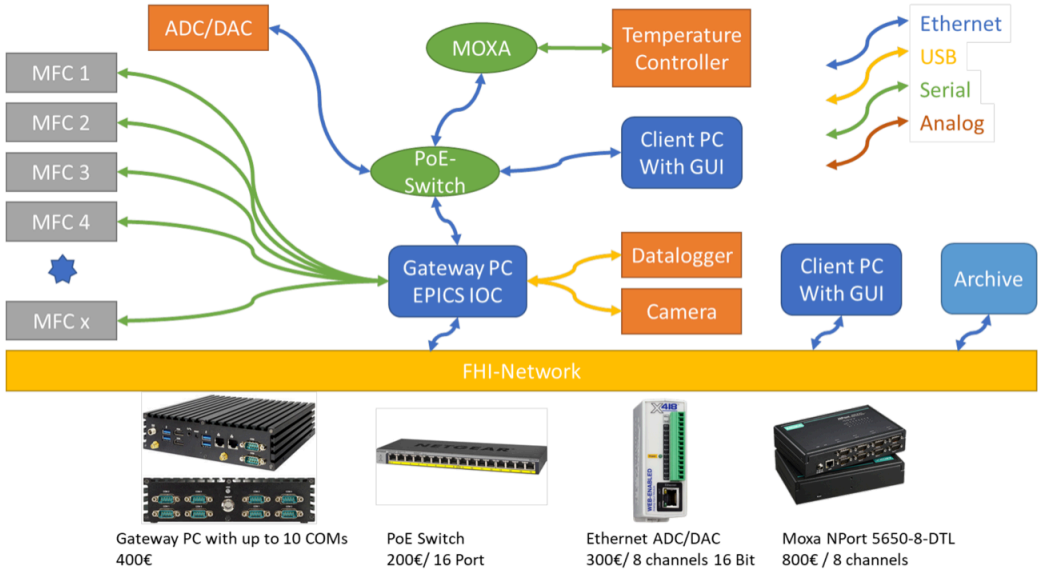
A pragmatic path from zero to a working IOC

EPICS

A MINIMAL, REALISTIC SETUP

1. Pick **one** small target IOC (Linux SBC or a repurposed PC) — no cluster, no container orchestration required
2. Start from an existing **support module** for your hardware bus (Modbus, Serial, GPIB, EtherCAT, …) before writing a driver
3. Keep the **database** as the single source of truth for what the lab controls
4. Add a lightweight client (CS-Studio, Phoebus, or a thin Python/Qt panel) — it can change later without touching the IOC

THE EDGE SERVER: ONE RECIPE, NOT A FRESH BUILD EVERY TIME



- Every new FHI small-lab setup is the same ~400 € gateway PC (up to 10 COM ports) running one EPICS IOC, reached over the FHI network by GUIs and an archive
- One `epics-edge-setup` script provisions it end to end — OS user, packages, procServ, EPICS base, the full FHI support-module stack — a lab plugs in instruments, it doesn't install Linux

CHAPTER 5

LESSONS FROM THE FIELD

What broke, what didn't

FIELD NOTES

SITUATION	PYTHON SERVICE	EPICS IOC
Power cut, unattended restart	needs a supervisor + venv still there	`systemd` restarts one binary
Student leaves, code un-maintained	dependency drift breaks it in months	still builds/runs years later
New detector, same bus	new script, new conventions	new record, same database model
Facility wants your data	custom export needed	already speaks CA/PVA

Real examples from FHI labs — the IOC as the thing that notices, not just the thing that logs:

- A UHV leak is caught by the IOC through the vacuum pressure sensor, and the section isolated using the valve pressure sensors
- The RGA filament can't be switched on by accident — the interlock checks system pressure first
- In the chemical reactor, the IOC shuts off the gas supply by monitoring the gas-detection system

CHAPTER 6

CONCLUSION & DISCUSSION

Small does not mean simple requirements

EPICS

TAKEAWAYS

- Small laboratories have **fewer hands**, not **lower reliability requirements** — if anything, the opposite
- The EPICS IOC in C/C++ gives a small lab a **compiled, deterministic, long-lived** control layer for the price of learning one framework
- Python remains the right tool **above** the IOC — for orchestration, analysis, and quick UIs — not as a replacement for the hardware-facing control process
- Bring your counter-examples — this is meant to start a discussion, not close one

THANK YOU

Questions & discussion welcome

Heinz Junkes · junkes@fhi-berlin.mpg.de · Fritz-Haber-Institut der Max-Planck-Gesellschaft

EPICS

FRITZ-HABER-INSTITUT
MAX-PLANCK-GESELLSCHAFT

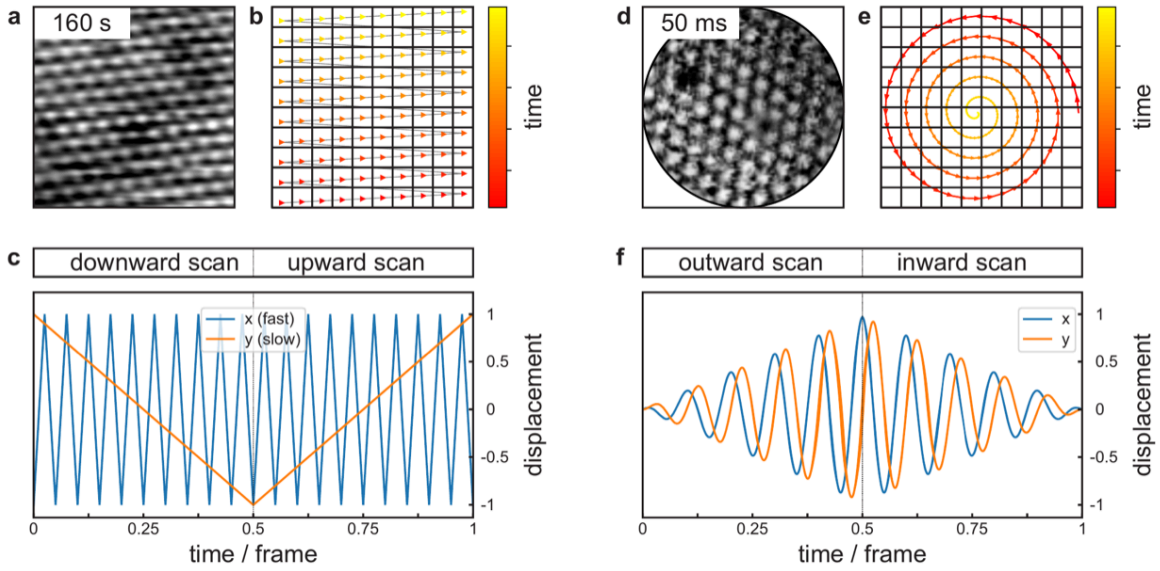
APPENDIX

REAL-TIME CONTROL AS THE EXPERIMENT ITSELF

Spiral high-speed STM — for discussion

EPICS

SAME SURFACE, SAME RESOLUTION — 3000× FASTER



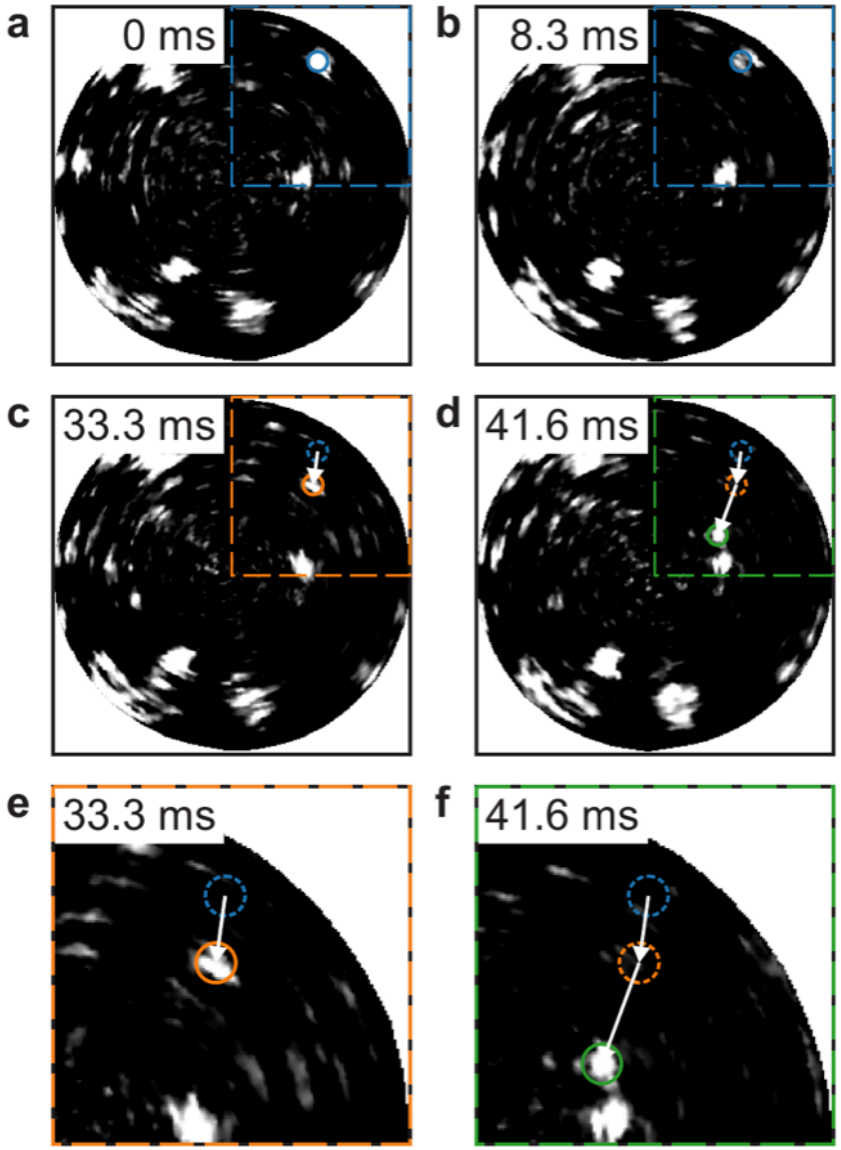
- Raster scan (left, 160 s): tip velocity reverses sharply at every line — the flyback wastes three of every four passes and rings the scanner
- Spiral scan (right, 50 ms): one continuous, smooth trajectory — no reversals, no flyback, same atomic-resolution image

GENERATING THE TRAJECTORY IS THE HARD REAL-TIME JOB

- The spiral is fully described by two lines: `x = t^a · cos(ω·t^b)`, `y = t^a · sin(ω·t^b)` — tuning `a`, `b`, `ω` moves between Archimedean and Fermat spirals, constant-linear- and constant-angular-velocity
- Generating and sampling that signal at speed **is** the control task: a VMEbus crate running **EPICS on RTEMS**, an FPGA sampling at up to 250 MHz, driving a Highland Technology arbitrary waveform generator into the scanner's piezo tube
- `pvapy` streams the resulting PV to a Python client for live display and HDF5 logging — Python sits **above** the loop, the same division of labor as everywhere else in this talk

THE PAYOFF: ATOMS CAUGHT MID-HOP

- Same Ru(0001) oxygen adlayer, now at 8.3 ms/frame (120 Hz) — a single oxygen atom is caught hopping between adsorption sites, frame to frame
- A ~100 s/frame raster scan could never have shown this — the atom, or the tip, would have moved long before the image finished
- Published: Gura, Yang, Brinker, Kalaß, Kirstaedter, Marschalik, Junkes, Heyde & Freund, *Appl. Phys. Lett.* **119**, 251601 (2021)



The same IOC — the same "boring, deterministic core" — that reads a pressure gauge can generate a real-time signal precise enough to watch atoms move.
Sophisticated science doesn' t need a different control architecture. It needs the same one, pushed harder.

SOURCES & FURTHER READING

- **hp-spm** — UHV-SPM EPICS IOC, 24 commercial instruments, one `.proto` / `.db` pair per device: `gitlab.fhi.mpg.de/fhi-automation/hp-spm`
- **epics-edge-setup** — startup script for the FHI EPICS edge server: `gitlab.fhi.mpg.de/epics-tools/epics-edge-setup`
- Gura, Yang, Brinker, Kalaß, Kirstaedter, Marschalik, Junkes, Heyde & Freund, "Spiral high-speed scanning tunneling microscopy: Tracking atomic diffusion on the millisecond timescale," *Appl. Phys. Lett.* **119**, 251601 (2021) — pubs.aip.org/aip/apl/article/119/25/251601
- Junkes, Freund, Gura, Heyde, Marschalik & Yang, "Experiment control with EPICS7 and symmetric multiprocessing on RTEMS," *ICALEPCS2017*, JACoW (2018), pp. 1762–1766
- EPICS Collaboration — documentation, training, and the full module ecosystem: epics-controls.org

Slides, sources, and figure credits: this deck, the hp-spm repository, and the papers above.