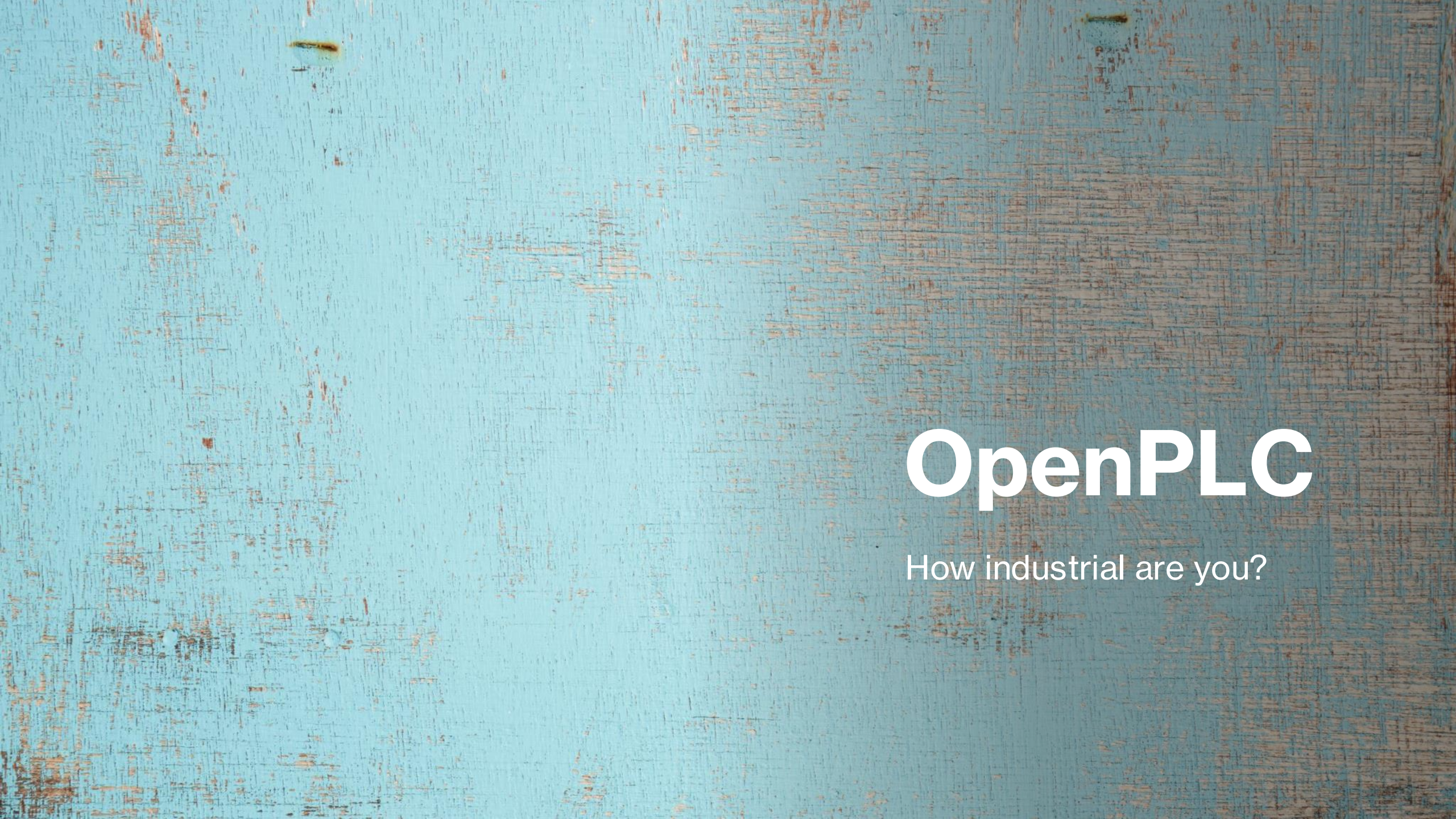

OpenPLC
How industrial are you?

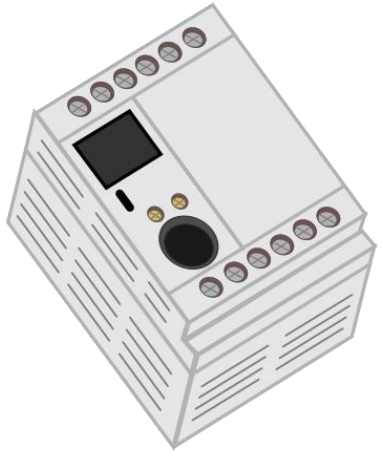
Contents

unibz

All images are CC, unless otherwise noted

PLC background

- PLC – Programmable Logic Controller
  - Product of the late ‘60s
  - Initially installed to replace bulk of relay matrices
  - Primarily targeted electricians and engineers

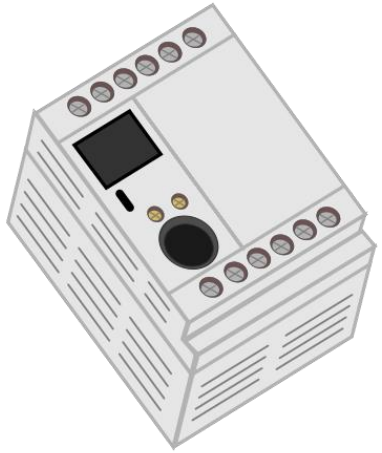


unibz

All images are CC, unless otherwise noted

PLC background

- PLC – Programmable Logic Controller
  - Product of the ‘70s
  - Initially installed to replace bulk of relay matrices
  - Primarily targeted electricians and engineers



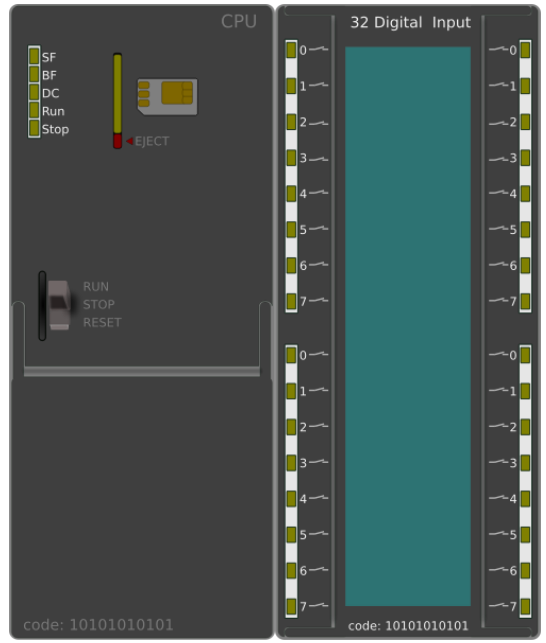
- Now, PLCs include and Advanced control logics
  - Distributed I/O
  - High speed feed-back control loops
  - Motion control, including robotics

unibz

All images are CC, unless otherwise noted

Modern PLC systems

- Proprietary Runtime
  - Middleware running on custom Linux or Windows systems
  - Open-source variants exist - for vendors - e.g., CoDeSys
  - Licensed per system or per size

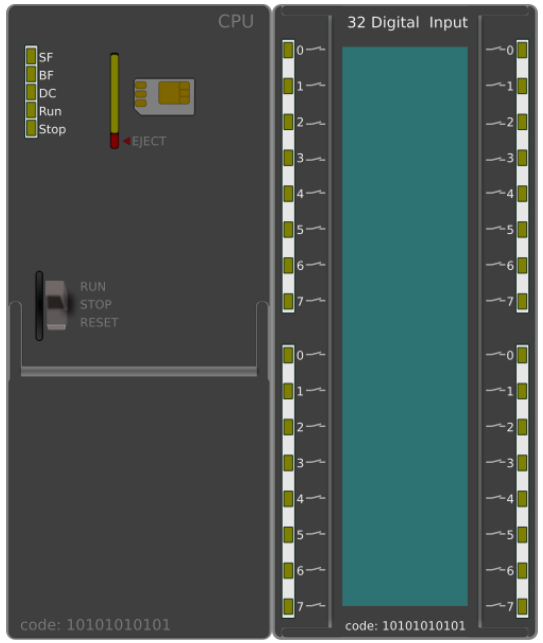


All images are CC, unless otherwise noted

Modern PLC systems

- Proprietary Runtime
  - Middleware running on custom Linux or Windows systems
  - Open-source variants exist - for vendors - e.g., CoDeSys
  - Licensed per system or per size

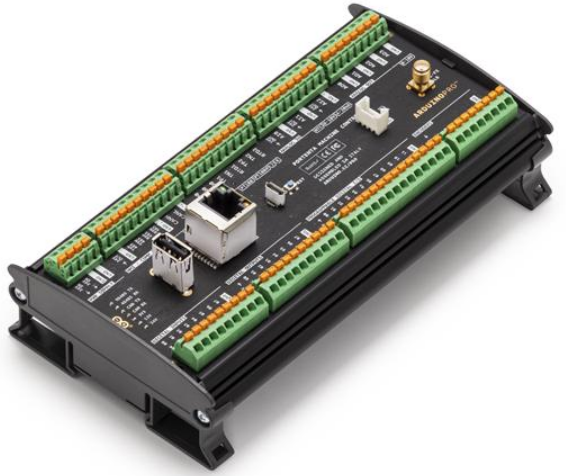
- Proprietary hardware binding SW to a product
  - e.g., Siemens S7 products work only with Simatic runtime
  - Rugged hardware made for extreme working conditions
  - Strong branding relying on distribution and availability

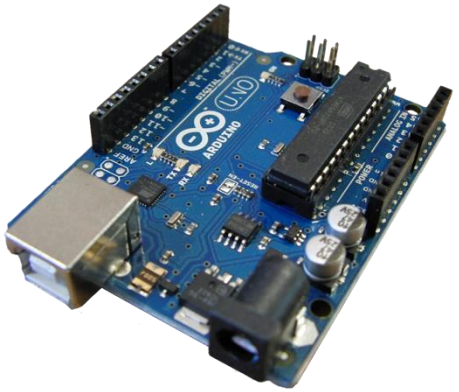


All images are CC, unless otherwise noted

OpenPLC

- Portable runtime system
  - Core is written in C
  - Typically comes with a Python-based webserver
  - Packaged with installer, easy to install on any generic device

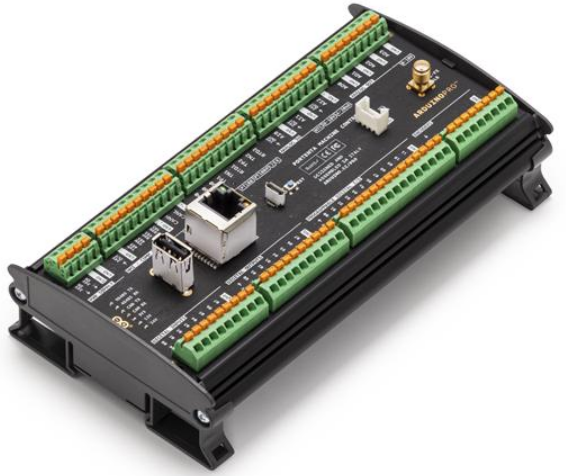


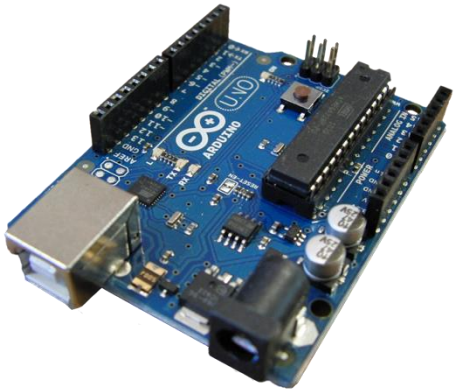


unibz

OpenPLC

- Portable runtime system
  - Core is written in C
  - Typically comes with a Python-based webserver
  - Packaged with installer, easy to install on any generic device

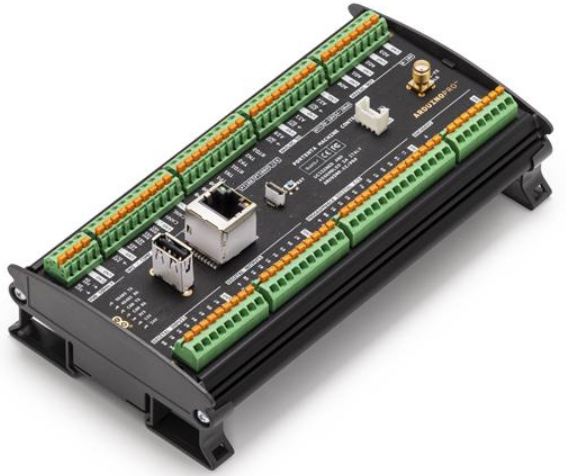




unibz

OpenPLC

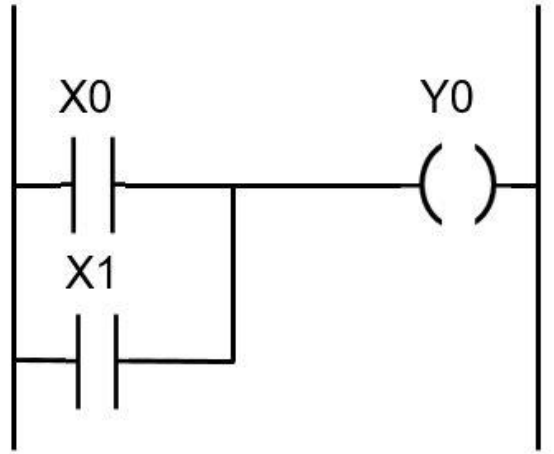
- Portable runtime system
  - Core is written in C
  - Typically comes with a Python-based webserver
  - Packaged with installer, easy to install on any generic device



- Works on Open Hardware
  - Microcontrollers, e.g., Arduino Uno/Nano/Mega
  - Single board computers, such as Raspberry PI
  - Sometimes also available as rugged version

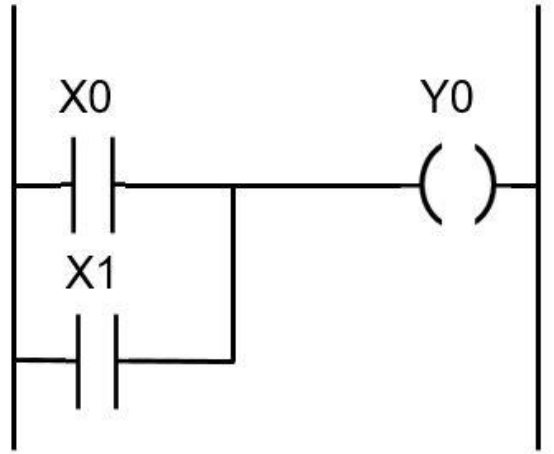
OpenPLC

- Home Automation – Programmable Logic Controller
  - Low cost, open hardware and open source
  - Easy to program thanks to intuitive languages
  - However, constrained due to limited I/O



OpenPLC

- Home Automation – Programmable Logic Controller
  - Low cost, open hardware and open source
  - Easy to program thanks to intuitive languages
  - However, constrained due to limited I/O



- Reevaluation of use due to shift to vPLC
  - May open a new target use
  - OpenPLC runtime is ready for containerization
  - Multi-instance ready on a general-purpose on-premise server

unibz

Performance comparison

- Compute comparison
  - vPLC instances Open vs Commercial
  - Run on Generic Purpose instance
  - Target hardware can also be Open, e.g., ARM64 systems

unibz

Performance comparison

- Compute comparison
  - vPLC instances Open vs Commercial
  - Run on Generic Purpose instance
  - Target hardware can also be Open, e.g., ARM64 systems

- Compute performance
  - Assess the determinism of a vPLC instance
  - Run compute only tasks
  - Assess interference of shared resources

unibz

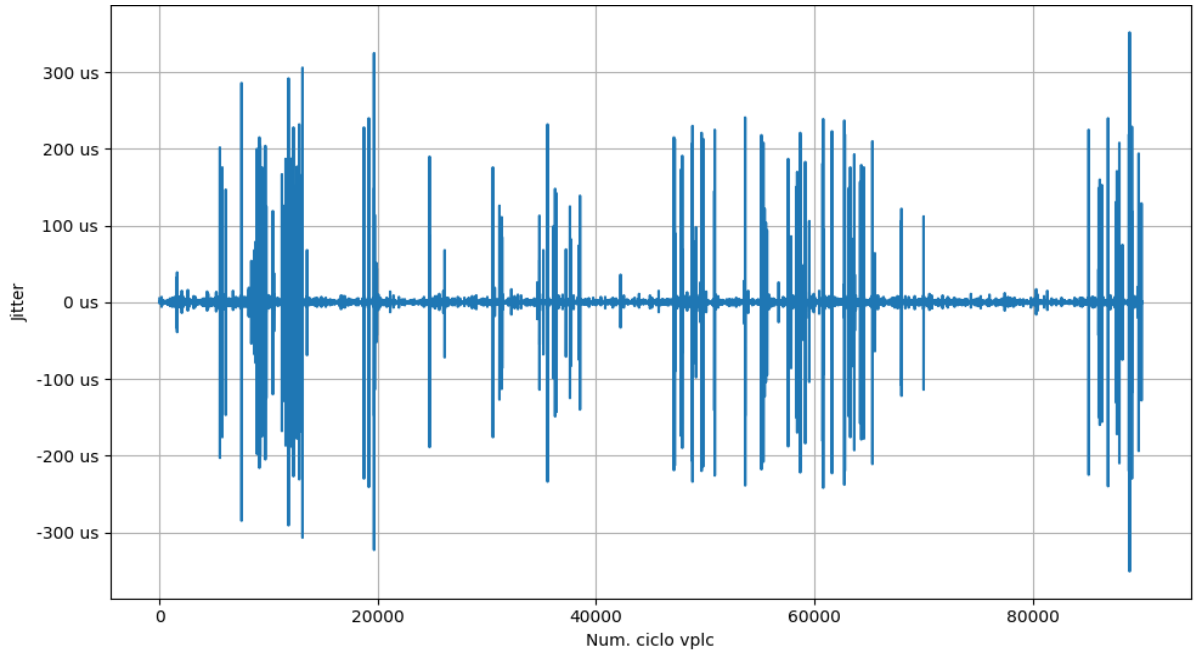
Performance comparison

- Compute performance
  - Compute Fibonacci numbers of a 10% load
  - Increase load in steps of 10
  - Repeat with 2 or more shared instances
  - Optimized Linux kernel with PREEMPT_RT

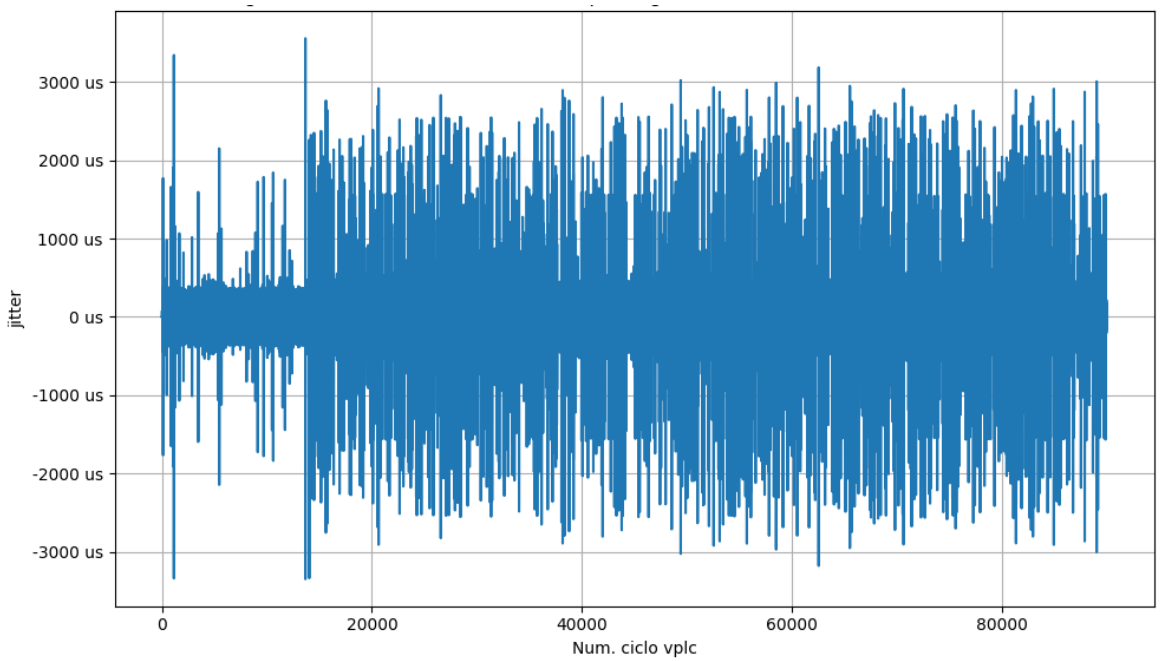
- Numbers performance
  - High Fibonacci number f(x) every 10 ms
  - 100 times for 10% load, 200x for 20%...

unibz

Performance comparison - compute



Without resource pinning



With resource pinning

unibz

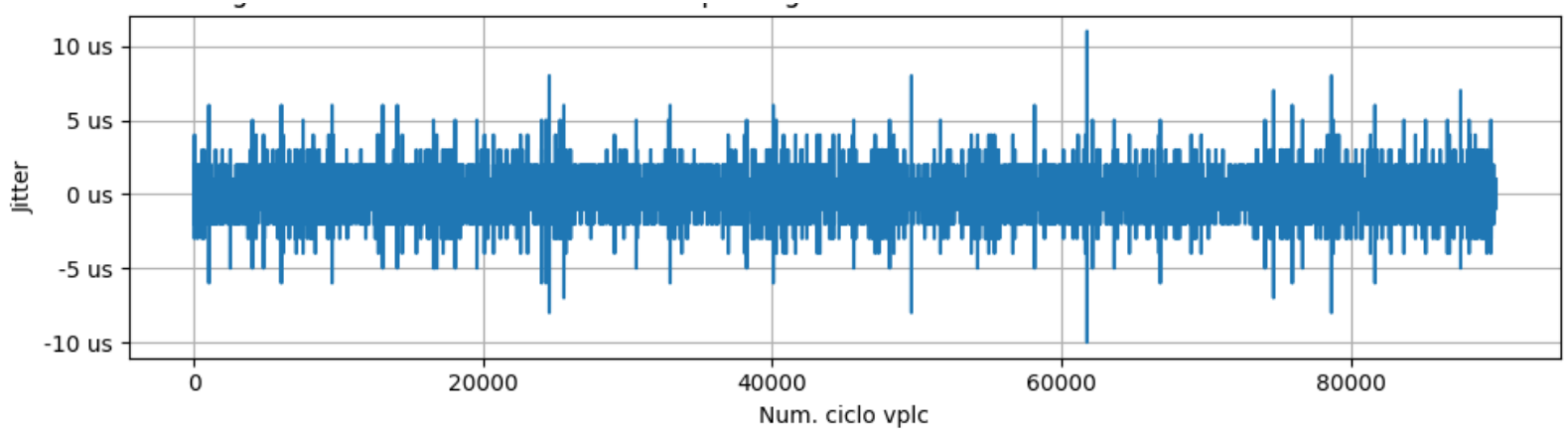
Performance comparison

- Compute first results
  - Resource pinning not as effective as with Commercial
  - Container includes webserver
  - -> Resource contention if pinned

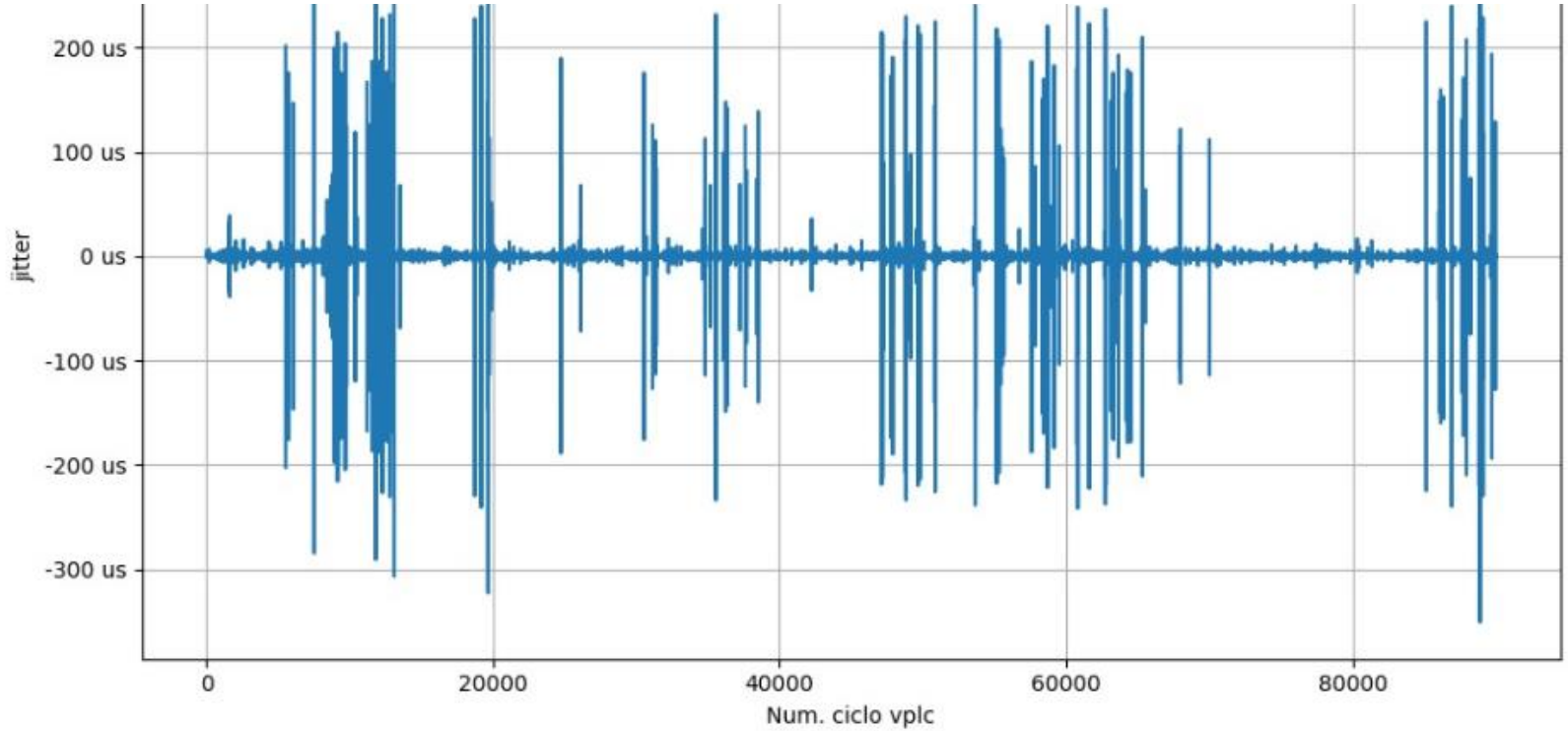
- vPLC must thus either
  - Run PLC core only
  - Be orchestrated to separate server from PLC core

Performance comparison - compute

CoDeSys 10% load



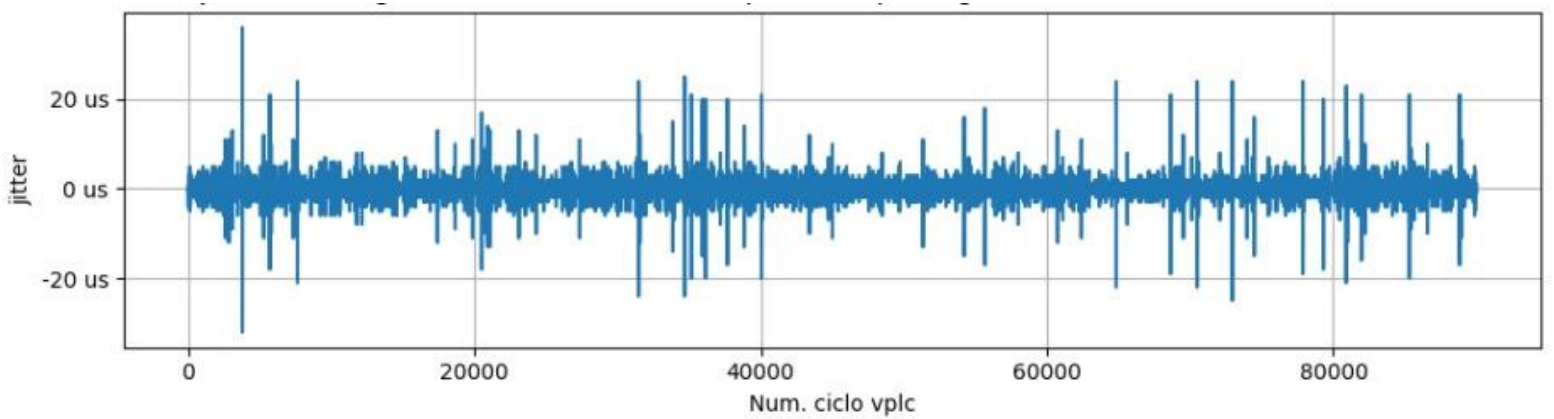
OpenPLC 10% load



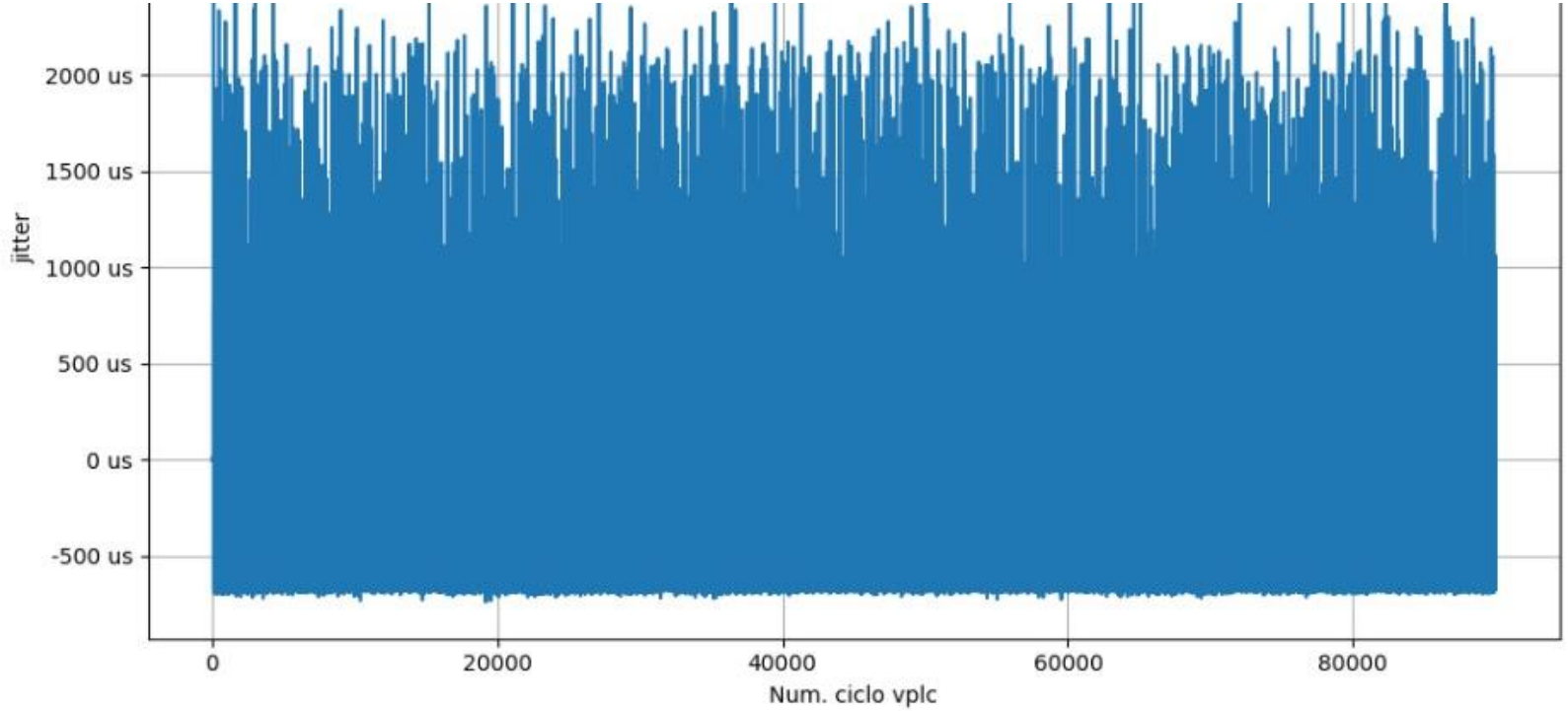
unibz

Performance comparison - compute

CoDeSys 30% load



OpenPLC 30% load



unibz

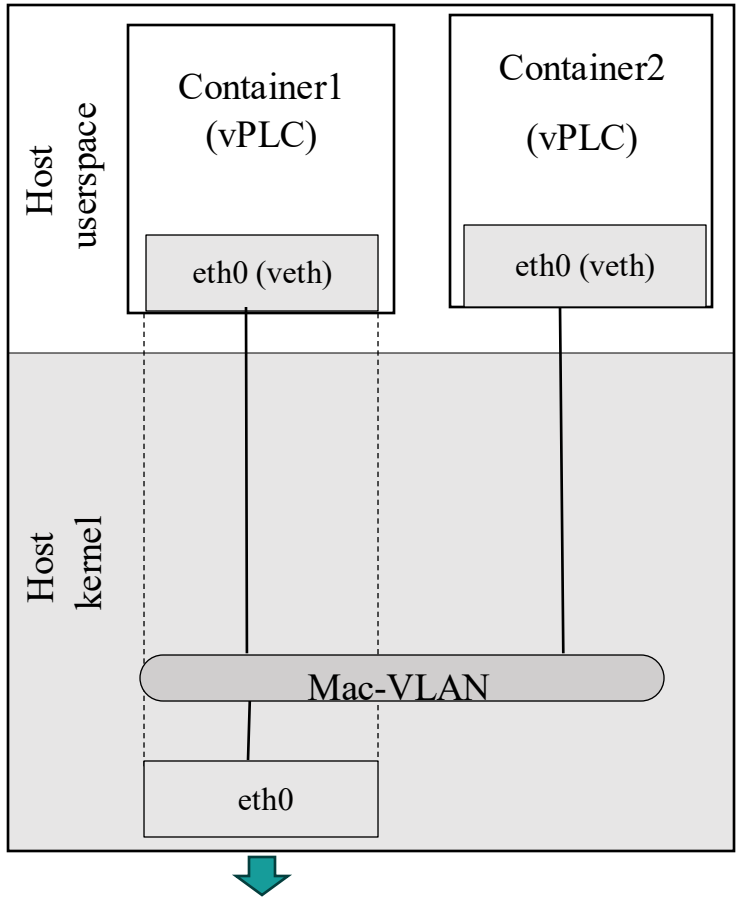
Performance comparison

- Compute performance lower
  - 1/3 of Fibonacci numbers with same CPU load
  - Jitter also inconsistent, 3x with same CPU load
  - Multiple instances multiply values almost linearly

- Results show that
  - Still possible to use
  - GP device has multiple of computing power
  - Jitter is main focus, has to be kept low

Networking

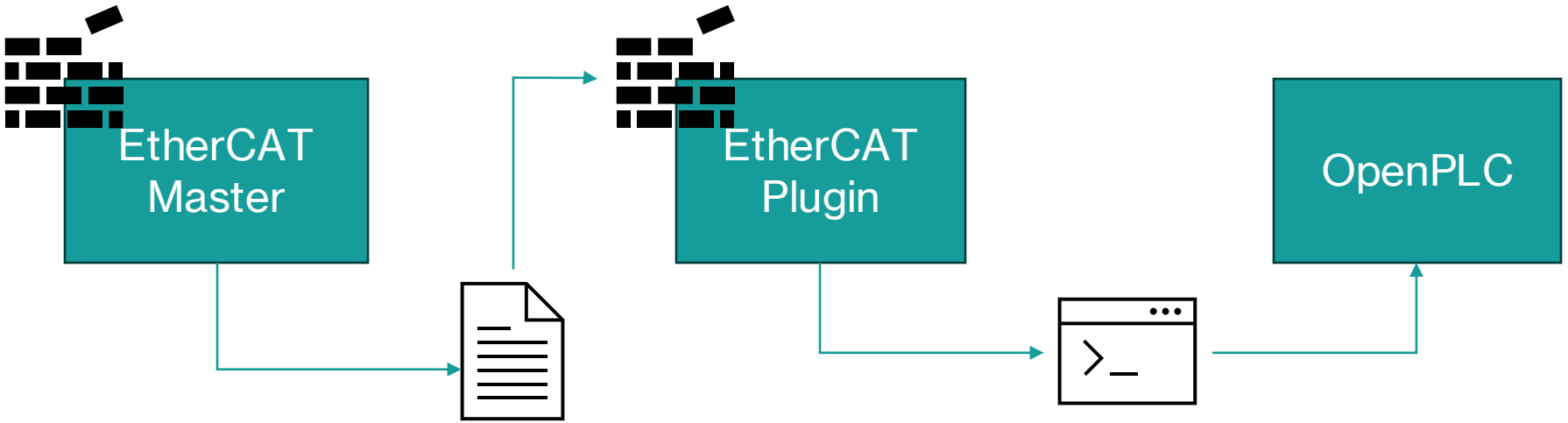
- Communication performance
  - High speed controls need right protocols
  - Default protocols not suitable
  - OpenPLC allows communication plugins, e.g., EtherCAT
- Port sharing
  - Multiple vPLC can share a port
  - MACvLAN showed minimal delay and jitter
  - Protocol agnostic and OSI/ISO lv.2 compatible



unibz

Networking

- Complicated installation and management
  - All three components compiled and installed separately
  - EtherCAT driver compiled JIT -> Docker installation cumbersome
  - Plugin requires running slave configuration, OpenPLC running slave mapping



unibz

Conclusions

- Compute performance
  - Lower performance than commercial not a problem
  - Jitter needs to be addressed and managed
  - Small scale industrial control already viable

unibz

Conclusions

- Compute performance
  - Lower performance than commercial not a problem
  - Jitter needs to be addressed and managed
  - Small scale industrial control already viable

- Network use
  - High performance stack possible
  - Compute jitter still problematic for <1ms control
  - Network stack use cumbersome

unibz

Future Work

- Compute performance
  - Orchestration and pinning of threads
  - Control group isolation
  - Runtime optimizations at kernel level

Future Work

- Compute performance
  - Orchestration and pinning of threads
  - Control group isolation
  - Runtime optimizations at kernel level

- Network use
  - Automation of EtherCAT stack use
  - Docker build scripts for driver installation
  - Native driver performance tests

Florian Hofer

info@florianhofer.it