Il mio setup Cursor per OPCUA Pipe Gateway

Guido Brugnara — Leader.IT · **Cursor Pro**

Cursor MeetUp Trento · 11 giugno 2026 · Afliant

luma.com/7rcm3za3

Il progetto (30 secondi)

`opcua_pipe_gateway` — client OPC-UA via STDIN/STDOUT (*"curl per il PLC"*)

- Implementazione duale: **Perl** (open62541) + **Python** (python-opcua)
- **Server di test** propri in entrambe le lingue → matrice interoperabilità
- Validazione su **KEPServerEX V6**, poi packaging per deploy

leader.it/Portal/OpcUApipeGateway

Il mio ambiente Cursor

Elemento	Scelta
Edizione	Cursor **Pro** (Agent con accesso completo agli strumenti)
Macchina dev	Desktop — workspace principale
Laptop meetup	E-book 14" (Intel N5000) — copia progetto per demo
Linguaggi	Perl, Python, Bash, C — codebase eterogeneo

Cursor come **pair programmer**, non autopilota — le decisioni industriali restano umane.

Istruzioni personalizzate (User Rules)

Regole globali che guidano ogni sessione:

- **Script:** preferenza **Perl** o **Bash**; commenti inline in **inglese**
- **Documentazione:** formato POD — italiano (`LEGGIMI.pod`) + inglese (`README.pod`)
- **Chat:** italiano con l'IA; deliverable bilingue dove serve
- **Modifiche codice:** scope minimo — stile esistente, no over-engineering
- **Trasparenza:** lavoro assistito da IA dichiarato negli header sorgente

*Le **User Rules** di Cursor — contesto persistente, senza ripeterle ogni chat.*

Configurazione workspace

Multi-root workspace (`Test_OPC_UA.code-workspace`):

```
folders:
  .                              ← ambiente dev e test
  ~/SVN/opcua_pipe_gateway       ← trunk distribuzione pubblica
```

Contesto chat: `@file` · `@folder` · `@workspace`

Documentazione: `SOLUZIONE_SICURA_MULTI_ROOT.md`, `PRESERVA_HISTORY_CURSOR.md`

Workflow: Plan → Agent → Test

1. **Plan mode —** architettura, scalette slide, analisi costi (senza modifiche)
2. **Agent mode —** codice, documentazione, script; l'IA esegue terminal e legge log
3. **Umano —** scenari di test, certificati, policy di sicurezza, approvazione finale

```
Client + Server paralleli → test 2×2 → analisi log con IA
  → validazione KEPServerEX → suite test senza errori
```

Esempi di prompt (reali) — 1/2

Template da lavoro reale — non demo live. Un obiettivo · un `@file` · chat focalizzata.

Porting

> Crea un equivalente Python di `@opcua_pipe_gateway.pl` — stessa CLI, exit code e formato I/O.

QA cross-language

> Esegui `@test_kepserverex_v6_demo.pl` e `.py` sullo stesso server; confronta output e correggi differenze.

Debug industriale

> Analizza `@OPC_diagnostic.8.log.txt` — perché KEPServerEX rifiuta il certificato client?

Esempi di prompt (reali) — 2/2

Stesso schema: un obiettivo focalizzato per chat.

Documentazione

> Scrivi documentazione POD in italiano per `@opcua_pipe_gateway.pl` seguendo lo stile esistente.

Questo meetup

> Pianifica slide Marp 10 minuti: setup Cursor prima, progetto OPC-UA come case study.

Come Cursor ha aiutato (concreto)

- **Porting:** client Perl → Python equivalente (stesso contratto CLI)
- **QA cross-language:** 4 combinazioni rivelano bug che un solo stack nasconde
- **Debug industriale:** log OPC, certificati, fix UserTokenPolicy
- **Documentazione:** POD, LEGGIMI, wiki, analisi costi/licenze
- **Queste slide:** Markdown Marp → PDF con **Marp for VS Code**

Estensioni e MCP

- **Marp for VS Code** (`marp-team.marp-vscode`) — slide → PDF
- Supporto linguaggio **Perl** — POD, navigazione `.pl`
- Nessuna config **MCP** specifica del progetto — **Agent tools** (terminal, search) per il lavoro quotidiano
- **Browser MCP** quando servono documentazione OPC-UA / vendor esterna

Setup semplice: User Rules + workspace + Agent.

Takeaway · Domande?

- **User Rules** + **multi-root workspace** + **Plan → Agent**
- **Matrice test + IA —** pattern riutilizzabile oltre OPC-UA
- **Human-in-the-loop** per domini industriali / sicurezza

Open source: Portale · SVN `opcua_pipe_gateway/tag/`

Slide in **inglese** · Talk in **italiano —** grazie!