

Mapo-Mono

Progetto Mapo MONO per gestione recupero dati, presentazione e (mini) storage locale, sviluppo iniziale per progetto Multiax USA - Jeldwen

Link google Drive area documenti:

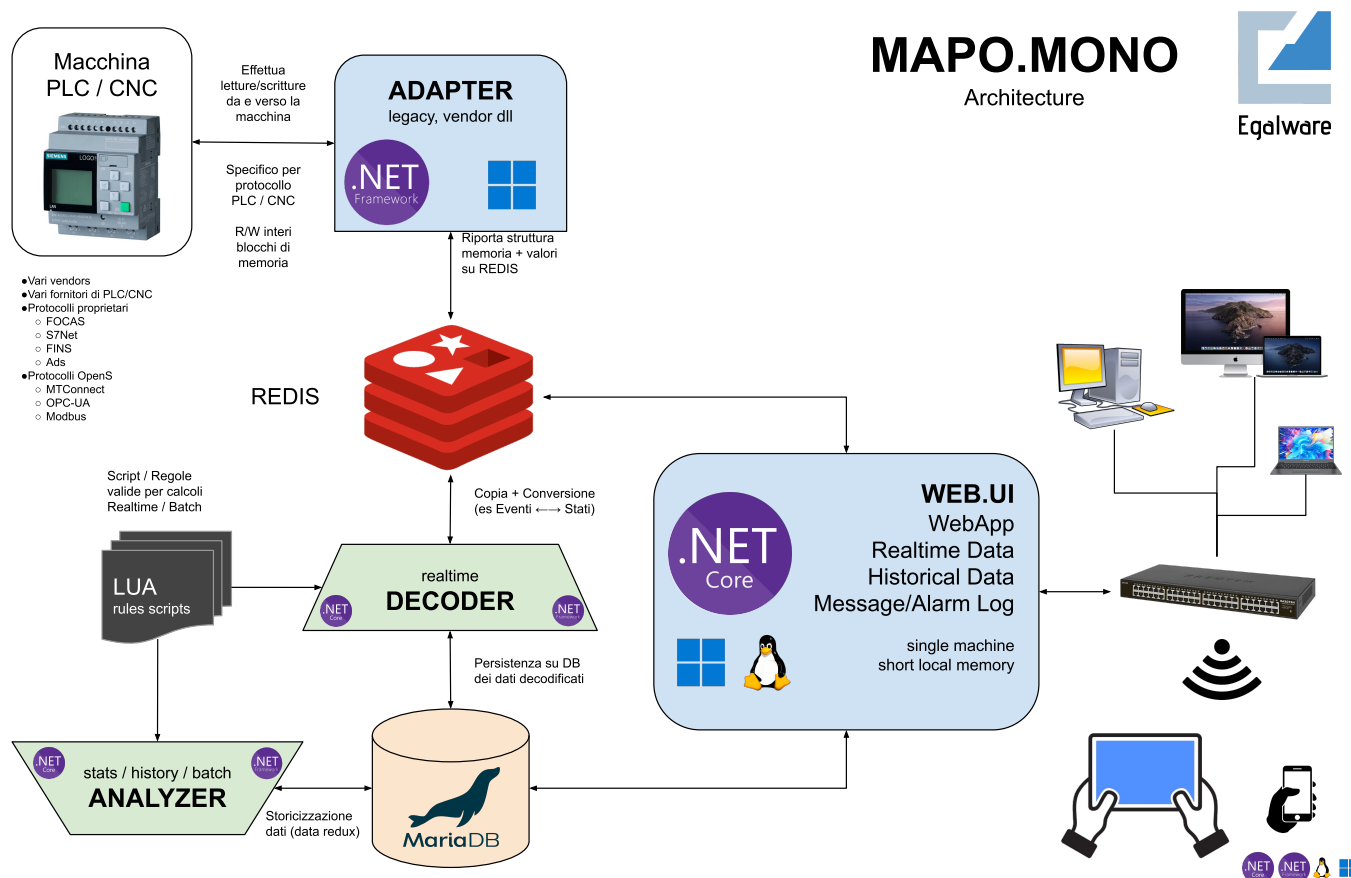
https://drive.google.com/drive/folders/1JSa52d85Yp93PEs9_nuil81yVftGFTO8?usp=sharing

- [Mapo-Mono](#)
- [Architettura](#)
 - [Architettura generale](#)
 - [Nomi applicazioni](#)
 - [Architettura Multiax](#)
- [Descrizione applicativi](#)
 - [MP.MONO.ADAPTER](#)
 - [LUA Rules Scripts](#)
 - [MP.MONO.DECODER](#)
 - [MP.MONO.ANALYZER](#)
 - [MP.MONO.UI](#)
 - [MP.MONO.SIM](#)
- [Formato Dati](#)
 - [RAW DATA](#)
 - [Processed Data](#)
- [Link](#)

Architettura

Architettura generale

Di seguito lo schema che indica l'architettura generale della soluzione e delle varie componenti in gioco.



Il sistema è pensato per essere installato su un architettura windows del PC bordo macchina o su un PC Linux a corredo (in toto o in parte).

- Acquisizione dati tramite adapter (che cambierà secondo ogni specifica macchina / PLC / CNC / protocollo)
- Pubblicazione su server REDIS
- Decodifica dati REDIS --> DB
- Presentazione web con dotNet core app
- Analyzer/storicizzazione dotnetcore
- Scripting x processing dati basato su LUA scripts (DECODER/ANALYZER)

Nomi applicazioni

Al momento (nov 2022) sono previsti vari livelli applicativi:

- D-MAINT (dashboard limitato alla sola gestione manutenzione)
- D-MONO (MP-MONO; dashboard completo)
- G-MAINT (sistema di rete limitato alla sola gestione manutenzione, ex WebGIM)
- G-SHOP (sistema integrato x tutto lo shopfloor dei vari singoli items delle amcchine = MAPO standard)

Architettura Multiax

Nello specifico caso di Jeldweb l'architettura prevede:

- PLC OSAI OPEN
- Presentazione dati tramite server OPC-UA di Osai
- Acquisizione dati tramite adapter
- Pubblicazione su server REDIS
- Decodifica dati REDIS --> DB
- Presentazione web con dotNet core app
- Analyzer/storizzazione dotnetcore
- Scripting x processing dati basato su LUA scripts (DECODER/ANALYZER)

Nel caso specifico di Multiax x Jeldwen si è proceduto con 1 singolo PC / 2 pc windows + linux (da verificare/completare)

Descrizione applicativi

Di seguito la descrizione funzionale di ogni applicativo del progetto

MP.MONO.ADAPTER

E' il primo programma che dialoga a 2 vie con la macchina per leggere/scrivere dati. Idealmente si occupa di tenere in sync le aree di memoria disponibili sulla macchina con le corrispettive aree di memoria in REDIS, in modo che tutta la soluzione successiva possa diventare "agnostica" riguardo al tipo di CNC/PLC/meccanismo di comunicazione. Ovviamente si tratta di dati grezzi, la loro interpretazione e processing, che può essere specifica x vendor o addirittura x gamma/modello/singolo impianto, sarà svolta dagli script LUA di seguito descritti.

In REDIS deve riportare sia lo schema della struttura dei dati in una posizione standard (che possa venire deserializzato) che i dati stessi, in questo modo a valle il sistema sarà in grado di leggere lo schema, capire quali dati sottoscrivere/gestire e da qui in avanti processare il tutto.

LUA Rules Scripts

Questo strato SW corrisponde alla parte che di volta in volta va personalizzata poiché dai dati grezzi si definiscono gli algoritmi che costruiscono i dati processati/consolidati.

E' l'equivalente, nell'architettura di MAPO MES completo, della definizione dei files rul che creano la tabella transizione microstati/eventi e allos tesso tempo sostituisce la seconda tabella transizione eventi/stati.

Gli script sono il set di regole che, dati i segnali/parametri grezzi, portano a consolidare i dati ad alto livello, e peremttono eventualmente il ricalcolo a posteriori (a patto di avere ancora le serie di dati grezzi di partenza).

Questi saranno quindi chiamati dal DECODER e dall'ANALYZER.

MP.MONO.DECODER

E' la componente sw che si occupa, in realtime (possibilmente per sottoscrizione notifiche evento cambio valori in redis o per polling molto rapido) di valutare le informazioni e tramite le regole definite dagli script LUA

- Trasformare informazioni grezze in Eventi e Stati
- Scrivere dati realtime elaborati in Redis per "consumo immediato"
- Consolidare su DB i dati ricevuti
- Salvare su DB/file anche i dati grezzi (per permettere successiva rielaborazione) - tipicamente questa possibilità (come finestra temporale mantenuta e quantità massima di dati storicizzabile) è massima in fase di tuning/setup e poi viene ridotta in produzione ad un periodo minimo di pochi giorni.

MP.MONO.ANALYZER

E' il sw che, periodicamente, si occupa di effettuare la conversione dei dati di dettaglio (da Redis e soprattutto dal DB), in dati che siano statistiche a consuntivo più sintetiche (es: consuntivi per commessa, statistiche orarie, ...).

Il processing avviene tramite gli script LUA condivisi con il DECODER e permette quindi una riquilifica a posteriori (a patto di avere ancora i dati grezzi e/o intermedi).

MP.MONO.UI

E' la parte dell'applicativo che viene vista dall'utente finale con cui questo interagisce. La sua funzione è presentare le informazioni realtime e di eprmettere il recupero, il display grafico e testuale di informazioni storizzate, secondo diverse categorie

- informazioni in realtime sullo stato macchina
- informazioni di produzione
- informazioni di stato macchina
- log allarmi
- diario eventi registrato da utente

Il tutto con un focus "locale" (è un singolo impianto/macchina) e limitato nel tempo x la quantità di dati di dettaglio e/o statistici mantenuti.

da installare dotnet 6 hosting:

```
choco install dotnet-6.0-windowshosting
```

MP.MONO.SIM

E' una console app pacchettizzata copme single file app (che richiede installazione dotNet framework...) fatta x simulare i dati in fase di sviluppo UI. Per testare è stata installata su IOB-WIN-SIMULA, come servizio tramite nssm.

Per installarla si prende il risultato della fase di pubblicazione di SingleApp.pubxml che prevede

- conf: release
- target framework: dotnet 6
- runtime: win-x64
- formato: single file senza framework 8da instalalre a parte
 - ad es con chocolatey: **choco install dotnet-6.0-runtime**
- Compilazione formato ReadyToRun

Questa folder va messa sul pc da cui avviare e poi dalla cartella dove c'è tutot (compreso file start.bat e nssm.exe) eseguire i seguenti comandi

```
nssm.exe install MP.MONO.SIM
```

e poi si seguono lke iscruzioni della procedura interattiva come descritto qui: <https://nssm.cc/usage>

Per la procedura di setup single exe vedere qui:

- <https://dotnetcoretutorials.com/2021/11/10/single-file-apps-in-net-6/>
- <https://docs.microsoft.com/en-us/dotnet/core/deploying/single-file/overview>

Per la procedura di creazione e gestione servizi (qui con nssm) vedere qui:

- <https://www.ben-morris.com/running-a-net-core-console-application-as-a-windows-service/>
- <https://www.initpals.com/net-core/net-core-console-app-as-a-windows-service/>
- <https://nssm.cc/>
- <https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/windows-service?view=aspnetcore-6.0&tabs=visual-studio>
- <https://docs.microsoft.com/en-us/dotnet/core/extensions/generic-host>

Formato Dati

Di seguito indicazione della struttura dati scambiati, sia come posizione (dove si trovato) che come definizione/formato.

RAW DATA

I Raw data provengono dall'adapter e devono essere inviati nel modo + rapido a REDIS in due modalità:

- invio come "setup preliminare" (esempio: conf memorie allarme, che ADAPTER scrive in REDIS)
- invio come hash serializzati (chiave + valore = dato serializzato)
- pubblicaizione in apposito canale PUB/SUB di REDIS per implementare una gestione eventi realtime multiclient rapida e da overhead contenuto

Ad esempio il dato grezzo di una variabile letta dal PLC/CNC e notificato è un dato grezzo che può rappresentare ad esempio

- codice stato macchina/impianto
- codice modo macchina/impianto
- codice stato allarmi (tipo bitmap)
- posizione asse

Processed Data

I dati processati rappresentano l'informazione finale destinata alla Web.UI e quindi sono elaborazioni a partire da eventi ricevuti per pub/sub o polling.

- vengono inviati alla UI con canali pub/sub dedicati (disaccoppiamento)
- vengono salvati come hash serializzati "pronti all'uso"

ad esempio i tracciati **MachineDTO** sono dati che rappresentano informazioni di configurazione statiche (costruttore, modello, sn macchina...) in aggiunta a informazioni dinamiche tipo enum per rappresentare MODO e STATO macchina.

Link

Link utili:

- OPC-UA
 - <https://opcfoundation.org/products/view/opc-ua-net-sdk-for-client-and-server>
 - <https://opcua.traeger.de/en/>
 - <https://libraries.io/nuget/OPCFoundation.NetStandard.Opc.Ua.Core>