

Simulatore Metrologico

Per poter sviluppare la soluzione di gestione del PLC metrologico è stato approntato un simulatore in Python3, tramite Snap7 library, che si comporta a grandi linee in modo analogo al PLC.

Si è lavorato aprtendo dalle librerie e dalle implementazioni di Davide Nardella da questi progetti:

- <https://snap7.sourceforge.net/>
- <https://github.com/davenardella/>
- <https://github.com/davenardella/snap7>
- <https://github.com/davenardella/Sharp7>
- <https://github.com/davenardella/snap7.examples>
- <https://github.com/davenardella/SnapTRAINER>
-

Modalità implementazione

Dopo aver approfondito lo stato del progetto snap7, si è inizialmente testato simulatore e client binari disponibili, insieme ad acquirettore base Egalware. Da qui si è scelto di usare le librerie per sviluppare un piccolo python script (adatto a ubuntu 20.04 e successivi, funzionante tramite WSL in locale) che simulasse il server S7 con un comportamento ciclico adatto a far funzionare il server asp.net core e debuggare

Specifiche Memorie

Le specifiche preliminari del simulatore (sviluppato inizialmente con gemini) sono state le seguenti:

1. DB100 composta di 64 valori iniziali (principalmente bitmap e numerici) + 4 valori string da 30 char per un totale di 192 byte
2. DB100.DBD0 parte con 4 byte che rappresentano solo bit di stato, ad esempio il primo bit segnala lo stato acquisizione (1) e attesa (0); il 9 bit (il primo del secondo byte) va a 1 quando termina l'acquisizione ed i dati sulal DB101 sono pronti e disponibili.
3. da DB100.DBD4 partono una serie di valori REAL che sono parametri / contatori così organizzati
 - DB100.DBD4: Real, valore 0 (rappresenta valore iniziale parametro x, in caso rettilineo lunghezza)
 - DB100.DBD8: REAL valore 100 (rappresenta valore finale parametro x, in caso rettilineo lunghezza)
 - DB100.DBD12: Real, valore fisso es +1.8 (rappresenta limite superiore)
 - DB100.DBD16: Real, valore fisso es -1.8 (rappresenta limite inferiore)
 - DB100.DBD20: Real, (parametro libero) - es tipo di lettura (1 = lineare, 2 = cerchio, 3 = arco...
 - DB100.DBD24: Real, (parametro libero) - es numero di letture previste/effettuate dato il tipo
 - DB100.DBD28: Real, (parametro libero) - es parametro x calcolo arco...
 - DB100.DBD32: Real, (parametro libero)
 - DB100.DBD36: Real, (parametro libero)
 - DB100.DBD40: Real, (parametro libero)
 - DB100.DBD56: Real, valore LIVE che che rappresenta la durata del processo in secondi oppure il valore di x raggiunto al momento del salvataggio
 - DB100.DBD60: REAL, valore LIVE letto (y)

4. da DB100.DBD64 partono 4 blocchi string da 30 char (32 byte totali in formato S7) che rappresentano

- DB100.DB64: string(30) Commessa
- DB100.DB96: string(30) Articolo
- DB100.DB128: string(30) IdControllo
- DB100.DB160: string(30) altro?

5. la DB101 viene popolata di 4000 REAL (o quanti ne vuoi/puoi) valori che corrispondono ai valori real ESATTAMENTE rilevati

Update memorie

Si vogliono aggiungere le seguenti are di memoria: DB900, DB901, DB920

Di seguito la mappa aggiornata delle DB impiegate: DB900, DB901, DB920.

Idealmente la DB900 contiene i dati della DB101, in formato DINT (quindi dovrebbero essere gli stessi dati moltiplicati per 1000 e arrotondati ad intero) e dove il primo dato conferma quanti valori sono stati scritti (del massimo di 4096 nella simulazione useremo i 20 creati per la DB101).

La DB901 prende il posto della DB100 come "stato del processo", ed i dati live anche qui diventano interi in micron per cui la x e la z live vanno moltiplicate per 1000 e arrotondate.

La DB920 è nuova e può essere gestita inserendo delle stringhe iniziali fisse (anche la stessa descrizione va bene) e per i valori DINT mettendo il numero della DB nel valore (es DBD160 --> metti 160). I due byte finali possono restare vuoti

DB900 (Dati di Processo) - 16kb

Indirizzo	Tipo Dati	Descrizione
DBD0-4	1 DINT	Numero di misure registrate nella DB (dimensione dell'array seguente di misure DINT)
DBD4-16387	4096 DINT	Array variabile fino a 4096 valori DINT (dimensione effettiva definita nel DBD0) - misure espresse in micron

NB: probabilmente va letto il primo valore per determinare dimensione e successivamente letto per blocchi il set di misure completo.

DB901 (Configurazione e Stato) - 38 Byte

Indirizzo	Tipo Dati	Descrizione
DBD0	Bit	Bit 0: Stato Ready Bit 1: Acquisizione (1=ON) Bit 2: Fine Acquisizione Bit 3: Dati Pronti su DB900
DBD1	Bit	ACK WRITE (da scrivere al PLC) per indicare lettura bit della cella precedente (stessa struttura interna)

Indirizzo	Tipo Dati	Descrizione
DBD2	Word	Sensore in uso: 1=Sensore SP1, 2=Sensore SP2, 3=Sensori SP1+SP2, 4=Sensori SP2-SP1
DBD4	Word	Tipo di misura: 0=Rettilinearità 1=Rotondità
DBD6	REAL	Freccia Bombatura: 0=Cilindro, >0=Bombatura Convessa, <0=Bombatura Concava
DBD10	Word	Numero di misure di rotondità richieste
DBD12	Word	Numero di misura attuale di rotondità
DBD14	REAL	Lunghezza Pezzo totale
DBD18	REAL	Lunghezza Misura pezzo
DBD22	DINT	Valore Attuale Misura (real time) in Micron
DBD26	REAL	Posizione longitudinale della misura Asse Z (real time)
DBD30	DINT	Numero totale di punti misurati
DBD34	DINT	Numero Pointer ultimo punto misurato

DB920 Gestione Certificato - 186 Byte

Indirizzo	Tipo Dati	Descrizione
DBD0	String(30)	Nome Cliente
DBD32	String(30)	Articolo Codice Pezzo
DBD64	String(30)	Matricola Pezzo
DBD96	String(30)	Nome Operatore
DBD128	String(30)	Nome Supervisore
DBD160	DINT	Codice Produzione PEP
DBD164	DINT	Numero di Certificato
DBD168	DINT	Massima Tolleranza Rettilinearità (micron)
DBD172	DINT	Massima Tolleranza Rotondità (micron)
DBD176	DINT	Errore Riscontrato di Rettilinearità (micron, max)
DBD180	DINT	Errore Riscontrato di Rotondità (micron, max)
DBD184	Bit	Bit 0: Attivazione Pagina Certificato (1=ON) Bit 1: Richiesta salvataggio certificato (1=ON)
DBD185	Bit	WRITE ACK (risposta a PLC per completamento azioni) Bit 0: Conferma Salvataggio Certificato Richiesta (1=ON)

Da valutare gestione in 2 blocchi, quello iniziale delle string e poi il resto di tipo misto

Uso script

Per eseguire su WSL basta installare python, eventuali pacchetti con pip ed è funzionante. `sudo apt install python3-pip sudo pip install python-snap7 sudo python3 server_s7.py`

nb: serve sudo x esecuzione su porta 102, bassa/protetta. Lo script da usare è la versione 3.8 se ubuntu 20.04 con python 3.8 oppure va aggiornato script (o usata vers 3.13 se fosse quella)

meglio però impiegare ambienti dedicati tramite uv (valido anche in ambiente windows) in questo modo (si da per scontato uv sia disponibile...)

- inizializzare ambiente venv con uv

```
uv venv
```

- installare pacchetto snap7

```
uv pip install python-snap7
```

- avviare script launcher che seleziona in base alla versione python disponibile

```
uv run .\launcher.py
```