

Upgrade sw IOB-PI clienti

ATTENZIONE: se si copia dalla versione standard, SUL PI VA DATO un dos2unix * x tutti i files..

Salt install##

Se il minion è taggato come iop-pi bast dare state apply, poiché dallo stato top.sls è configurato di fare gli stati

- iob-pi.sls
- iob-setup.sls

approfondimenti

Valutato python-rq x gestione code con redis tramite esecuzione workers python gestiti in autonomia:

<https://python-rq.org/> <https://python-rq.org/docs/workers/>

Testato ma al momento non è rispondente ai requisiti del programma per cui si rischia un blocco inf ase di invio se server non OK, per ora implementata copia "manuale" della coda verso redis per salvaguardare dati accumulati (vers 2.6.x)

MD5 zip##

creazione MD5: da powershell

```
Get-FileHash .\Mapo.zip -Algorithm MD5
```

ReadParallela IOB-PI

Overview

readParallela.py is a high-frequency data acquisition script designed for Raspberry Pi. It interfaces with a custom GPIO board to read 12 parallel digital inputs, applies various signal processing filters, and queues events for asynchronous transmission to a remote server via Redis and HTTP.

Panoramica

readParallela.py è uno script di acquisizione dati ad alta frequenza progettato per Raspberry Pi. Interfaccia una scheda GPIO personalizzata per leggere 12 ingressi digitali paralleli, applica vari filtri di elaborazione del segnale e mette in coda gli eventi per la trasmissione asincrona a un server remoto tramite Redis e HTTP.

Core Functionality

1. Signal Acquisition & Processing

The script polls 12 GPIO pins at a configurable interval (**SAMPLETIME**). For each input, it applies a three-stage processing pipeline:

- **Inversion:** Reverses the signal logic if configured for a specific bit.
- **Short Signal Filtering (Debounce):** Filters out brief electrical spikes or noise by requiring a signal to remain stable for a specific number of cycles (`MAX_COUNTER_FILTER`).
- **Blinking Logic:** Detects and manages "blinking" signals, allowing for specific timing-based event detection.

2. Data Packaging

Once processed, the 12 bits are aggregated into a single integer value and converted into a Hexadecimal string.

3. Event Queuing & Transmission

To ensure high-speed reading is not interrupted by network latency:

- **Redis Queue:** Detected changes (and timeout events) are timestamped and pushed into a Redis list (`IOB`).
- **Asynchronous Sender:** A dedicated background thread (`svuotaCoda`) periodically dequeues items from Redis and performs HTTP requests to a central server.
- **Reliability:** The script implements a "retry" mechanism and monitors server availability (`URLALIVE`) to manage offline periods.

4. Timeout Management

The script tracks two types of timeouts:

- **Short Timeout:** Ensures that a signal state is recorded even if it changes quickly.
- **Long Timeout:** Provides a fallback to ensure state consistency over longer periods.

Funzionalità Principali

1. Acquisizione e Elaborazione Segnali

Lo script interroga 12 pin GPIO a un intervallo configurabile (`SAMPLETIME`). Per ogni ingresso, applica una pipeline di elaborazione a tre stadi:

- **Inversione:** Inverte la logica del segnale se configurata per un bit specifico.
- **Filtraggio Segnali Brevi (Debounce):** Filtra i picchi elettrici o il rumore richiedendo che il segnale rimanga stabile per un numero specifico di cicli (`MAX_COUNTER_FILTER`).
- **Gestione Blinking:** Rileva e gestisce segnali "blinking", permettendo il rilevamento di eventi basati su timing specifici.

2. Impacchettamento Dati

Una volta elaborati, i 12 bit vengono aggregati in un unico valore intero e convertiti in una stringa Esadecimale.

3. Code e Trasmissione Eventi

Per garantire che la lettura ad alta velocità non sia interrotta dalla latenza di rete:

- **Coda Redis:** Le variazioni rilevate (e gli eventi di timeout) vengono datate e inserite in una lista Redis (**IOB**).
- **Invio Asincrono:** Un thread dedicato in background (**svuotaCoda**) estrae periodicamente gli elementi dalla coda Redis e esegue richieste HTTP a un server centrale.
- **Affidabilità:** Lo script implementa un meccanismo di "retry" e monitora la disponibilità del server (**URLALIVE**) per gestire i periodi offline.

4. Gestione Timeout

Lo script monitora due tipi di timeout:

- **Timeout Breve:** Assicura che lo stato di un segnale venga registrato anche se cambia rapidamente.
- **Timeout Lungo:** Fornisce un sistema di fallback per garantire la coerenza dello stato su periodi più lunghi.

Implementation Details (Refactored Version)

The script has been modernized with the following improvements:

- **Object-Oriented Design:** Logic is encapsulated within a **ReadParallelaIOB** class, eliminating reliance on dangerous global variables.
- **Efficient Bitwise Operations:** Replaced manual **if** checks with bitwise operators for faster bit reconstruction.
- **Modern Networking:** Replaced **urllib** with the **requests** library for more robust and readable HTTP communication.
- **Optimized Configuration:** Uses loops to load per-bit settings from the configuration file.

Dettagli di Implementazione (Versione Refactored)

Lo script è stato modernizzato con i seguenti miglioramenti:

- **Design Orientato agli Oggetti:** La logica è incapsulata all'interno della classe **ReadParallelaIOB**, eliminando la dipendenza da variabili globali pericolose.
- **Operazioni Bitwise Efficienti:** Sostituzione dei controlli manuali **if** con operatori bitwise per una ricostruzione dei bit più veloce.
- **Networking Moderno:** Sostituzione di **urllib** con la libreria **requests** per una comunicazione HTTP più robusta e leggibile.
- **Configurazione Ottimizzata:** Utilizzo di cicli per caricare le impostazioni per singolo bit dal file di configurazione.

Configuration

Settings are managed via **IOB.cfg**, including:

- **[time]:** Sampling rates and timeouts.
- **[id]:** Machine identifier.
- **[web]:** URLs for data transmission and connectivity checks.
- **[blink], [invert], [filter]:** Per-bit configuration for signal processing.
- **[log]:** Logging verbosity and file paths.

Configurazione

Le impostazioni sono gestite tramite `IOB.cfg`, tra cui:

- `[time]`: Frequenze di campionamento e timeout.
- `[id]`: Identificativo macchina.
- `[web]`: URL per la trasmissione dati e controlli di connettività.
- `[blink]`, `[invert]`, `[filter]`: Configurazione per singolo bit per l'elaborazione dei segnali.
- `[log]`: Livello di logging e percorsi dei file.

Dependencies

- `RPi.GPIO`: For hardware interfacing.
- `redis`: For local event queuing.
- `requests`: For web communication.
- `configparser`: For configuration management.

Dipendenze

- `RPi.GPIO`: Per l'interfacciamento hardware.
- `redis`: Per la gestione della coda locale.
- `requests`: Per la comunicazione web.
- `configparser`: Per la gestione della configurazione.