

# MAPO-IOB-WIN-PY

---

Progetto per porting script Python della comunicazione con schede IOB seriali (versione iniziale 2006).

Funzionato fino al 2024 tramite PC winXP locali e seriale locale, poi passati a Moxa + seriali virtuali e IOB dedicato (dove chiama Moxa sia x part program e d\_cx che x IOB-WIN python)

## Specifiche tecniche

il progetto prevede il sorgente python che viene poi compilato, tramite PyInstaller, in un EXE windows che è monolitico e si occupa di tutto il programma (semplificando la gestione)

E' un progetto separato e a se stante rispetto ai normali IOB-WIN-\* perché

- non è compilato da VisualStudio
- non usa dotNet Framework (ma appunto python)
- non comunica via rete direttamente ma tramite seriale (remotata via Moxa dal 2024)

## Setup preliminare

Va selezionato in VSCode l'interprete python corretto (3.11 al momento della stesura), tramite **CTRL+Shift+P** e poi *Python: Select Interpreter*

Vanno installate eventuali dipendenze (es **pip install redis**) x moduli non standard

## Modifiche principali

Le modifiche principali per passare dalla versione WinXp (interpretata) alla nuova (compilata) hanno riguardato

- migrazione Python2 (2.7) --> Python3 (3.11)
- impiego di VSCode come editor e debugger soluzione
- revisione gestione parser e lettura configurazione
- nuova configurazione logfile (con rotate gestito dalla libreria)
- introduzione librerie x Redis
- gestione oggetto stato IOB da serializzare su Redis x comunicazione WatchDog con IOB-WIN
- intrusione compilazione del python con PyInstaller
- configurazione compilazione + copia conf tramite script bat (**createExe.bat**)

## Creazione exe

Impieghiamo PyInstaller.

sito: <https://pyinstaller.org/en/stable/usage.html>

Deve essere aggiunto tramite pip sulla macchina di build

```
pip install PyInstaller
```

Viene generato exe con il comando PyInstaller che nella versione minimale è qualcosa come

```
python -m PyInstaller IOB-WIN-PSER-TEST.py
```

Se si volesse generare un unico file exe + icona la sintassi diventa

```
python -m PyInstaller --onefile --icon SteamWare.ico IOB-WIN-PSER.py
```

Recupero ed estrazione file versione

```
pyi-grab_version.exe .\dist\IOB-WIN-PSER-vers.exe
```

Reimpostazione file versione

```
pyi-set_version.exe file_version_info_new.txt dist\IOB-WIN-PSER.exe
```

Il file exe + le varie librerie accessorie saranno salvate in **dist/nome\_progetto/**

NB: bisogna poi a mano copiare i file di conf, ma non nella cartella principale ma in **\_internal**, ovvero **dist/nome\_progetto/\_internal**

Esempio bat finale:

```
REM script creazione EXE da python a mano

REM setup preliminare virtual environment x lavorare sul progetto se mancasse
REM python -m venv venv

REM attivazione virtual env specifico
REM .\venv\Scripts\Activate.ps1
REM install componente
REM pip install PyInstaller

REM estrazione file version (come file_version_info.txt)
REM pyi-grab_version.exe .\dist\IOB-WIN-PSER-vers.exe

@REM python -m PyInstaller --onefile --icon SteamWare.ico IOB-WIN-PSER-TEST.py
python -m PyInstaller --onefile --icon SteamWare.ico IOB-WIN-PSER.py

REM effettua copia delle conf x poter avviare...
xcopy CONF\*.cfg dist\CONF\
```

```
REM editing file versione con valori vers corrente x dataOra
REM reinserimento info versione da file editato (file_version_info_new.txt)
REM pyi-set_version.exe file_version_info_new.txt dist\IOB-WIN-PSER.exe
```

## Integrazione in yaml di CI/CD

A partire da gennaio 2025 è stato separato il progetto e costruito un apposito pipeline di compilazione dello stesso.

Per farlo nel file yaml sono specificati i comandi per

- impiegare il corretto virtual environment
- compilare l'eseguibile dallo script python
- sistemare il numero di versione nel file template
- impostare il manifest della versione windows nell'exe finale
- zippare ed inviare a nexus versione compilata
- registrare sul liman disponibilità della release compilata

## Lancio dell'eseguibile

L'eseguibile è richiamabile direttamente o tramite BAT, si aspetta in ingresso un parametro, ovvero il nome del file \*.cfg con cui avviarsi, che conterrà

- configurazioni base x gestione letture varie
- parametri connessioni porta comm(virtuale tramite MOXA)
- parametri accessori x log, ...

## Integrazione con IOB-MAN

Per poter venire integrata con IOB-MAN è necessario che ci sia una specifica modalità di comunicazione, tramite Redis, che ritorna lo stato dell'adapter al manager per effettuare le procedure di watchdog ed evitare riavvio automatico per mancata comunicazione.

Si è impiegata questa calsse helper serializzata e salvata su redis con un thread separato

```
class IobWinStatus:
    def toJSON(self):
        return json.dumps(self, default=lambda o: o.__dict__, sort_keys=True,
indent=4)
    #stream = 'cse' # Class Variable
    def __init__(self,codIob,iobType):
        self.CodIob = codIob
        self.IobType = iobType
        self.counterIOB = 0
        self.counterMAC = 0
        self.freeNotes = ''
        self.lastDataIn = ''
        self.lastDataOut = ''
        self.lastUpdate = ''
        self.online = False
```

```

self.queueAllLen      = 0
self.queueEvLen       = 0
self.queueFlLen       = 0
self.queueMsLen       = 0
self.queueRawTransfLen = 0
self.queueUllLen      = 0
self.semIn            = 'ND'
self.semOut           = 'ND'

```

Periodicamente si aggiornano indicazioni dei valori **lastDataIn** (Out,...) e il boolean di **online** e poi serializza su redis

## Configurazione IOB-MAN

Vers <= 3.6

IOB Man fino alla versione 3.6 va configurato in modo che vada correttamente a lanciare gli eseguibili **IOB-WIN-PSER** (ovvero del compilato Python-Seriale), andando ad agire sul file \*.config:

```

<appSettings>
  ...
  <add key="appNameExt" value="IOB-WIN-PSER" />
  ...
  <add key="targetExe" value="C:\Temp\CNC_XP\IOB-WIN-PSER\IOB-WIN-PSER.exe" />
  <add key="TargetNLogConf" value="C:\Temp\CNC_XP\IOB-WIN-PSER\CONF\NLog.config"
/>
  <add key="TargetLogDir" value="C:\Temp\CNC_XP\IOB-WIN-PSER\logs\" />
  ...
  <add key="BaseArg" value="" />
  ...
</appSettings>

```

ottenendo alla fine qualcosa come

```

<appSettings>
  <add key="appName" value="IOB-MAN" />
  <add key="appNameExt" value="IOB-WIN-PSER" />
  <add key="uiPeriod" value="100" />
  <add key="checkPeriod" value="3000" />
  <add key="forceCheckPeriodMult" value="20" />
  <add key="autoRestartTimeoutMin" value="15" />
  <add key="autoStartProc" value="true" />
  <add key="closeOnChildUpdate" value="true" />
  <add key="targetExe" value="C:\Temp\CNC_XP\IOB-WIN-PSER\IOB-WIN-PSER.exe" />
  <add key="TargetNLogConf" value="C:\Temp\CNC_XP\IOB-WIN-PSER\CONF\NLog.config"
/>
  <add key="TargetLogDir" value="C:\Temp\CNC_XP\IOB-WIN-PSER\logs\" />
  <add key="ApiUrl" value="https://liman.egalware.com/ELM.Api/" />
  <add key="BaseArg" value="" />

```

```

    <add key="ArgsConfFile" value="/CONF/process.json" />
    <!--Gestione riavvio periodico: ora e min di avvio, periodo ripetizione in
minuti -->
    <add key="fullRestartHour" value="0" />
    <add key="fullRestartMinute" value="30" />
    <add key="fullRestartIntervMin" value="1440" />
    <add key="waitForExitMsec" value="250" />
    <!--gestione REDIS-->
    <add key="RedisConn" value="localhost,abortConnect=false,ssl=false" />
    <add key="RedisConnAdmin" value="localhost,abortConnect=false,ssl=false" />
    <add key="redisDb" value="10" />
    <add key="ClientSettingsProvider.ServiceUri" value="" />
</appSettings>

```

## Vers >= 3.7

Dalla versione 3.7 è permesso il lancio di vari applicativi differenti da un unico IOB-MAN. Per farlo correttamente va comunque imposta il (nuovo) file di configurazione json con le apposite sezioni di chiamata, ad esempio

```

{
  "ListIOB": {
    "L003": "IOB-WIN-PSER",
    "03": "d_cx",
  }
  "ListTarget": {
    "IOB-WIN-PSER": {
      "ExeName": "IOB-WIN-PSER",
      "ExePath": "C:\\Steamware\\IOB-WIN-PSER\\IOB-WIN-PSER.exe",
      "LogDir": "C:\\Steamware\\IOB-WIN-PSER\\logs\\",
      "NLogPath": "C:\\Steamware\\IOB-WIN-PSER\\CONF\\NLog.config",
      "BaseArgs": ""
    },
    "d_cx": {
      "ExeName": "d_cx",
      "ExePath": "C:\\Steamware\\d_cx\\d_cx.exe",
      "LogDir": "C:\\Steamware\\d_cx\\logs\\",
      "NLogPath": "C:\\Steamware\\d_cx\\NLog.config",
      "BaseArgs": ""
    }
  }
}

```

Dove nella prima aprte è indicato ogni IOB e quale exe chiamare, mentre nella seconda parte sono specificati i parametri di avvio dell'exe e in particolare i **BaseArgs** con cui avviare (per IOB-WIN-PSER sono vuoti come per d\_cx a differenza dei normali IOB-WIN-\*)

## Approfondimenti

Link vari impiegati per la preparazione

Di seguito pagine link varie usate x sistemare progetto

- <https://pyinstaller.org/>
- <https://pyinstaller.org/en/stable/usage.html>
- <https://datatofish.com/executable-pyinstaller/>

Valutati ma non impiegati

- <https://github.com/Nuitka/Nuitka>
- <https://pypi.org/project/py2exe/>

Version

Date	Vers	Note
2006..	1.0..	Prima versione in DotNet, PC locale WinXP con stati locali
2018..	2.0..	Versione Python locale vers 2.7, sempre WinXP
2024.06.01	2.8.2.*	Versione Python vers 3, Compilata con PyInstaller, per gestione tramite IOB-MAN + Moxa
2025.01.27	2.9..	Separazione progetto a se stante, versione con gestione del numero versione x File e Applicazione (x integrazione con LiMan e EgwAppControlCenter)