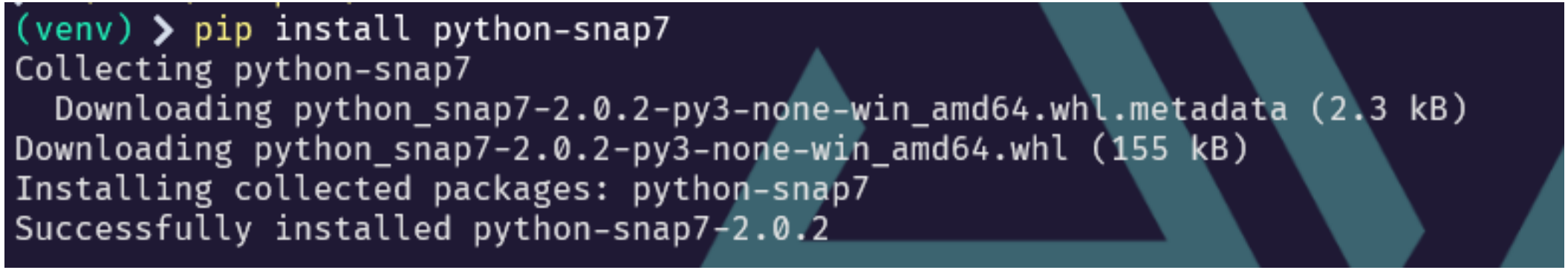
Práctica GET/PUT Python

José Pablo Hernández Alonso – JPHAJP

Portafolio Temas Selectos de Mecatrónica

https://jphajp.github.io/Simens_PLC_Comms/index.html

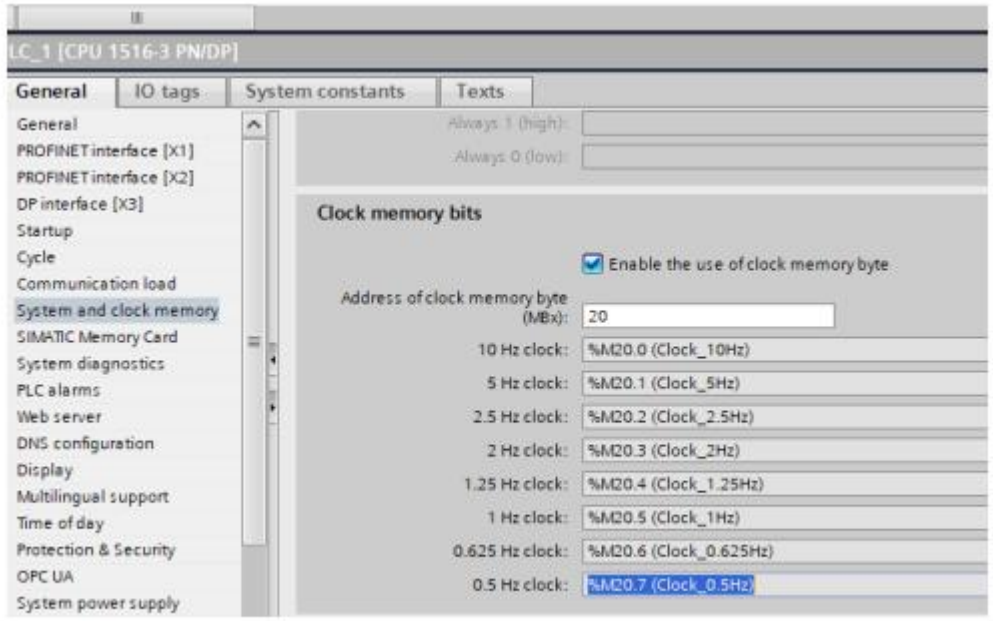
Instalación snap7 en computadora

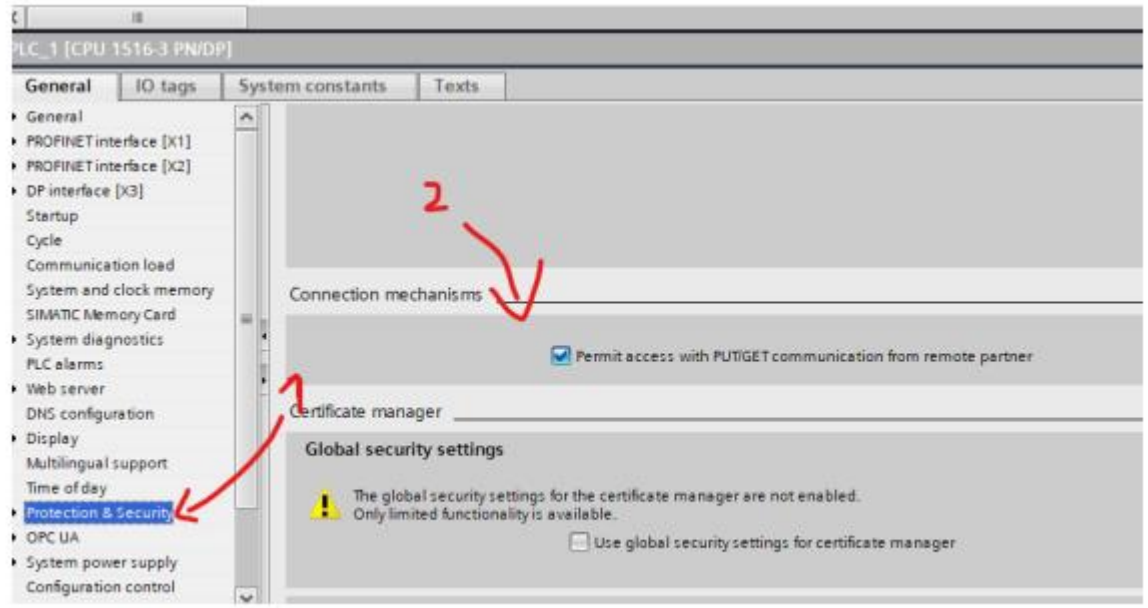


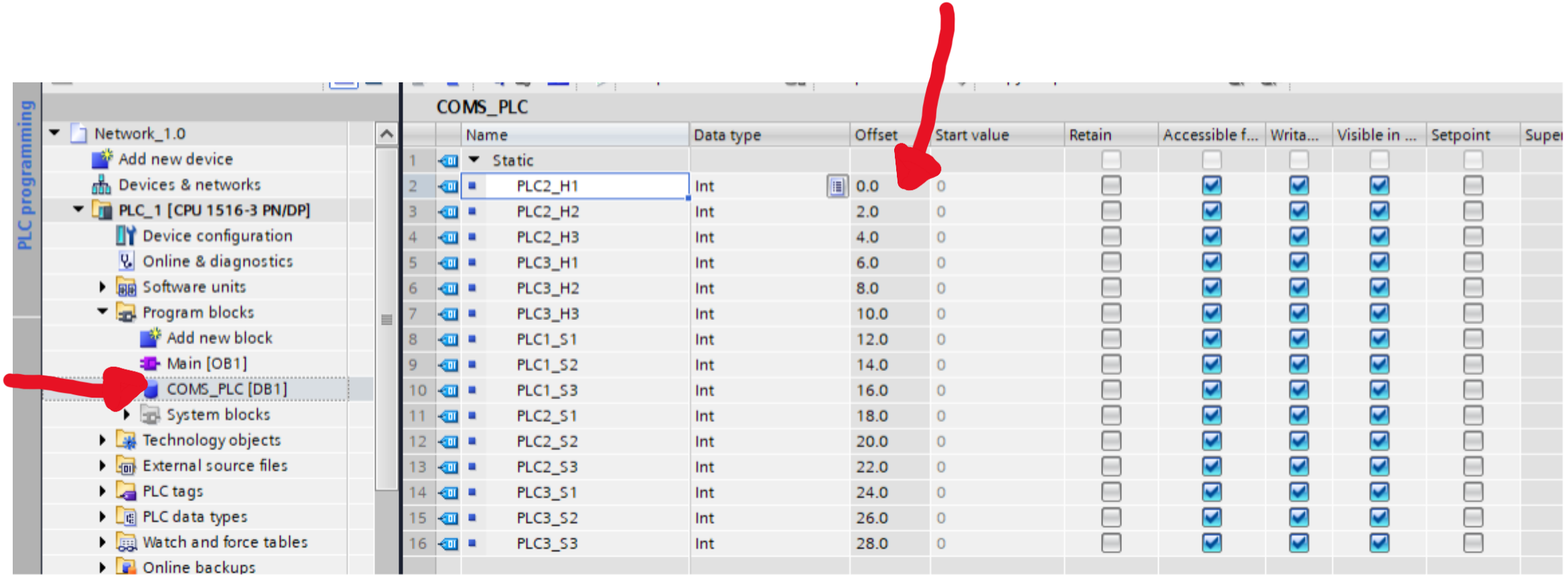
Comando:

pip install python-snap7

Configuración de reloj y permiso PUT/GET







Crear una DB, sin optimización para conseguir tener valores de offset.

```python
from snap7 import Client
from snap7.util import get_bool, set_bool, get_uint, set_uint
# 1) Define aquí tus PLCs
PLCS = [
    {'name': 'PLC1', 'ip': '192.168.0.1', 'rack': 0, 'slot': 1}
    #{'name': 'PLC2', 'ip': '192.168.0.2', 'rack': 0, 'slot': 1},
    # … añade tantos como necesites
]
# 2) Define aquí los “jobs” que quieres hacer:
#    cada dict indica: en qué PLC ('plc'),
#    qué DB (‘db’), byte de inicio (‘byte’),
#    y si es BOOL → bit (‘bit’), si es UINT → usa el campo ‘type’: 'uint'
#    campo 'write_value' es el valor que vas a poner.
JOBS = [
    # Ejemplo: leer/escribir un BOOL en DB2.DBX0.0 de PLC1
    {'plc': 'PLC1', 'db': 2, 'byte': 0, 'bit': 0, 'type': 'bool',
'write_value': True},
    # Ejemplo: leer/escribir un UINT en DB2.DBW2  de PLC1
    {'plc': 'PLC1', 'db': 2, 'byte': 2,          'type': 'uint','write_value':
2}
    # Otro BOOL en PLC2.DB5.DBX10.3
    #{'plc': 'PLC2', 'db': 5, 'byte': 10, 'bit': 3, 'type': 'bool',
'write_value': False},
    # … añade los que hagan falta
]
def connect_plc(cfg):
    plc = Client()
    plc.connect(cfg['ip'], cfg['rack'], cfg['slot'])
    print(f"{cfg['name']}: conectado")
    return plc
def disconnect_plc(plc, name):
    plc.disconnect()
    print(f"{name}: desconectado")
```

Ver códigos completos en Github

```python
def process_job(plc, job):
    db, byte = job['db'], job['byte']
    if job['type'] == 'bool':
        # 1 byte contiene tu bool
        buf = plc.db_read(db, byte, 1)
        old = get_bool(buf, 0, job['bit'])
        print(f"→ Antes BOOL DB{db}.DBX{byte}.{job['bit']} = {old}")
        # escribe nuevo
        set_bool(buf, 0, job['bit'], job['write_value'])
        plc.db_write(db, byte, buf)
        print(f"← Después BOOL = {job['write_value']}\n")
    elif job['type'] == 'uint':
        # 2 bytes para uint
        buf = plc.db_read(db, byte, 2)
        old = get_uint(buf, 0)
        print(f"→ Antes UINT DB{db}.DBW{byte} = {old}")
        # genera un buffer limpio y escribe
        buf2 = bytearray(2)
        set_uint(buf2, 0, job['write_value'])
        plc.db_write(db, byte, buf2)
        print(f"← Después UINT = {job['write_value']}\n")
    else:
        raise ValueError("Tipo no soportado")
def main():
    # 1. Conecta todos
    clients = {}
    for cfg in PLCS:
        clients[cfg['name']] = connect_plc(cfg)
    # 2. Ejecuta los jobs
    for job in JOBS:
        plc = clients[job['plc']]
        process_job(plc, job)
    # 3. Desconecta todos
    for name, plc in clients.items():
        disconnect_plc(plc, name)

if __name__ == '__main__':
    main()
```

Este código de Python permite leer y escribir datos en uno o más PLCs Siemens S7 usando la biblioteca snap7, de forma automática y flexible.

PLCS = [{'name': 'PLC1', 'ip': '192.168.0.1', 'rack': 0, 'slot': 1}]
Aquí defines los PLCs a los que te vas a conectar.

- Cada PLC tiene:
  - ip: dirección IP del PLC
  - rack y slot: son valores que indican dónde está el CPU dentro del hardware del PLC (normalmente rack=0 y slot=1 para S7-1200/1500).

JOBS = [{'plc': 'PLC1', 'db': 2, 'byte': 0, 'bit': 0, 'type': 'bool', 'write_value': True}, {'plc': 'PLC1', 'db': 2, 'byte': 2, 'type': 'uint','write_value': 2}]
Cada tarea indica:

- plc: en qué PLC se aplicará.
- db: número de bloque de datos (DB) en el PLC.
- byte y bit:
  - Para bool: especifica byte y bit, como DB2.DBX0.0 (byte 0, bit 0).
  - Para uint: solo byte, como DB2.DBW2 (word en byte 2).
- type: tipo de dato a modificar (bool, uint...).
- write_value: valor que se va a escribir.

Funciones principales

connect_plc(cfg)

- Se conecta al PLC usando la IP, rack y slot.

disconnect_plc(plc, name)

- Cierra la conexión con el PLC.process_job(plc, job)
- Según el tipo de dato:
  - Si es bool: lee 1 byte del DB, cambia 1 bit, y lo vuelve a escribir.
  - Si es uint: lee 2 bytes, cambia el valor como unsigned int, y lo vuelve a escribir.
- Usa get_bool, set_bool, get_uint, set_uint de snap7.util.

- Ejemplos:

{'plc': 'PLC1', 'db': 2, 'byte': 0, 'bit': 0, 'type': 'bool', 'write_value': True}
modificará el bit 0 del byte 0 del DB2 en el PLC1 y lo pondrá en True (1).

{'plc': 'PLC1', 'db': 2, 'byte': 2, 'type': 'uint','write_value': 2}
escribirá el número 2 como unsigned int en los bytes 2 y 3 del DB2 del PLC1.

Documentación para TIA PORTAL 15.1