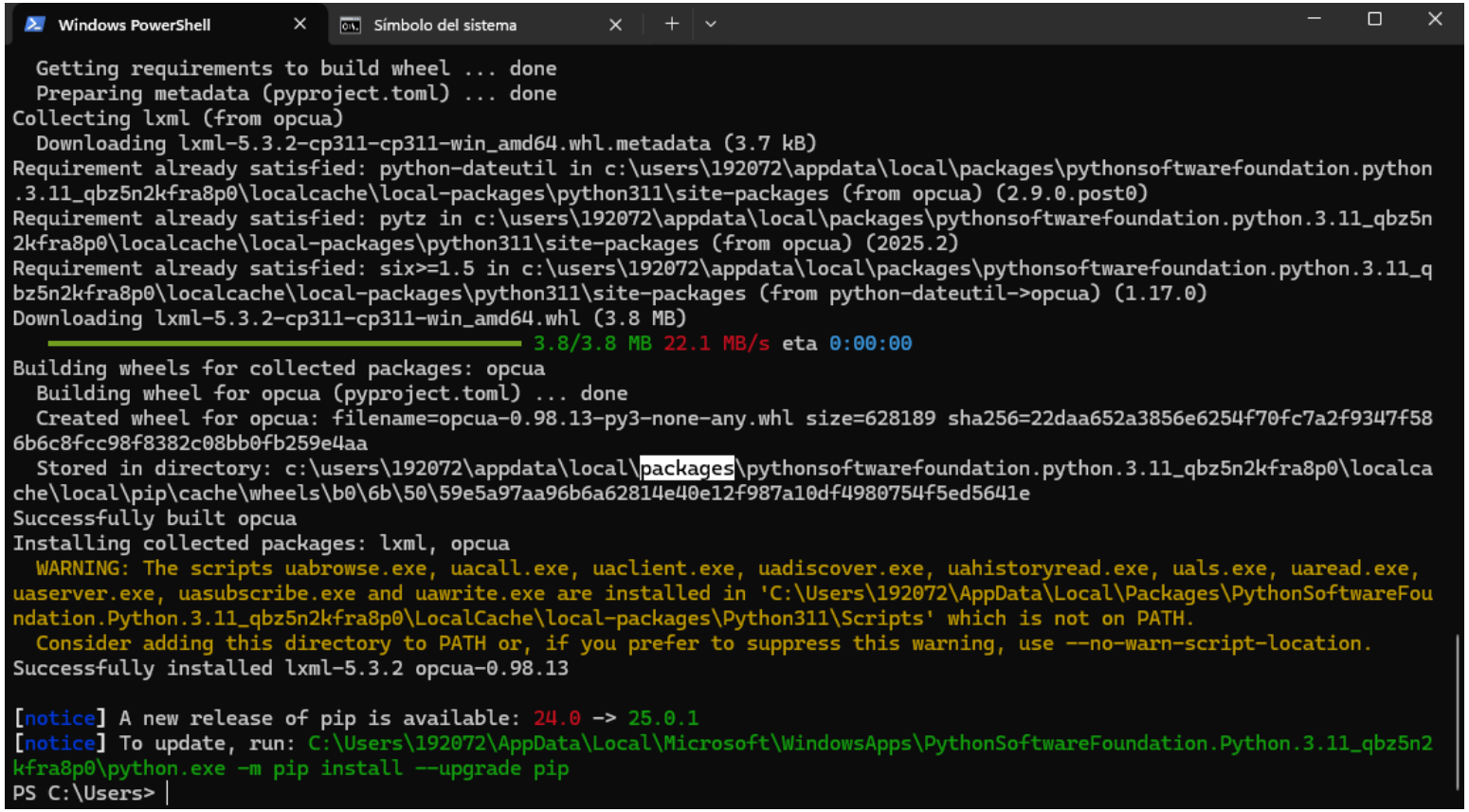
Práctica OPC Interfaz - python

José Pablo Hernández Alonso – JPHAJP

Portafolio Temas Selectos de Mecatrónica

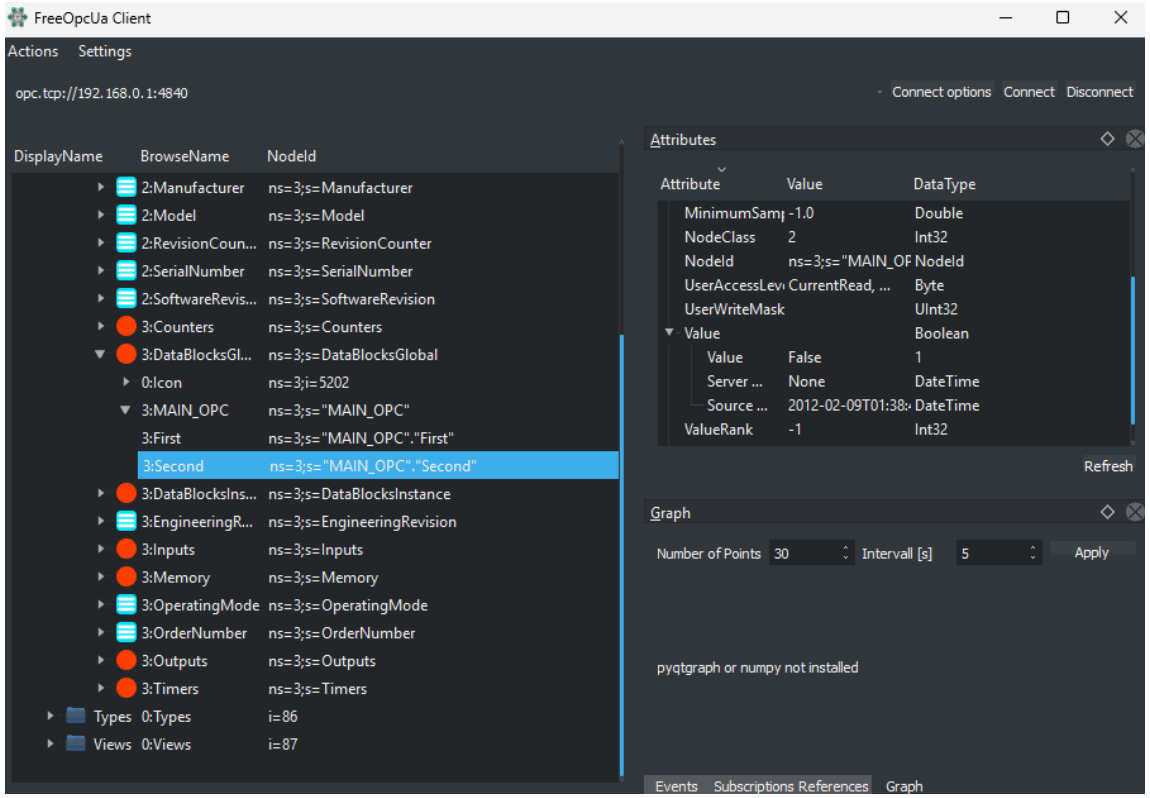
https://jphajp.github.io/Simens_PLC_Comms/index.html

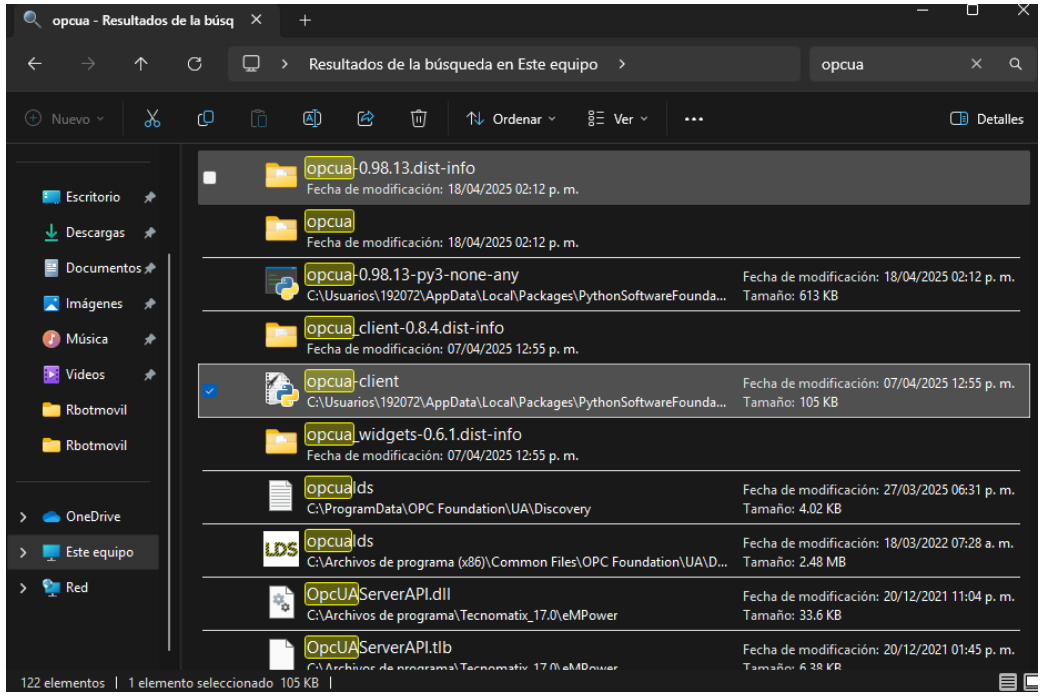
Instalación OPC en computadora



pip install opcua

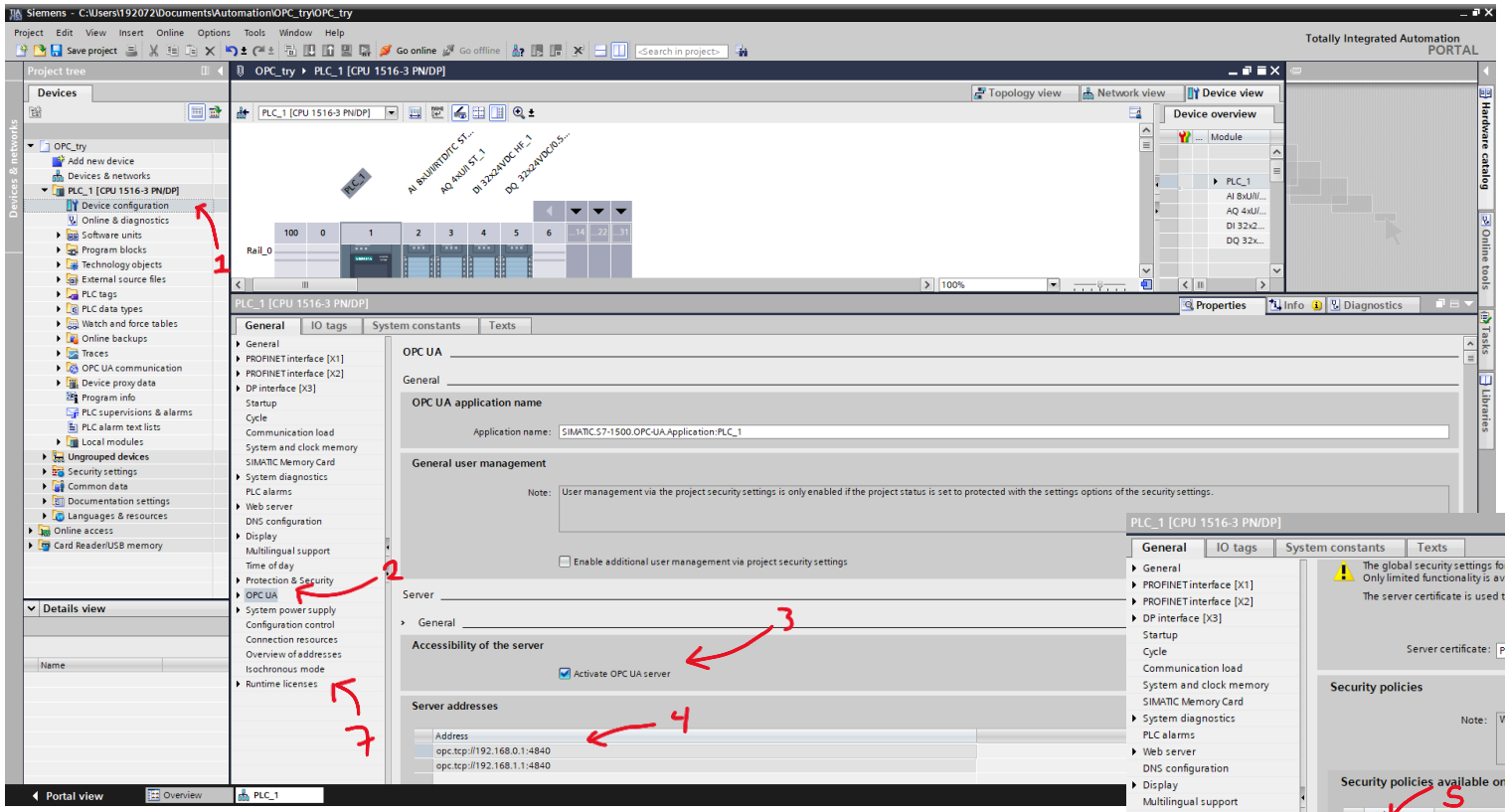
Inicializar cliente de OPC

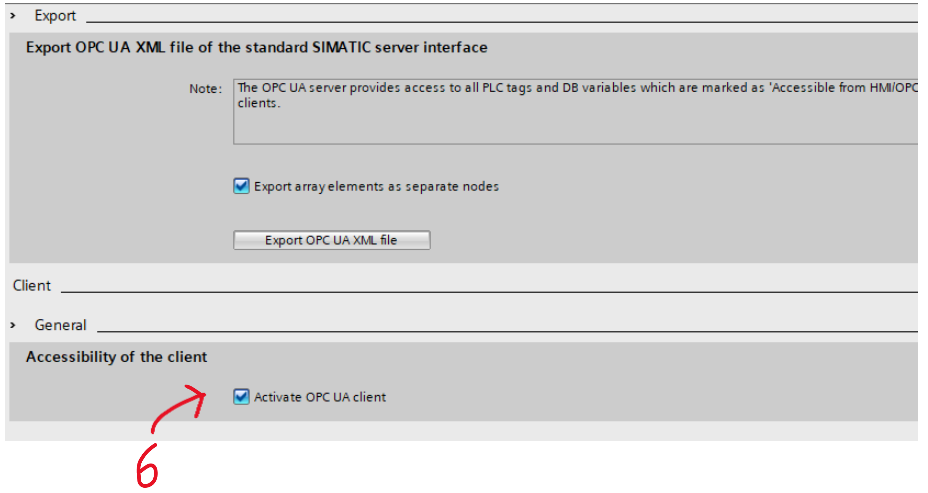


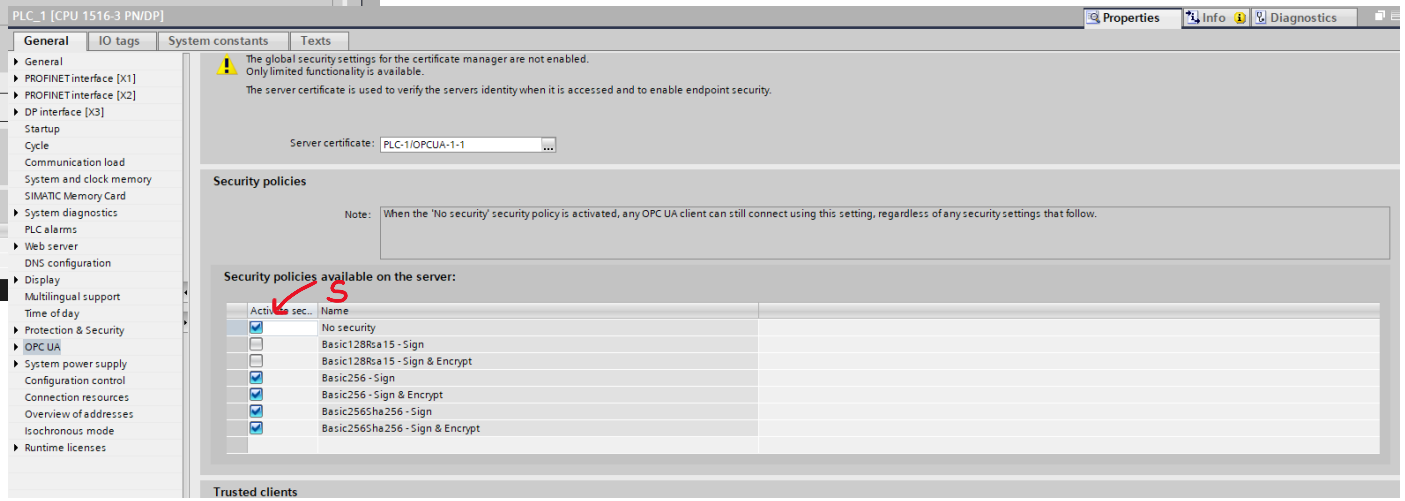


C:\Users\192072\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\Scripts

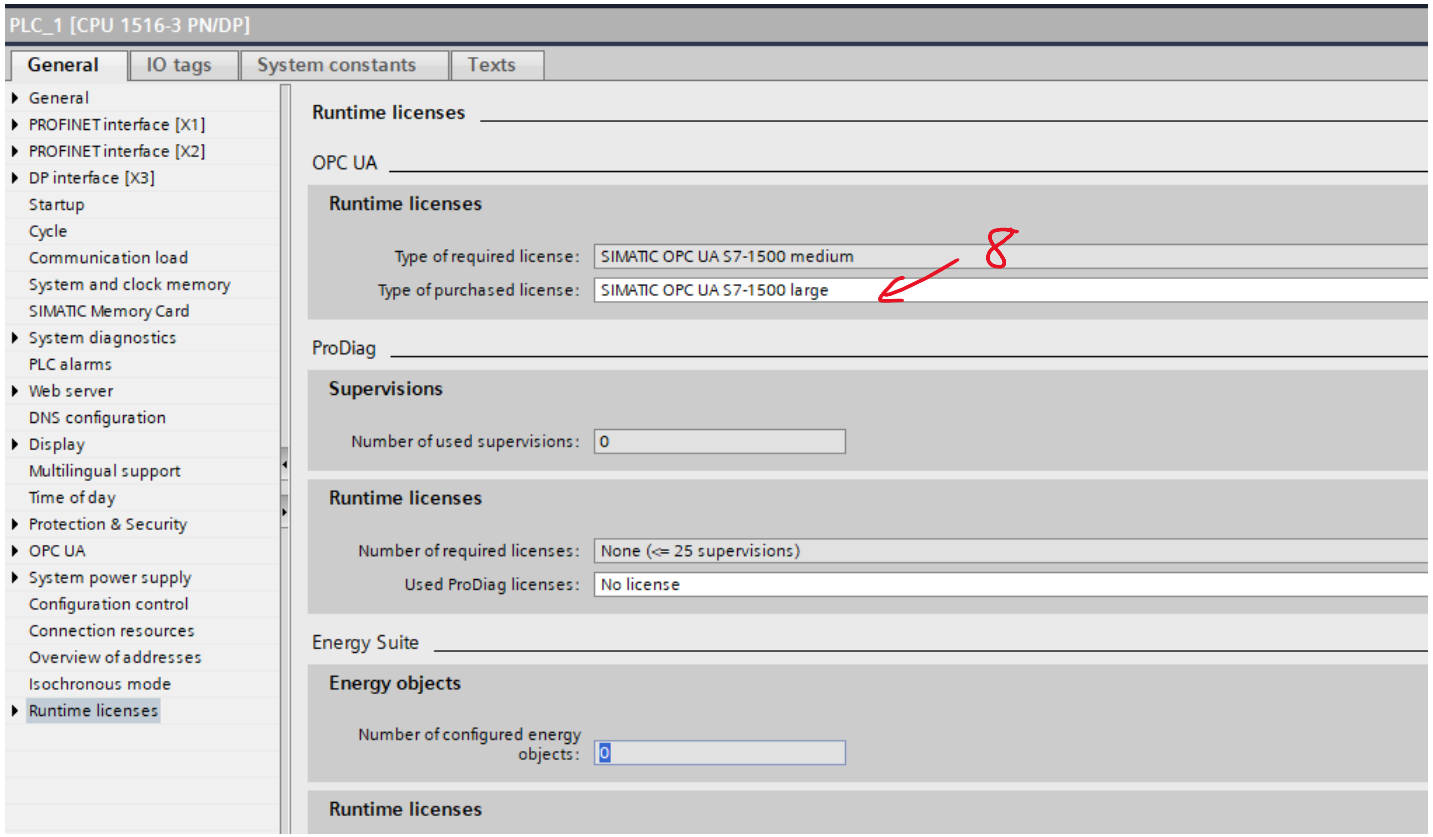
Configuración de TIA PORTAL S7-1500





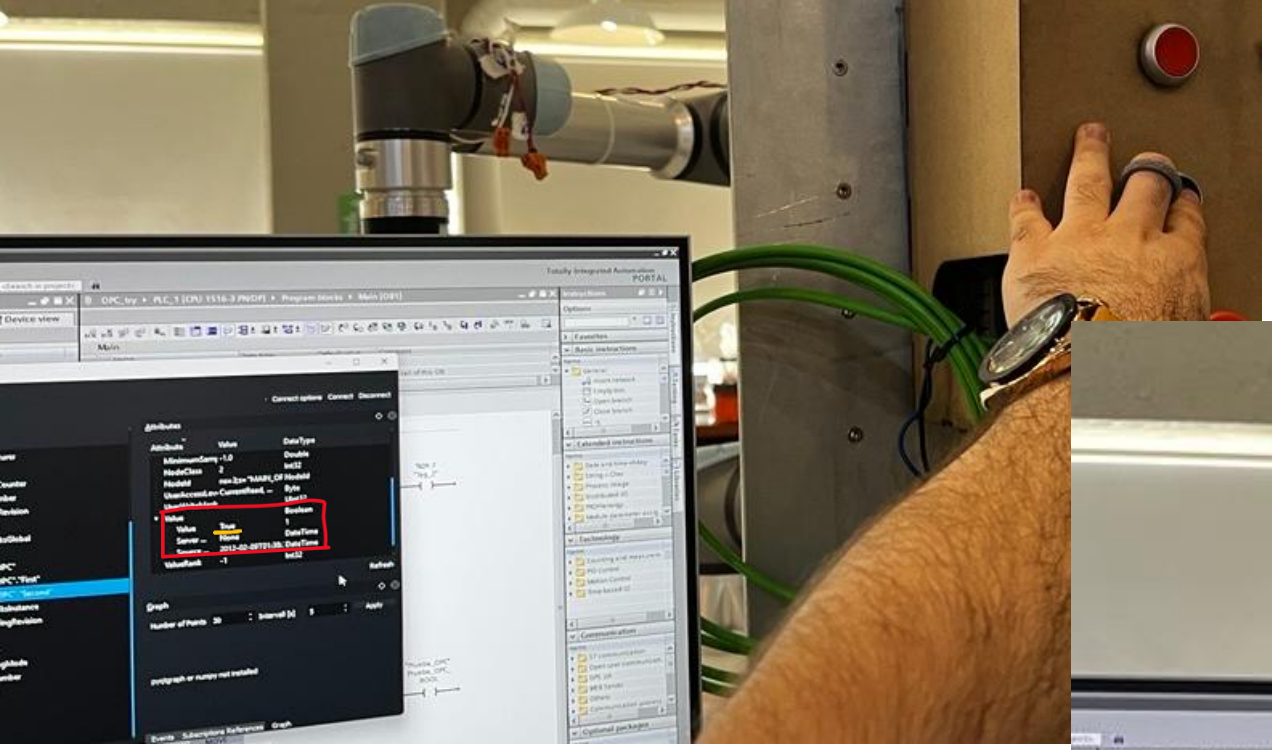


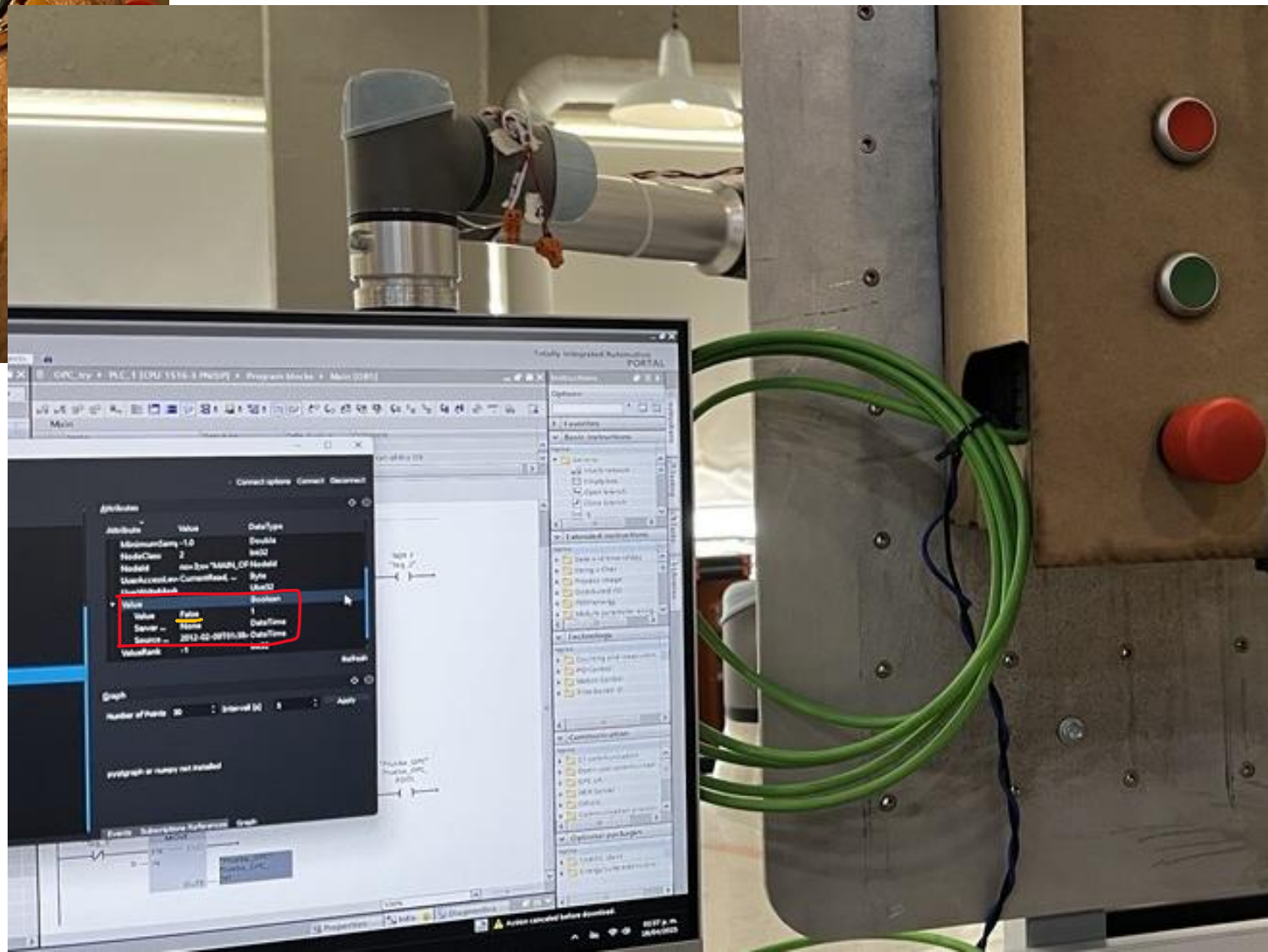
*Para el **paso 4** revisar la conexión física de Ethernet en X1 o X2.*


PLC_1 [CPU 1516-3 PN/DP]
General
IO tags
System constants
Texts
General
PROFINET interface [X1]
PROFINET interface [X2]
DP interface [X3]
Startup
Cycle
Communication load
System and clock memory
SIMATIC Memory Card
System diagnostics
PLC alarms
Web server
DNS configuration
Display
Multilingual support
Time of day
Protection & Security
OPC UA
System power supply
Configuration control
Connection resources
Overview of addresses
Isochronous mode
Runtime licenses
Runtime licenses
OPC UA
Runtime licenses
Type of required license:
SIMATIC OPC UA S7-1500 medium
Type of purchased license:
SIMATIC OPC UA S7-1500 large
8
ProDiag
Supervisions
Number of used supervisions:
0
Runtime licenses
Number of required licenses:
None (<= 25 supervisions)
Used ProDiag licenses:
No license
Energy Suite
Energy objects
Number of configured energy objects:
0
Runtime licenses

```python
from flask import Flask, jsonify, request, render_template
import json
from opcua import Client  # Asegúrate de tener instalada la biblioteca python-opcua

app = Flask(__name__)
DB_FILE = 'Web/Reportes/Teoria/T6/assets/simple_scada/db.json'
# Configura aquí el endpoint OPC (ajusta la IP y puerto según tu entorno)
OPC_ENDPOINT = "opc.tcp://localhost:4840"

def read_db():
    with open(DB_FILE, 'r', encoding='utf-8') as f:
        return json.load(f)

def write_db(data):
    with open(DB_FILE, 'w', encoding='utf-8') as f:
        json.dump(data, f, indent=4)

def opc_set_value(node_id, value):
    """Envía el valor al PLC vía OPC UA."""
    try:
        client = Client(OPC_ENDPOINT)
        client.connect()
        node = client.get_node(node_id)
        node.set_value(value)
        client.disconnect()
        print(f"OPC: Se envió {value} a {node_id}")
        return True
    except Exception as e:
        print("Error escribiendo en OPC:", e)
        return False

def opc_read_value(node_id):
    """Lee el valor del PLC vía OPC UA."""
    try:
        client = Client(OPC_ENDPOINT)
        client.connect()
        node = client.get_node(node_id)
        value = node.get_value()
        client.disconnect()
        print(f"OPC: Se leyó {value} de {node_id}")
        return value
    except Exception as e:
        print("Error leyendo desde OPC:", e)
        return None

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/api/data', methods=['GET'])
def get_data():
    # Intentamos leer el valor actual del foco desde el PLC
    opc_value = opc_read_value("ns=2;s=PLC1.Foco")
    data = read_db()
    if opc_value is not None:
        data['foco'] = opc_value
        write_db(data)
    print("GET /api/data - Estado actual del foco:", data)
    return jsonify(data)

@app.route('/api/foco', methods=['POST'])
def update_foco():
    req_data = request.get_json()
    print("POST /api/foco - Datos recibidos:", req_data)
    if 'foco' in req_data:
        data = read_db()
        data['foco'] = req_data['foco']
        # Se envía el nuevo valor al PLC mediante OPC UA
        if opc_set_value("ns=2;s=PLC1.Foco", req_data['foco']):
            print("Llamada OPC exitosa")
        else:
            print("Error en la llamada OPC")
        write_db(data)
        print("Foco actualizado a:", req_data['foco'])
        return jsonify({"status": "success", "foco": data['foco']})
    else:
        print("Error: No se recibió el parámetro 'foco'")
        return jsonify({"status": "error", "message": "Parámetro 'foco' no recibido"}), 400

if __name__ == '__main__':
    app.run(debug=True)
```

Ver códigos completos en Github





Documentación para TIA PORTAL 15.1