Schloms Process GmnH

Job Interview Task

How to Approach the Problem

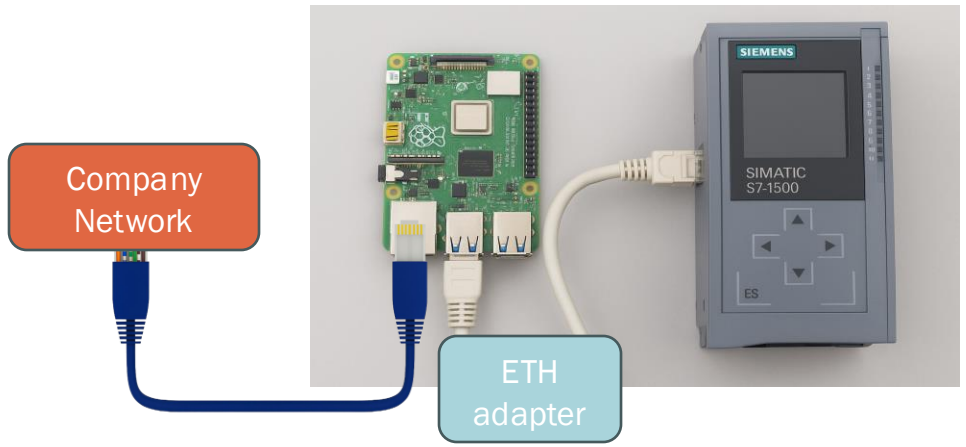
Configure Raspberry Pi Networking

- Connect eth0 to the company network.
- Connect eth1 (USB Ethernet adapter) directly to the PLC.
- Assign static IP 192.168.0.10 to eth1.

Security considerations

- Will have to ensure the PLC allows OPC-UA connections from the Pi.
- Use authentication if required (username/password or certificates).
- We can also optionally, set up firewall rules on the raspberry pi to isolate traffic.

- PLC (OPC-UA): 192.168.0.5 (isolated network)
- Raspberry Pi:
  - eth0: 10.80.10.30 (company network)
  - eth1: 192.168.0.10 (PLC network via USB-Ethernet adapter)
- InfluxDB Server: 10.80.10.1



How to write on raspberry pi

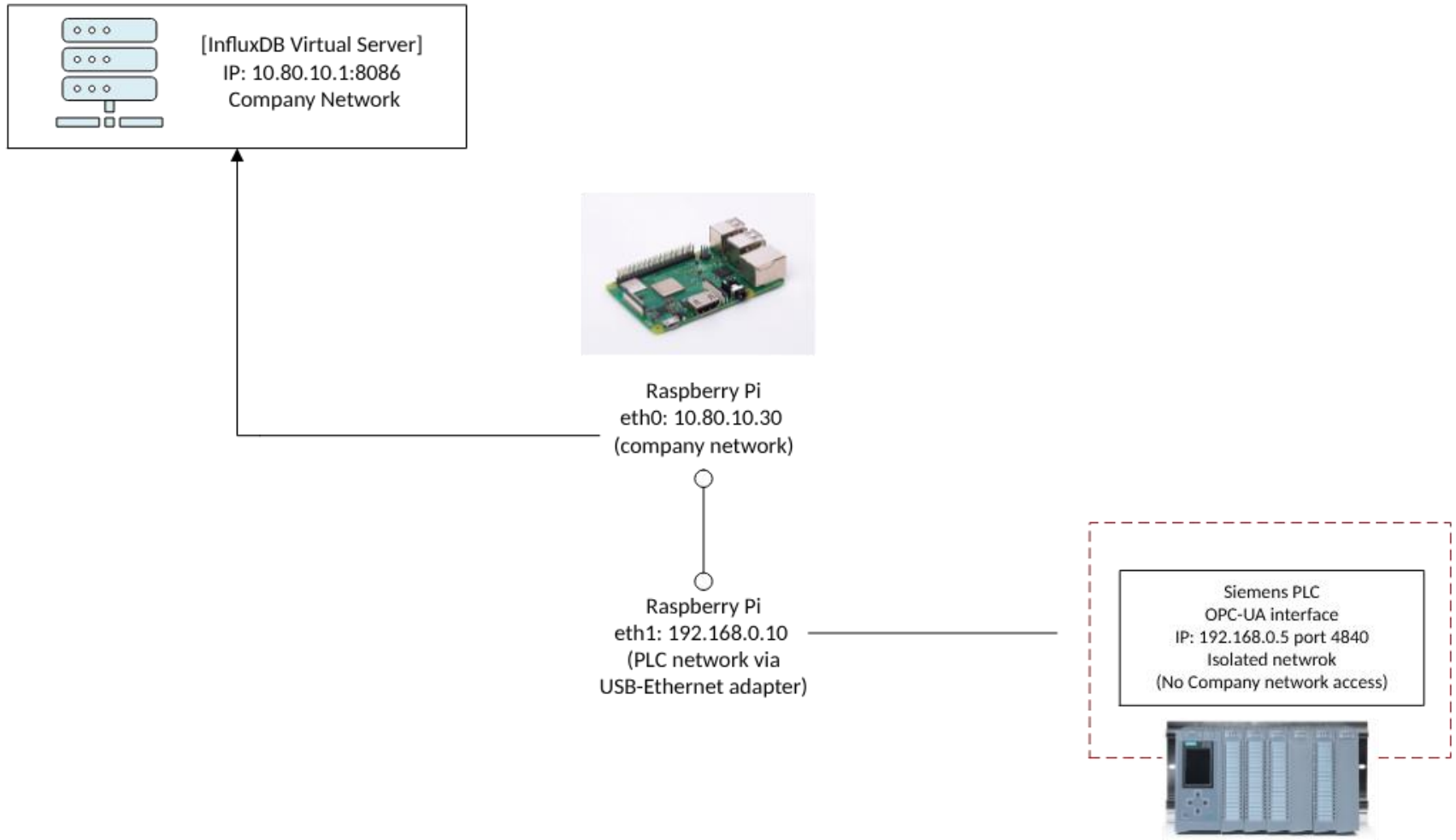
interface eth0
- static ip_address=10.80.10.30/24
- static routers=10.80.10.1
- static domain_name_servers=8.8.8.8

interface eth1
- static ip_address=192.168.0.10/24

Network Plan

Network Architecture

- Raspberry Pi acts as a network bridge.
  - Two separate network segments.
- Siemens PLC isolated from company network.
- Data flow: PLC --> Raspberry Pi --> InfluxDB virtual server
- Pi has only 1 built in ethernet port
- The ETH adapter gives the raspberry pi a second ethernet port.
  - Port 1: Company network (10.80.10.30)
  - Port 2: PLC network (192.168.0.x)



How to Approach the Problem in simulation

```
# OPC-UA server URL
opcua_url = "opc.tcp://jannen-ThinkPad:53530/OPCUA/SimulationServer"
# opcua_url = "opc.tcp://192.168.0.5:4840"

# NodeIDs to read from the OPC-UA server
node_ids = [
    "ns=3;i=5",
    "ns=3;i=6",
    "ns=3;i=7",
    "ns=3;i=8",
    "ns=3;i=9"
]
```

Goals

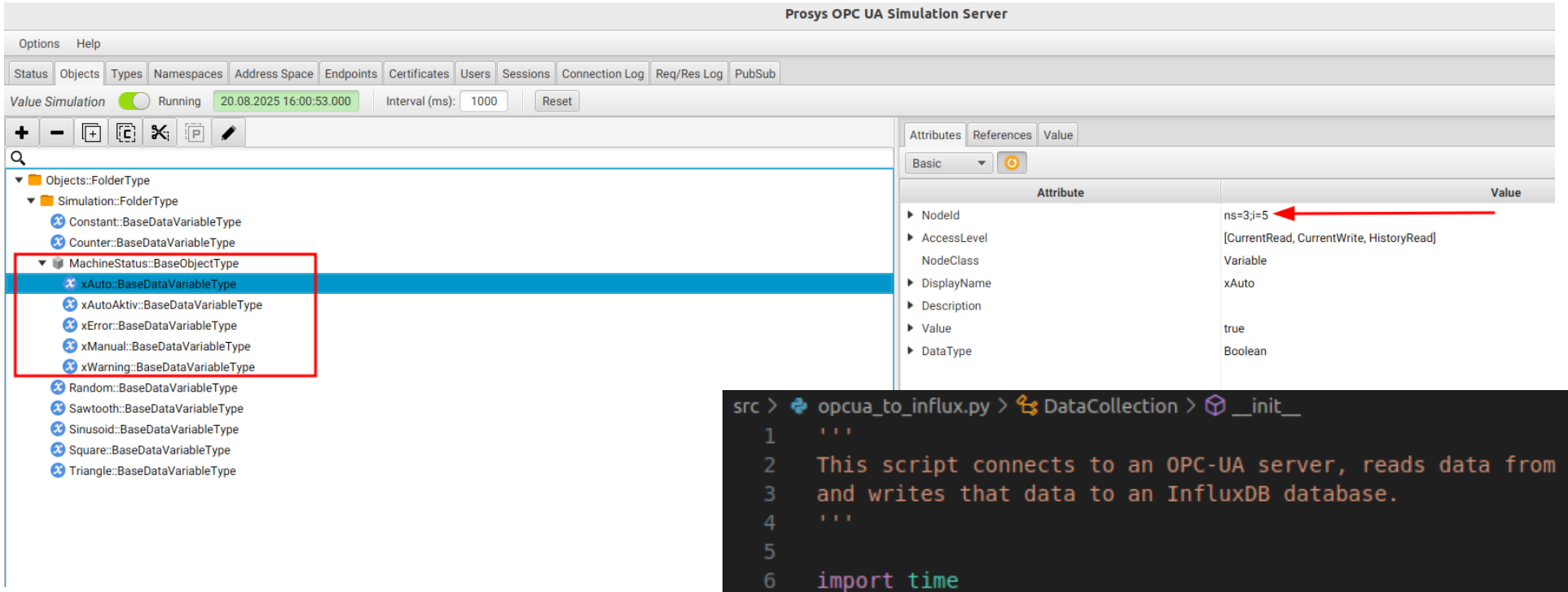
- Machine status from PLC with OPC-UA standard.
  - To achieve this, simulate machine status data using Prosys OPC UA simulation server.
- Read this data using a Python script on a Raspberry Pi.
  - Run the python script on local PC
- Store the data in InfluxDB for monitoring and analysis.
  - Use docker to run InfluxDB virtual server.

Prepare the Simulation

- Create a MachineStatus object in Prosys.
- Add variables (xAuto, xError, ..) with NodeIDs ns=3;i=5 to ns=3;i=9.
- Set data types to Boolean.

Develop the Script

- Connect to OPC UA server.
- Read values every 1 second.
- Write to InfluxDB using appropriate tags and measurements.



This script connects to an OPC-UA server to read machine status data from specific nodes and writes that data to an InfluxDB time-series database every second.

- Connects to the OPC-UA server (simulated or real PLC).
- Uses the opcua Python library to interact with the server.
- self.node_ids represent the variables under the MachineStatus object.
- Each node is read individually in the loop.

src > opcua_to_influx.py > DataCollection > __init__

```python
'''
This script connects to an OPC-UA server, reads data from a specified node,
and writes that data to an InfluxDB database.
'''

import time
from opcua import Client
from influxdb_client import InfluxDBClient, Point, WritePrecision
from influxdb_client.client.write_api import SYNCHRONOUS

class DataCollection:
    def __init__(self):
        # OPC-UA server URL
        self.opcua_url = "opc.tcp://jannen-ThinkPad:53530/OPCUA/SimulationServer"
        # opcua_url = "opc.tcp://192.168.0.5:4840"

        # NodeIDs to read from the OPC-UA server
        self.node_ids = [
            "ns=3;i=5",
            "ns=3;i=6",
            "ns=3;i=7",
            "ns=3;i=8",
            "ns=3;i=9"
        ]
```

```
# InfluxDB configuration
self.influx_host = "http://localhost:8086"
# influx_host = "10.80.10.1"  # Company InfluxDB server
self.token = "UDb3euK28xX5A027jQStLnv4yJ9Hmsbx43TpCUUftiHsfV6EY4Q2RNJu1u600Ldi-R1jWcVOsluzoU8yVVGBoQ=="
self.org = "jannen" # Replace with company or organization name
# bucket = "plc_data"
self.bucket = "machine_status"
```

- **Uses the InfluxDB v2 client with token-based authentication.**
- **Writes data to a specific bucket (machine_status) under an organization (jannen).**

```
def run(self):
    # Connect to OPC-UA server
    opc_client = Client(self.opcua_url)
    try:
        opc_client.connect()
        if opc_client.is_connected():
            print("Connected to OPC-UA Server")
        else:
            print("OPC-UA Server connection failed.")

    # Exit the run method gracefully
    except Exception as e:
        print(f"Error connecting to OPC-UA Server: {e}")


    # Connect to InfluxDB
    try:
        influx_client = InfluxDBClient(url=self.influx_host, token=self.token, org=self.org)
        # health check
        health = requests.get(f"{self.influx_host}/health")
        if health.ok and health.json().get("status") == "pass":
            write_api = influx_client.write_api(write_options=SYNCHRONOUS)
            print("Connected to InfluxDB and health check passed.")
        else:
            print("InfluxDB health check failed.")
    except Exception as e:
        print(f"Failed to connect to InfluxDB: {e}")
        return
```

Establishing Connection

- Tries to connect to the OPC-UA server and InfluxDB.
- If it fails, it catches the exception and prints an error message.
- Then it exits the method without crashing the whole program.

```python
try:
    while True:
        json_body = []
        # Read values from the specified nodes
        for node_id in self.node_ids:
            node = opc_client.get_node(node_id)
            value = node.get_value()
            print(f"Read {node_id}: {value}")

            if value is None:
                print(f"Warning: Node {node_id} returned None")
                continue

            json_body.append({
                "measurement": "machine_status",
                "tags": {
                    "node_id": node_id
                },
                "fields": {
                    "value": int(value) if isinstance(value, bool) else float(value)
                }
            })

            # Write to InfluxDB
            point = Point("machine_status").tag("node_id", node_id) \
                .field("value", int(value) if isinstance(value, bool) else float(value)) \
                .time(time.time_ns(), WritePrecision.NS)
            write_api.write(bucket=self.bucket, org=self.org, record=point)

        print("Data written to InfluxDB")

        # Wait before next read
        print("Waiting for 1 second before next read...")
        time.sleep(1)

finally:
    opc_client.disconnect()
    influx_client.close()
    print("Disconnected from OPC-UA Server and InfluxDB")
    print("Data collection completed.")
    print("Exiting...")
```

- **Reads values every second.**
- **Converts boolean values to integers for consistency.**
- **Uses both json_body and Point objects to write to InfluxDB.**
- **Skips nodes that return None to avoid writing invalid data.**

Video : data_collection_sim.webm

Thank you

Jannen

+49 15210317952

Thyman_bathlo@yahoo.com

https://github.com/Jannen06