Technical Report

Secure Monitoring System for Oil/Gas Pumping Station

Code Samples for System Integration

1. PLC Structured Text (IEC 61131-3)

```
(* PLC Program: Read 4–20 mA inputs, scale to engineering units, handle alarms *)

PROGRAM PumpStationMonitor
VAR
  // Raw analog inputs (0…20 mA mapped to 0…32767)
  AI_Pressure1  : INT;
  AI_Flow1      : INT;
  AI_Temp1      : INT;

  // Scaled values
  Pressure1     : REAL;   // bar
  Flow1         : REAL;   // m³/h
  Temp1         : REAL;   // °C

  // Alarm flags
  AlarmPressureHigh : BOOL := FALSE;
  AlarmFlowLow      : BOOL := FALSE;
  AlarmOverTemp     : BOOL := FALSE;
END_VAR

// Scaling function: rawInt → scaledReal
```

```
FUNCTION Scale4to20mA : REAL
VAR_INPUT
   rawValue    : INT;
   minPV       : REAL;  // e.g. 0.0
   maxPV       : REAL;  // e.g. sensor span
END_VAR
VAR
   scaleFactor : REAL;
END_VAR
scaleFactor := (maxPV - minPV) / 32767.0;
Scale4to20mA := minPV + (TO_REAL(rawValue) * scaleFactor);
END_FUNCTION

// Main cyclic task
Pressure1 := Scale4to20mA(AI_Pressure1, 0.0, 100.0);   // 0–100 bar
Flow1     := Scale4to20mA(AI_Flow1,     0.0, 500.0);   // 0–500 m³/h
Temp1     := Scale4to20mA(AI_Temp1,    -40.0, 150.0);  // –40…150 °C

// Alarm logic
IF Pressure1 > 80.0 THEN
   AlarmPressureHigh := TRUE;
END_IF

IF Flow1 < 20.0 THEN
   AlarmFlowLow := TRUE;
END_IF

IF Temp1 > 120.0 THEN
   AlarmOverTemp := TRUE;
```

```
END_IF

END_PROGRAM
```

2. OPC UA Server (Python with freeopcua)

```
# OPC UA Server: Publish Pressure, Flow, Temperature

from opcua import ua, Server
import time

if __name__ == "__main__":
    server = Server()
    server.set_endpoint("opc.tcp://0.0.0.0:4840/")
    server.set_server_name("QaterjiPumpStation_OPCUA")

    # Register namespace
    uri = "http://qaterji.local/OPCUA/"
    idx = server.register_namespace(uri)

    # Create objects folder
    objects = server.get_objects_node()
    station = objects.add_object(idx, "PumpStation")

    # Create variables
    pv_pressure = station.add_variable(idx, "Pressure1_bar", 0.0)
    pv_flow     = station.add_variable(idx, "Flow1_m3h",    0.0)
    pv_temp     = station.add_variable(idx, "Temp1_C",      0.0)
```

```
    # Allow clients to write
    pv_pressure.set_writable()
    pv_flow.set_writable()
    pv_temp.set_writable()

    server.start()
    print("OPC UA server started at opc.tcp://0.0.0.0:4840/")

    try:
        while True:
            # In real use, read from PLC via Modbus/OPC/etc.
            # Here we simulate with dummy values or fetch via OPC DA
            val_p = 50.0  # replace with actual read
            val_f = 300.0
            val_t = 85.0

            pv_pressure.set_value(ua.Variant(val_p, ua.VariantType.Float))
            pv_flow.set_value(ua.Variant(val_f, ua.VariantType.Float))
            pv_temp.set_value(ua.Variant(val_t, ua.VariantType.Float))

            time.sleep(1)
    finally:
        server.stop()
        print("OPC UA server stopped")
```

3. MQTT Client for Alerts (Python with paho-mqtt)

```
# MQTT Publisher: Send alarm messages to broker
```

```
import paho.mqtt.client as mqtt
import json

BROKER_HOST = "mqtt.qaterji.local"
BROKER_PORT = 1883
TOPIC      = "pumpstation/alarms"

client = mqtt.Client("AlarmPublisher")
client.connect(BROKER_HOST, BROKER_PORT, keepalive=60)

def publish_alarm(alarm_type, value, sensor_id):
   payload = {
      "timestamp": time.strftime("%Y-%m-%d %H:%M:%S"),
      "alarm":    alarm_type,
      "value":    value,
      "sensorId":  sensor_id
   }
   client.publish(TOPIC, json.dumps(payload), qos=1)
   print(f"Published alarm: {payload}")

# Example usage
if __name__ == "__main__":
   import time
   publish_alarm("PressureHigh", 85.3, "Pressure1")
   time.sleep(2)
   publish_alarm("OverTemp", 122.0, "Temp1")
```

4. SQL Schema for Event Logging

```
-- Create table for event logging
CREATE TABLE EventLog (
   EventID     INT IDENTITY(1,1) PRIMARY KEY,
   TimeStamp   DATETIME    NOT NULL DEFAULT GETDATE(),
   AlarmType   VARCHAR(50) NOT NULL,
   ValueAtEvent FLOAT      NOT NULL,
   SensorID    VARCHAR(20) NOT NULL
);

-- Index to speed up queries by time
CREATE INDEX IDX_EventLog_Time ON EventLog(TimeStamp);

-- Sample insert (triggered by application)
INSERT INTO EventLog (AlarmType, ValueAtEvent, SensorID)
VALUES ('FlowLow',  15.8, 'Flow1');
```

5. Sample HMI Tag Configuration (JSON)

```
{
  "HMI_Project": "PumpStationMonitor",
  "Tags": [
    {
      "Name": "Pressure1_bar",
      "Address": "PLC1.AnalogInput1",
      "DataType": "Real",
      "Scale": {"MinIn": 4.0, "MaxIn": 20.0, "MinOut": 0.0, "MaxOut": 100.0}
    },
    {
      "Name": "Flow1_m3h",
      "Address": "PLC1.AnalogInput2",
```

```
    "DataType": "Real",
    "Scale": {"MinIn": 4.0, "MaxIn": 20.0, "MinOut": 0.0, "MaxOut": 500.0}
   },
   {
    "Name": "Temp1_C",
    "Address": "PLC1.AnalogInput3",
    "DataType": "Real",
    "Scale": {"MinIn": 4.0, "MaxIn": 20.0, "MinOut": -40.0, "MaxOut": 150.0}
   }
  ],
  "Alarms": [
   {"Tag": "Pressure1_bar", "HighLimit": 80.0},
   {"Tag": "Flow1_m3h",    "LowLimit": 20.0},
   {"Tag": "Temp1_C",      "HighLimit": 120.0}
  ]
}
```