Si-Ware Systems

Assessment Project

Technical Assessment document “Product engineer position”

Real-Time Production Line Sensor Dashboard with Remote Access & Notifications

Objective

Develop a Python desktop application designed for a production line environment. The system must monitor at least 5 sensors simultaneously, update data in real time, trigger alarms when limits are exceeded, provide remote data access, and include optional advanced maintenance & notification features.

Note: Applicants are allowed to use AI tools to support development, research, and documentation, however, all submitted work must be fully comprehensible and defendable by the applicant.

Functional Requirements (Mandatory)

1. Real-Time Sensor Monitoring

- Monitor at least 5 independent sensors concurrently.

- Read data from a sensor simulator (serial/TCP/Modbus).

- Each sensor must provide:

  - Current value (float/int)
  - Timestamp
  - Status (OK / Faulty Sensor)

- GUI updates at least 2 times per second.

2. Multithreading / Concurrency

- Worker threads for sensor communication.

- GUI fully responsive.

- Thread-safe communication (queues/signals/events).

- No direct GUI updates from worker threads.

3. GUI Requirements

The GUI must include:

Main Dashboard

- Table with all sensors:
    - Sensor name
    - Latest value
    - Timestamp
    - Status (color-coded: green/yellow/red)
- Small real-time plot per sensor (rolling 10–20 seconds).
- Global system status indicator.

Alarm Log

- Displays triggered alarms with:
    - Time
    - Sensor name
    - Value
    - Type of alarm

4. Alarm Logic

For each sensor:

- Define **LOW** and **HIGH** limits.
- When the reading goes outside limits:
    - Generate an alarm entry
    - Highlight the sensor row in red
    - Update alarm log

5. Hardware Communication / Simulator

Applicants must implement one of the following protocols:

- **Serial** (ASCII or binary packets)
- **TCP socket**
- **Modbus/TCP register reads**

Applicant must provide a **sensor simulator** "Script that mimics real sensor" that:

- Publishes values for sensors readings
- Occasionally triggers alarm conditions
- Uses the same protocol chosen for the app

Example sensor types: temperature, vibration, speed, pressure, optical counters, etc.

Hint: Applicant can parse saved sensor data offline and use implemented protocol to share it

Bonus Features

These features are optional but will score additional points.

Bonus A — Remote Maintenance Console

Add a separate tab/window called **Maintenance Console** that allows:

1. Live Log Viewer

- Display system logs collected by a background thread.

2. Remote Commands (choose at least 2)

Examples:

- Restart sensor simulator
- Force sensor refresh
- Run a self-test
- Request detailed snapshot of values
- Clear alarms

3. Access Control

- Require password/token for this section.

4. Event Streaming (extra bonus)

- Implement a WebSocket feed for real-time logs or sensor values.

Bonus B — Alarm Notification System

Add one or more **notification mechanisms** to alert engineers when alarms occur.

Possible notification methods:

1. **Email alerts**
    - Using SMTP (Gmail/Outlook/custom)
2. **SMS alerts**
    - Using Twilio / Nexmo
3. **Slack or Microsoft Teams messages**
    - Via incoming webhook
4. **Webhook POST to a remote server**
    - Send alarm JSON payload
5. **Desktop notifications**
    - System tray or popup

Deliverables

1. GitHub repository with:

- All source code
- Sensor simulator
- Config files
- requirements.txt

2. README containing:

- Setup steps
- Running instructions
- Protocol description (serial/TCP/Modbus format)
- API documentation

3. Short demo video (2–5 minutes) Demonstrate:

- Real-time updates
- Alarm triggering

- Remote API access
- Bonus maintenance console (if implemented)

4. Basic unit tests

- Sensor parsing
- Alarm logic
- API output

5. Presentation Slides

Evaluation Criteria (100)

Core Functionality – 50 pts

- 5 sensors monitored
- Correct multithreading
- Real-time GUI responsiveness
- Alarm system works

Architecture & Code Quality – 20 pts

- Clean structure
- Documentation + comments
- Thread-safety

GUI & UX – 10 pts

- Clear, responsive dashboard
- Smooth updates

Simulator Quality – 10 pts

- Matches protocol

Bonus Features – 10 pts

- Bonus Implementation

Thank you!

We appreciate your time. Let’s stay connected and build something great together.

Mohamed Osama

mohamed.osama@si-ware.com

www.si-ware.com

Si-Ware Systems