COMCS.Lda System

1201942 - Fábio Silva
1250441- David Xavier

isep | Instituto Superior de **Engenharia** do Porto

Departamento de Engenharia Eletrotécnica

Instituto Superior de Engenharia do Porto

2021

Introduction

The objective of this project is the development of a robust environmental monitoring solution for COMCS.Lda, an organization managing sensitive products across industrial units and centralized warehousing. As product quality is susceptible to minor fluctuations in environmental conditions, the maintenance of strict temperature and humidity levels is critical throughout production, storage, and transport stages. To address this, the proposed solution implements a distributed system capable of constant monitoring, real-time visualization, and anomaly detection. The architecture integrates a Node-RED Command Centre, a dedicated Alert Server, and distributed sensing clients using ESP32 and Raspberry Pi Pico devices.

System Architecture

Command Centre and Visualization

The central monitoring node is implemented using Node-RED to serve as the primary Command Centre for the work cell. This component is responsible for capturing data via the MQTT protocol. The dashboard provides real-time situational awareness by displaying instant and historical graphs. Furthermore, the system includes a dedicated visual indicator to display alerts forwarded by the Alert Server, ensuring immediate operator awareness of critical deviations.

Alert Server Implementation

The Alert Server is developed in C and utilizes the json-c and Paho MQTT libraries. The server operates on a continuous UDP listening loop, capturing datagrams from multiple monitoring clients. Upon receipt of a packet, the system parses the JSON payload and enforces a strict validation schema. The server explicitly verifies that the temperature lies within the 0–50°C range and humidity within 20–80%. Any reading outside these bounds triggers a validation error and the packet is discarded to prevent data pollution.

Sensing Clients and Communication

The data acquisition layer consists of ESP32 and Raspberry Pi Pico devices equipped with DHT11 sensors. The firmware architecture utilizes the FreeRTOS real-time operating system

to manage concurrency efficiently, separating sensor acquisition from network communication via thread-safe queues. This ensures that network latency does not impact the sampling frequency. The clients format payloads according to the SmartData model and transmit them via MQTT (QoS 1) to the Command Centre and via UDP to the Alert Server.

Reliability and Quality of Service (QoS)

UDP Reliability and Acknowledgements

While UDP is inherently connectionless, the project requirements dictated a mechanism for implementing some kind of QoS. The implemented C server supports this by sending an explicit application-layer acknowledgement (ACK_OK) back to the client's IP and port immediately after a valid packet is processed. This handshake allows the sensing clients to verify successful data capture.

Fault Tolerance and Rolling Window

To mitigate the risk of data loss during network interruptions, a Status Rolling Window mechanism was implemented on the sensing clients. In the event of communication failure, data is diverted to local SPIFFS storage. The Alert Server complements this by implementing a clean_inactive_clients routine, which activates if a client is silent for more than 60 seconds, freeing its slot and ensuring the differential calculation logic does not rely on obsolete data when the connection is eventually restored.

Compilation, Installation, and Setup

Alert Server Deployment

The Alert Server was developed in C to run on a Linux environment. The source code, contained within the file main.c, requires the GNU Compiler Collection (GCC) for compilation. Given the dependencies on external libraries for JSON parsing and MQTT communication, the compilation process must explicitly link the json-c and paho-mqtt3cs libraries.

The server can be compiled using the following command:
gcc main.c -o main -ljson-c -lpaho-mqtt3cs

MQTT Infrastructure and Authentication

To facilitate the Publish/Subscribe messaging architecture required by the Command Centre and the Alert Server, a HiveMQ MQTT broker was deployed.

To ensure secure data segregation and access control, specific user credentials were created for each active component in the system. Distinct users were provisioned for the UDP Alert Server (to publish alerts) and for both sensing devices (to publish sensor data).

System Credentials:
    - UDP Server User: alert-server / COMCSPassword1.
    - ESP32 Client User: comcs2526cagg11/ COMCSPassword1.
    - Pico Client User: raspberry / COMCSPassword1.

Firmware Compilation (Sensing Clients)

The firmware for the sensing layer, comprising the ESP32 and Raspberry Pi Pico, was developed and compiled using the Arduino IDE. Libraries utilized include WiFi.h for network connectivity on the ESP32, PubSubClient for MQTT communication,and the DHT sensor library for environmental data acquisition.

The compilation process involved:
    1. Installing the ESP32 and RP2040 board definitions via the Board Manager.
    2. Configuring the WiFi SSID, HiveMQ broker and its credentials (created in section 4.2), and the target UDP server IP.
    3. Flashing the compiled binary to the respective devices via USB.

Command Centre Setup (Node-RED)

To create the command centre, we use Node-RED running on a virtual machine to simulate the Node-RED server. To set up and run Node-RED, follow these steps:

1. Start the virtual machine.
2. Run the command node-red on the server.
3. Open the Node-RED interface in a web browser using http://<domain-or-IP>:1880.
4. Import the Node-RED flow configuration.
5. Deploy the changes to generate the dashboard. Access the dashboard by opening: http://<domain-or-IP>:1880/ui

This will display the Command Centre graphs.

Validation and Result

Validation tests confirmed the server's logical correctness. When two clients reported temperatures of 24.0ºC and 27.0ºC respectively, the server correctly calculated a difference of 3.0ºC, exceeding the 2.0ºC threshold. Logs confirmed the generation of the SensorMismatch alert and its subsequent arrival at the MQTT broker and verified as an alert in the Node-Red UI. The relative humidity was also tested and produced the same results.

Conclusion

The proposed solution successfully establishes a reliable monitoring infrastructure for COMCS.Lda, meeting the critical objective of maintaining product quality through constant environmental oversight. By integrating a Node-RED Command Centre with a high-performance C-based Alert Server, the system effectively bridges the gap between real-time operational visualization and critical anomaly detection. The validation tests confirmed the system's ability to instantly identify temperature and humidity differentials exceeding the strict safety thresholds, ensuring that any deviation compromising the industrial process is immediately flagged to operators.

Furthermore, the robustness of the solution was verified through the successful application of fault-tolerance mechanisms. The "rolling window" strategy proved essential for data persistence, eliminating information gaps during network instability, while the Guaranteed Delivery QoS policies ensured that critical alerts were never lost. The seamless interoperability achieved between the FreeRTOS-based sensing clients and the server infrastructure, underpinned by strict SmartData standardization, demonstrates a scalable and resilient architecture ready for deployment in sensitive industrial environments.