**Smart Factory Ventilation System**

## Introduction

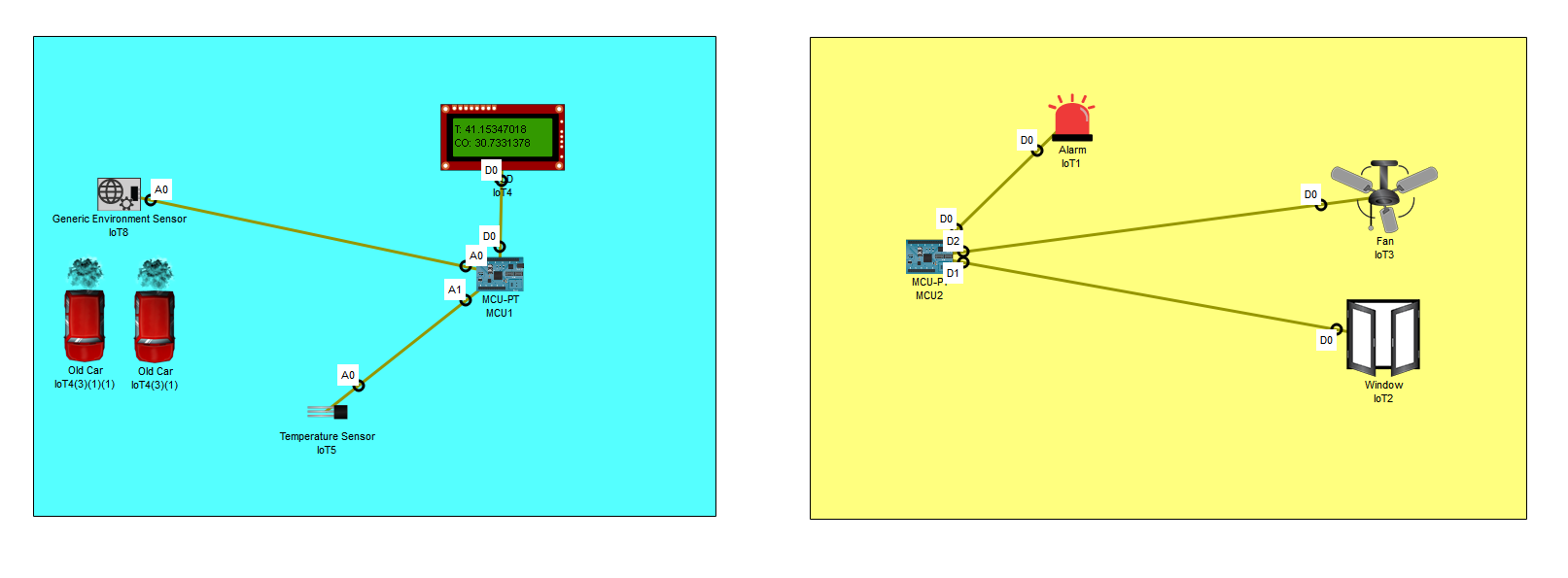
This report presents the implementation of a smart factory ventilation system using Python sockets and Cisco Packet Tracer. The system consists of two MCUs, MCU1 and MCU2, that interact with a cloud-based server to monitor and regulate air quality. The server processes sensor data and sends control commands to maintain optimal environmental conditions. The implementation adheres to the problem statement's requirements, ensuring real-time monitoring and control through a graphical dashboard.

## System Overview

The implemented system consists of the following components:

* **MCU1**: Reads temperature and CO levels from a sensor and transmits the data to the cloud.
* **MCU2**: Receives control commands from the cloud and actuates the fan and window accordingly.
* **Cloud Server**: Collects sensor data, processes it, visualizes readings, and triggers alerts based on predefined thresholds.

The cloud service is implemented using Python and handles communication through TCP sockets. The real-time data visualization is achieved using Matplotlib.



## Implementation Details

* **Programming Languages & Tools Used**:
  + Python for server implementation
  + Cisco Packet Tracer for edge device simulation
  + Matplotlib for live data visualization
  + TCP socket programming for communication
* **Sensor Data Handling**:
  + Temperature and CO levels are read periodically.
  + Data is transmitted to the cloud using a structured protocol.
  + Alerts are triggered if readings exceed set thresholds.
* **Control Mechanism**:
  + The server continuously monitors sensor data.
  + If CO levels or temperature exceed thresholds, the server sends control commands to MCU2.
  + MCU2 then activates the fan and opens the window to mitigate air quality issues.

## Types of Sensors & Thresholds

* **CO Sensor (Simulated using a generic environmental sensor)**
  + Threshold: **20 ppm**
* **Temperature Sensor**
  + Threshold: **35°C**

## Protocol Messages

The system uses the following protocol messages:

* **Sensor Data Format (MCU1 → Cloud)**: count, temperature, co\_level

Example: 1,30.5,10.2

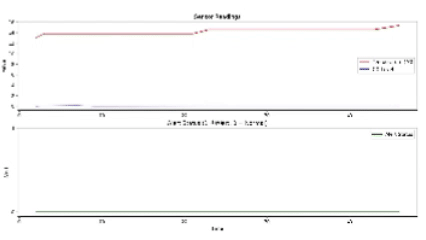
* **Control Commands (Cloud → MCU2)**:
  + 1 (Activate fan and open window)
  + 0 (Deactivate fan and close window)

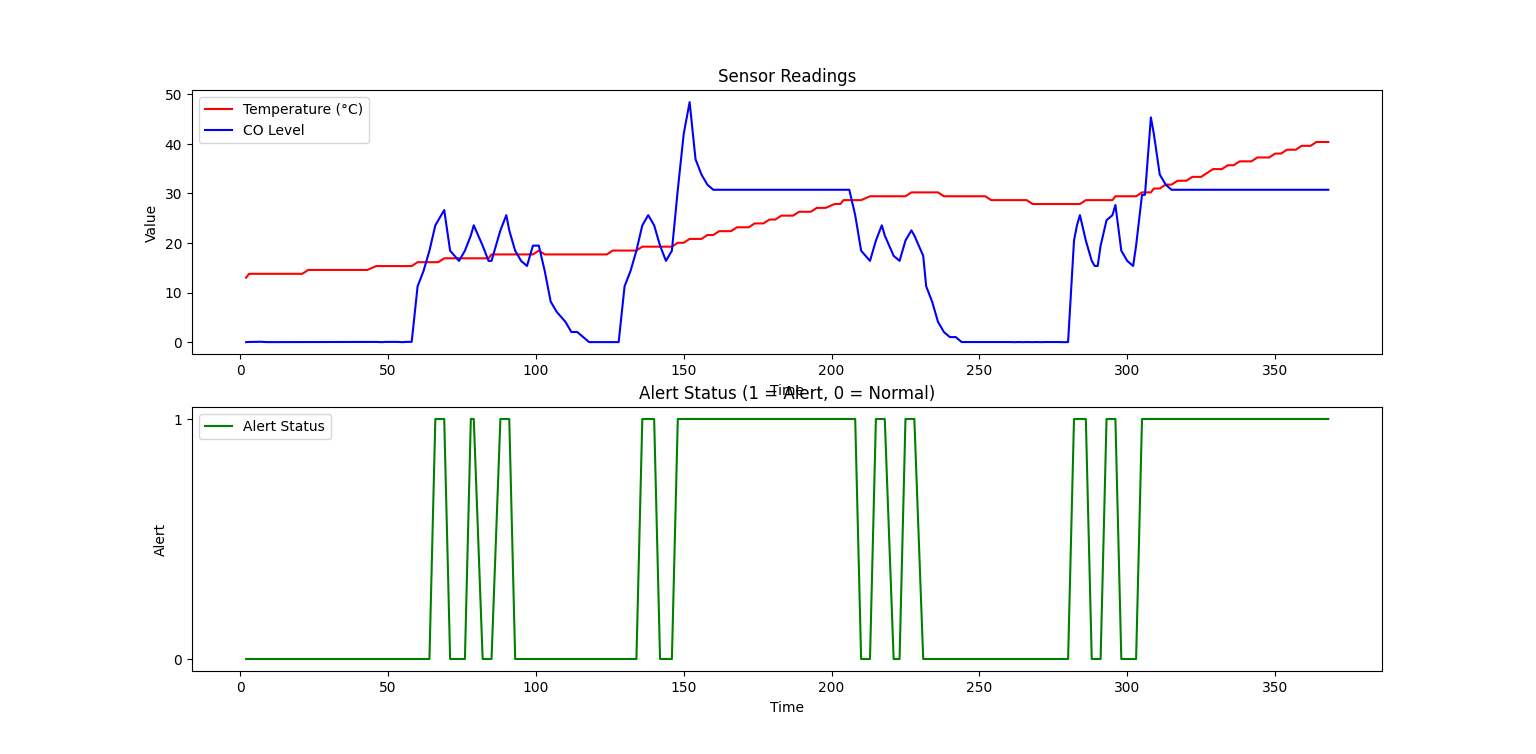
## Challenges & Solutions

1. **Handling Multiple Clients Simultaneously**:
   * Initially, only one client could connect at a time.
   * Solution: The alert server was modified to open and close dynamically to allow multiple connections.
2. **CO Sensor Replacement**:
   * The Packet Tracer environment lacked a CO sensor.
   * Solution: A generic environmental sensor was used, with water level values reinterpreted as CO readings.

## Results & Observations

* The server successfully receives and processes sensor data.
* The Matplotlib dashboard continuously updates with real-time sensor readings.
* CO and temperature levels are managed effectively by activating actuators.
* All system functionalities, including alerts, data visualization, and control actions, perform as expected.





## Conclusion

The implemented smart factory ventilation system demonstrates an effective use of Python socket programming and real-time data visualization. The system successfully maintains air quality by detecting and mitigating CO levels and high temperatures. Future improvements could include implementing MQTT for more efficient messaging and integrating actual CO sensors when hardware is available.