Universidad del Valle

LAB-2

1. Overview

In this assignment, you will design, implement, and evaluate two embedded applications using the **ESP32 platform**. Both challenges require integrating sensors, network communication protocols, backend data handling, and structured software design. You must follow a rigorous engineering workflow, including requirements analysis, architecture definition, software/hardware integration, and testing.

1. Challenge 1 – ESP32 Standalone Web Server

Design and implement a **standalone HTTP web server** running on the ESP32.
The system must:

Functional Requirements

1. **Environmental Monitoring**
   - Read and display **temperature** and **relative humidity** using the DHT22 sensor.
   - Refresh values periodically (configurable interval, recommended 1–5 seconds).
2. **User Input Interface**
   - Provide a web interface allowing the user to **set a motor speed setpoint (RPM)**.
   - Show the entered setpoint in the **Serial Monitor** in real time.
   - Motor hardware is *not* required—focus on interface and data flow.
3. **Web UI Requirements**
   - The server must be **mobile-responsive** and accessible from any device on the same network.
   - UI must clearly display:
     - Sensor readings
     - Input field for RPM setpoint
     - Timestamp of last update

2. Challenge 2 – ESP32 MQTT-Based IoT System

Design and implement a system where the ESP32 communicates using the MQTT protocol. The system must:

Functional Requirements

1. **Telemetry Publishing**
   - Publish temperature and humidity readings periodically (topic naming must follow good structure, e.g., esp32/sensors/dht22).
2. **Data Logging or Visualization**
   - Use an MQTT dashboard to show sensor values over time.
3. **Setpoint Subscription**
   - Subscribe to a topic (e.g., esp32/control/rpm) for motor RPM setpoint.
   - Display the received setpoint via Serial Monitor.
4. **Broker Connectivity**
   - Use a cloud broker (e.g., HiveMQ Cloud).
   - Document your topic structure and message format.
5. **Additional Notes**

- Motor hardware is not required.

- Ensure QoS selection is justified.
- All messages must follow a clean JSON structure.

3. Deliveries

- **Code repository in GitHub(well organized, README with build/run steps).**
- A short technical report including (could be in the repository):
    - Requirement Analysis, System Analysis and Design (Architecture & Components), Hardware Design and Integration, Firmware Design and Development, Testing and Validation.
    - **Comparative Analysis**
        - Compare the two approaches:
            - Complexity
            - Scalability
            - Real-time performance
            - Suitability for industrial IoT systems
            - Reliability and maintainability