I am doing home automation project. I using esp32, asp.net core for backend, anguler for frontend, mySql for database. I already done the esp32 setup part. The communication between esp32 and asp.net is happen though mqtt protocol. In asp.net core we can get temperature every 5 seconds by subscribing **"home/room1/temperature/read" topic. And we can control a light by publishing “ON” or “OFF” payloads to topic "home/room1/light1/command". And we can get the switch status by subscribeing "home/room1/light1/status" topic. I supposed to develop this to industry level in future adding more functionalities.**

**I want to display temperature realtime in anguler frontend using signalR. And control the light and display the light state in a dashboard in frontend. For now give guidance to setup asp.net core backend only to achive this goals.**

**I am using MQTTnet version ="4.3.7.1207"  
give guidance to make asp.net core back end using best/common practices in web development**

**mosquitto\_pub -h 192.168.8.169 -t "home/room1/light1/command" -m "OFF"**

**mosquitto\_sub -h 192.168.8.169 -t "home/room1/light1/status"**

**mosquitto\_sub -h 192.168.8.169 -t "home/room1/temperature/read"**

**1. ESP32 Setup:**

* **MQTT Client**: On the ESP32, you can use the **PubSubClient** library to handle MQTT communication.

ESP32 will

Subscribe :

home/light/control : for controlling devices (e.g., GPIO pins for lights)

publish:

home/light/status : publish device status

home/sensor/temperature : Send sensor data

MQTT topic Structure : <system> / <location> / <device> / <sensor/actuator> / <action>

e.g. :

 **Turning on a light**:

* **Command**: home/livingroom/light1/command (Payload: ON)
* **Status**: home/livingroom/light1/status (Payload: ON or OFF)

 **Getting a temperature reading from a sensor**:

* **Sensor data**: home/livingroom/temperature/read (Payload: 25.3)

 **Fan control**:

* **Command**: home/bedroom/fan1/command (Payload: ON, OFF, SPEED 3)
* **Status**: home/bedroom/fan1/status (Payload: ON, OFF, SPEED 3)

MQTT Brocker : mosquito

By default, Mosquitto will be installed to C:\Program Files\mosquitto. The configuration file is located at C:\Program Files\mosquitto\mosquitto.conf.

To start the service:

net start mosquitto

To stop the service:

net stop mosquitto

subscribe to a topic:

mosquitto\_sub -h localhost -t "test/topic"

publish a message

mosquitto\_pub -h localhost -t "test/topic" -m "Hello MQTT"

Configure the MQTT Broker:

Locate the Configuration File.( C:\Program Files\mosquitto\mosquitto.conf)

To change the default port (1883 for MQTT) and ip address, add this line:

listener 1883 192.168.8.169

to allows anyone to connect to the MQTT broker without needing a username or password.

allow\_anonymous true

**netstat -a** : to display all active network connections and the ports on which the computer is listening.

2. **ASP.NET Core Backend**

**Packages:**

**MQTTnet**

**MQTTnet.Extensions.ManagedClient**

**Microsoft.AspNetCore.SignalR**

**3. Angular Frontend:**

* **UI Design**: Create a clean and responsive interface using Angular to allow users to control devices (e.g., lights, fans) and view sensor data (e.g., temperature, humidity).
* **API Integration**: Use Angular services to make HTTP requests to your ASP.NET Core API to control devices or fetch sensor data.
* **Real-Time Updates**: For real-time updates, integrate **WebSockets** or use MQTT over WebSockets for the frontend to listen directly to MQTT topics.

Example Features:

* **Dashboard**: Show real-time sensor data like temperature, humidity, etc.
* **Control Panel**: Buttons or switches to turn devices on/off.
* **Notifications**: Alerts if certain thresholds are exceeded (e.g., temperature too high).

ng new frontend

npm install @microsoft/signalr

ng generate service signalr