Electric Vehicle Battery Management System with IoT

Here's a Python-based program for an Electric Vehicle Battery Management System (BMS) with IoT. This system monitors battery parameters like voltage, temperature, current, and state of charge (SOC) and transmits data to an IoT cloud for real-time monitoring. It uses MQTT (Message Queuing Telemetry Transport) to send data to an IoT platform such as ThingsBoard, AWS IoT, or Adafruit IO.

FEATURES:

- Reads battery parameters (simulated data for now, but can be integrated with real sensors)
- Sends data to an IoT cloud platform via MQTT
- Implements alerts for overvoltage, overtemperature, and overcurrent.

Python Code for EV BMS with IoT
We'll use:

- paho-mqtt for MQTT communication
- random for sensor simulation

Python code:

```
import paho.mqtt.client as mqtt
import random
import time

# MQTT Configuration
MQTT_BROKER = "mqtt.example.com"  # Replace with actual broker URL
MQTT_PORT = 1883
MQTT_TOPIC = "ev/battery"
MQTT_CLIENT_ID = "EV_BMS_Client"

# Battery Thresholds
MAX_VOLTAGE = 4.2  # Max voltage per cell (V)
MAX_TEMPERATURE = 60  # Max temperature (°C)
MAX_CURRENT = 100  # Max current (A)

# Simulated Battery Data
battery_data = {
    "voltage": 3.7,  # Voltage per cell (V)
    "temperature": 25,  # Temperature in Celsius
    "current": 50,  # Current in Amperes
    "soc": 80  # State of Charge (%)
}
```

```
# MQTT Callback Functions
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT Broker!")
    else:
        print(f"Failed to connect, return code {rc}")

def send_data(client):
    """Simulate battery data and send to MQTT broker."""
    battery_data["voltage"] = round(random.uniform(3.5, 4.3), 2)
    battery_data["temperature"] = round(random.uniform(20, 65), 1)
    battery_data["current"] = round(random.uniform(30, 110), 1)
    battery_data["soc"] = round(random.uniform(20, 100), 1)

    alert = "None"
    if battery_data["voltage"] > MAX_VOLTAGE:
        alert = "Overvoltage!"
    elif battery_data["temperature"] > MAX_TEMPERATURE:
        alert = "Overtemperature!"
    elif battery_data["current"] > MAX_CURRENT:
        alert = "Overcurrent!"

    message = {
        "voltage": battery_data["voltage"],
        "temperature": battery_data["temperature"],
        "current": battery_data["current"],
        "soc": battery_data["soc"],
        "alert": alert
    }

    client.publish(MQTT_TOPIC, str(message))
    print(f"Data Sent: {message}")

# MQTT Client Setup
client = mqtt.Client(MQTT_CLIENT_ID)
client.on_connect = on_connect
client.connect(MQTT_BROKER, MQTT_PORT, 60)

# Main Loop
client.loop_start()
while True:
    send_data(client)
    time.sleep(5)  # Send data every 5 seconds
```

How It Works:

- The script establishes a connection with the MQTT broker.

- It generates simulated battery parameters every 5 seconds.
- If values exceed thresholds, an alert is triggered.
- The data is published to the MQTT topic (ev/battery).
- An IoT platform can subscribe to the topic for real-time monitoring.