Smart Railway Coach Monitoring System (ESP8266 Based)

🔧 System Overview

✅ Problems Being Solved:

- **Lack of real-time fire detection**
- **No way to monitor overcrowding**
- **No derailment detection**
- **Delayed emergency response**
- **Manual passenger counting**

🛠️ Components Needed

Purpose	Sensor/Module	Description
Fire Detection	Flame sensor / MQ-2	Detect fire or smoke
Overcrowding	IR Sensor (x2)	Count passengers via entry/exit
Derailment Detection	MPU6050 (Accelerometer)	Detect abnormal motion
Control + WiFi	ESP8266	Core microcontroller with WiFi
Local Alert	Buzzer + LED	Local alerts in emergencies
Cloud Platform	ThingSpeak	Real-time data monitoring
Display (Optional)	OLED or LCD	Coach-level data display

□ Circuit Connections

1. 🔥 **Fire Detection (Flame Sensor)**:
   - **VCC** → 3.3V
   - **GND** → GND
   - **OUT** → D32
2. 🚪 **Passenger Counting (IR Sensors)**:
   - **IR Sensor 1 OUT** → D33
   - **IR Sensor 2 OUT** → D25
3. 🚂 **Derailment Detection (MPU6050)**:
   - **VCC** → 3.3V

   - **GND** → GND
   - **SDA** → D2
   - **SCL** → D1
4. 🔊 **Buzzer**:
   - **VCC** → 3.3V
   - **GND** → GND
   - **Signal** → D5
5. 💡 **LED**:
   - **Positive** → D6 (via 220Ω resistor)
   - **Negative** → GND

☁ ThingSpeak Cloud Setup

1. Go to **ThingSpeak**.
2. Create a new channel.
3. Add 3 fields:
   - **Field 1**: Fire (to monitor flame detection)
   - **Field 2**: Passenger Count (to monitor overcrowding)
   - **Field 3**: Derailment (to monitor abnormal motion)
4. Copy your **API Key** (to use in the code).

💻 Required Libraries for Arduino IDE

Install the following via **Library Manager**:

- **ESP8266WiFi.h** (for WiFi functionality)
- **ESP8266HTTPClient.h** (for HTTP communication)
- **Wire.h** (for I2C communication with MPU6050)
- **MPU6050 by Electronic Cats** (for accelerometer integration)

💾 Code File Names

Feature	Code File Name
Main Project Code	SmartCoach_ESP8266.ino

□ Main Code (ESP8266 Version)

cpp

CopyEdit

```cpp
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <Wire.h>
#include <MPU6050.h>

// WiFi Credentials
const char* ssid = "Your_SSID";
const char* password = "Your_PASSWORD";

// ThingSpeak API Key
String apiKey = "Your_API_KEY";
const char* server = "http://api.thingspeak.com/update";

// Pin Definitions
#define FLAME_SENSOR 32
#define IR_ENTRY 33
#define IR_EXIT 25
#define BUZZER 5
#define LED 6

MPU6050 mpu;
int passengerCount = 0;

void setup() {
  Serial.begin(115200);
  pinMode(FLAME_SENSOR, INPUT);
  pinMode(IR_ENTRY, INPUT);
  pinMode(IR_EXIT, INPUT);
  pinMode(BUZZER, OUTPUT);
  pinMode(LED, OUTPUT);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) delay(500);

  Wire.begin();
  mpu.initialize();
}

void loop() {
  // Read Sensors
  int flame = digitalRead(FLAME_SENSOR);
  int irIn = digitalRead(IR_ENTRY);
  int irOut = digitalRead(IR_EXIT);

  // Fire Detection Logic
  if (flame == LOW) {
    digitalWrite(BUZZER, HIGH);  // Alert for fire
    digitalWrite(LED, HIGH);     // Turn on LED for fire
  } else {
    digitalWrite(BUZZER, LOW);   // Turn off buzzer
    digitalWrite(LED, LOW);      // Turn off LED
  }

  // Passenger Counting Logic
  if (irIn == LOW) passengerCount++;  // Passenger enters
  if (irOut == LOW && passengerCount > 0) passengerCount--;  // Passenger exits
```

```
 // Derailment Detection
 mpu.getAcceleration();
 int x = abs(mpu.getAccelerationX());
 int y = abs(mpu.getAccelerationY());
 int z = abs(mpu.getAccelerationZ());
 bool derail = (x > 15000 || y > 15000 || z > 25000);

 // Send Data to ThingSpeak
 if (WiFi.status() == WL_CONNECTED) {
  HTTPClient http;
  String url = String(server) + "?api_key=" + apiKey +
               "&field1=" + String(flame) +
               "&field2=" + String(passengerCount) +
               "&field3=" + String(derail);
  http.begin(url);
  http.GET();
  http.end();
 }

 delay(2000); // Wait for 2 seconds before sending data again
}
```

⚙️ Code to Run First

- Upload the **SmartCoach_ESP8266.ino** file to the ESP8266 board. This will demonstrate all the sensor operations (fire detection, passenger counting, derailment detection) and cloud integration with **ThingSpeak**.

📽️ Demonstration Plan

1. **Introduce Components**: Briefly explain each sensor and its role in the system.
2. **Simulate Sensors**:
    - Use a lighter at a safe distance to simulate **fire detection**.
    - Block **IR sensors** to simulate **passenger entry/exit**.

   - Shake the **ESP8266** to simulate **derailment**.
3. **Show Alerts**: Demonstrate how the **LED** and **Buzzer** work as local alerts for **fire detection**.
4. **Show Real-Time Data**: Display **real-time data** from the **ThingSpeak dashboard** to monitor fire, passenger count, and derailment status.

▯ Optional Mini-Model Build

1. Use a **cardboard box** or **plastic container** to simulate the **train coach**.
2. Fix the **IR sensors** at the **“door”** of the coach.
3. Place **buzzer** and **LED** inside the box for visual and sound alerts.
4. Power the **ESP8266** using **USB** or a **laptop**.
5. Display the **real-time data** on a **laptop screen via Wi-Fi** using the **ThingSpeak dashboard**.

🚀 Ideas to Improve and Add Real-Life Problem Solving:

Real-Time Train Monitoring:

- **GPS Integration**: Integrate a **GPS module** to track the **train's real-time location** and display it on a dashboard.
- **Speed and Velocity Monitoring**: Monitor **train speed** and sudden **changes in velocity** to predict potential hazards.

Passenger Safety:

- **Air Quality Monitoring**: Add sensors to monitor **CO2 levels** and ensure a safe environment for passengers.
- **AI-Driven Behavior Analysis**: Implement **AI** to detect **abnormal behavior** or security threats among passengers.

Automatic Emergency Response:

- **Alerting Emergency Services**: Set up automatic alerts to **emergency services** in case of **fire, derailment,** or **overcrowding**.
- **Activate Safety Systems**: Trigger safety systems like **fire suppression** or **automated braking** in case of emergencies.

Predictive Maintenance:

- **Train Component Monitoring**: Monitor **wheels, engines,** and other critical components for **early fault detection**.
- **Data Analytics**: Use **data analytics** to predict component failures and schedule **preventive maintenance**.

Passenger Comfort Monitoring:

- **Temperature and Humidity Sensors**: Use these sensors to monitor the **coach environment** and ensure **passenger comfort**.

🚀 Future Work

- **Integrate AI** to analyze **train operations**, improve **safety**, and provide **real-time alerts** to both the **train crew** and **passengers**.