```
#define BLYNK_TEMPLATE_ID "TMPL3gIOe8BtA"
#define BLYNK_TEMPLATE_NAME "Elevator Smart Surviellance"
#define BLYNK_AUTH_TOKEN "lg3ir2LBohoRiJnhLopOkQqH7I18qOYf"

// Include necessary libraries
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>

// Create ADXL345 instance
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);

// Variables to store previous accelerometer values
float prevX = 0, prevY = 0, prevZ = 0;

// Acceptable vibration ranges for normal operation
float normalX_min = -1, normalX_max = 0.6;
float normalY_min = -1.8, normalY_max = 0;
float normalZ_min = 9.5, normalZ_max = 11.6;

// WiFi credentials
char ssid[] = "Redmi 12C";          // Your WiFi SSID
char pass[] = "redmi12c";           // Your WiFi Password

// Timer to control event frequency
unsigned long lastNotificationTime = 0;
const unsigned long notificationCooldown = 3000;  // 3 seconds cooldown

void setup() {
```

```
  // Start the Serial Monitor
  Serial.begin(115200);

  // Initialize the ADXL345 sensor
  if (!accel.begin()) {
    Serial.println("Failed to detect ADXL345. Please check wiring.");
    while (1);
  }

  // Set range to +-2G for more sensitivity
  accel.setRange(ADXL345_RANGE_2_G);

  // Print initialization message
  Serial.println("ADXL345 Initialized");
  Serial.println("Monitoring vibrations...");

  // Connect to WiFi
  Serial.println("Connecting to WiFi...");
  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting...");
  }

  Serial.println("Connected to WiFi");

  // Initialize Blynk
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
}
```

```
void loop() {
  // Run Blynk
  Blynk.run();

  // Check if WiFi is still connected
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("WiFi Disconnected! Attempting to reconnect...");
    WiFi.begin(ssid, pass);
    delay(1000);
    return;
  }

  // Create an event to hold the sensor data
  sensors_event_t event;
  accel.getEvent(&event);

  // Calculate the change in accelerometer data
  float deltaX = abs(event.acceleration.x - prevX);
  float deltaY = abs(event.acceleration.y - prevY);
  float deltaZ = abs(event.acceleration.z - prevZ);

  // Send X, Y, Z values to Blynk virtual pins
  Blynk.virtualWrite(V1, event.acceleration.x);  // X-axis
  Blynk.virtualWrite(V2, event.acceleration.y);  // Y-axis
  Blynk.virtualWrite(V3, event.acceleration.z);  // Z-axis

  // Debugging: Print sensor data to Serial Monitor
  Serial.print("X: "); Serial.println(event.acceleration.x);
  Serial.print("Y: "); Serial.println(event.acceleration.y);
  Serial.print("Z: "); Serial.println(event.acceleration.z);
  Serial.print("deltaX: "); Serial.println(deltaX);
```

```
  Serial.print("deltaY: "); Serial.println(deltaY);
  Serial.print("deltaZ: "); Serial.println(deltaZ);

  // Check if current values are outside the normal operation range
  bool outOfRange = (event.acceleration.x < normalX_min || event.acceleration.x > normalX_max) ||
                    (event.acceleration.y < normalY_min || event.acceleration.y > normalY_max) ||
                    (event.acceleration.z < normalZ_min || event.acceleration.z > normalZ_max);

  if (outOfRange) {
    unsigned long currentTime = millis();

    // Only trigger event if the cooldown period has passed
    if (currentTime - lastNotificationTime > notificationCooldown) {
      Serial.println("Abnormal Vibration Detected! Logging Event...");

      // Trigger an event in Blynk (to log and notify via automation)
      Blynk.logEvent("motor_fault_alert", "Abnormal Vibration Detected!\nSomething is going wrong in the elevator's motor section.");

      Serial.println("Event Logged to Blynk!");

      lastNotificationTime = currentTime;  // Update the last notification time
    } else {
      Serial.println("Notification Cooldown: Skipping event");
    }
  }

  // Update previous accelerometer values for comparison in the next loop
  prevX = event.acceleration.x;
  prevY = event.acceleration.y;
  prevZ = event.acceleration.z;
```

```
  // Delay before reading the next values
  delay(2500);  // Adjust delay as needed
}
```