To modify your MPU6050 motion detection code to connect via Bluetooth to your phone (e.g. using a serial Bluetooth app), you can use the BluetoothSerial library built into the ESP32 platform. This allows the ESP32-C3 to emulate a classic Bluetooth Serial connection (SPP), which you can read with apps like:

- Serial Bluetooth Terminal (Android)
- LightBlue Explorer (iOS)
- Bluetooth Serial Controller (iOS)

Here's your modified code with Bluetooth Serial support added:

✅ Modified Code: MPU6050 with Bluetooth Serial on ESP32-C3

cpp — Copy — Edit

```cpp
#include <Wire.h>          // I2C library
#include "I2Cdev.h"        // Needed for MPU6050
#include "MPU6050.h"       // MPU6050 sensor
#include "BluetoothSerial.h" // Bluetooth library for ESP32

MPU6050 mpu;
BluetoothSerial SerialBT;  // Create Bluetooth serial object

// LED connected to D1 (GPIO3 / A1 on XIAO ESP32-C3)
const int ledPin = 3;

// Acceleration threshold to trigger the LED
const int accelThreshold = 15000;

// Variables for sensor data
int16_t ax, ay, az;
int16_t gx, gy, gz;

void setup() {
  // Start both regular and Bluetooth serial
  Serial.begin(115200);
  SerialBT.begin("XIAO-MPU6050"); // Name the Bluetooth device

  while (!Serial); // Wait for serial monitor (optional)

  Serial.println("Initializing I2C devices...");
  SerialBT.println("Initializing I2C devices...");

  // Initialize I2C (SDA = GPIO6, SCL = GPIO7)
  Wire.begin(6, 7);

  // Initialize MPU6050
  mpu.initialize();
  Serial.println("Testing MPU6050 connection...");
  SerialBT.println("Testing MPU6050 connection...");

  if (mpu.testConnection()) {
    Serial.println("MPU6050 connection successful");
    SerialBT.println("MPU6050 connection successful");
  } else {
    Serial.println("MPU6050 connection failed! Check wiring.");
    SerialBT.println("MPU6050 connection failed! Check wiring.");
    while (1); // Halt on error
  }

  // Setup LED
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);

  Serial.println("Setup complete. Move the sensor to trigger the LED.");
  SerialBT.println("Setup complete. Move the sensor to trigger the LED.");
}

void loop() {
  // Get accelerometer and gyroscope values
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  // Send debug data to both Serial Monitor and Bluetooth
  String output = "ax: " + String(ax) + "\tay: " + String(ay) + "\taz: " + String(az);
  Serial.println(output);
  SerialBT.println(output);

  // Trigger LED if threshold exceeded
  if (abs(ax) > accelThreshold) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }

  delay(50);
}
```

✅ What You'll See

When you run this sketch and pair your ESP32-C3 with your phone:

The error message you're seeing indicates **undefined references to various Bluetooth functions**, like `esp_spp_write`, `esp_bt_gap_register_callback`, etc. These are part of the **Classic Bluetooth (SPP)** stack in the ESP32.

✅ The root issue:

You're trying to compile code that uses **Classic Bluetooth** (`BluetoothSerial`) on an **ESP32-C3** — but **ESP32-C3 only supports Bluetooth Low Energy (BLE), not Classic Bluetooth.**

❌ Incompatible setup:

- You're using:
  - **Board:** ESP32-C3
  - **Library:** `BluetoothSerial` (Classic Bluetooth, not supported on ESP32-C3)
- Result:
  - The linker can't find implementations of functions like `esp_spp_write`, `esp_bt_gap_*`, etc., because those are for Classic Bluetooth, which isn't included in the ESP32-C3 toolchain.

✅ How to fix it:

Option 1: Use a board that supports Classic Bluetooth

If your project *requires* `BluetoothSerial`, **switch to an ESP32 board that supports Classic Bluetooth**, like:

- **ESP32-WROOM-32**
- **ESP32-WROVER**

To do this:

1. In Arduino IDE, go to **Tools > Board**.
2. Select something like `ESP32 Dev Module` or `ESP32 WROOM`.
3. Recompile.

Option 2: Switch to BLE (Bluetooth Low Energy)

If you're open to using **BLE instead of Classic Bluetooth**, use the **ESP32-C3's BLE libraries**, such as:

- `BLEDevice`
- `NimBLE-Arduino` (lighter and works well on ESP32-C3)

But you'd need to **rewrite your code** to use BLE services and characteristics instead of `BluetoothSerial`.

✅ What You Can Do Instead:

If You Need Bluetooth:

Use BLE (Bluetooth Low Energy) instead of `BluetoothSerial`. You can try one of these libraries:

- NimBLE-Arduino – lightweight and works with ESP32-C3
- `ESP32 BLE Arduino` (built-in in the ESP32 board support)

If You Want to Replace `BluetoothSerial`, Try This:

Here's a minimal BLE server example that works with ESP32-C3:

cpp Copy Edit

```cpp
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>

BLECharacteristic *pCharacteristic;
bool deviceConnected = false;

class MyServerCallbacks : public BLEServerCallbacks {
  void onConnect(BLEServer* pServer) {
    deviceConnected = true;
  }
  void onDisconnect(BLEServer* pServer) {
    deviceConnected = false;
  }
};

void setup() {
  Serial.begin(115200);

  BLEDevice::init("ESP32C3-BLE");
  BLEServer *pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());

  BLEService *pService = pServer->createService("0000180A-0000-1000-8000-00805F9B34FB");

  pCharacteristic = pService->createCharacteristic(
                    "00002A57-0000-1000-8000-00805F9B34FB",
                    BLECharacteristic::PROPERTY_READ |
                    BLECharacteristic::PROPERTY_WRITE
                  );

  pCharacteristic->setValue("Hello from ESP32-C3!");
  pService->start();

  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->start();
  Serial.println("BLE Server is up.");
}

void loop() {
  if (deviceConnected) {
    // Do something when connected
  }
}
```