

IoT Lab1: 基础设备控制

3220102866 陈奕萱

1 实验目的和要求

- 了解常用的硬件平台(esp32-wroom-32)
- 理解RIOT操作系统的多线程控制原理和方法
- 掌握基于GPIO引脚的LED灯控制技术
- 学习通过I2C总线读取IMU传感器数据的方法
- 识别设备运动状态并根据运动状态展示不同LED颜色的功能

2 实验内容

基于ESP32硬件以及RIOT系统，根据完成以下实验：

1. IMU传感器读取及LED控制实验
 - 通过GPIO控制LED灯状态，展示不同颜色
 - 读取IMU惯性传感器数据，并打印至串口
 - 识别设备运动状态，根据运动状态展示不同灯颜色

3 实验背景

3.1 RIOT操作系统

RIOT(<https://github.com/RIOT-OS/RIOT>) 是一个开源的微控制器操作系统，旨在满足物联网(IoT)设备和其他嵌入式设备的需求。它支持一系列通常在物联网(IoT)中发现的设备:8位，16位和32位微控制器。RIOT基于以下设计原则:节能、实时功能、内存占用小、模块化和统一的API访问，独立于底层硬件(该API提供部分POSIX遵从性)。

3.2 MPU6050惯性传感器介绍

MPU6050是一款六轴运动处理传感器，它集成了3轴MEMS陀螺仪和3轴MEMS加速度计，以及一个可扩展的数字运动处理器DMP。这种传感器能够检测物体在空间中的姿态变化，包括俯仰角(Pitch)、翻滚角(Roll)和偏航角(Yaw)。MPU6050通过I2C接口读取数据，经过数据处理后，可以提供姿态角的精确测量。它广泛应用于平衡车、无人机、智能手环、手机等设备中，用于保持设备平衡和稳定，以及实现姿态控制。MPU6050模块的引脚功能包括两组I2C信号线SDA/SCL和XDA/XCL，其中SDA/SCL用于与外部主机通讯，而XDA/XCL则用于MPU6050与其他I2C传感器通讯。模块的I2C设备地址可以通过AD0引脚的电平控制，提供灵活的地址配置。

4 主要仪器设备

- PC
- ESP32-WROOM-32、MPU6050惯性传感器、LED RGB灯。

5 实验题目简答

5.1 列举生活中常见的一些传感器，其应用场景有哪些？

1. 气体传感器（例如CO₂传感器、甲烷传感器）

- 空气质量监测：用于检测室内外空气中的CO₂浓度，应用于智能家居空气净化器或气象站。
- 工业安全：甲烷传感器用于检测工业环境或矿井中的可燃气体，预防气体泄漏导致的爆炸风险。
- 汽车排放控制：用于监测尾气中的有害气体，如CO₂、NO_x，确保汽车排放符合环保标准。

2. 心率传感器（光电容积脉搏波描记法，PPG）

- 智能手表和健身手环：通过手腕测量脉搏，实时监测心率并提供运动和健康数据分析。
- 医疗设备：如心率监测仪，用于医院或家庭中的心脏健康监测，帮助医生随时了解患者的心脏情况。
- 运动训练设备：跑步机或自行车等设备中集成心率监控功能，帮助用户调整运动强度。

3. LIDAR（光探测和测距传感器）

- 自动驾驶汽车：LIDAR用于创建高精度的3D环境地图，帮助自动驾驶车辆识别周围的物体并导航。
- 无人机：用于测绘、地形测量和避障，在复杂环境中提供精准的高度和距离感知。
- 机器人：用于SLAM（即时定位与地图构建），帮助机器人在不熟悉的环境中自主移动。

4. 霍尔传感器

- 电动车和电机：用于检测转速和位置，确保电机的精确控制。
- 智能手机：在一些手机中用于检测手机套的开关，从而自动唤醒或关闭屏幕。
- 汽车安全系统：用于车速传感器，检测车轮的转动速度，以辅助ABS（防抱死刹车系统）和车身稳定控制系统。

5. 电容式触摸传感器

- 智能手机和平板电脑：用于触控屏幕感应，帮助用户实现精准的手指输入。
- 家电控制面板：电容式触控传感器被广泛应用于智能家居设备，如微波炉、洗衣机等的控制界面，替代传统的按钮。

5.2 MPU6050惯性传感器SDA和SCL引脚的作用是什么？请详细介绍I2C是如何通信以及如何通过GPIO控制LED灯？

1. MPU6050惯性传感器的SDA和SCL引脚作用：组成了I2C（Inter-Integrated Circuit）通信协议的物理层

- SDA（数据线）：这是I2C通信协议中的数据传输线，用于在主设备和从设备之间传输数据。在MPU6050中，SDA引脚用于传输加速度计和陀螺仪的数据。
- SCL（时钟线）：这是I2C通信协议中的时钟信号线，用于同步数据传输。在MPU6050中，SCL引脚用来确保主设备和从设备在同一时间执行数据的发送或接收。

2. I2C通信原理：

- 起始条件 (Start Condition)：
 - 主设备通过将SDA从高电平拉到低电平（当SCL保持高电平）来发出起始信号，通知所有从设备开始通信。
- 从设备地址传输：
 - 主设备发送7位从设备地址，接着发送第8位表示读（1）或写（0）操作。
 - 所有从设备会监听这条消息，匹配地址的从设备会做出响应。
- 应答信号 (ACK/NACK)：
 - 匹配地址的从设备在接收到数据后会发送一个ACK（应答信号），表示已成功接收到数据。
 - 如果没有匹配或设备无法处理，则发送NACK（非应答信号）。
- 数据传输：
 - 主设备和从设备按字节传输数据，发送一个字节后，接收设备会返回ACK/NACK信号。
 - 读操作时，主设备读取从设备的数据；写操作时，主设备向从设备发送数据。
- 停止条件 (Stop Condition)：
 - 当通信完成后，主设备通过将SDA从低电平拉回高电平（当SCL保持高电平）来发送停止信号，结束通信。

3. 如何通过GPIO控制LED灯

- 配置GPIO引脚为输出模式：
 - 首先，需要将用于控制LED灯的GPIO引脚配置为输出模式。微控制器中的GPIO通常可以配置为输入或输出。
- 控制LED开关：
 - 点亮LED：通过将GPIO引脚设为高电平，提供正电压，LED将点亮。
 - 关闭LED：将GPIO引脚设为低电平，LED熄灭。
- 在实际的应用中，例如MPU6050与微控制器进行通信时，可以通过I2C总线读取传感器数据（如角速度、加速度等），并根据这些数据来控制GPIO引脚状态。

5.3 RIOT系统如何创建线程？线程之间如何通信？列举出其他线程通信方法。

1. 在RIOT中创建线程

在RIOT系统中，线程可以通过`thread_create()`函数来创建。该函数的主要参数包括线程栈、栈大小、线程优先级、线程处理函数等。以下是创建线程的基本步骤：

- 线程栈：为每个线程分配一个栈空间，用来保存该线程的局部变量、函数调用等信息。
- 栈大小：指定栈的大小。
- 优先级：设定线程的优先级，优先级越高，线程越先执行。
- 线程处理函数：指定线程的执行函数。
- 参数：传递给线程函数的参数。

```
thread_create(stack1, sizeof(stack1),
             THREAD_PRIORITY_MAIN - 1, THREAD_CREATE_STACKTEST,
             thread_func, NULL, "thread_1");
```

2. 线程之间的通信

RIOT系统提供了多种线程间通信机制，最常用的方式是通过消息传递机制。RIOT中的每个线程都有一个消息队列，可以通过`msg_send()`、`msg_receive()`函数在线程之间发送和接收消息。以下是线程消息通信机制：

- `msg_t`: 消息结构，表示从一个线程发送到另一个线程的消息。
- `msg_send()`: 将消息发送到目标线程。
- `msg_receive()`: 接收消息。

```
#include "thread.h"
#include "msg.h"
#include <stdio.h>

#define MAIN_QUEUE_SIZE 8
msg_t main_msg_queue[MAIN_QUEUE_SIZE];

void *thread_func(void *arg) {
    (void)arg;
    msg_t msg;
    while (1) {
        msg_receive(&msg); // 接收消息
        printf("Received message: %d\n", msg.content.value);
    }
    return NULL;
}

int main(void) {
    char stack[THREAD_STACKSIZE_DEFAULT];

    // 创建线程
    thread_create(stack, sizeof(stack),
                 THREAD_PRIORITY_MAIN - 1, 0,
                 thread_func, NULL, "receiver_thread");

    // 初始化主线程的消息队列
    msg_init_queue(main_msg_queue, MAIN_QUEUE_SIZE);

    // 向线程发送消息
    msg_t msg;
    msg.content.value = 42; // 消息内容
    msg_send(&msg, thread_getpid());

    while (1) {
        printf("Main thread is running!\n");
        xtimer_sleep(2);
    }
}
```

3. 其他线程通信方法

除了消息传递，RIOT还支持其他多种线程间的通信方法：

1. Mutex (互斥锁)

- 功能：用于保护共享资源，防止多个线程同时访问同一资源。
- 使用场景：当多个线程需要访问同一全局变量或其他临界资源时，可以使用互斥锁来确保线程安全。

```
mutex_t lock;
mutex_init(&lock);

// 在线程中使用互斥锁保护共享资源
mutex_lock(&lock);
// 访问共享资源
mutex_unlock(&lock);
```

2. Semaphore (信号量)

- 功能：信号量是一种用于控制线程同步的机制。可以用来控制访问某个共享资源的线程数量，或者用于线程之间的同步。
- 使用场景：多个线程之间的协调，比如实现一个线程等待另一个线程完成特定任务。

```
semaphore_t sem;
semaphore_init(&sem, 0);

// 等待信号量
semaphore_wait(&sem);

// 释放信号量
semaphore_post(&sem);
```

3. Thread Flags (线程标志)

- 功能：线程标志是一种轻量级的线程间通信方式，允许一个线程设置另一个线程的标志，通知它执行某个任务。
- 使用场景：可以用来触发特定事件，适用于简单的线程同步需求。

```
thread_flags_set(thread_pid, THREAD_FLAG(0)); // 设置线程标志
```

4. Condition Variables (条件变量)

- 功能：条件变量和互斥锁一起使用，用于让线程等待某个条件变为真，然后再继续执行。
- 使用场景：在特定条件满足时，通知其他线程继续执行。

```
condition_variable_t cond;
mutex_t lock;

// 线程A
mutex_lock(&lock);
while (条件不满足) {
    condition_wait(&cond, &lock);
```

```

    }
    // 继续执行
    mutex_unlock(&lock);

    // 线程B
    mutex_lock(&lock);
    // 改变条件并通知
    condition_signal(&cond);
    mutex_unlock(&lock);

```

总结

在RIOT系统中，线程通过 `thread_create()` 创建，并可以通过消息传递、互斥锁、信号量等方式进行通信。这些通信机制帮助多个线程安全、有效地共享资源并协同工作。

6 实验数据记录和处理

以下实验记录均需结合屏幕截图，进行文字标注和描述（看完请删除本句）。

6.1 代码截图

1. ledcontroller.cpp

LED 灯光赋值和初始化

```

LEDController::LEDController(uint8_t gpio_r, uint8_t gpio_g, uint8_t gpio_b){
    // input your code
    led_gpio[0] = gpio_r;
    led_gpio[1] = gpio_g;
    led_gpio[2] = gpio_b;

    for (int i = 0; i < 3; i++) {
        gpio_init(led_gpio[i], GPIO_OUT); // 初始化所有引脚为输出模式
        gpio_write(led_gpio[i], 0); // 默认关闭所有灯
    }
}

```

LED 灯光定义和 LED 状态定义

```

void LEDController::change_led_color(uint8_t color){
    // input your code
    switch (color) {
        case COLOR_NONE:
            gpio_write(led_gpio[0], 0); // 红色关闭
            gpio_write(led_gpio[1], 0); // 绿色关闭
            gpio_write(led_gpio[2], 0); // 蓝色关闭
            break;
        case COLOR_RED:
            gpio_write(led_gpio[0], 1); // 红色打开
            gpio_write(led_gpio[1], 0); // 绿色关闭
            gpio_write(led_gpio[2], 0); // 蓝色关闭
            break;
        case COLOR_GREEN:
            gpio_write(led_gpio[0], 0); // 红色关闭
            gpio_write(led_gpio[1], 1); // 绿色打开
            gpio_write(led_gpio[2], 0); // 蓝色关闭
            break;
        case COLOR_YELLOW:
            gpio_write(led_gpio[0], 1); // 红色打开
            gpio_write(led_gpio[1], 1); // 绿色打开
            gpio_write(led_gpio[2], 0); // 蓝色关闭
            break;
        case COLOR_BLUE:
            gpio_write(led_gpio[0], 0); // 红色关闭
            gpio_write(led_gpio[1], 0); // 绿色关闭
            gpio_write(led_gpio[2], 1); // 蓝色打开
            break;
        case COLOR_MAGENTA:
            gpio_write(led_gpio[0], 1); // 红色打开
            gpio_write(led_gpio[1], 0); // 绿色关闭
            gpio_write(led_gpio[2], 1); // 蓝色打开
            break;
        case COLOR_CYAN:
            gpio_write(led_gpio[0], 0); // 红色关闭
            break;
    }
}

```

2. main.cpp

LED 线程，传感器获取的运动状态与灯光状态一一对应，进行灯光控制

```

void *_led_thread(void *arg)
{
    (void) arg;
    LEDController led(LED_GPIO_R, LED_GPIO_G, LED_GPIO_B);
    led.change_led_color(0);
    while(1){
        // Input your codes
        // Wait for a message to control the LED
        // Display different light colors based on the motion state of the device.
        msg_t msg;
        msg_receive(&msg);

        if (msg.content.value == Stationary) {
            led.change_led_color(COLOR_NONE);
        } else if (msg.content.value == Moving_X) {
            led.change_led_color(COLOR_CYAN);
        } else if (msg.content.value == Moving_Y) {
            led.change_led_color(COLOR_WHITE);
        } else if (msg.content.value == Tilted) {
            led.change_led_color(COLOR_RED);
        } else if (msg.content.value == Rotating) {
            led.change_led_color(COLOR_BLUE);
        } else if (msg.content.value == Moving) {
            led.change_led_color(COLOR_GREEN);
        } else if (msg.content.value == Shaking) {
            led.change_led_color(COLOR_YELLOW);
        } else {
            led.change_led_color(COLOR_NONE);
        }
    }
    return NULL;
}

```



detectmovement 模块，使用传感器获取的数据来进行状态判定

```

MoveState _detect_movement(MPU6050Data &data)
{
    // 计算加速度在X轴、Y轴和Z轴上的大小
    float acceleration_x = fabs(data.ax);
    float acceleration_y = fabs(data.ay);
    //float acceleration_z = fabs(data.az); // 新增Z轴加速度

    // 检测设备的总加速度
    float total_acceleration = sqrt(data.ax * data.ax + data.ay * data.ay + data.az * data.az);
    // 计算陀螺仪总角速度
    float total_rotation = sqrt(data.gx * data.gx + data gy * data.gy + data.gz * data.gz);

    // 判断水平静止：Z轴加速度接近0，X轴和Y轴加速度接近0，且没有旋转
    if (fabs(total_acceleration - g_acc) < ACC_THRESHOLD*3 && acceleration_x < ACC_THRESHOLD*3 && acceleration_y < ACC_THRESHOLD*3 && total_rotation < ROT_THRESHOLD/2) {
        return Stationary;
    }
    // 判断倾斜静止：设备的加速度在X、Y、Z轴上较为稳定，且没有旋转
    else if (fabs(total_acceleration - g_acc) < ACC_THRESHOLD*3 && total_rotation < ROT_THRESHOLD/2) {
        return Tilted;
    }
    // 判断X轴平移：X轴加速度大于阈值，且没有旋转
    else if (acceleration_x > ACC_THRESHOLD && acceleration_y < ACC_THRESHOLD && total_rotation < ROT_THRESHOLD) {
        return Moving_X;
    }
    // 判断Y轴平移：Y轴加速度大于阈值，且没有旋转
    else if (acceleration_y > ACC_THRESHOLD && acceleration_x < ACC_THRESHOLD && total_rotation < ROT_THRESHOLD) {
        return Moving_Y;
    }
    // 判断旋转：总角速度大于阈值
    else if (total_rotation > ROT_THRESHOLD) {
        return Rotating;
    }
    // 判断移动状态：总加速度明显偏离重力加速度
    else if (fabs(total_acceleration - g_acc) > ACC_THRESHOLD) {
        return Moving;
    }
}

```



IMU 线程，处理直接获取的传感器数据，换算成物理数据方便状态判定

```

void *_imu_thread(void *arg)
{
    (void) arg;
    MPU6050 mpu;
    mpu.initialize();

    // 配置陀螺仪加速度计全量程范围
    uint8_t gyro_fs = mpu.getFullScaleGyroRange();
    uint8_t accel_fs_g = mpu.getFullScaleAccelRange();

    // 默认转换因子
    float gyro_fs_convert = 1.0;
    float accel_fs_real = 1.0;

    // 根据陀螺仪量程范围设置转换因子
    switch (gyro_fs) {
        case MPU6050_GYRO_FS_250:
            gyro_fs_convert = 131.0;
            break;
        case MPU6050_GYRO_FS_500:
            gyro_fs_convert = 65.5;
            break;
        case MPU6050_GYRO_FS_1000:
            gyro_fs_convert = 32.8;
            break;
        case MPU6050_GYRO_FS_2000:
            gyro_fs_convert = 16.4;
            break;
        default:
            printf("[IMU_THREAD] Unknown GYRO_FS: 0x%u\n", gyro_fs);
    }
}

```



```

// 根据加速度计量程范围设置转换因子
switch (accel_fs_g) {
    case MPU6050_ACCEL_FS_2:
        accel_fs_real = g_acc * 2;
        break;
    case MPU6050_ACCEL_FS_4:
        accel_fs_real = g_acc * 4;
        break;
    case MPU6050_ACCEL_FS_8:
        accel_fs_real = g_acc * 8;
        break;
    case MPU6050_ACCEL_FS_16:
        accel_fs_real = g_acc * 16;
        break;
    default:
        printf("[IMU_THREAD] Unknown ACCEL_FS: 0x%x\n", accel_fs_g);
}
}

// 计算加速度计转换因子
float accel_fs_convert = 32768.0 / accel_fs_real;

// 初始化原始传感器数据的变量
int16_t ax, ay, az, gx, gy, gz;

// 延迟一秒以确保传感器稳定
delay_ms(1000);

```

IMU 线程主循环，周期性读取传感器数据并进行输出

```

// 主循环: 周期性读取传感器数据并进行输出
while (1) {
    // 读取传感器数据
    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    // 将原始数据转换为实际值
    MPU6050Data data;
    data.ax = ax / accel_fs_convert;
    data.ay = ay / accel_fs_convert;
    data.az = az / accel_fs_convert;
    data.gx = gx / gyro_fs_convert;
    data.gy = gy / gyro_fs_convert;
    data.gz = gz / gyro_fs_convert;

    // 输出传感器数据
    printf("-----\n");
    printf("[IMU_THREAD] Accelerometer (X,Y,Z): (%.02f, %.02f, %.02f) m/s^2\n", data.ax, data.ay, data.az);
    printf("[IMU_THREAD] gyroscope (X,Y,Z): (%.02f, %.02f, %.02f) °/s\n", data.gx, data.gy, data.gz);

    // 检测运动状态
    Movestate state = detectMovement(data);

    // 向LED线程发送消息通知当前的运动状态
    msg_t msg;
    msg.content.value = (int)state;
    msg_send(&msg, _led_pid); // 发送消息到LED控制线程

    // 200ms的延时
    delay_ms(200);
}
return NULL;
}

```

main 文件的主程序，进行线程创建

```

int main(void)
{
    _led_pid = thread_create(stack_for_led_thread, sizeof(stack_for_led_thread), THREAD_PRIORITY_MAIN - 2,
                           THREAD_CREATE_STACKTEST, _led_thread, NULL,
                           "led_controller_thread");
    if (_led_pid <= KERNEL_PID_UNDEF) {
        printf("[MAIN] Creation of receiver thread failed\n");
        return 1;
    }
    thread_create(stack_for_imu_thread, sizeof(stack_for_imu_thread), THREAD_PRIORITY_MAIN - 1,
                  THREAD_CREATE_STACKTEST, _imu_thread, NULL,
                  "imu_read_thread");
    printf("[Main] Initialization successful - starting the shell now\n");
    while(1)
    {
        ztimer_sleep(ZTIMER_USEC, 1000000); // 1秒延时
    }
    return 0;
}

```

6.2 串口惯性传感器数据及运动状态实验结果截图

1 Stationary

输出数据中，Z轴加速度接近 g ，X轴和Y轴加速度接近 0，且没有旋转

```

2024-09-18 20:17:37,225 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.06, 0.67, 0.06) °/s
2024-09-18 20:17:37,219 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.06, 0.67, 0.06) °/s
2024-09-18 20:17:37,425 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.38, -0.13, 9.87) m/s^2
2024-09-18 20:17:37,428 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.12, 0.67, 0.06) °/s
2024-09-18 20:17:37,422 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.12, 0.67, 0.06) °/s
2024-09-18 20:17:37,428 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.36, -0.10, 9.88) m/s^2
2024-09-18 20:17:37,531 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.06, 0.85, -0.12) °/s
2024-09-18 20:17:37,526 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.39, -0.12, 9.84) m/s^2
2024-09-18 20:17:37,834 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.24, 0.55, 0.06) °/s
2024-09-18 20:17:38,029 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.24, 0.55, 0.06) °/s
2024-09-18 20:17:38,034 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.40, -0.10, 9.93) m/s^2
2024-09-18 20:17:38,037 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.06, 0.67, 0.06) °/s
2024-09-18 20:17:38,232 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.06, 0.67, 0.06) °/s
2024-09-18 20:17:38,241 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.40, -0.09, 9.93) m/s^2
2024-09-18 20:17:38,245 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.12, 0.79, 0.06) °/s
2024-09-18 20:17:38,435 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.44, -0.17, 9.83) m/s^2
2024-09-18 20:17:38,444 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.37, 0.85, -0.06) °/s
2024-09-18 20:17:38,438 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.37, 0.85, -0.06) °/s
2024-09-18 20:17:38,444 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.34, -0.12, 9.85) m/s^2
2024-09-18 20:17:38,447 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.12, 0.61, -0.06) °/s
2024-09-18 20:17:38,487 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.45, -0.07, 9.81) m/s^2
2024-09-18 20:17:38,856 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.24, 0.49, 0.12) °/s
2024-09-18 20:17:39,045 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.24, 0.61, 0.06) °/s
2024-09-18 20:17:39,050 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.41, -0.11, 9.84) m/s^2
2024-09-18 20:17:39,053 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.24, 0.61, 0.06) °/s
2024-09-18 20:17:39,053 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.24, 0.61, 0.06) °/s
2024-09-18 20:17:39,253 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.44, -0.09, 9.88) m/s^2
2024-09-18 20:17:39,257 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.06, 0.79, -0.12) °/s
2024-09-18 20:17:39,451 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.06, 0.79, -0.12) °/s
2024-09-18 20:17:39,457 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.39, -0.14, 9.84) m/s^2
2024-09-18 20:17:39,466 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.30, 0.61, -0.06) °/s
2024-09-18 20:17:39,468 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.30, 0.61, -0.06) °/s
2024-09-18 20:17:39,660 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.43, -0.07, 9.83) m/s^2
2024-09-18 20:17:39,663 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.24, 0.73, -0.12) °/s
2024-09-18 20:17:39,857 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.24, 0.73, -0.12) °/s
2024-09-18 20:17:39,066 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.43, -0.07, 9.83) m/s^2
2024-09-18 20:17:39,063 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.24, 0.73, -0.12) °/s
2024-09-18 20:17:39,063 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.24, 0.73, -0.12) °/s
2024-09-18 20:17:39,857 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.44, -0.10, 9.88) m/s^2
2024-09-18 20:17:39,866 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.18, 0.98, -0.06) °/s

```

2 Moving_X

X轴加速度大于阈值，Y轴加速度小于阈值，且没有旋转

```

2024-09-18 19:31:59,561 # [IMU_THREAD] Accelerometer (X,Y,Z): (-0.17, 0.22, 9.71) m/s^2
2024-09-18 19:31:59,565 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.79, 2.67, 0.06) °/s
2024-09-18 19:31:59,565 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.79, 2.67, 0.06) °/s
2024-09-18 19:31:59,764 # [IMU_THREAD] Accelerometer (X,Y,Z): (1.42, 0.51, 9.85) m/s^2
2024-09-18 19:31:59,768 # [IMU_THREAD] Gyroscope (X,Y,Z): (-1.16, -1.48, 24.15) °/s
2024-09-18 19:31:59,967 # [IMU_THREAD] Gyroscope (X,Y,Z): (-1.16, -1.48, 24.15) °/s
2024-09-18 19:31:59,967 # [IMU_THREAD] Accelerometer (X,Y,Z): (2.83, 0.63, 10.05) m/s^2
2024-09-18 19:31:59,971 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.24, 0.49, 29.39) °/s
2024-09-18 19:32:00,165 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.24, 0.49, 29.39) °/s
2024-09-18 19:32:00,171 # [IMU_THREAD] Accelerometer (X,Y,Z): (-0.55, -0.25, 9.87) m/s^2
2024-09-18 19:32:00,174 # [IMU_THREAD] Gyroscope (X,Y,Z): (2.01, 0.79, -12.80) °/s

```

3 Moving_Y

X轴加速度小于阈值，Y轴加速度大于阈值，且没有旋转

```

2024-09-18 19:28:21,242 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.32, -1.56, 9.79) m/s^2
2024-09-18 19:28:21,215 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.12, -1.34, 1.83) °/s
2024-09-18 19:28:21,510 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.12, -1.34, 1.83) °/s
2024-09-18 19:28:21,515 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.30, 1.18, 9.91) m/s^2
2024-09-18 19:28:21,519 # [IMU_THREAD] Gyroscope (X,Y,Z): (3.96, 2.26, -15.24) °/s
2024-09-18 19:28:21,713 # [IMU_THREAD] Gyroscope (X,Y,Z): (3.96, 2.26, -15.24) °/s
2024-09-18 19:28:21,718 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.78, 1.01, 9.68) m/s^2
2024-09-18 19:28:21,722 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.18, 0.49, -0.49) °/s
2024-09-18 19:28:21,916 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.18, 0.49, -0.49) °/s
2024-09-18 19:28:21,917 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.38, -0.86, 10.08) m/s^2
2024-09-18 19:28:21,925 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.00, 2.13, 17.13) °/s
2024-09-18 19:28:22,119 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.00, 2.13, 17.13) °/s
2024-09-18 19:28:22,125 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.47, -2.14, 10.11) m/s^2
2024-09-18 19:28:22,128 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.06, 1.83, 30.18) °/s
2024-09-18 19:28:22,323 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.06, 1.83, 30.18) °/s

```

4 Tilted

输出数据中，设备的加速度在X、Y、Z轴上较为稳定，且没有旋转

```

2024-09-18 19:08:34,100 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.43, -0.85, 0.98) °/s
2024-09-18 19:08:34,295 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.43, -0.85, 0.98) °/s
2024-09-18 19:08:34,300 # [IMU_THREAD] Accelerometer (X,Y,Z): (3.89, -0.35, 9.13) m/s^2
2024-09-18 19:08:34,304 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.49, -7.74, 0.24) °/s
2024-09-18 19:08:34,498 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.49, -7.74, 0.24) °/s
2024-09-18 19:08:34,503 # [IMU_THREAD] Accelerometer (X,Y,Z): (3.86, -0.31, 9.06) m/s^2
2024-09-18 19:08:34,506 # [IMU_THREAD] Gyroscope (X,Y,Z): (1.40, 11.10, -0.73) °/s
2024-09-18 19:08:34,761 # [IMU_THREAD] Gyroscope (X,Y,Z): (1.40, 11.10, -0.73) °/s
2024-09-18 19:08:34,765 # [IMU_THREAD] Accelerometer (X,Y,Z): (3.66, -0.25, 9.34) m/s^2
2024-09-18 19:08:34,769 # [IMU_THREAD] Gyroscope (X,Y,Z): (4.33, 3.48, 13.05) °/s
2024-09-18 19:08:34,905 # [IMU_THREAD] Gyroscope (X,Y,Z): (4.33, 3.48, 13.05) °/s
2024-09-18 19:08:34,910 # [IMU_THREAD] Accelerometer (X,Y,Z): (4.06, -0.47, 8.30) m/s^2
2024-09-18 19:08:34,913 # [IMU_THREAD] Gyroscope (X,Y,Z): (15.30, -31.77, 44.02) °/s
2024-09-18 19:08:35,108 # [IMU_THREAD] Gyroscope (X,Y,Z): (15.30, -31.77, 44.02) °/s
2024-09-18 19:08:35,113 # [IMU_THREAD] Accelerometer (X,Y,Z): (3.93, -0.42, 9.27) m/s^2
2024-09-18 19:08:35,117 # [IMU_THREAD] Gyroscope (X,Y,Z): (-1.04, -5.67, -3.78) °/s
2024-09-18 19:08:35,311 # [IMU_THREAD] Gyroscope (X,Y,Z): (-1.04, -5.67, -3.78) °/s
2024-09-18 19:08:35,317 # [IMU_THREAD] Accelerometer (X,Y,Z): (4.03, -0.22, 9.47) m/s^2
2024-09-18 19:08:35,326 # [IMU_THREAD] Gyroscope (X,Y,Z): (-1.77, -15.43, -6.34) °/s
2024-09-18 19:08:35,514 # [IMU_THREAD] Gyroscope (X,Y,Z): (-1.77, -15.43, -6.34) °/s
2024-09-18 19:08:35,526 # [IMU_THREAD] Accelerometer (X,Y,Z): (4.75, -0.33, 8.83) m/s^2
2024-09-18 19:08:35,529 # [IMU_THREAD] Gyroscope (X,Y,Z): (-2.26, -20.98, -4.76) °/s
2024-09-18 19:08:35,722 # [IMU_THREAD] Accelerometer (X,Y,Z): (4.88, -0.13, 8.82) m/s^2
2024-09-18 19:08:35,727 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.30, 0.12, 0.12) °/s
2024-09-18 19:08:35,921 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.30, 0.12, 0.12) °/s
2024-09-18 19:08:35,926 # [IMU_THREAD] Accelerometer (X,Y,Z): (4.95, -0.26, 8.70) m/s^2
2024-09-18 19:08:35,936 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.30, 1.89, -0.30) °/s
2024-09-18 19:08:36,124 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.30, 1.89, -0.30) °/s
2024-09-18 19:08:36,130 # [IMU_THREAD] Accelerometer (X,Y,Z): (4.97, -0.21, 8.62) m/s^2
2024-09-18 19:08:36,133 # [IMU_THREAD] Gyroscope (X,Y,Z): (-1.83, 5.06, -3.72) °/s
2024-09-18 19:08:36,327 # [IMU_THREAD] Gyroscope (X,Y,Z): (-1.83, 5.06, -3.72) °/s
2024-09-18 19:08:36,333 # [IMU_THREAD] Accelerometer (X,Y,Z): (4.53, -0.28, 9.13) m/s^2
2024-09-18 19:08:36,336 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.30, 1.10, -0.18) °/s
2024-09-18 19:08:36,530 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.30, 1.10, -0.18) °/s
2024-09-18 19:08:36,536 # [IMU_THREAD] Accelerometer (X,Y,Z): (4.76, -0.24, 8.58) m/s^2
2024-09-18 19:08:36,540 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.00, 1.83, 0.43) °/s
2024-09-18 19:08:36,735 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.00, 1.83, 0.43) °/s
2024-09-18 19:08:36,737 # [IMU_THREAD] Accelerometer (X,Y,Z): (4.69, -0.26, 8.66) m/s^2
2024-09-18 19:08:36,742 # [IMU_THREAD] Gyroscope (X,Y,Z): (-1.83, 5.06, -3.72) °/s
2024-09-18 19:08:36,942 # [IMU_THREAD] Accelerometer (X,Y,Z): (4.62, -0.22, 8.87) m/s^2
2024-09-18 19:08:36,946 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.73, 0.61, -0.85) °/s

```

5 Rotating

输出数据中，设备的总角速度大于阈值

```
2024-09-18 19:11:42,458 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.52, 0.11, 0.88) m/s^2
2024-09-18 19:11:42,462 # [IMU_THREAD] Gyroscope (X,Y,Z): (-5.79, -46.89, -1.34) °/s
2024-09-18 19:11:42,656 # ...
2024-09-18 19:11:42,664 # [IMU_THREAD] Accelerometer (X,Y,Z): (2.62, -0.23, 9.74) m/s^2
2024-09-18 19:11:42,664 # [IMU_THREAD] Gyroscope (X,Y,Z): (1.89, -42.01, 4.82) °/s
2024-09-18 19:11:42,859 # ...
2024-09-18 19:11:42,868 # [IMU_THREAD] Accelerometer (X,Y,Z): (3.61, -0.42, 9.58) m/s^2
2024-09-18 19:11:42,868 # [IMU_THREAD] Gyroscope (X,Y,Z): (-1.10, 16.16, -2.26) °/s
2024-09-18 19:11:43,862 # ...
2024-09-18 19:11:43,868 # [IMU_THREAD] Accelerometer (X,Y,Z): (4.27, -0.48, 8.23) m/s^2
2024-09-18 19:11:43,872 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.12, -13.11, 1.46) °/s
2024-09-18 19:11:43,266 # ...
2024-09-18 19:11:43,272 # [IMU_THREAD] Accelerometer (X,Y,Z): (4.55, -0.24, 8.64) m/s^2
2024-09-18 19:11:43,275 # [IMU_THREAD] Gyroscope (X,Y,Z): (-2.01, -8.11, -2.99) °/s
2024-09-18 19:11:43,469 # ...
2024-09-18 19:11:43,475 # [IMU_THREAD] Accelerometer (X,Y,Z): (4.62, -0.25, 8.91) m/s^2
2024-09-18 19:11:43,479 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.43, -7.38, 0.61) °/s
2024-09-18 19:11:43,672 # ...
2024-09-18 19:11:43,678 # [IMU_THREAD] Accelerometer (X,Y,Z): (5.08, -0.44, 8.85) m/s^2
2024-09-18 19:11:43,681 # [IMU_THREAD] Gyroscope (X,Y,Z): (1.95, -16.46, 4.63) °/s
2024-09-18 19:11:43,676 # ...
```

6 Moving

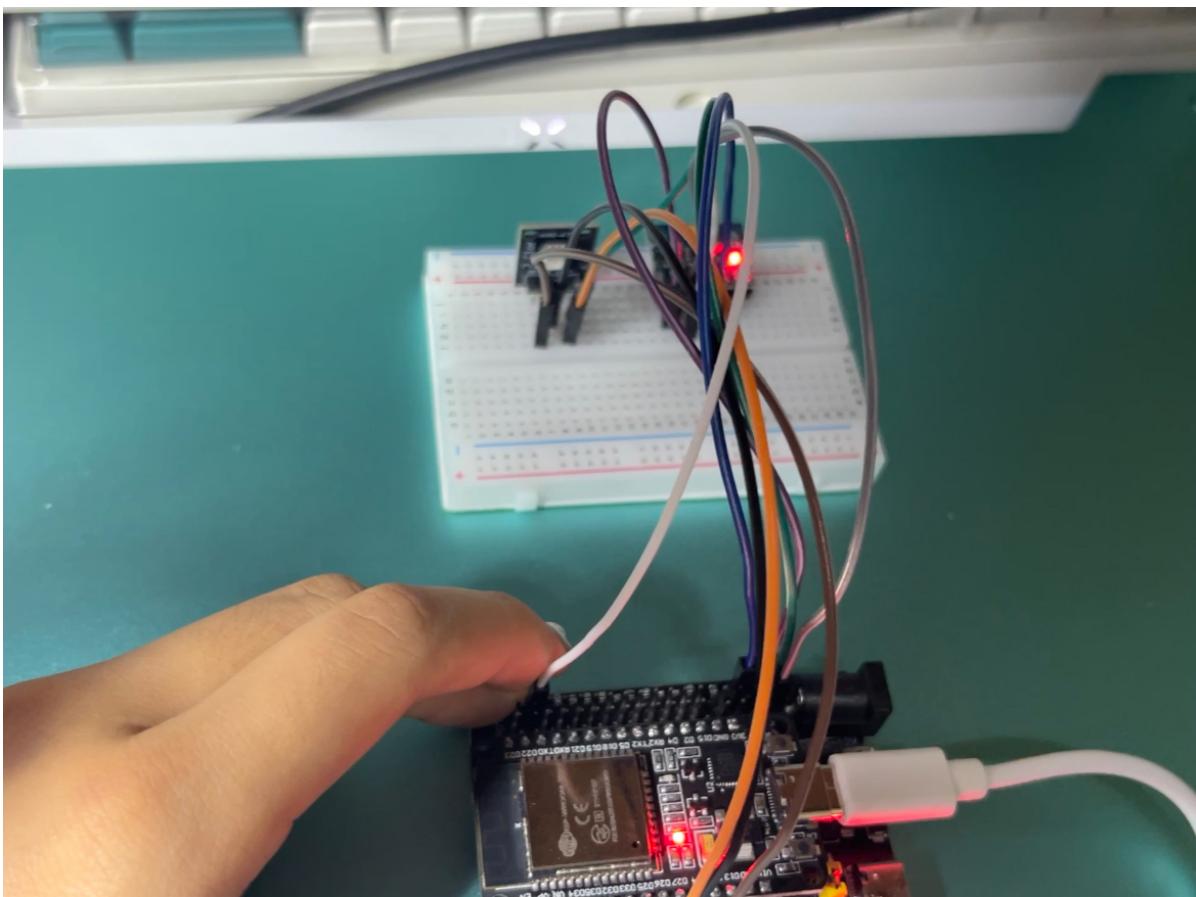
输出数据中，设备的总加速度明显偏离重力加速度

```
2024-09-18 19:13:44,593 # [IMU_THREAD] Gyroscope (X,Y,Z): (0.24, 0.67, -39.15) °/s
2024-09-18 19:13:44,787 # ...
2024-09-18 19:13:44,793 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.46, 0.86, 9.78) m/s^2
2024-09-18 19:13:44,796 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.91, -0.18, -0.18) °/s
2024-09-18 19:13:44,800 # ...
2024-09-18 19:13:44,996 # [IMU_THREAD] Accelerometer (X,Y,Z): (-1.13, -1.91, 10.01) m/s^2
2024-09-18 19:13:45,000 # [IMU_THREAD] Gyroscope (X,Y,Z): (-1.22, -0.91, -26.28) °/s
2024-09-18 19:13:45,194 # ...
2024-09-18 19:13:45,204 # [IMU_THREAD] Accelerometer (X,Y,Z): (1.13, 2.43, 10.02) m/s^2
2024-09-18 19:13:45,203 # [IMU_THREAD] Gyroscope (X,Y,Z): (4.57, 5.55, 14.63) °/s
2024-09-18 19:13:45,397 # ...
2024-09-18 19:13:45,403 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.42, -0.06, 9.81) m/s^2
2024-09-18 19:13:45,406 # [IMU_THREAD] Gyroscope (X,Y,Z): (-1.46, -1.34, -0.12) °/s
2024-09-18 19:13:45,601 # ...
2024-09-18 19:13:45,606 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.97, 0.21, 10.13) m/s^2
2024-09-18 19:13:45,610 # [IMU_THREAD] Gyroscope (X,Y,Z): (-2.50, -0.36, -16.83) °/s
2024-09-18 19:13:45,804 # ...
2024-09-18 19:13:45,805 # [IMU_THREAD] Accelerometer (X,Y,Z): (-0.08, -2.35, 10.01) m/s^2
2024-09-18 19:13:45,813 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.49, 0.67, 16.46) °/s
2024-09-18 19:13:46,007 # ...
2024-09-18 19:13:46,014 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.14, 0.89, 9.79) m/s^2
2024-09-18 19:13:46,017 # [IMU_THREAD] Gyroscope (X,Y,Z): (1.77, 0.73, -10.18) °/s
2024-09-18 19:13:46,211 # ...
2024-09-18 19:13:46,216 # [IMU_THREAD] Accelerometer (X,Y,Z): (3.90, 3.66, 9.55) m/s^2
2024-09-18 19:13:46,219 # [IMU_THREAD] Gyroscope (X,Y,Z): (-2.87, -0.61, 39.45) °/s
2024-09-18 19:13:46,414 # ...
2024-09-18 19:13:46,419 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.64, -0.09, 10.18) m/s^2
2024-09-18 19:13:46,423 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.18, 2.80, 0.18) °/s
2024-09-18 19:13:46,617 # ...
2024-09-18 19:13:46,622 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.49, -0.15, 9.92) m/s^2
2024-09-18 19:13:46,626 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.18, 0.98, 0.08) °/s
2024-09-18 19:13:46,820 # ...
2024-09-18 19:13:46,826 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.06, -0.59, 9.88) m/s^2
2024-09-18 19:13:46,830 # [IMU_THREAD] Gyroscope (X,Y,Z): (1.04, 1.71, 14.21) °/s
2024-09-18 19:13:47,023 # ...
2024-09-18 19:13:47,026 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.55, 0.38, -9.49) m/s^2
2024-09-18 19:13:46,833 # [IMU_THREAD] Gyroscope (X,Y,Z): (1.04, 1.71, 14.21) °/s
2024-09-18 19:13:47,029 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.55, 0.38, 9.49) m/s^2
2024-09-18 19:13:47,032 # [IMU_THREAD] Gyroscope (X,Y,Z): (1.04, 1.83, 5.67) °/s
2024-09-18 19:13:47,227 # ...
2024-09-18 19:13:47,232 # [IMU_THREAD] Accelerometer (X,Y,Z): (0.37, -0.01, 9.93) m/s^2
2024-09-18 19:13:47,235 # [IMU_THREAD] Gyroscope (X,Y,Z): (-0.18, 0.73, -0.12) °/s
```

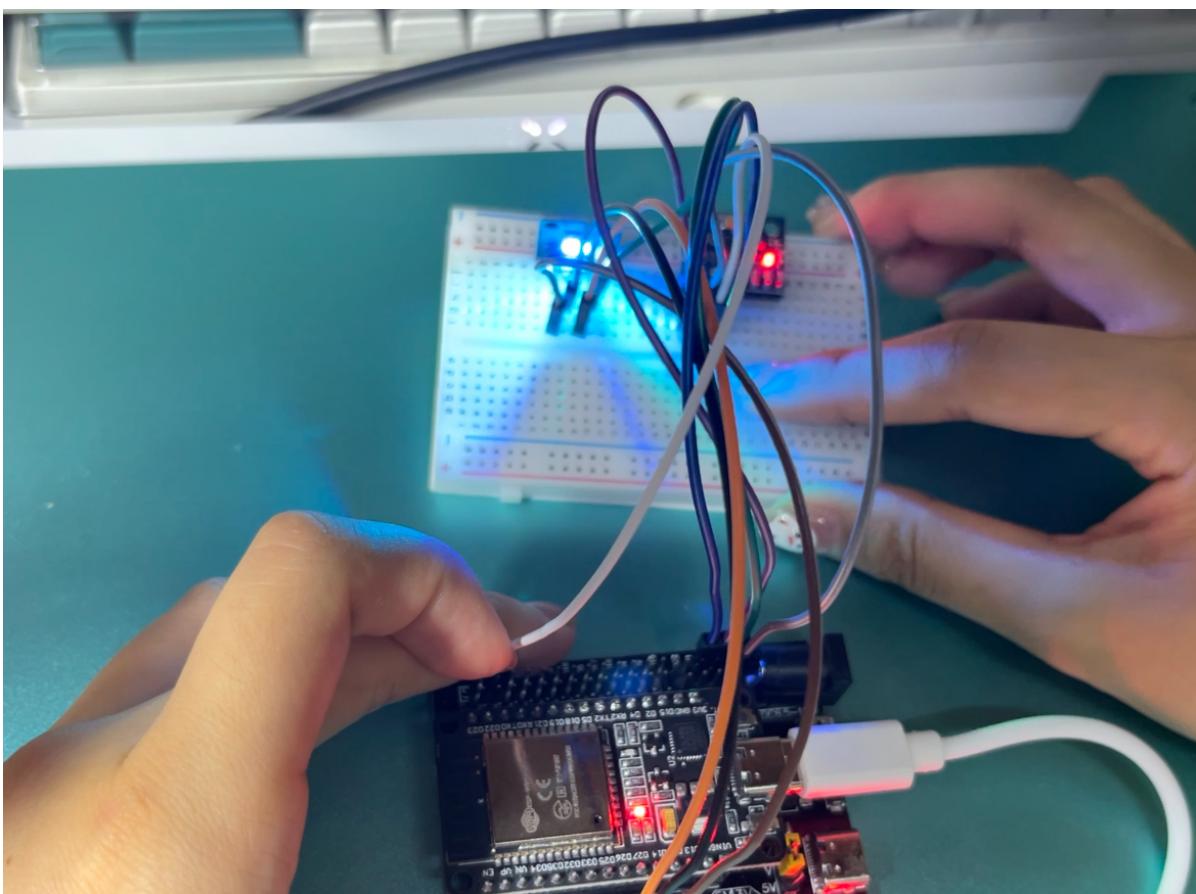
6.3 LED灯颜色照片

由于大部分 LED 灯都在运动过程中亮色，所以我采用了视频录制的方式，以下照片由视频中截取获得。

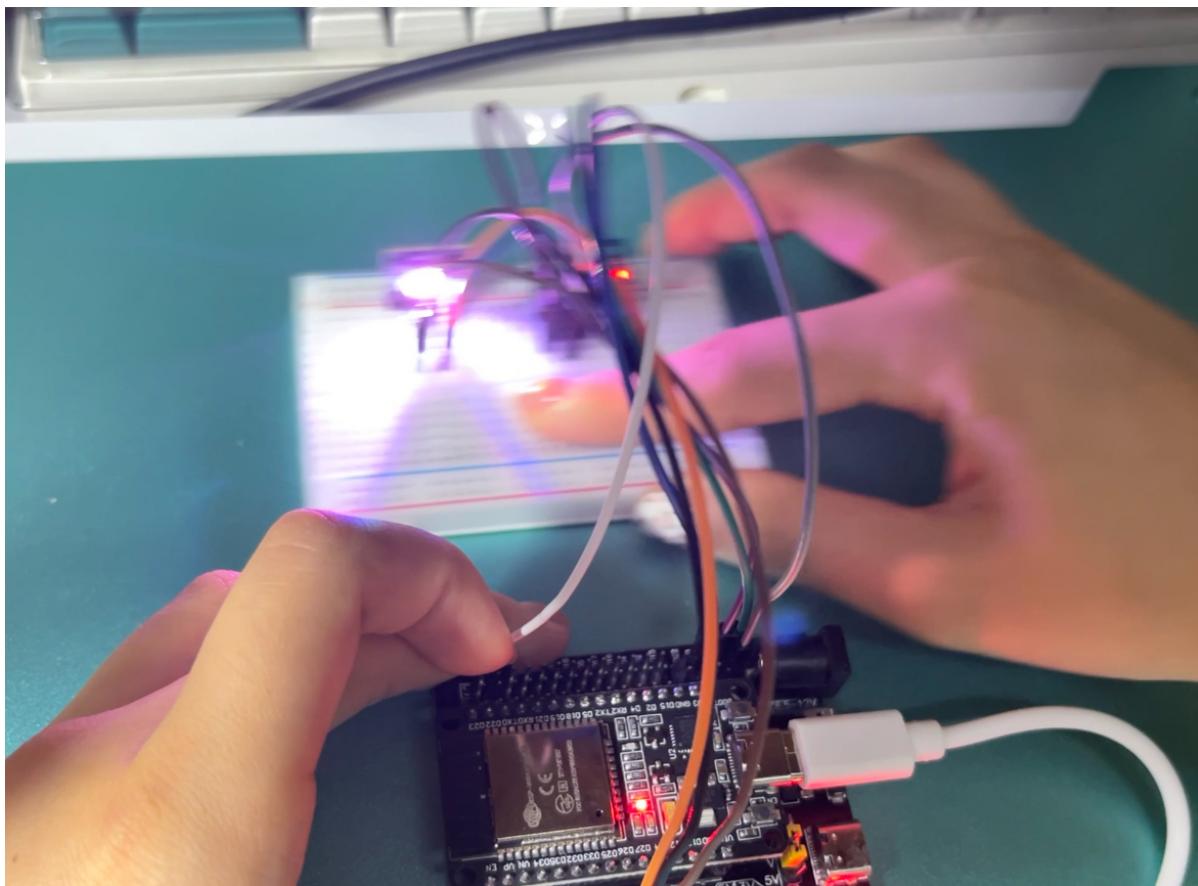
1 Stationary: NONE



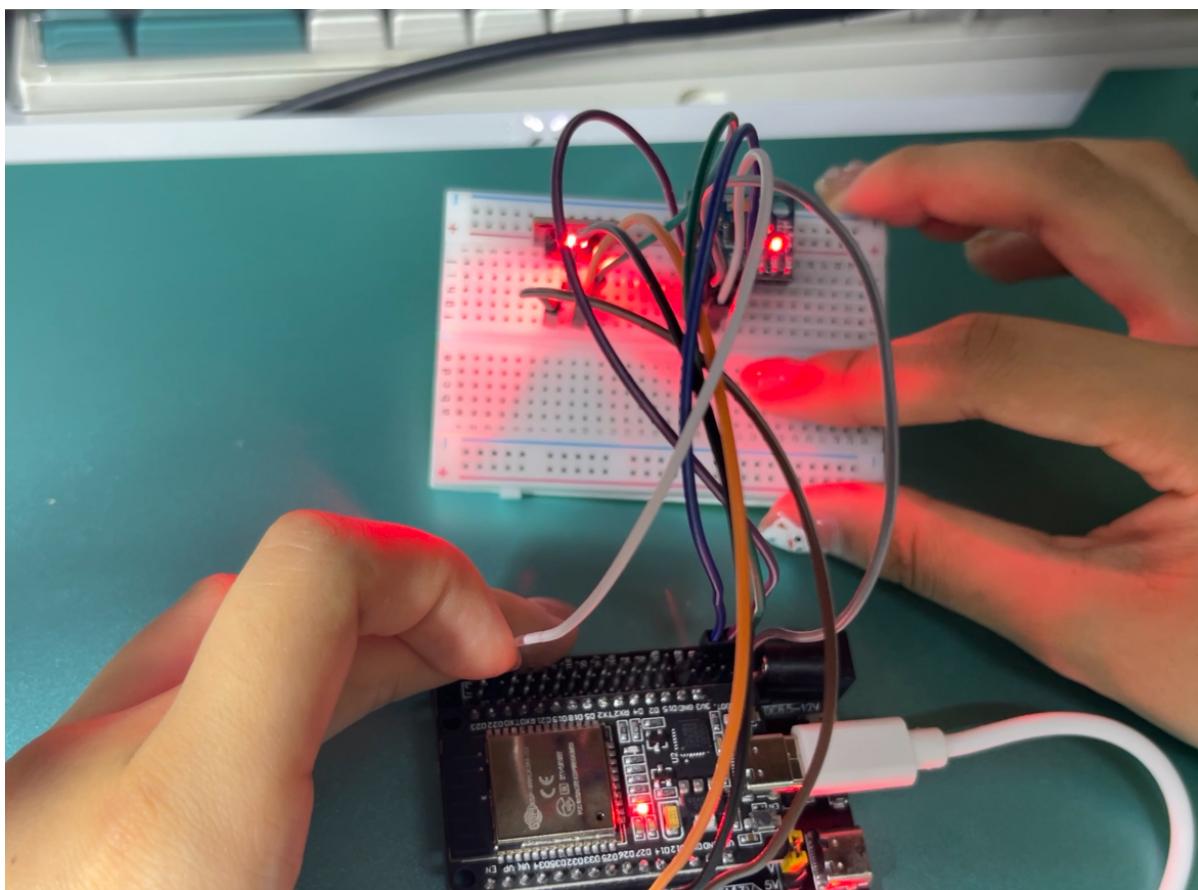
2 Moving_X: CYAN



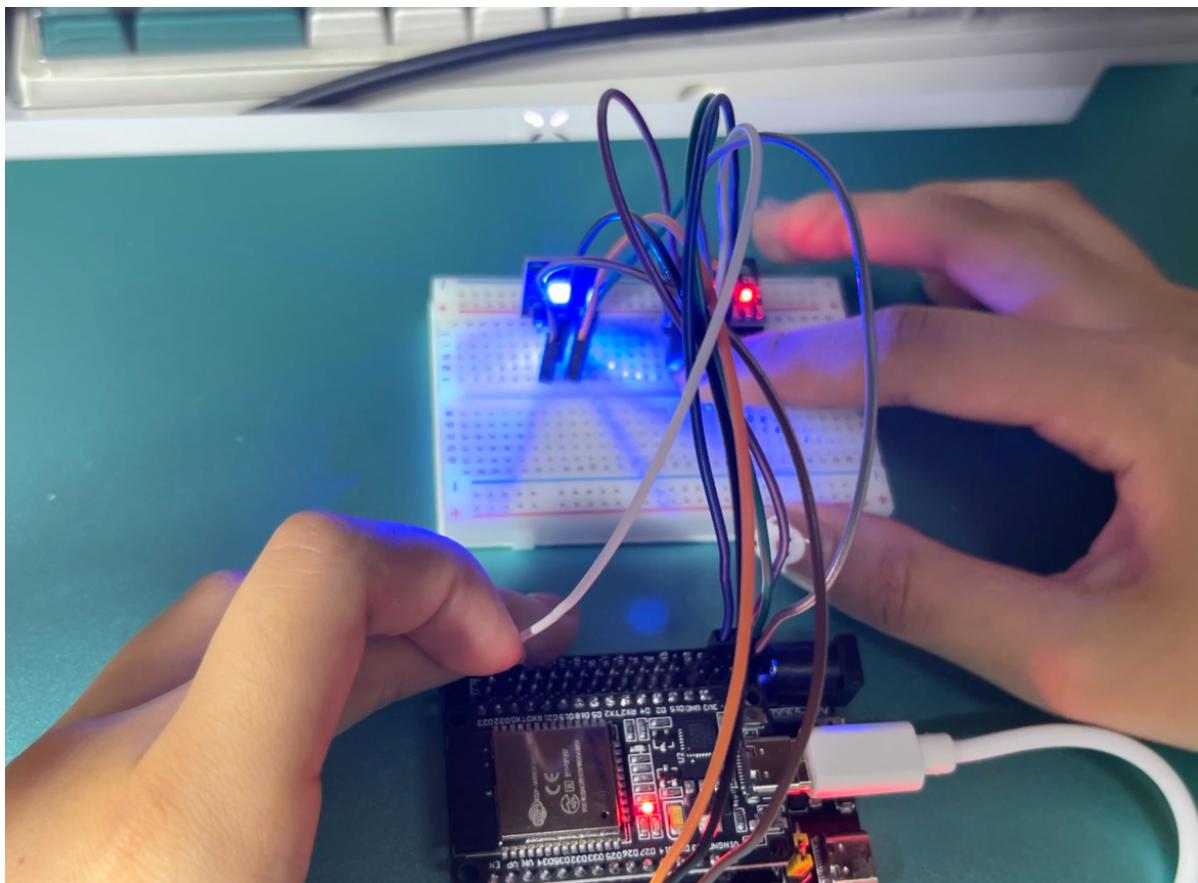
3 Moving_Y: WHITE



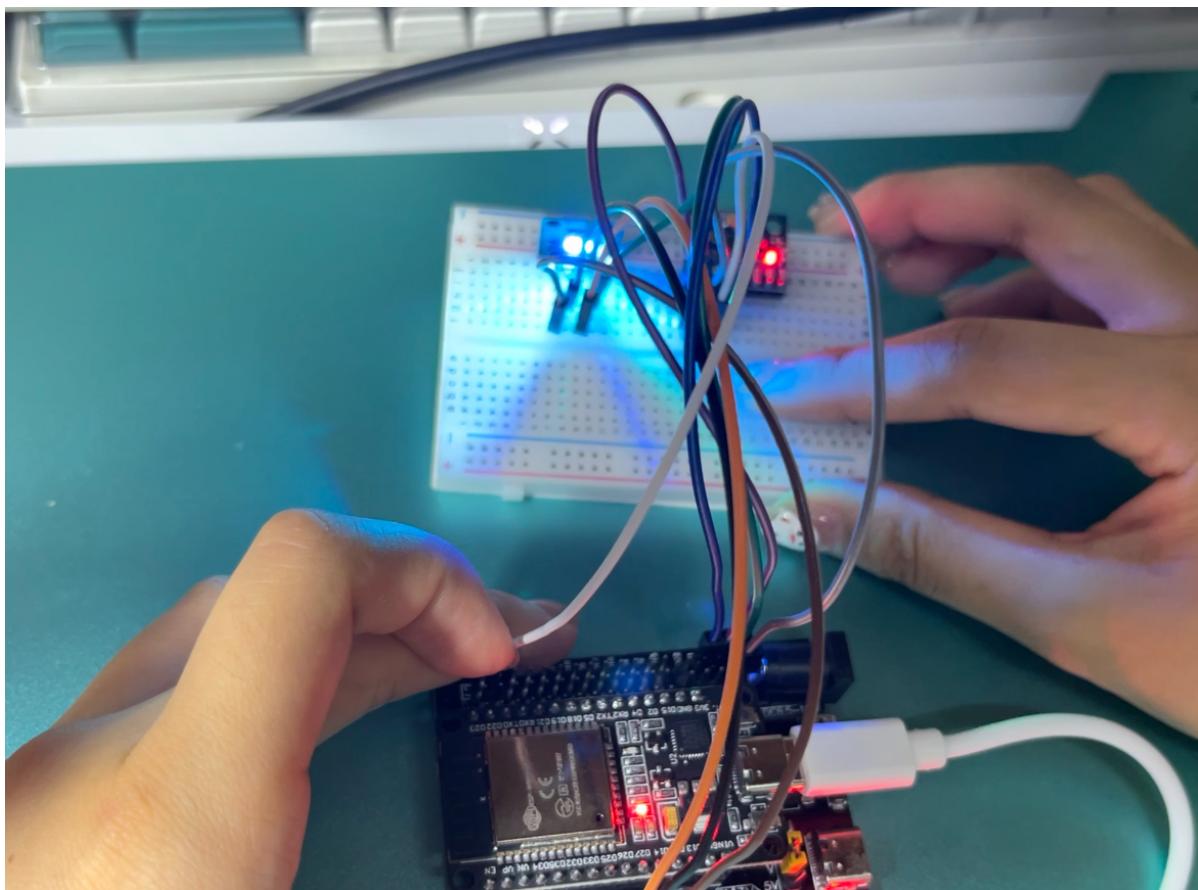
4 Tilted: RED



5 Rotating: BLUE



6 Moving: GREEN



7 实验结果与分析

7.1 基于该实验，总结开发一个完整的传感器应用基本流程。

1. 需求分析与应用设计

- 明确应用场景：确定传感器的具体用途（例如环境监测、运动检测等）。
- 选择传感器类型：根据需求选择合适的传感器（如温湿度传感器、压力传感器等）。
- 制定功能需求：列出应用需要实现的主要功能，例如数据采集、数据处理、显示等。

2. 硬件选型与连接

- 选择开发平台：选择适合的开发平台或控制器，如Arduino、ESP32等。
- 传感器硬件选型：根据应用场景选择相应的传感器，确保其性能满足需求。
- 传感器接线：根据传感器和控制器的接口类型（如I2C、SPI、UART）进行硬件连接。

3. 传感器校准与测试

- 初步测试：利用开发板和基础代码，测试传感器能否正常工作，读取的数值是否合理。
- 传感器校准：对传感器进行校准，确保其输出数据的准确性，必要时需要参考标准设备或数据进行调校。

4. 编写软件

- 开发驱动代码：编写或引入传感器的驱动程序，确保控制器能正确读取传感器数据。
- 数据处理算法：编写算法处理从传感器获取的数据，例如过滤噪声、平滑数据、数据归一化等。
- 用户交互界面：如果应用涉及交互，可以设计用户界面或通过终端显示数据。
- 数据存储与传输：根据需求决定是否将数据保存到本地存储，或者通过Wi-Fi、蓝牙等传输到云端或其他设备。

5. 调试与优化

- 逐步调试：对硬件、软件进行逐步调试，确保应用稳定运行，传感器数据准确采集和处理。
- 性能优化：根据调试结果，优化数据处理算法、减少功耗、提高应用运行效率。

6. 部署与维护

- 部署应用：将应用部署到最终环境，确保传感器能够长期稳定运行。
- 后续维护：定期检查传感器的工作状态，维护软件的更新，处理可能出现的故障。

8 Appendix

8.1 ledcontroller.cpp

```
#include "ztimer.h"
#include "ledcontroller.hh"
#include "periph/gpio.h"
#include <stdio.h>

LEDController::LEDController(uint8_t gpio_r, uint8_t gpio_g, uint8_t gpio_b){
    printf("LED Controller initialized with (RGB: GPIO%d, GPIO%d, GPIO%d)\n",
gpio_r, gpio_g, gpio_b);
    // input your code
    led_gpio[0] = gpio_r;
    led_gpio[1] = gpio_g;
    led_gpio[2] = gpio_b;

    for (int i = 0; i < 3; i++) {
        gpio_init(led_gpio[i], GPIO_OUT); //初始化所有引脚为输出模式
        gpio_write(led_gpio[i], 0); // 默认关闭所有灯
    }
}

void LEDController::change_led_color(uint8_t color){
    // input your code
    switch (color) {
        case COLOR_NONE:
            gpio_write(led_gpio[0], 0); // 红色关闭
            gpio_write(led_gpio[1], 0); // 绿色关闭
            gpio_write(led_gpio[2], 0); // 蓝色关闭
            break;
        case COLOR_RED:
            gpio_write(led_gpio[0], 1); // 红色打开
            gpio_write(led_gpio[1], 0); // 绿色关闭
            gpio_write(led_gpio[2], 0); // 蓝色关闭
            break;
        case COLOR_GREEN:
            gpio_write(led_gpio[0], 0); // 红色关闭
            gpio_write(led_gpio[1], 1); // 绿色打开
            gpio_write(led_gpio[2], 0); // 蓝色关闭
            break;
        case COLOR_YELLOW:
            gpio_write(led_gpio[0], 1); // 红色打开
            gpio_write(led_gpio[1], 1); // 绿色打开
            gpio_write(led_gpio[2], 0); // 蓝色关闭
            break;
        case COLOR_BLUE:
            gpio_write(led_gpio[0], 0); // 红色关闭
            gpio_write(led_gpio[1], 0); // 绿色关闭
            gpio_write(led_gpio[2], 1); // 蓝色打开
            break;
        case COLOR_MAGENTA:
            gpio_write(led_gpio[0], 1); // 红色打开
            gpio_write(led_gpio[1], 0); // 绿色关闭
            gpio_write(led_gpio[2], 1); // 蓝色打开
            break;
    }
}
```

```

        case COLOR_CYAN:
            gpio_write(led_gpio[0], 0); // 红色关闭
            gpio_write(led_gpio[1], 1); // 绿色打开
            gpio_write(led_gpio[2], 1); // 蓝色打开
            break;
        case COLOR_WHITE:
            gpio_write(led_gpio[0], 1); // 红色打开
            gpio_write(led_gpio[1], 1); // 绿色打开
            gpio_write(led_gpio[2], 1); // 蓝色打开
            break;
        default:
            printf("Unknown color value\n");
            break;
    }
}

```

8.2 main.cpp

```

#include <stdio.h>
#include <stdlib.h>
#include <cmath>
#include <string>
#include <log.h>
#include <errno.h>
#include "clk.h"
#include "board.h"
#include "periph_conf.h"
#include "timex.h"
#include "ztimer.h"
#include "periph/gpio.h"
#include "thread.h"
#include "msg.h"
#include "shell.h"
#include "xtimer.h"
#include "ledcontroller.hh"
#include "mpu6050.h"

#define THREAD_STACKSIZE          (THREAD_STACKSIZE_IDLE)
static char stack_for_led_thread[THREAD_STACKSIZE];
static char stack_for_imu_thread[THREAD_STACKSIZE];

static kernel_pid_t _led_pid;
#define LED_MSG_TYPE_ISR         (0x3456)
#define LED_GPIO_R GPIO26
#define LED_GPIO_G GPIO25
#define LED_GPIO_B GPIO27
struct MPU6050Data
{
    float ax, ay, az;
    float gx, gy, gz;
};

enum MoveState{Stationary, Tilted, Rotating, Moving, Shaking, Moving_X, Moving_Y, None};

void delay_ms(uint32_t sleep_ms)
{

```

```

    ztimer_sleep(ZTIMER_USEC, sleep_ms * US_PER_MS);
    return;
}
void *_led_thread(void *arg)
{
    (void) arg;
    LEDController led(LED_GPIO_R, LED_GPIO_G, LED_GPIO_B);
    led.change_led_color(0);
    while(1){
        msg_t msg;
        msg_receive(&msg);

        if (msg.content.value == Stationary) {
            led.change_led_color(COLOR_NONE);
        } else if (msg.content.value == Moving_X) {
            led.change_led_color(COLOR_CYAN);
        } else if (msg.content.value == Moving_Y) {
            led.change_led_color(COLOR_WHITE);
        } else if (msg.content.value == Tilted) {
            led.change_led_color(COLOR_RED);
        } else if (msg.content.value == Rotating) {
            led.change_led_color(COLOR_BLUE);
        } else if (msg.content.value == Moving) {
            led.change_led_color(COLOR_GREEN);
        } else if (msg.content.value == Shaking) {
            led.change_led_color(COLOR_YELLOW);
        } else {
            led.change_led_color(COLOR_NONE);
        }
    }
    return NULL;
}

// 加速度和陀螺仪阈值
#define ACC_THRESHOLD 1.5 // 加速度静止阈值
#define ROT_THRESHOLD 50 // 角速度静止阈值
#define g_acc 9.81         // 重力加速度

MoveState detectMovement(MPU6050Data &data)
{
    // 计算加速度在X轴、Y轴和Z轴上的大小
    float acceleration_x = fabs(data.ax);
    float acceleration_y = fabs(data/ay);
    //float acceleration_z = fabs(data.az); // 新增Z轴加速度

    // 检测设备的总加速度
    float total_acceleration = sqrt(data.ax * data.ax + data.ay * data.ay +
data.az * data.az);
    // 计算陀螺仪总角速度
    float total_rotation = sqrt(data.gx * data.gx + data.gy * data.gy + data.gz *
data.gz);

    // 判断水平静止: Z轴加速度接近g, X轴和Y轴加速度接近0, 且没有旋转
}

```

```

        if (fabs(total_acceleration - g_acc) < ACC_THRESHOLD*3 && acceleration_x <
ACC_THRESHOLD*3 && acceleration_y < ACC_THRESHOLD*3 && total_rotation <
ROT_THRESHOLD/2) {
            return Stationary;
        }
        // 判断倾斜静止：设备的加速度在X、Y、Z轴上较为稳定，且没有旋转
        else if (fabs(total_acceleration - g_acc) < ACC_THRESHOLD*3 && total_rotation < ROT_THRESHOLD/2) {
            return Tilted;
        }
        // 判断X轴平移：X轴加速度大于阈值，且没有旋转
        else if (acceleration_x > ACC_THRESHOLD && acceleration_y < ACC_THRESHOLD &&
total_rotation < ROT_THRESHOLD) {
            return Moving_X;
        }
        // 判断Y轴平移：Y轴加速度大于阈值，且没有旋转
        else if (acceleration_y > ACC_THRESHOLD && acceleration_x < ACC_THRESHOLD &&
total_rotation < ROT_THRESHOLD) {
            return Moving_Y;
        }
        // 判断旋转：总角速度大于阈值
        else if (total_rotation > ROT_THRESHOLD) {
            return Rotating;
        }
        // 判断移动状态：总加速度明显偏离重力加速度
        else if (fabs(total_acceleration - g_acc) > ACC_THRESHOLD) {
            return Moving;
        }
        else {
            return Moving;
        }
    }

void *_imu_thread(void *arg)
{
    (void) arg;
    MPU6050 mpu;
    mpu.initialize();

    // 配置陀螺仪和加速度计全量程范围
    uint8_t gyro_fs = mpu.getFullScaleGyroRange();
    uint8_t accel_fs_g = mpu.getFullScaleAccelRange();

    // 默认转换因子
    float gyro_fs_convert = 1.0;
    float accel_fs_real = 1.0;

    // 根据陀螺仪量程范围设置转换因子
    switch (gyro_fs) {
        case MPU6050_GYRO_FS_250:
            gyro_fs_convert = 131.0;
            break;
        case MPU6050_GYRO_FS_500:
            gyro_fs_convert = 65.5;
            break;
        case MPU6050_GYRO_FS_1000:

```

```

        gyro_fs_convert = 32.8;
        break;
    case MPU6050_GYRO_FS_2000:
        gyro_fs_convert = 16.4;
        break;
    default:
        printf("[IMU_THREAD] Unknown GYRO_FS: 0x%x\n", gyro_fs);
}

// 根据加速度计量程范围设置转换因子
switch (accel_fs_g) {
    case MPU6050_ACCEL_FS_2:
        accel_fs_real = g_acc * 2;
        break;
    case MPU6050_ACCEL_FS_4:
        accel_fs_real = g_acc * 4;
        break;
    case MPU6050_ACCEL_FS_8:
        accel_fs_real = g_acc * 8;
        break;
    case MPU6050_ACCEL_FS_16:
        accel_fs_real = g_acc * 16;
        break;
    default:
        printf("[IMU_THREAD] Unknown ACCEL_FS: 0x%x\n", accel_fs_g);
}

// 计算加速度计转换因子
float accel_fs_convert = 32768.0 / accel_fs_real;

// 初始化原始传感器数据的变量
int16_t ax, ay, az, gx, gy, gz;

// 延迟一秒以确保传感器稳定
delay_ms(1000);

// 主循环: 周期性读取传感器数据并进行输出
while (1) {
    // 读取传感器数据
    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    // 将原始数据转换为实际值
    MPU6050Data data;
    data.ax = ax / accel_fs_convert;
    data/ay = ay / accel_fs_convert;
    data.az = az / accel_fs_convert;
    data.gx = gx / gyro_fs_convert;
    data.gy = gy / gyro_fs_convert;
    data.gz = gz / gyro_fs_convert;

    // 输出传感器数据
    printf("-----\n");
    printf("[IMU_THREAD] Accelerometer (X,Y,Z): (%.02f, %.02f, %.02f)\n",
m/s^2\n", data.ax, data.ay, data.az);
    printf("[IMU_THREAD] Gyroscope (X,Y,Z): (%.02f, %.02f, %.02f) °/s\n",
data.gx, data.gy, data.gz);
}

```

```
// 检测运动状态
MoveState state = detectMovement(data);

// 向LED线程发送消息通知当前的运动状态
msg_t msg;
msg.content.value = (int)state;
msg_send(&msg, _led_pid); // 发送消息到LED控制线程

// 200ms的延时
delay_ms(200);
}

return NULL;
}

static const shell_command_t shell_commands[] = {
{ NULL, NULL, NULL }
};

int main(void)
{
    _led_pid = thread_create(stack_for_led_thread, sizeof(stack_for_led_thread),
THREAD_PRIORITY_MAIN - 2,
                        THREAD_CREATE_STACKTEST, _led_thread, NULL,
                        "led_controller_thread");
    if (_led_pid <= KERNEL_PID_UNDEF) {
        printf("[MAIN] Creation of receiver thread failed\n");
        return 1;
    }
    thread_create(stack_for_imu_thread, sizeof(stack_for_imu_thread),
THREAD_PRIORITY_MAIN - 1,
                        THREAD_CREATE_STACKTEST, _imu_thread, NULL,
                        "imu_read_thread");
    printf("[Main] Initialization successful - starting the shell now\n");
    while(1)
    {
        ztimer_sleep(ZTIMER_USEC, 1000000); // 1秒延时
    }
    return 0;
}
```

