2025

南京宇微电子科技有限公司

MPU6050_6 轴模块 V1.0

2025 年 9 月 15 日修订

FU KUN

目录

版本修订

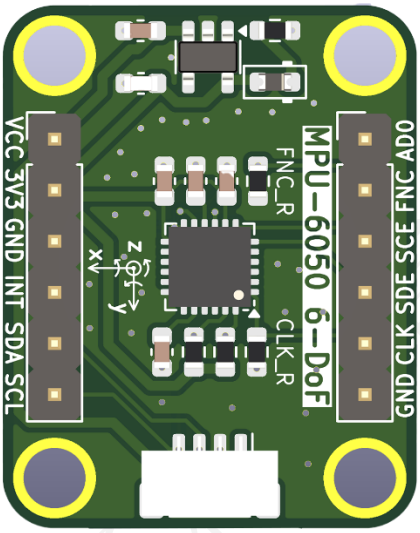
时间	版本号	修订内容
2025 年 9 月 15 日	V1.0	初版

客户须知

本文档为产品使用参考所编写，文档版本可能随时更新，恕不另行通知。本文中提供的所有使用方法、说明及建议仅供参考，不构成任何承诺或保证。使用本产品及本文档内容所产生的结果，由用户自行承担风险。本公司对因使用本文档或产品而导致的任何直接或间接损失，不承担任何责任。

一、概览

这款模块搭载了一颗 MPU6050 六轴传感器芯片，是一款由 InvenSense 公司开发的高性能六轴运动传感器，集成了三轴加速度计和三轴陀螺仪，广泛应用于运动追踪、姿态检测和导航系统。它采用 MEMS 技术，支持 I2C 通信接口，最高速率达 400kHz，具有高精度和低功耗特点。MPU6050 可测量 ±2g 至 ±16g 的加速度和 ±250°/s 至 ±2000°/s 的角速度，内置 16 位 ADC 提供高分辨率数据。此外，它还包含一个数字运动处理器（DMP），可进行复杂运动算法处理，减轻主控计算负担。支持宽电压供电，输入 3.3~6.5V，并支持一路 3.3V 稳压输出。

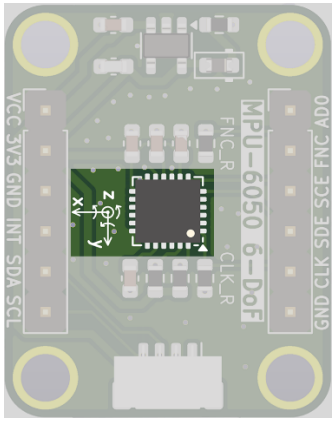


模块参数	
主芯片	MPU6050
性能	加速度： ±2g 至 ±16g 角速度：±250°/s 至 ±2000°/s
供电	输入 VCC：3.3~6.5V 输出 3V3：3.3V
接口	IIC
尺寸	约 20mm×26mm×1.6mm（具体以实物为准）
重量	约 g
配件	• 板卡×1 • 用户手册×1（电子版）

二、板卡分区介绍

2.1 MPU6050 主芯片

数据手册：MPU6050



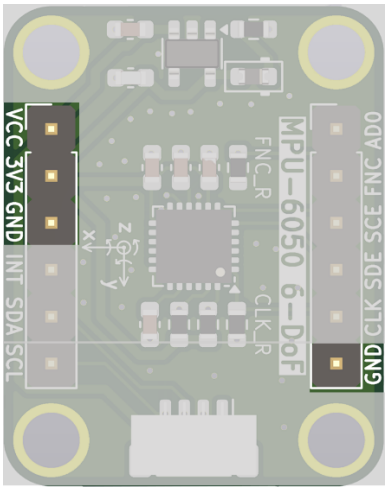
MPU6050 芯片输出加速度和角速度数据三轴分量，加速度的正方向为坐标轴指示方向，绕某一轴旋转角速度的正方向为旋转箭头绕行方向。

三轴加速度：沿 X、Y、Z 三个轴的加速度值，单位通常为 g（重力加速度，1g≈9.8 m/s²）。原始数据以 16 位有符号整数形式输出，具体值取决于 MPU6050 配置的量程（±2g、±4g、±8g、±16g）。例如，若量程为 ±2g，输出值为 -32768 到 +32767，对应 -2g 到 +2g。

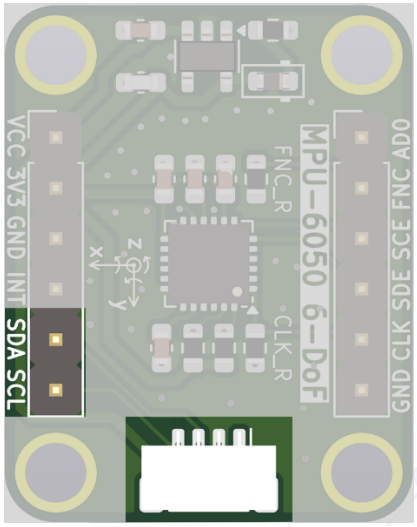
陀螺仪数据：沿 X、Y、Z 三个轴的角速度，单位通常为 °/s（度每秒）。同样以 16 位有符号整数形式输出，量程可选（±250°/s、±500°/s、±1000°/s、±2000°/s）。例如，若量程为 ±250°/s，输出值为 -32768 到 +32767，对应 -250°/s 到 +250°/s。

2.2 电源供电

VCC 外接 3.3~6.5V 供电，经过 LDO 稳压为 3.3V 给 MPU6050 供电，并且可通过 3V3 接口输出 3.3V 给其它模块使用。或者通过 3V3 接口输入 3.3V 电压，给 MPU6050 供电，VCC 悬空。



2.3 IIC 主通信接口

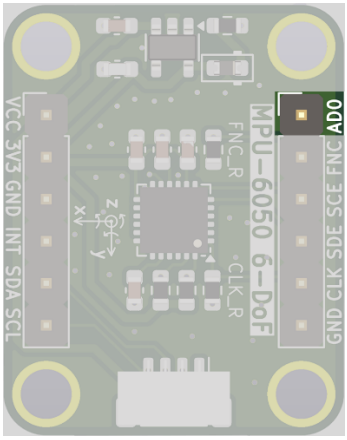


MPU6050 的 I2C 接口支持标准/快速模式（100/400 kHz），从地址 0x68/0x69，支持突发读取和中断配置，具备辅助 I2C 接口扩展功能，适合低功耗、多设备场景，通信需注意上拉电阻和时序配置。

2.4 通信地址变更

MPU6050 的 I2C 从设备地址由 AD0 引脚的电平决定：

- 默认地址：当 AD0 引脚接低电平（GND）时，I2C 地址为 0x68（7 位地址格式）。
- 变更地址：将 AD0 引脚接高电平（3.3V）时，地址变为 0x69。



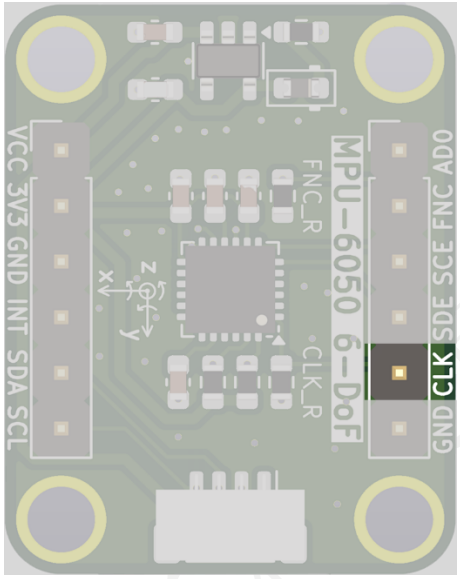
2.5 外部时钟输入

MPU6050 默认使用内部 8 MHz 时钟，其 CLKIN 引脚支持 30 kHz 至 60 MHz 的外部时钟单端

输入，电压需与 VDD 兼容（2.375V 至 3.46V），通过 PWR_MGMT_1 寄存器的 CLKSEL 位配置。通过 I2C 接口配置 PWR_MGMT_1 寄存器（地址 0x6B）的 CLKSEL 位（位 2:0）选择时钟源：

- CLKSEL = 000：内部 8 MHz 振荡器（默认）。
- CLKSEL = 100：外部时钟输入（通过 CLKIN 引脚）。

其他值可能用于 PLL 或其他模式，具体参考数据手册。

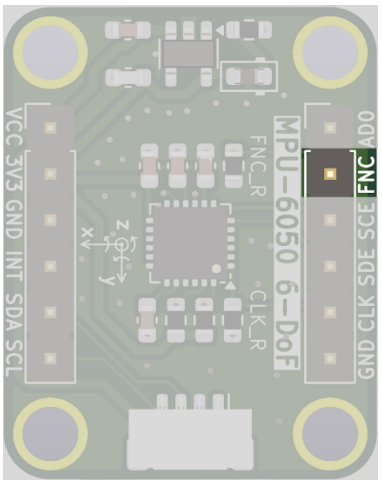


2.6 外部同步信号输入

FSYNC 引脚允许输入外部同步信号，以触发 MPU6050 的传感器数据采样（加速度计和陀螺仪），实现与外部系统（如摄像头或其他传感器）的时间同步。用于需要精确时间对齐的场景，如多传感器融合、运动捕获或与外部设备协同工作。FSYNC 信号的上升沿或下降沿（通过寄存器配置）触发传感器数据采样。每次触发后，MPU6050 将当前加速度计和陀螺仪数据存储到寄存器或 FIFO。

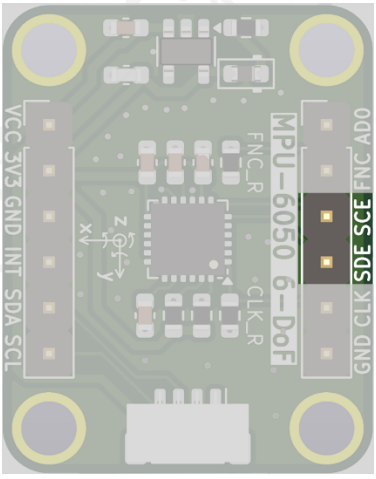
通过 I2C 接口配置 CONFIG 寄存器（地址 0x1A）的 EXT_SYNC_SET 位（位 5:3）启用外部同步并选择触发模式：

- 000：禁用 FSYNC（默认）。
- 001–111：启用 FSYNC，支持不同输入信号类型（如 TEMP_OUT_L[0]、GYRO_XOUT_L[0] 等）或触发边沿，具体参考数据手册。



2.7 辅助 IIC 通信接口

MPU6050 的辅助 I2C 接口（Auxiliary I2C Interface）用于连接外部传感器（如磁力计），允许 MPU6050 作为主设备通过 I2C 总线读取外部从设备的数据。辅助 I2C 接口通过 AUX_DA（辅助数据线，引脚 23）和 AUX_CL（辅助时钟线，引脚 24）连接外部 I2C 从设备（如磁力计 HMC5883L），实现扩展功能（如 9 轴运动跟踪）。



辅助 I2C 接口通过 MPU6050 的寄存器配置，主要涉及以下寄存器：

I2C_MST_EN（寄存器 0x6A，USER_CTRL 位 5）：设置为 1，启用辅助 I2C 主控功能。

I2C_MST_CLK（寄存器 0x6A，I2C_MST_CTRL 位 3:0）：配置辅助 I2C 的时钟频率（基于内部时钟，分频后支持 348 kHz、400 kHz 等）。常见配置：0x0D（400 kHz）。

SLV0_ADDR（寄存器 0x25）：设置外部从设备的 I2C 地址（7 位地址，最高位为读/写标志）。

示例：若外部设备地址为 0x1E（如 HMC5883L），写入 0x1E（读模式需置位最高位）。

SLV0_REG（寄存器 0x26）：指定外部从设备的寄存器地址（要读取或写入的数据寄存器）。

SLV0_CTRL（寄存器 0x27）：

配置从设备操作：

- 位 7：启用/禁用从设备 0（1 为启用）。
- 位 6：读/写选择（1 为读，0 为写）。
- 位 3:0：数据长度（1 到 255 字节）。

I2C_SLV0_DO（寄存器 0x63）：用于写入外部从设备的数据（写操作时）。

EXT_SENS_DATA_XX（寄存器 0x49 至 0x60）：存储从外部从设备读取的数据，供主设备通过主 I2C 接口访问。

I2C_MST_DELAY_CTRL（寄存器 0x67）：配置从设备数据读取的延迟，优化同步。

三、应用

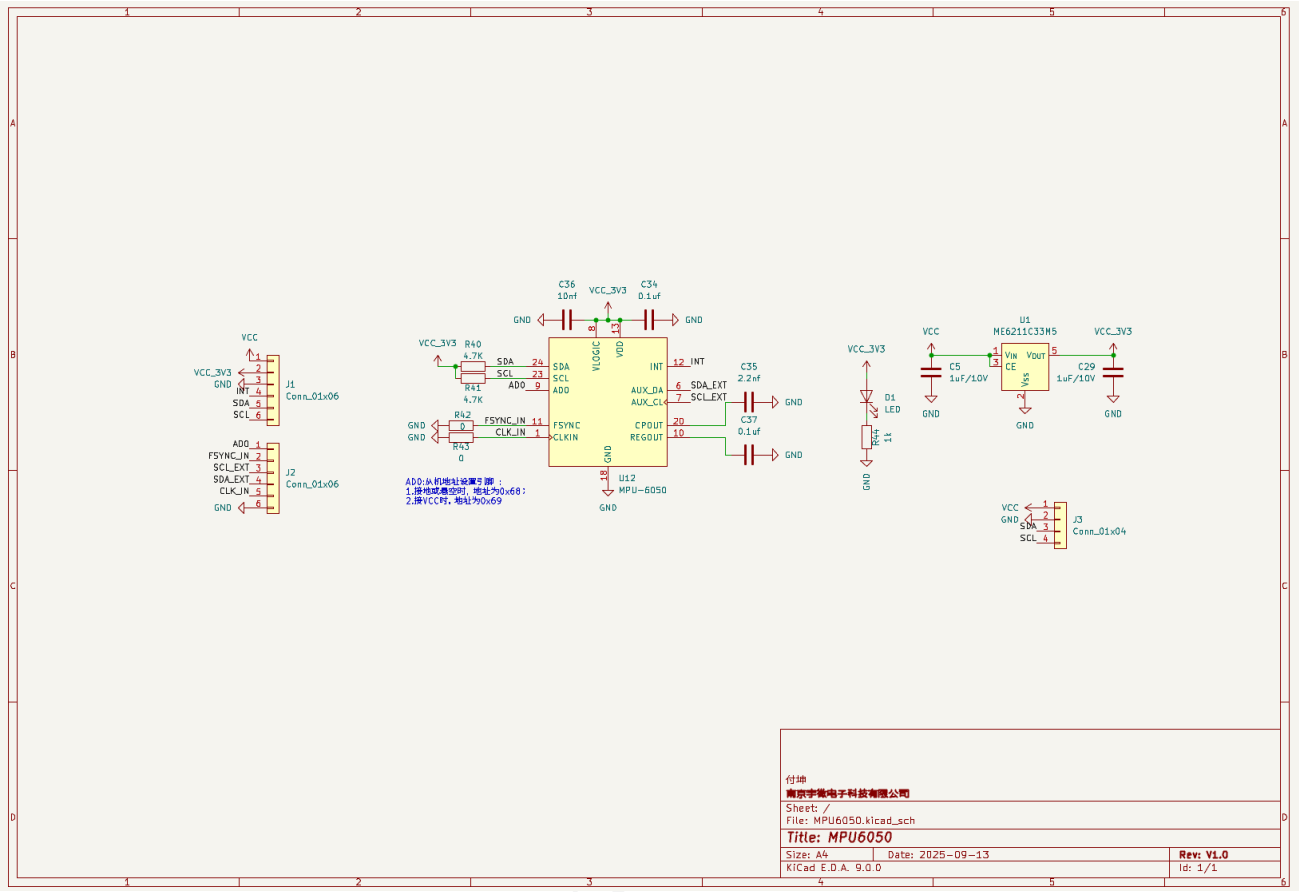
3.1 移动目标姿态检测

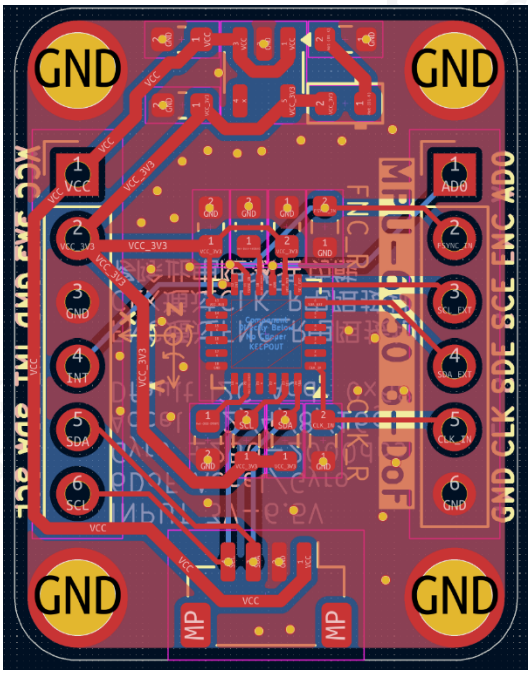
MPU6050 在移动目标姿态检测中应用广泛，其三轴加速度计和三轴陀螺仪可实时捕捉目标的加速度和角速度，结合算法（如卡尔曼滤波或 DMP）计算姿态角（俯仰、滚转、偏航），适用于无人机、机器人、车辆和可穿戴设备。以下是其在移动目标姿态检测中的具体应用。

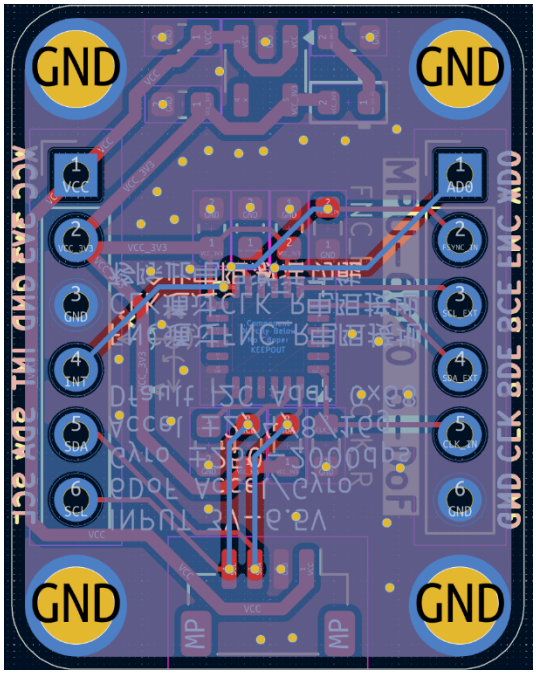
3.2 目标震动监测

MPU6050 在目标振动监测中应用广泛，其三轴加速度计可精准捕捉目标的振动信号，适用于工业设备、建筑结构和消费电子等领域。通过 I2C 接口，MPU6050 实时输出加速度数据，结合 STM32 或 Arduino 平台，可分析振动频率、幅度和方向，用于设备健康监测或故障预警。例如，在工业电机中，MPU6050 检测异常振动，数据经 USB VCP 传输至 PC 进行分析，判断轴承磨损或失衡。低功耗设计适合长期监测，±2g 至 ±16g 的量程适应多种场景。配合滤波算法，可降低噪声，提高精度。局限性在于高频振动可能受限于采样率（最高 1 kHz）。通过辅助 I2C 接口，还可扩展磁力计，实现更复杂环境监测。

四、原理图与 PCB 布线


付坤
南京宇微电子科技有限公司
Sheet: /
File: MPU6050.kicad_sch
Title: MPU6050
Size: A4
Date: 2025-09-13
Rev: V1.0
KiCad E.D.A. 9.0.0
Id: 1/1
MPU-6050
U12
J1
Conn_01x06
J2
Conn_01x06
J3
Conn_01x04
D1
LED
U1
ME6211C33M5
C5
1uF/10V
C29
1uF/10V
C34
0.1uf
C35
2.2nf
C36
10nf
C37
0.1uf
R40
4.7K
R41
4.7K
R42
0
R43
0
R44
1k
AD0:从机地址设置引脚：
1.接地或悬空时，地址为0x68；
2.接VCC时，地址为0x69


GND
GND
GND
GND
MP
MP
MPU-6050 6-DoF
VCC
GND
INT
SDA
SCL
ADO
FSYNC
CLK_IN


GND
GND
GND
GND
MP
MP
MPU-6050 6-DoF
VCC
GND
INT
SDA
SCL
ADO
FSYNC
CLK_IN

五、编程指南

5.1 姿态解算的数学原理

5.1.1 MPU6050 数据输出

这款芯片可以输出三轴旋转角度增量、三轴加速度增量和温度共 7 个物理量，每个物理量 2 个字节，共 14 个字节的数据。

5.1.2 旋转的数学原理

我们定义三轴旋转角度增量分别为，绕着 x 轴旋转，从 x 增长的方向逆向看过去为逆时针方向为正方向的旋转角速度为ω_α；绕着 y 轴旋转，从 y 增长的方向逆向看过去为逆时针方向为正方向的旋转角速度增量为ω_β；绕着 z 轴旋转，从 z 增长的方向逆向看过去为逆时针方向为正方向的旋转角度增量为ω_γ。

我们研究物体转动的规律，即研究与物体绑定的移动空间坐标系相对于全局坐标系的转动规律。基于空间直角坐标系下的物体转动，为了方便研究，定义一个转动是按照先绕 x 轴、再绕 y 轴、最后绕 z 轴旋转的顺序完成转动。根据三角函数之间的关系，按照 x、y、z 轴依次转动的旋转矩阵为

绕 x 轴正向转动后的新坐标与转动前坐标的关系

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ ycos\alpha - zsin\alpha \\ zcos\alpha + ysin\alpha \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

绕 y 轴正向转动后的新坐标与转动前坐标的关系

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = \begin{bmatrix} x'cos\beta + z'sin\beta \\ y' \\ z'cos\beta - x'sin\beta \end{bmatrix} = \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

绕 z 轴正向转动后的新坐标与转动前坐标的关系

$$\begin{bmatrix} x''' \\ y''' \\ z''' \end{bmatrix} = \begin{bmatrix} x''cos\gamma - sin\gamma \\ y''cos\gamma + x''sin\gamma \\ z'' \end{bmatrix} = \begin{bmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix}$$

每一次转动的参考系均为前一次转动结束的坐标系，而转动结束的新坐标系又将成为下一次转动的参考坐标系，因此我们定义一个起始的做坐标系为全局坐标系，其中的任意点或者向量均可以用包含(x, y, z)的参量表示，包括空间直角坐标系的三条法向量。

$$\begin{cases} \hat{x} = (1,0,0) \\ \hat{y} = (0,1,0) \\ \hat{z} = (0,0,1) \end{cases}$$

如果我们将绕着三条坐标轴的旋转矩阵分别用T_x、T_y和T_z表示，则一次完整的旋转可表示为

$$\begin{bmatrix} x''' \\ y''' \\ z''' \end{bmatrix} = T_z T_y T_x \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

由于 MPU6050 是按照一定的时间间隔δt采样输出，因此一次旋转的角度增量为$\omega_i \delta t$，累加更新三轴法向量，最终得到得旋转的总角度。

在 MPU6050 芯片静止或做匀速直线运动时，加速度计输出的加速度值为重力加速度 g，合方向竖直向下。当 MPU6050 斜放置时，加速度计输出的三个方向的值可能不为零，因为竖直向下的重力加速度 g 按照一定的姿态分布到三个移动坐标轴上。

我们以三轴加速度建立空间直角坐标系，假设转动后的三轴的加速度输出分别为(a_x, a_y, a_z)，令转动前的三轴加速度输出分别是$(0,0,g)$，因此

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = T_z T_y T_x \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

将三个变换矩阵带入上式易得

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} -sin\beta \\ cos\beta sin\alpha \\ cos\beta cos\alpha \end{bmatrix}$$

MPU 在空间旋转的姿态角不仅可以由陀螺仪经过离散求和获得，也可以通过加速度计或者 x/y 轴的旋转角度，最终输出的姿态角度建议使用加权平均处理输出。注意，这种求解方式仅在合外力为 0 时有效，且可以用加速度计计算的姿态角校准陀螺仪计算的欧拉角。

5.1.3 平移的数学原理

我们定义三轴加速度增量分别为，沿着芯片 x 轴方向的加速度增量 δg_x，沿着芯片 y 轴方向的加速度增量 δg_z，沿着芯片 z 轴方向的加速度增量 δg_z。MPU6050 输出的 3 轴加速度可以用来计算速度和位移，速度的计算公式如下

$$\overrightarrow{v_{n+1}} = \overrightarrow{v_n} + \overrightarrow{a_n}\delta t$$

这里的a_n是最新输出的加速度值，v_n是上一次计算的在a_n输出时刻的速度值，v_{n+1}是计算在a_n输出开始经过δt时间后的速度。间隔δt的两个速度在时域上互相迭代，我们定义在初始 0 时刻的边界条件为

$$a_0 = 0m/s^2$$

$$v_0 = 0m/s$$

因为a_n是沿坐标轴三个分量的形式输出，所以

$$v_{(n+1)x} = v_{nx} + a_{nx}\delta t$$

$$v_{(n+1)y} = v_{ny} + a_{ny}\delta t$$

$$v_{(n+1)z} = v_{nz} + a_{nz}\delta t$$

假设，在 MPU6050 芯片从静止启动后接收到第一组加速度数据为$t = 0s$时刻，且各向初速度分量均为为 0，则

$$v_{2x} = a_{1x}\delta t$$

$$v_{2y} = a_{1y}\delta t$$

$$v_{2z} = a_{1z}\delta t$$

这样一直迭代下去，就可以计算出任意离散时刻沿着运动坐标系坐标轴的速度分量。

位移的表达式如下

$$\overrightarrow{x_{n+1}} = \overrightarrow{v_n}\delta t + \frac{1}{2}\overrightarrow{a_n}\delta t^2$$

这里的x_{n+1}是本次加速度输出之后经过δt时刻的位移，按照运动坐标系展开

$$x_{(n+1)x}\hat{x} = v_{nx}\hat{x}\delta t + \frac{1}{2}a_{nx}\hat{x}\delta t^2$$

$$x_{(n+1)y}\hat{y} = v_{ny}\hat{y}\delta t + \frac{1}{2}a_{ny}\hat{y}\delta t^2$$

$$x_{(n+1)z}\hat{z} = v_{nz}\hat{z}\delta t + \frac{1}{2}a_{nz}\hat{z}\delta t^2$$

将速度的计算结果和加速度带入上式，可得移动坐标系整体的位移分量模值，方向由坐标轴的单位矢量经过上一节的平移和旋转获得。

以上是理想情况，事实上在地球上有重力加速度的干扰，加速度计有重力加速度的干扰，在静止时总在竖直向下的方向有个加速度$\vec{g}[0,0,1]$，而随着姿态的变化，重力加速度分量将遍布三个轴，并与受外界力产生的加速度$\overrightarrow{g_f}$融合在一起，必须要结合转动角度消去这个干扰。所以，MPU6050 输出的加速度、重力加速度分量和有运动的变化产生的加速度有如下关系。

$$\overrightarrow{g_{outputx}} = \vec{g} + \overrightarrow{g_f}$$

按照运动坐标系展开

$$g_{outputx}\hat{x} = g_x\hat{x} + g_{fx}\hat{x}$$

$$g_{outputy}\hat{y} = g_y\hat{y} + g_{fy}\hat{y}$$

$$g_{outputz}\hat{z} = g_z\hat{z} + g_{fz}\hat{z}$$

根据陀螺仪计算的角速度，给出当前体坐标与全球坐标的角度，然后根据三角函数可以求解分布在体坐标系各轴上的重力加速度，进而求得体坐标系各轴的外力加速度，最后带入速度和位移的计算公式，进行迭代计算。

5.1.4 欧拉角的万向锁问题和四元数

欧拉角（Euler Angle）可以表示 3D 空间中任何方向的 3 个值，由莱昂哈德·欧拉（Leonhard Euler）在 18 世纪提出。一共有 3 种欧拉角：俯仰角（Pitch）、偏航角（Yaw）和滚转角（Roll）。俯仰角是表示往上或往下看的角，偏航角表示往左和往右看的角，滚转角是表示翻滚的角。为了便于描述，我们假定的绕轴转动为

- 俯仰角（Pitch）=绕 x 轴旋转
- 偏航角（Yaw）=绕 y 轴旋转
- 滚转角（Roll）=绕 z 轴旋转

欧拉角旋转的计算需规定一个顺序，比如先计算绕 x 的旋转分量，再计算绕 y 的旋转分量，最后计算绕 z 的旋转分量，最终将三个旋转矩阵组合成这次旋转矩阵。这里有个常见的理解误区：将计算过程映射到目标的实际运动，欧拉角计算最终的结果是告诉我们。我们设想一种情况，规定 xyz 为旋转矩阵的计算顺序，在某次旋转中，其 y 矩阵分量的旋转角度为 90°，将 5.1.2 节旋转矩阵分量带入得

$$\begin{bmatrix} x''' \\ y''' \\ z''' \end{bmatrix} = T_{z(\gamma)} T_{y\left(\beta=\frac{\pi}{2}\right)} T_{x(\alpha)} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -sin(\gamma-\alpha) & cos\big((\gamma-\alpha)\big) \\ 0 & cos\big((\gamma-\alpha)\big) & sin\big((\gamma-\alpha)\big) \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$= \begin{bmatrix} -ysin(\gamma-\alpha) + zcos(\gamma-\alpha) \\ zsin(\gamma-\alpha) + ycos(\gamma-\alpha) \\ -x \end{bmatrix}$$

总旋转矩阵仅依赖于$(\gamma-\alpha)$（一个组合角度），而不是 γ和α分别独立。原本三个独立参数 α,β,γ 现在退化为两个有效参数，一个是固定的 $\beta=\frac{\pi}{2}$，另一个是$(\gamma-\alpha)$。这意味着，$\left(\gamma=\frac{\pi}{3},\ \alpha=0\right)$与$\left(\gamma=\frac{2\pi}{3},\ \alpha=\frac{\pi}{3}\right)$产生相同的旋转矩阵，也就是说当中间旋转角度$\beta=\frac{\pi}{2}$时，调整滚转$\alpha$的效果等同于反向调整偏航$\gamma$，系统根据旋转矩阵调整飞行器姿态时，无法正确输出控制信号，这是固有的“万向锁定”（Gimbal Lock）问题。

上文中提到的用欧拉角计算，有万向节死锁问题，为了解决这个问题，我们介绍基四元数的姿态表征。四元数（Quaternion）是一种数学工具，用于表示三维空间中的旋转，广泛应用于姿态估计、计算机图形学和机器人领域，尤其能避免欧拉角的“万向锁定”（Gimbal Lock）问题。四元数可表示为

$$q_{(w,x,y,z)} = w + xi + yj + zk$$

其中，w是标量部分（实部），一个实数。$xi+yj+zk$是向量部分（虚部），表示三维向量。ijk是虚单位，满足$i^2=j^2=k^2=ijk=-1$。标量部分w是四元数中不含虚单位的实数项。其 4 个分量具有单位约束$|q|=1$，所有仅有 3 个自由度，可用于表示绕一个单位向量的 3 维旋转角度θ，假设这个单位向量为$\vec{n}=(n_x,n_y,n_z)$，则

$$q=cos\left(\frac{\theta}{2}\right)+sin\left(\frac{\theta}{2}\right)(n_x i+n_y j+n_z k)$$

可以确保只要$\vec{n}$为单位向量，则$|q|$约束为 1

从欧拉角（Z-Y-X 顺序）计算四元数时，使用半角公式

$$w=\cos\left(\frac{\theta_z}{2}\right)\cos\left(\frac{\theta_y}{2}\right)\cos\left(\frac{\theta_x}{2}\right)+\sin\left(\frac{\theta_z}{2}\right)\sin\left(\frac{\theta_y}{2}\right)\sin\left(\frac{\theta_x}{2}\right)$$

$$x=\sin\left(\frac{\theta_z}{2}\right)\cos\left(\frac{\theta_y}{2}\right)\cos\left(\frac{\theta_x}{2}\right)-\cos\left(\frac{\theta_z}{2}\right)\sin\left(\frac{\theta_y}{2}\right)\sin\left(\frac{\theta_x}{2}\right)$$

$$y=\cos\left(\frac{\theta_z}{2}\right)\sin\left(\frac{\theta_y}{2}\right)\cos\left(\frac{\theta_x}{2}\right)+\sin\left(\frac{\theta_z}{2}\right)\cos\left(\frac{\theta_y}{2}\right)\sin\left(\frac{\theta_x}{2}\right)$$

$$z=\cos\left(\frac{\theta_z}{2}\right)\cos\left(\frac{\theta_y}{2}\right)\sin\left(\frac{\theta_x}{2}\right)-\sin\left(\frac{\theta_z}{2}\right)\sin\left(\frac{\theta_y}{2}\right)\cos\left(\frac{\theta_x}{2}\right)$$

这些公式基于三角恒等式，数学上保证$w^2+x^2+y^2+z^2=1$。MPU6050 输出的角速度w_k是离散的，因此需要进行累加

$$q_{k+1}=q_k+\frac{1}{2}q_k\otimes w_k\Delta t$$

在实际计算中（如数值积分或传感器数据融合），由于浮点误差或外部输入，四元数可能偏离单位模值。为了保证$|q|=1$，程序运行一段时间后需要进行归一化：

$$|q|=\sqrt{w^2+x^2+y^2+z^2}$$

$$q_{norm}=\frac{q}{|q|}=\frac{(w,x,y,z)}{\sqrt{w^2+x^2+y^2+z^2}}$$

四元数（Z-Y-X 顺序）转欧拉角的公式为

$$\alpha=atan2\big(wx+yz,1-2(x^2+y^2)\big)$$
$$\beta=arcsin\big(2(wy-zx)\big)$$
$$\gamma=atan2\big(2(wz+xy),1-2(y^2+z^2)\big)$$

5.1.5 Madgwick 滤波器

经典滤波算法，如高通、低通、带通等，可以滤除掺杂在信号中的杂波，互补滤波时卡尔曼滤波的简化版本，用来滤除白噪声等近似服从高斯分布的随机误差噪声，本质上是一种最优化思想。

陀螺仪和加速计在空间姿态的计算结果上互有重叠，采取何种加权平均方式能将误差控制在最小呢？举个例子，使用陀螺仪计算的值抽象成T_1，误差为ϵ_1，加速度计计算的值抽象为T_2，误差为ϵ_2，其中$\epsilon_1 \sim N(0,\sigma_1^2)$，$\epsilon_2 \sim N(0,\sigma_2^2)$，$E[\epsilon_1\epsilon_2]=0$，则

$$T_1 = T + \epsilon_1$$

$$T_2 = T + \epsilon_2$$

所以

$$T_{est2} = \alpha T_1 + (1-\alpha)T_2$$

方差为

$$\begin{aligned} &D[T_{est2}] \\ &= D[\alpha T_1 + (1-\alpha)T_2] \\ &= \alpha^2 D[T_1] + (1-\alpha)^2 D[T_2] \\ &= \alpha^2 {\epsilon_1}^2 + (1-\alpha)^2 {\epsilon_2}^2 \end{aligned}$$

方差的最小值

$$\begin{aligned} &min\{D[T_{est2}]\} \\ &= \alpha^2 {\epsilon_1}^2 + (1-\alpha)^2 {\epsilon_1}^2, \alpha|\frac{{\epsilon_2}^2}{{\epsilon_1}^2 + {\epsilon_2}^2} \\ &= \frac{{\epsilon_1}^2 {\epsilon_2}^2}{{\epsilon_1}^2 + {\epsilon_2}^2} \end{aligned}$$

也就是说使用加权平均的方法，能找到一个最优解，使得误差最小。

以角度更新为例，加速度计计算出的绝对角度为θ_{accelk}，陀螺仪计算出的角度增量为$\theta_{degk} \times \delta t$，累积的绝对角度为$\theta_{k-1} + \theta_{degk} \times \delta t$，采用合适的$\alpha$计算出最优的绝对角度$\theta_k$。

$$\theta_k = \alpha\theta_{accelk} + (1-\alpha)\left(\theta_{k-1} + \theta_{degk} \times \delta t\right)$$

以上是一种最优化思想，下面我们简要介绍 Madgwick 滤波器在 MPU6050 数据优化中的应用。

我们在读取 MPU6050 实时输出的一组角加速度数据$[w_x, w_y, w_z]$和加速度数据$[a_x, a_y, a_z]$，使用当前的四元数将世界坐标重力加速度值[0,0,0,1]转化成体坐标$q \otimes [0,0,0,1] \otimes q^*$，计算过程中忽略标量部分，

因为误差只关心向量，结果减去$[0, a_x, a_y, a_z]$，即为误差。

$$f = q \otimes [0,0,0,1] \otimes q^* - [0, a_x, a_y, a_z] = \begin{bmatrix} 2(wy - xy) - a_x \\ 2(xy + wy) - a_y \\ 1 - 2(x^2 + y^2) - a_z \end{bmatrix}$$

其中，加速度数据转化为四元数，标量部分$w = 0$，q^*是四元数的共轭$[w, -x, -y, -z]$，$\otimes$是四元数的乘法。我们现在的目标是最小化这个误差矩阵。

梯度下降是一种优化方法，通过沿目标函数的负梯度方向迭代更新参数，逐步逼近最小值。寻找误差函数矩阵f的最小值，我们希望找到使误差最小的四元数q。梯度下降更新规则为：

$$q_{k+1} = q_k + \left(\frac{1}{2} q_k \otimes w_{q,k} - \beta \frac{\nabla_q |f|^2}{|\nabla_q |f|^2|} \right) \Delta t$$

其中，μ是步长（学习率），在 Madgwick 滤波器中为$\beta \Delta t$，其中β是校正增益，$\frac{\nabla_q |f|^2}{|\nabla_q |f|^2|}$控制修正方向，$\Delta t$是采样间隔。$\nabla_q |f|^2$是目标函数(误差函数$f$的范数)对 q 的梯度。误差函数$|f|^2 = {f_x}^2 + {f_y}^2 + {f_z}^2$，其梯度

$$\nabla_q |f|^2 = 2f \frac{\partial f}{\partial q}$$

计算误差函数f对四元数的雅可比矩阵$\frac{\partial f}{\partial q}$：

$$J = \frac{\partial f}{\partial q} = \begin{bmatrix} \frac{\partial f_x}{\partial w} & \frac{\partial f_x}{\partial x} & \frac{\partial f_x}{\partial y} & \frac{\partial f_x}{\partial z} \\ \frac{\partial f_y}{\partial w} & \frac{\partial f_y}{\partial x} & \frac{\partial f_y}{\partial y} & \frac{\partial f_y}{\partial z} \\ \frac{\partial f_z}{\partial w} & \frac{\partial f_z}{\partial x} & \frac{\partial f_z}{\partial y} & \frac{\partial f_z}{\partial z} \end{bmatrix}$$

对每个分量求偏导，得

$$J = \begin{bmatrix} 2y & -2z & 2w & -2x \\ 2z & 2y & 2x & 2w \\ 0 & -4x & -4y & 0 \end{bmatrix}$$

梯度为

$$\nabla_q |f|^2 = 2J^T f = 2 \begin{bmatrix} 2yf_x + 2zf_y \\ -2zf_x + 2yf_y - 4xf_z \\ 2wf_x + 2xf_y - 4yf_z \\ -2xf_x + 2wf_y \end{bmatrix}$$

所以基于加速度计梯度下降的微调四元数如下式所示，

对加速度计和陀螺仪数据进行融合，并归一化

$$q_{k+1} = q_k + \left(\frac{1}{2} q_k \otimes w_{q,k} - \beta \frac{\nabla_q |f|^2}{|\nabla_q |f|^2|}\right) \Delta t$$

$$q_{k+1} = \frac{q_{k+1}}{|q_{k+1}|}$$

5.2 基于 MPU6050 的运动检测 USB 串口输出程序

以下是为 STM32F103C8T6 微控制器设计的基于 HAL 库的 MPU6050 输出三轴加速度计数据、三轴陀螺仪数据和温度数据的程序。

5.2.1 结构体和寄存器定义

```
typedef struct {
    float Accel_X;
    float Accel_Y;
    float Accel_Z;
} MPU6050_ACCEL_t;

typedef struct {
    float Gyro_X;
    float Gyro_Y;
    float Gyro_Z;
} MPU6050_GYRO_t;

/* Private define ------------------------------------------------------------*/
#define MPU6050_ADDR          0xD0    // MPU6050 I2C 地址 (0x68 << 1, AD0 接地)
#define REG_SMPLRT_DIV        0x19    // 采样率分频器
#define REG_GYRO_CONFIG       0x1B    // 陀螺仪配置
#define REG_ACCEL_CONFIG      0x1C    // 加速度计配置
#define REG_ACCEL_XOUT_H      0x3B    // 加速度 X 高字节
#define REG_GYRO_XOUT_H       0x43    // 陀螺仪 X 高字节
#define REG_PWR_MGMT_1        0x6B    // 电源管理 1
#define REG_WHO_AM_I          0x75    // 设备 ID 寄存器

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
void MPU6050_Init(void);
void MPU6050_Read_All(MPU6050_ACCEL_t *Mpu_Accel, MPU6050_GYRO_t *Mpu_Gyro, float *Temp);
```

此代码片段是为 STM32F103 微控制器使用 HAL 库通过 I2C 接口驱动 MPU6050 传感器（加速度计、陀螺仪、温度传感器）的部分代码，适用于读取加速度、角速度和温度数据。

结构体定义：

- MPU6050_ACCEL_t：存储加速度计数据（单位：g），包含 X、Y、Z 三个轴的浮点数值。
- MPU6050_GYRO_t：存储陀螺仪数据（单位：°/s），包含 X、Y、Z 三个轴的浮点数值。

5.2.2 MPU6050 初始化函数

```
void MPU6050_Init(void)
{
    uint8_t Data;
    char buffer[64];

    // 检查设备 ID
    uint8_t Check;
    HAL_StatusTypeDef status = HAL_I2C_Mem_Read(&hi2c1, MPU6050_ADDR, REG_WHO_AM_I,
I2C_MEMADD_SIZE_8BIT, &Check, 1, 1000);
    if (status == HAL_OK && Check == 0x68)
    {
        snprintf(buffer, sizeof(buffer), "MPU6050 ID: 0x%02X\r\n", Check);
        CDC_Transmit_FS((uint8_t*)buffer, strlen(buffer));
    }
    else
    {
        snprintf(buffer, sizeof(buffer), "MPU6050 not found! Status: %d, ID: 0x%02X\r\n", status, Check);
        CDC_Transmit_FS((uint8_t*)buffer, strlen(buffer));
        while (1); // 停止程序
    }

    // 复位设备
    Data = 0x80;
    HAL_I2C_Mem_Write(&hi2c1,          MPU6050_ADDR,          REG_PWR_MGMT_1,
I2C_MEMADD_SIZE_8BIT, &Data, 1, 1000);
    HAL_Delay(100);

    // 唤醒设备，设置时钟源为 PLL
    Data = 0x01; // PLL with X axis gyro reference
    HAL_I2C_Mem_Write(&hi2c1,          MPU6050_ADDR,          REG_PWR_MGMT_1,
I2C_MEMADD_SIZE_8BIT, &Data, 1, 1000);

    // 设置采样率 1KHz (8kHz / (7+1))
    Data = 0x07;
    HAL_I2C_Mem_Write(&hi2c1,          MPU6050_ADDR,          REG_SMPLRT_DIV,
I2C_MEMADD_SIZE_8BIT, &Data, 1, 1000);
```

```
    // 加速度计 ±2g
    Data = 0x00;
    HAL_I2C_Mem_Write(&hi2c1, MPU6050_ADDR, REG_ACCEL_CONFIG,
I2C_MEMADD_SIZE_8BIT, &Data, 1, 1000);

    // 陀螺仪 ±250°/s
    Data = 0x00;
    HAL_I2C_Mem_Write(&hi2c1, MPU6050_ADDR, REG_GYRO_CONFIG,
I2C_MEMADD_SIZE_8BIT, &Data, 1, 1000);
}
```

本函数是 MPU6050 的初始化函数，先通过 IIC 查询地址，若地址正确则开始对 MPU6050 初始化。MPU6050 需要初始化的关键项目有

- 复位：向电源管理寄存器 1（REG_PWR_MGMT_1, 0x6B）写入 0x80，复位 MPU6050 的所有寄存器。
- 唤醒设备，设置时钟源为 PLL：向 REG_PWR_MGMT_1（0x6B）写入 0x01，选择 PLL（以 X 轴陀螺仪为参考）作为时钟源，并唤醒传感器（退出睡眠模式）。
- 设置采样率：向采样率分频寄存器（REG_SMPLRT_DIV, 0x19）写入 0x07（0~255），配置采样率为 1kHz（8kHz / (7+1)）。开启内置数字滤波，则内部时钟频率由 8k 降到 1k。
- 加速度灵敏度：向加速度计配置寄存器（REG_ACCEL_CONFIG, 0x1C）写入 0x00，设置量程为 ±2g（灵敏度 16384 LSB/g），量程（REG_ACCEL_CONFIG 的 AFS_SEL 位），加速度（g）= 原始值（LSB） / 灵敏度（LSB/g）：
    - ±2g（AFS_SEL = 0b00）：灵敏度 16384 LSB/g。
    - ±4g（AFS_SEL = 0b01）：灵敏度 8192 LSB/g。
    - ±8g（AFS_SEL = 0b10）：灵敏度 4096 LSB/g。
    - ±16g（AFS_SEL = 0b11）：灵敏度 2048 LSB/g。
- 陀螺仪灵敏度：向陀螺仪配置寄存器（REG_GYRO_CONFIG, 0x1B）写入 0x00，设置量程为 ±250°/s（灵敏度 131 LSB/°/s）。灵敏度由量程决定，量程越大，灵敏度越低（单位物理量的 LSB 值越小）。量程（REG_GYRO_CONFIG 的 FS_SEL 位），角速度（°/s） = 原始值（LSB） / 灵敏度（LSB/°/s）：
    - ±250°/s（FS_SEL = 0b00）：灵敏度 131 LSB/°/s。
    - ±500°/s（FS_SEL = 0b01）：灵敏度 65.5 LSB/°/s。

- ±1000°/s（FS_SEL = 0b10）：灵敏度 32.8 LSB/°/s。
- ±2000°/s（FS_SEL = 0b11）：灵敏度 16.4 LSB/°/s。

5.2.3 MPU6050 读取数据函数

```
void MPU6050_Read_All(MPU6050_ACCEL_t *Mpu_Accel, MPU6050_GYRO_t *Mpu_Gyro, float *Temp)
{
    uint8_t Read_Data[14];
    char buffer[64];
    HAL_StatusTypeDef status = HAL_I2C_Mem_Read(&hi2c1, MPU6050_ADDR,
REG_ACCEL_XOUT_H,
                                                I2C_MEMADD_SIZE_8BIT, Read_Data, 14, 1000);
    if (status == HAL_OK)
    {
        // 加速度计数据 (±2g, 16384 LSB/g)
        Mpu_Accel->Accel_X = (int16_t)(Read_Data[0] << 8 | Read_Data[1]) / 16384.0f;
        Mpu_Accel->Accel_Y = (int16_t)(Read_Data[2] << 8 | Read_Data[3]) / 16384.0f;
        Mpu_Accel->Accel_Z = (int16_t)(Read_Data[4] << 8 | Read_Data[5]) / 16384.0f;

        // 温度数据
        *Temp = ((int16_t)(Read_Data[6] << 8 | Read_Data[7]) / 340.0f) + 36.53f;

        // 陀螺仪数据 (±250°/s, 131 LSB/°/s)
        Mpu_Gyro->Gyro_X = (int16_t)(Read_Data[8] << 8 | Read_Data[9]) / 131.0f;
        Mpu_Gyro->Gyro_Y = (int16_t)(Read_Data[10] << 8 | Read_Data[11]) / 131.0f;
        Mpu_Gyro->Gyro_Z = (int16_t)(Read_Data[12] << 8 | Read_Data[13]) / 131.0f;
    }
    else
    {
        snprintf(buffer, sizeof(buffer), "I2C Error: %d\r\n", status);
        CDC_Transmit_FS((uint8_t*)buffer, strlen(buffer));
        Mpu_Accel->Accel_X = Mpu_Accel->Accel_Y = Mpu_Accel->Accel_Z = 0.0f;
        Mpu_Gyro->Gyro_X = Mpu_Gyro->Gyro_Y = Mpu_Gyro->Gyro_Z = 0.0f;
        *Temp = 0.0f;
    }
}
```

这里我们仅展示 while 循环的内容，设定一个速度“50”前向转动占空比。

5.2.4 main 主函数

```
int main(void)
{
```

```
    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();
    MX_CRC_Init();
    MX_I2C1_Init();
    MX_USB_DEVICE_Init();
    HAL_Delay(1000); // 等待 USB 枚举

    MPU6050_Init();

    MPU6050_ACCEL_t Accel;
    MPU6050_GYRO_t Gyro;
    float Temp;

    while (1)
    {
        MPU6050_Read_All(&Accel, &Gyro, &Temp);
        char buffer[128];
        snprintf(buffer, sizeof(buffer),
                 "Accel: X=%.2f, Y=%.2f, Z=%.2f | Temp=%.2f | Gyro: X=%.2f, Y=%.2f, Z=%.2f\r\n",
                 Accel.Accel_X, Accel.Accel_Y, Accel.Accel_Z, Temp,
                 Gyro.Gyro_X, Gyro.Gyro_Y, Gyro.Gyro_Z);
        while (CDC_Transmit_FS((uint8_t*)buffer, strlen(buffer)) == USBD_BUSY) {
            HAL_Delay(1);
        }
        HAL_Delay(500);
    }
}
```

5.2 自主编写姿态解算程序及官方 DMP 库的使用方法

将下一版更新，敬请期待！

六、联系我们

若需任何帮助，请邮件联系我们：info@fukunlab.com

样品购买：淘宝店铺-宇微电子