**作者**：HANDS FREE TEAM

**交流 QQ 群**：521037187

# HANDS FREE 理念

当你热爱某种事物，你可能会想办法去弥补它身上的缺点，即使意味着会牺牲你一点点，我们只是一群呆在大学里幼稚青年，但我们也有着对机器人事业的向往。

Hands Free，顾名思义解放双手。我们想做到的是能够搭建一个共享的平台，一个友好的易于共同开发的框架。Hands Free 从单片机平台开始，逐步地扩展到了相应的其他周边，为的是让整个”机器人”的开发过程降低耦合，尽可能地减少一些底层的开发环节，在开发过程中提供了一个更好的交流方式。Hands Free 其理念的核心是优化开发过程的同时，让设计的 idea 的分享过程更加 Free，是乐于分享的，鼓励分享的。

Hands Free Team 是一个奉献于机器人事业的团队，我们希望国内能有一个优秀的开源项目,我们不是生产加工一体化的公司，也没有公司的乱七八糟的规则，我们也不想花时间去纠结于成本，利润的关系，我们只想奉献我们优秀的设计，优秀的代码。

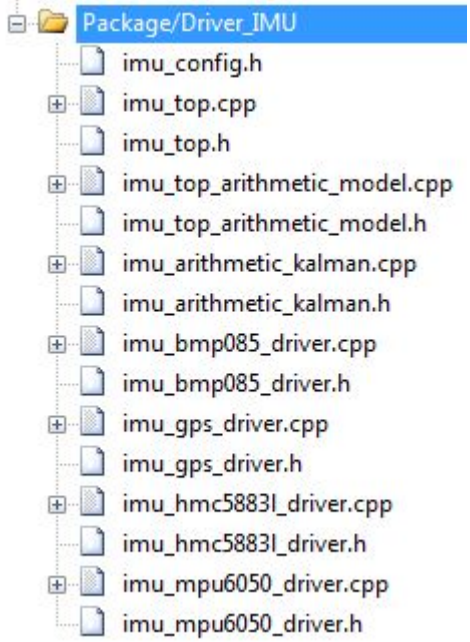
HANDS FREE 理念总结成一句话：创造一个机会共同成长。

如果你觉得“哎呦不错”的话，就一起加入进来吧!!!

# HANDS FREE IMU 算法包手册

## Package Name Driver_IMU

## Test Project Test_Package_Driver_IMU




## Intro

测试平台： HANDS_FREE_ControlUnit_V1

BSP： i2c.c/.h 模拟 I2C：HF_Simulation_I2C1

玩过四轴，写过 IMU 的朋友可能类似的经历，IMU 算法对四轴飞行器来说至关重要，如果不加任何融合算法的条件下，仅仅靠陀螺仪积分来进行姿态解算的话，它的效果可以暂时满足四轴飞行器的飞行的，但是时间久了就会发现机身倾斜，所以 IMU 算法包的核心工作是抑制积分漂移的同时，又要满足一定的动态特性。

为了将加速度计加入进来抑制积分漂移，首先是要对加速度计的值进行滤波。STM32F1 在计算能力上有所欠缺，为了满足控制频率的要求，因此现在大多采用向量叉值误差修正陀螺仪角速度的方法。在 STM32F4 的平台上我们考虑了更大运算量的一阶扩展卡尔曼的算法。这里更多的是提供一个学习和了解的方式，相对于 PIXHAWK 等成熟飞控的多阶扩展卡尔曼滤波，效果应该是较差的，这个程序还没有在四轴飞行器上有过验证，但是通过简单测试，应该能够满足四轴飞行器的基本飞行需要。写算法的工作者能力有限，大多地方还是使用工程实践的方法，没有较为合理的理论理解，如有错误，请多包涵。

By Chenyingbing Mawenke

# 1 Function Description

## >>> imu_config.h

编译配置文件

```
[常用配置]
#define SIMULAT_I2C_ID_IMU      1               模拟 I2C 接口号
#define SYSTEM_SUPPORT_HMC_DTASHOW     0u       是否使能打印 HMC
                                                数据用于参数修正
#define HMC_Correct_X           -0.11f          HMC 矫正值 X
#define HMC_Correct_Y           0.08f           HMC 矫正值 Y
#define HMC_Correct_Z           -0.48f          HMC 矫正值 Z
#define K_Amplify_AccCov        300.0f          ACC 协方差增益
#define K_Amplify_GyroCov       0.1f            GYRO 协方差增益
```

## >>> imu_top.cpp/.h

用户函数接口文件

```
[常用函数]
void IMU_Top_Init(void);
                和 IMU 有关的模拟 IIC 接口   各传感器的初始化程序
void IMU_Top_Call(void);
                1ms 调用一次    不断更新 IMU 数据
```

## >>> imu_top_arithmetic_model.cpp/.h

算法函数接口文件

```
[用户接口变量]
变量类:       ARITHMETIC_MODEL
全局变量名:   imu_arithmetic_model

   >> unsigned char Fusion_En;
              使能融合算法标志位 1 有效 0 时仅进行陀螺角速度积分
   >> unsigned char Fusion_State;
                    融合状态查看，若为 0 则因设备失效算法失效
   >> unsigned char Imu_Top_fusion_HavingInitial;
                              是否已初始化角度，输出 1 时已初始化
   >> FUSION_IPType_PRY_DGREE Target_Angle;
                                        目标姿态角 由用户输入
   >> FUSION_IPType_PRY_DGREE  ControlVector_Position;
                              姿态角 位置环 误差控制向量 输出
```

(pitch,roll 在 0-90° 有效)

>> FUSION_IPType_PRY_DGREE ControlVector_Velocity;

姿态角 速度环 误差控制向量 即陀螺仪角速度 输出

>> FUSION_IPType_PRY_DGREE Initial_Angle_dgree;

初始的姿态角

>> IMU_QUATERNION Fus_Quaternion;

融合的四元数

>> FUSION_IPType_PRY_DGREE Fus_Angle;

融合的姿态角

>> FUSION_IPType_PRY_DGREE AccHmcMes_Angle_dgree;

测量的姿态角

### >>> imu_arithmetic_kalman.cpp/.h

KALMAN 算法文件

定义了一个 KALMAN 类，包含了卡尔曼滤波算法中的一些基本变量，如：卡尔曼协方差矩阵 Pk，卡尔曼增益 Kg 等。

### >>> imu_bmp085_driver.cpp/.h

BMP085 设备定义文件

定义了一个 bmp085 的基本驱动函数，变量接口等。本程序未使用，其功能未测试？

### >>> imu_gps_driver.cpp/.h

GPS 设备定义文件

定义了一个 gps 的基本驱动函数，变量接口等。本程序未使用，其功能未测试？

### >>> imu_hmc5883l_driver.cpp/.h

磁罗盘设备定义文件

定义了一个 hmc5883 的基本驱动函数，变量接口等。

### >>> imu_mpu6050_driver.cpp/.h

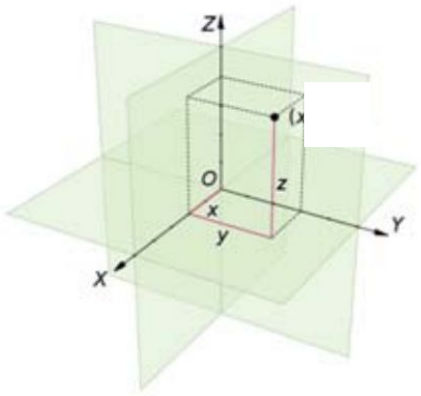
加速度/陀螺仪设备定义文件

定义了一个 mpu6050 的基本驱动函数，变量接口等。

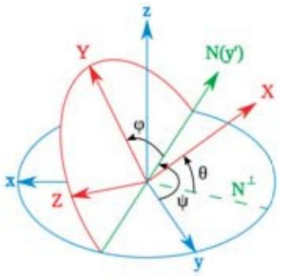
# 2 Arithmetic

## >>> 导航系规定

定义不同的 XYZ 轴会导致 PITCH,ROLL,YAW 的定义不同，正负不同。




定义$\psi$、$\theta$、$\varphi$分别为绕 Z 轴、Y 轴、X 轴的旋转角度，如果用 Tait-Bryan angle 表示，分别为 Yaw、Pitch、Roll。




## >>> 姿态转换矩阵

姿态转换矩阵：参考系 n 到载体系 b(载体系 b 到参考系 n 为其转置)

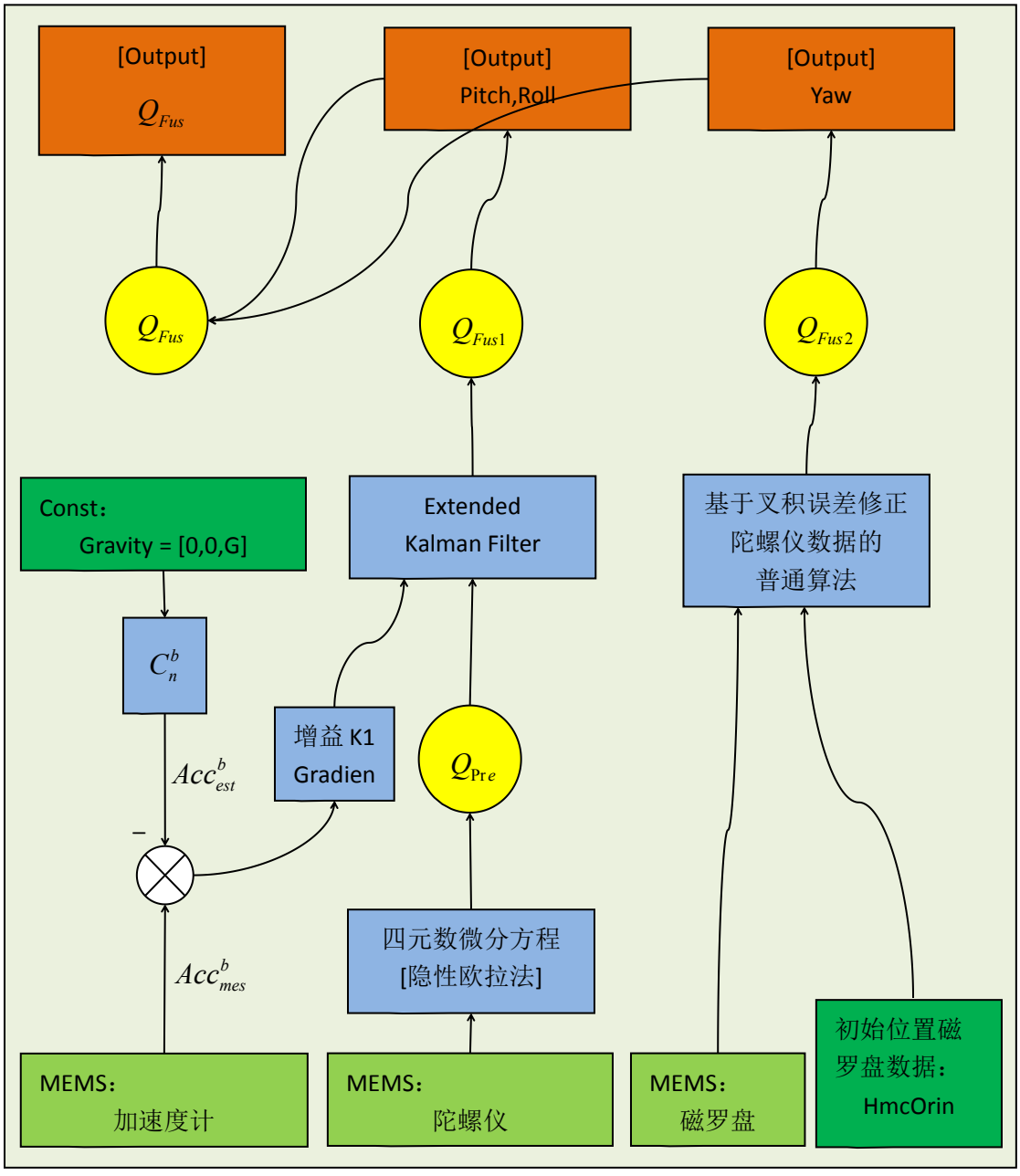
### >> 欧拉角表示

$$\begin{bmatrix} \cos\theta\cos\Psi & \cos\theta\sin\Psi & -\sin\theta \\ \sin\varphi\sin\theta\cos\Psi-\cos\varphi\sin\Psi & \sin\varphi\sin\theta\sin\Psi+\cos\varphi\cos\Psi & \sin\varphi\cos\theta \\ \cos\varphi\sin\theta\cos\Psi+\sin\Psi\sin\varphi & \cos\varphi\sin\theta\sin\Psi-\sin\varphi\cos\Psi & \cos\varphi\cos\theta \end{bmatrix}$$

## >> 四元数表示

$$\begin{matrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{matrix}$$

## >>> 算法流程图



## >>> 扩展卡尔曼滤波基本公式

详见维基百科：

https://en.wikipedia.org/wiki/Extended_Kalman_filter

基本公式：

$$\dot{x}(t) = f(x(t), u(t)) + w(t), w(t) \sim N(0, Q(t))$$
$$z(k) = h(x_k) + v_k, v_k \sim N(0, R_k)$$

>>

式中：

$$x_k = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \qquad z_k = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$$

>>

$$\dot{x}(t) = f(x(t), u(t)) + w(t), w(t) \sim N(0, Q(t))$$

$$\begin{bmatrix} \dot{q}_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_2 & -q_3 \\ q_0 & -q_1 & q_3 \\ q_0 & q_1 & -q_2 \end{bmatrix} \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} + w(t)$$

>>

$$z(k) = h(x_k) + v_k, v_k \sim N(0, R_k)$$

$$\begin{bmatrix} ax \\ ay \\ az \end{bmatrix}_{estimate} = C_n^b \begin{bmatrix} 0 \\ 0 \\ G \end{bmatrix} + v(t)$$

>>

$$\frac{\partial f}{\partial x}=\begin{bmatrix}1 & -g_x & -g_y & -g_z\\ g_x & 1 & g_y & -g_z\\ g_x & -g_y & 1 & g_z\\ g_x & g_y & -g_z & 1\end{bmatrix}$$

>>

$$\frac{\partial h}{\partial x}=\begin{bmatrix}-2q_2G & 2q_3G & -2q_0G & 2q_1G\\ 2q_1G & 2q_0G & 2q_3G & 2q_2G\\ 2q_0G & -2q_1G & -2q_2G & 2q_3G\end{bmatrix}$$