> github主页：https://github.com/snqx-lqh
>
> 本项目github地址：https://github.com/snqx-lqh/STM32F103C8T6HalDemo
>
> 欢迎交流

文献参考以及说明

移植MPU9250，参考下面的文章，以及正点原子的移植项目工程文件。但是移植到最后使用还是有部分问题，问题内容将在最后进行说明。

https://blog.csdn.net/weixin_45682654/article/details/136244101

https://blog.csdn.net/Rare_Hunter/article/details/134200468

使用的是HAL库，关于I2C初始化一类的处理，这里不做讲解，需要保证你有以下接口的IIC函数。且能正常使用

```
int mpu9250_write_bytes(uint8_t addr,uint8_t reg,uint8_t len,uint8_t *data);
int mpu9250_read_bytes(uint8_t addr,uint8_t reg,uint8_t len,uint8_t *data);
```

简单描述一下我这两个函数的封装逻辑。首先封装了一个封装的HAL库。

```
int mpu9250_write_bytes(uint8_t addr,uint8_t reg,uint8_t len,uint8_t *data)
{
    u_i2c1_write_bytes(addr, reg, data ,len);
    return 0;
}

int mpu9250_read_bytes(uint8_t addr,uint8_t reg,uint8_t len,uint8_t *data)
{
    u_i2c1_read_bytes(addr,reg,data,len);
    return 0;
}
```

然后每个函数内部的实现函数如下：

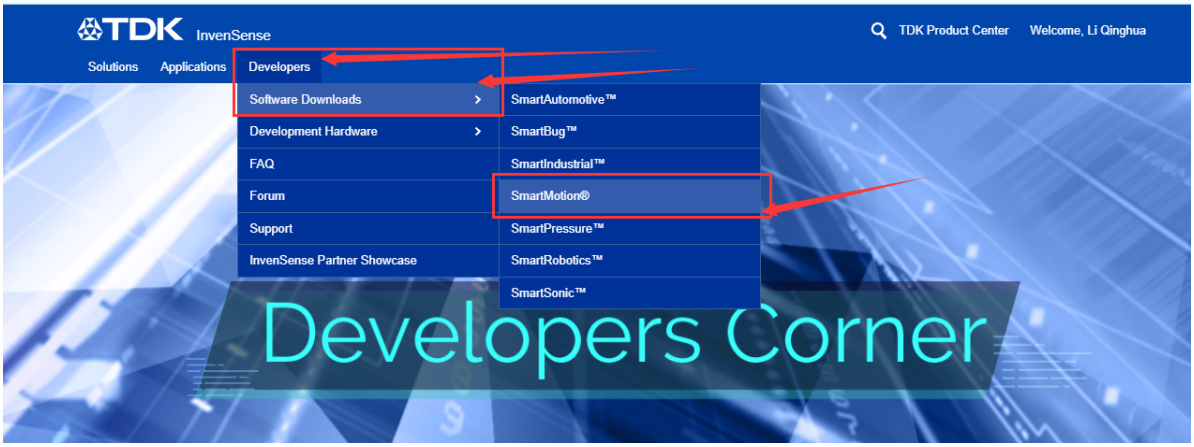
```
void u_i2c1_write_bytes(unsigned char add,unsigned char reg,unsigned char
*data,unsigned char len)
{
    HAL_I2C_Mem_Write(&hi2c1, (add<<1), reg, I2C_MEMADD_SIZE_8BIT,
data,len,HAL_MAX_DELAY);
}

void u_i2c1_read_bytes(unsigned char add,unsigned char reg,unsigned char
*data,unsigned char len)
{
    HAL_I2C_Mem_Read(&hi2c1, (add<<1), reg,I2C_MEMADD_SIZE_8BIT, data, len,
HAL_MAX_DELAY);
}
```

具体的实现方法可以去我的开源代码里面查看，开源代码中不仅包含DMP库的解算，还有使用卡尔曼滤波，互补滤波，Mahony姿态解算以及Madgwick的解算方式。

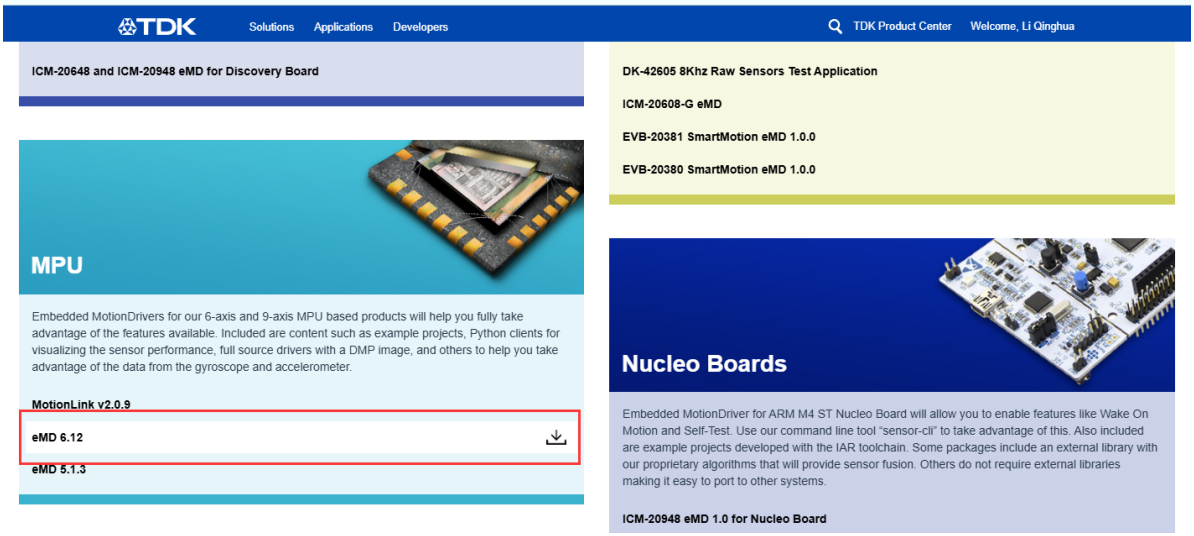
下载DMP库

进入官网：https://invensense.tdk.com/

先注册，注册完成后需要邮箱里面点击他发给你的链接进行一个验证。注册密码需要有特殊字符、大小写、数字。登录完成后点击下图。



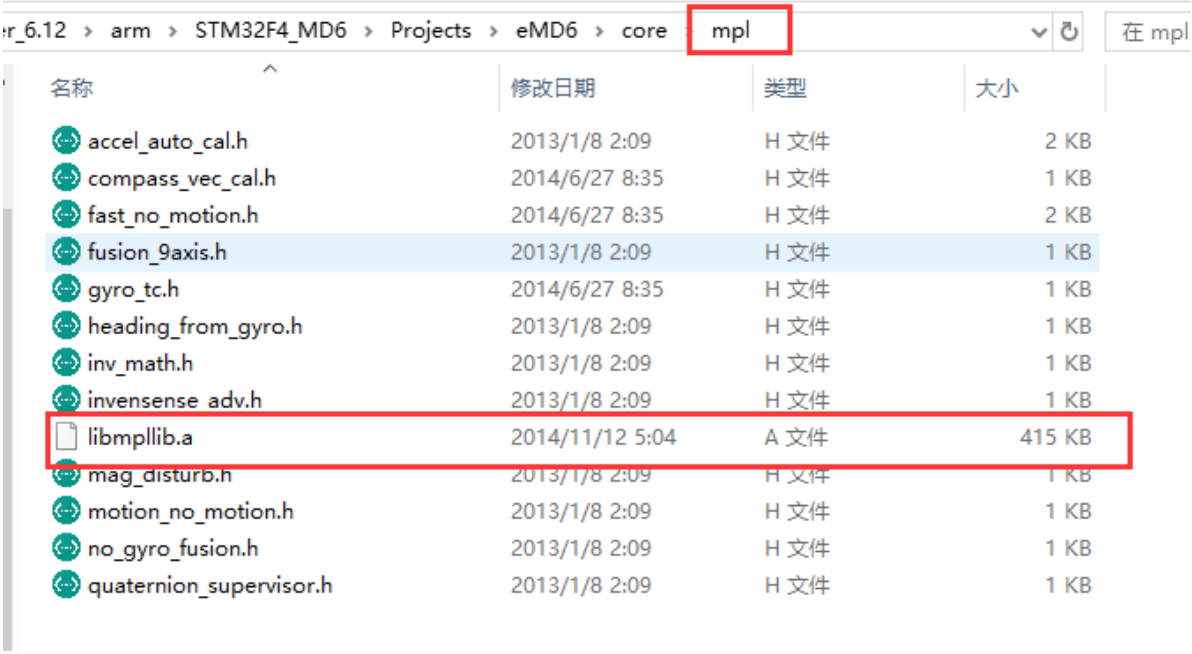
点进去后滑动到最底下，找到MPU栏目。点击压缩包下载。



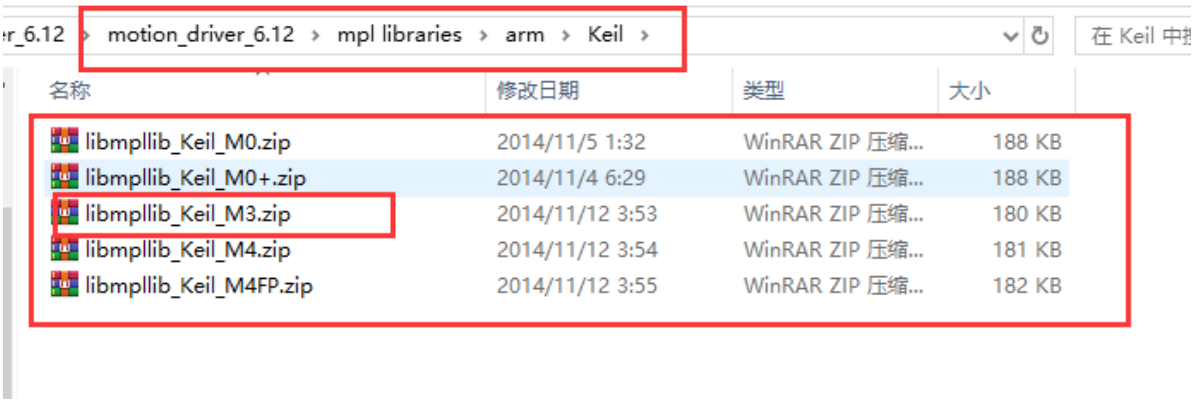
DMP移植

在下载的项目包中，找到和DMP解算相关的代码。将以下内容中全部文件都加入在项目中去，处理好的文件内容可以直接下载我的开源文件查看，以下步骤只是为了方便解释内容。

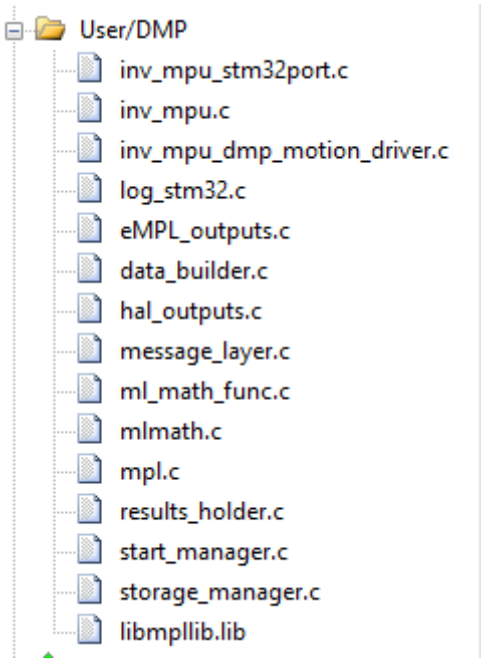
加载的时候，在`mpl`文件夹中有一个`libmpllib.a`文件，该文件需要用实际的库替换，



如图所示，找到对应的lib文件，去替换，我这里使用的是STM32F103，所以应该使用M3的库文件，解压后将其替换刚刚的`libmpllib.a`文件。



所有文件加载进去后，目录如下，其中`inv_mpu_stm32port.c`文件暂时不用管，该部分内容只是为了提供一个计算的接口，后面会做说明。



接下来就是代码处理，首先是`inv_mpu.c`，我会使用注释表示要添加的内容。在开头的部分，我们将I2C处理的头文件包含进来，我这里I2C处理部分封装在了`#include "mpu6050.h"`，`STM32_MPU6050`，是为了定义我们自己的操作函数。

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "inv_mpu.h"

/*********************用户处理部分 start**********************/
#include "mpu9250.h"
#include "usart.h"

#define STM32_MPU9250  //开启自定义的函数宏定义
#define MPU9250        //使用MPU9250相关的处理函数
/*********************用户处理部分 end  **********************/
```

然后紧接着添加I2C处理部分的代码。下面多余的部分只是为了方便定位说明，在40几行左右。这部分的处理，需要注意`i2c_write`和`i2c_read`的函数格式，格式内容在源文件注释里面有。

```
/* The following functions must be defined for this platform:
 * i2c_write(unsigned char slave_addr, unsigned char reg_addr,              ###
  格式
 *      unsigned char length, unsigned char const *data)
 * i2c_read(unsigned char slave_addr, unsigned char reg_addr,               ###
  格式
 *      unsigned char length, unsigned char *data)
 * delay_ms(unsigned long num_ms)
 * get_ms(unsigned long *count)
 * reg_int_cb(void (*cb)(void), unsigned char port, unsigned char pin)
 * labs(long x)
 * fabsf(float x)
 * min(int a, int b)
 */
/*********************用户处理部分 修改函数宏定义 start**********************/
#if defined STM32_MPU9250
#include "log.h"
unsigned char *mpl_key = (unsigned char*)"eMPL 5.1";
#define i2c_write   mpu9250_write_bytes
#define i2c_read    mpu9250_read_bytes
#define delay_ms    HAL_Delay
#define get_ms      mget_ms

#define log_i       printf
#define log_e       printf
/* labs is already defined by TI's toolchain. */
/* fabs is for doubles. fabsf is for floats. */
#define fabs        fabsf
#define min(a,b)    ((a<b)?a:b)

static inline int reg_int_cb(struct int_param_s *int_param)
{
    return NULL;
}
```

```
//空函数,未用到.
static void mget_ms(unsigned long *time)
{

}
/*********************用户处理部分 修改函数宏定义 end**********************/
#elif defined EMPL_TARGET_STM32F4
```

然后在`inv_mpu.h`中，给`int_param_s`添加一个指针变量。

```
struct int_param_s {
#if defined EMPL_TARGET_MSP430 || defined MOTION_DRIVER_TARGET_MSP430
    void (*cb)(void);
    unsigned short pin;
    unsigned char lp_exit;
    unsigned char active_low;
#elif defined EMPL_TARGET_UC3L0
    unsigned long pin;
    void (*cb)(volatile void*);
    void *arg;
#elif defined EMPL_TARGET_STM32F4
    void (*cb)(void);
/***********  添加   *************/
#else
    void (*cb)(void);
/*********************************/
#endif
};
```

同样在`inv_mpu_dmp_motion_driver.c`文件中也修改为自己的操作函数，并将一个`__no_operation()`函数注释。

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "inv_mpu.h"
#include "inv_mpu_dmp_motion_driver.h"
#include "dmpKey.h"
#include "dmpmap.h"

/*********************用户处理部分 start**********************/
#include "mpu9250.h"

#define STM32_MPU9250  //开启自定义的函数宏定义
#define MPU9250        //使用MPU9250相关的处理函数
/*********************用户处理部分 end  **********************/

/* The following functions must be defined for this platform:
 * i2c_write(unsigned char slave_addr, unsigned char reg_addr,
 *      unsigned char length, unsigned char const *data)
 * i2c_read(unsigned char slave_addr, unsigned char reg_addr,
 *      unsigned char length, unsigned char *data)
 * delay_ms(unsigned long num_ms)
```

```
 * get_ms(unsigned long *count)
 */
/*********************用户处理部分 修改函数宏定义 start**********************/
#if defined STM32_MPU9250

#define delay_ms       HAL_Delay
#define get_ms         mget_ms
#define log_i          printf
#define log_e          printf

static void mget_ms(unsigned long *time)
{

}
/*********************用户处理部分 修改函数宏定义 end**********************/
#elif defined EMPL_TARGET_STM32F4
```

```
641  int dmp_set_accel_bias(long *bias)
642  {
643      long accel_bias_body[3];
644      unsigned char regs[12];
645      long long accel_sf;
646      unsigned short accel_sens;
647
648      mpu_get_accel_sens(&accel_sens);
649      accel_sf = (long long)accel_sens << 15;
650      //__no_operation();
651
652      accel_bias_body[0] = bias[dmp.orient & 3];
653      if (dmp.orient & 4)
654          accel_bias_body[0] *= -1;
655      accel_bias_body[1] = bias[(dmp.orient >> 3) & 3];
656      if (dmp.orient & 0x20)
657          accel_bias_body[1] *= -1;
658      accel_bias_body[2] = bias[(dmp.orient >> 6) & 3];
659      if (dmp.orient & 0x100)
660          accel_bias_body[2] *= -1;
661
```

然后修改`log_stm32.c`文件，首先，在开头处，将他包含的`stm32f4xx.h`去除，并修改成包含了stm32f1处理库的`main.h`，如果你是标准库，就是其他的文件。

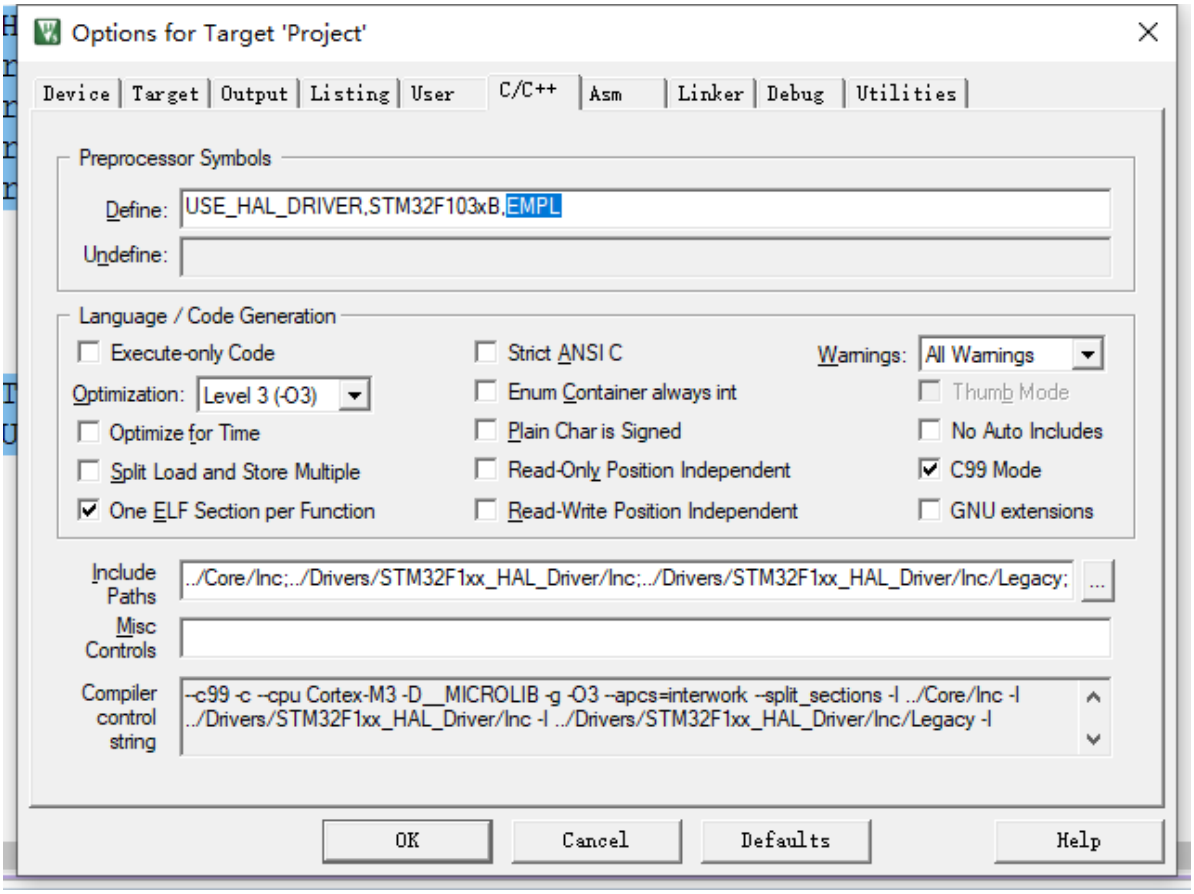
```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>

#include "packet.h"
#include "log.h"
+ #include "main.h"
+ #include "usart.h"
```

然后修改下面函数中的三个函数的fputs的代码。

```
int _MLPrintLog (int priority, const char* tag, const char* fmt, ...)
{
```

```
+    FILE * noUse;
     /****************/
     //  原代码部分  //
     /****************/
     for (ii = 0; ii < length; ii += (PACKET_LENGTH-5)) {
#define min(a,b) ((a < b) ? a : b)
         this_length = min(length-ii, PACKET_LENGTH-5);
         memset(out+3, 0, 18);
         memcpy(out+3, buf+ii, this_length);
         for (i=0; i<PACKET_LENGTH; i++) {
/*************** 修改处 ****************/
+          fputc(out[i],noUse);
/***************************************/
         }
     }
     va_end(args);
     return 0;
}

void eMPL_send_quat(long *quat)
{
+    FILE * noUse;
     /****************/
     //  原代码部分  //
     /****************/
     for (i=0; i<PACKET_LENGTH; i++) {
/*************** 修改处 ****************/
+       fputc(out[i],noUse);
/***************************************/
     }
}

void eMPL_send_data(unsigned char type, long *data)
{
+    FILE * noUse;
     /****************/
     //  原代码部分  //
     /****************/
     for (i=0; i<PACKET_LENGTH; i++) {
/*************** 修改处 ****************/
+       fputc(out[i],noUse);
/***************************************/
     }
}
```

接着需要在项目配置的宏定义处，加上`EMPL`。



DMP使用

然后新建一个方便其他文件使用的接口文件`inv_mpu_stm32port.c`。

```
#include "inv_mpu_stm32port.h"

#include <math.h>
#include "inv_mpu.h"
#include "mpl.h"
#include "invensense.h"
#include "invensense_adv.h"
#include "data_builder.h"
#include "eMPL_outputs.h"
#include "mltypes.h"
#include "mpu.h"
#include "log.h"
#include "packet.h"
#include "inv_mpu.h"
#include "inv_mpu_dmp_motion_driver.h"
#include "stdio.h"


#define DEFAULT_MPU_HZ  (100)
#define COMPASS_READ_MS (100)
#define Q30  1073741824.0f
#define Q16  65536.0f

/* The sensors can be mounted onto the board in any orientation. The mounting
 * matrix seen below tells the MPL how to rotate the raw data from thei
 * driver(s).
 * TODO: The following matrices refer to the configuration on an internal test
 * board at Invensense. If needed, please modify the matrices to match the
```

```
 * chip-to-body matrix for your particular set up.
 */
/* (使用Ai简易翻译了一下原注释)
 * 传感器可以以任何方向安装到板上。
 * 下面的安装矩阵告诉MPL如何从驱动程序旋转原始数据。
 * TODO: 下面的矩阵指的是Invensense内部测试板上的配置。
 * 如果需要，请修改矩阵以匹配您特定设置的芯片到本体矩阵。
*/
static signed char gyro_orientation[9] = {-1, 0, 0,
                                           0,-1, 0,
                                           0, 0, 1};
//磁力计方向设置
static signed char comp_orientation[9] = { 0, 1, 0,
                                           1, 0, 0,
                                           0, 0,-1};
/* These next two functions converts the orientation matrix (see
 * gyro_orientation) to a scalar representation for use by the DMP.
 * NOTE: These functions are borrowed from Invensense's MPL.
 */
/* (使用Ai简易翻译了一下原注释)
 * 以下这两个函数将方向矩阵（参见gyro_orientation）转换为标量表示，以供DMP使用。
 * 注释：这些函数是从Invensense的MPL借用的。
*/
static unsigned short inv_row_2_scale(const signed char *row)
{
    unsigned short b;

    if (row[0] > 0)
        b = 0;
    else if (row[0] < 0)
        b = 4;
    else if (row[1] > 0)
        b = 1;
    else if (row[1] < 0)
        b = 5;
    else if (row[2] > 0)
        b = 2;
    else if (row[2] < 0)
        b = 6;
    else
        b = 7;      // error
    return b;
}

static unsigned short inv_orientation_matrix_to_scalar(
    const signed char *mtx)
{
    unsigned short scalar;
    /*
       XYZ  010_001_000 Identity Matrix
       XZY  001_010_000
       YXZ  010_000_001
       YZX  000_010_001
       ZXY  001_000_010
       ZYX  000_001_010
     */
    scalar = inv_row_2_scale(mtx);
    scalar |= inv_row_2_scale(mtx + 3) << 3;
```

```
    scalar |= inv_row_2_scale(mtx + 6) << 6;
    return scalar;
}

/**
  * @brief   自检测试
  * @param
  * @retval  void
 **/
static int run_self_test(void)
{
    int result;
    long gyro[3], accel[3];

    result = mpu_run_self_test(gyro, accel);
    //result = mpu_run_6500_self_test(gyro, accel,0);
    if (result == 0x7) {
        /* Test passed. We can trust the gyro data here, so let's push it down
         * to the DMP.
         */
        float sens;
        unsigned short accel_sens;
        mpu_get_gyro_sens(&sens);
        gyro[0] = (long)(gyro[0] * sens);
        gyro[1] = (long)(gyro[1] * sens);
        gyro[2] = (long)(gyro[2] * sens);
        dmp_set_gyro_bias(gyro);
        mpu_get_accel_sens(&accel_sens);
        accel[0] *= accel_sens;
        accel[1] *= accel_sens;
        accel[2] *= accel_sens;
        dmp_set_accel_bias(accel);
    } else {
        return -1;
    }

    return 0;
}

/**
  * @brief   初始化MPU6050的DMP相关配置
  * @param
  * @retval  void
 **/
int mpu_dmp_init(void)
{
    uint8_t res=0;
    struct int_param_s int_param;
    unsigned char accel_fsr;
    unsigned short gyro_rate, gyro_fsr;
    unsigned short compass_fsr;

    if(mpu_init(&int_param)==0) //初始化MPU9250
    {
        res=inv_init_mpl();     //初始化MPL
        if(res)return 1;
        inv_enable_quaternion();
        inv_enable_9x_sensor_fusion();
```

```
        inv_enable_fast_nomot();
        inv_enable_gyro_tc();
        inv_enable_vector_compass_cal();
        inv_enable_magnetic_disturbance();
        inv_enable_eMPL_outputs();
        res=inv_start_mpl();    //开启MPL
        if(res)return 1;
        res=mpu_set_sensors(INV_XYZ_GYRO|INV_XYZ_ACCEL|INV_XYZ_COMPASS);//设置所需
要的传感器
        if(res)return 2;
        res=mpu_configure_fifo(INV_XYZ_GYRO | INV_XYZ_ACCEL);   //设置FIFO
        if(res)return 3;
        res=mpu_set_sample_rate(DEFAULT_MPU_HZ);                //设置采样率
        if(res)return 4;
        res=mpu_set_compass_sample_rate(1000/COMPASS_READ_MS);  //设置磁力计采样率
        if(res)return 5;
        mpu_get_sample_rate(&gyro_rate);
        mpu_get_gyro_fsr(&gyro_fsr);
        mpu_get_accel_fsr(&accel_fsr);
        mpu_get_compass_fsr(&compass_fsr);
        inv_set_gyro_sample_rate(1000000L/gyro_rate);
        inv_set_accel_sample_rate(1000000L/gyro_rate);
        inv_set_compass_sample_rate(COMPASS_READ_MS*1000L);
        inv_set_gyro_orientation_and_scale(
            inv_orientation_matrix_to_scalar(gyro_orientation),
(long)gyro_fsr<<15);
        inv_set_accel_orientation_and_scale(
            inv_orientation_matrix_to_scalar(gyro_orientation),
(long)accel_fsr<<15);
        inv_set_compass_orientation_and_scale(
            inv_orientation_matrix_to_scalar(comp_orientation),
(long)compass_fsr<<15);


        res=dmp_load_motion_driver_firmware();                  //加载dmp固件
        if(res)return 6;

res=dmp_set_orientation(inv_orientation_matrix_to_scalar(gyro_orientation));//设
置陀螺仪方向
        if(res)return 7;
        res=dmp_enable_feature(DMP_FEATURE_6X_LP_QUAT|DMP_FEATURE_TAP|
   //设置dmp功能

 DMP_FEATURE_ANDROID_ORIENT|DMP_FEATURE_SEND_RAW_ACCEL|DMP_FEATURE_SEND_CAL_GYRO
|
            DMP_FEATURE_GYRO_CAL);
        if(res)return 8;
        res=dmp_set_fifo_rate(DEFAULT_MPU_HZ);  //设置DMP输出速率(最大不超过200Hz)
        if(res)return 9;
        res=run_self_test();        //自检
        if(res)return 10;
        res=mpu_set_dmp_state(1);   //使能DMP
        if(res)return 11;
    }
    return 0;
}

/**
```

```
  * @brief   读取四元数值并计算得到实际的角度值
  * @param
  * @retval  void
 **/
int mpu_dmp_get_data(float *pitch, float *roll, float *yaw)
{
    float q0 = 1.0f, q1 = 0.0f, q2 = 0.0f, q3 = 0.0f;
    short gyro[3];
    short accel[3];
    long quat[4];
    unsigned long timestamp;
    short sensors;
    unsigned char more;
    if(dmp_read_fifo(gyro, accel, quat, &timestamp, &sensors, &more))
    {
        return -1;
    }

    if(sensors & INV_WXYZ_QUAT)
    {
        q0 = quat[0] / Q30;
        q1 = quat[1] / Q30;
        q2 = quat[2] / Q30;
        q3 = quat[3] / Q30;

        *pitch = asin(-2 * q1 * q3 + 2 * q0 * q2) * 57.3; // pitch
        *roll = atan2(2 * q2 * q3 + 2 * q0 * q1, -2 * q1 * q1 - 2 * q2 * q2 + 1)
* 57.3; // roll
        *yaw = atan2(2 * (q0 * q3 + q1 * q2), q0 * q0 + q1 * q1 - q2 * q2 - q3 *
q3) * 57.3; // yaw
    }

    return 0;
}

int mpu_mpl_get_data(float *pitch,float *roll,float *yaw)
{
    unsigned long sensor_timestamp;
    short gyro[3], accel_short[3],compass_short[3],sensors;
    unsigned char more;
    long compass[3],accel[3],quat[4],temperature;
    long data[9];
    inv_time_t timestamp;
    int8_t accuracy;

    if(dmp_read_fifo(gyro, accel_short, quat, &sensor_timestamp,
&sensors,&more))return 1;

    if(sensors&INV_XYZ_GYRO)
    {
        inv_build_gyro(gyro,sensor_timestamp);          //把新数据发送给MPL
        mpu_get_temperature(&temperature,&sensor_timestamp);
        inv_build_temp(temperature,sensor_timestamp);   //把温度值发给MPL，只有陀螺仪
需要温度值
    }

    if(sensors&INV_XYZ_ACCEL)
    {
```

```
        accel[0] = (long)accel_short[0];
        accel[1] = (long)accel_short[1];
        accel[2] = (long)accel_short[2];
        inv_build_accel(accel,0,sensor_timestamp);      //把加速度值发给MPL
    }

    if (!mpu_get_compass_reg(compass_short, &sensor_timestamp))
    {
        compass[0]=(long)compass_short[0];
        compass[1]=(long)compass_short[1];
        compass[2]=(long)compass_short[2];
        inv_build_compass(compass,0,sensor_timestamp); //把磁力计值发给MPL
    }
    inv_execute_on_data();
    inv_get_sensor_type_euler(data,&accuracy,&timestamp);

    *roll  = (data[0]/Q16);
    *pitch = -(data[1]/Q16);
    *yaw   = -data[2] / Q16;
    return 0;
}
```

并且在`inv_mpu_stm32port.h`中进行声明。

```
#ifndef _INV_MPU_STM32PORT_H
#define _INV_MPU_STM32PORT_H

#include "main.h"

int mpu_dmp_init(void);
int mpu_dmp_get_data(float *pitch, float *roll, float *yaw);
int mpu_mpl_get_data(float *pitch,float *roll,float *yaw);

#endif
```

然后就可以直接使用了

```
static void cal_with_dmp()
{
    int count = 0;
    unsigned long timestamp;
    mpu_dmp_init();
    while(1)
    {
        HAL_Delay(10);

if(mpu_dmp_get_data(&mpu9250_data.anglePitch,&mpu9250_data.angleRoll,&mpu9250_da
ta.angleYaw)==0)
        {

mpu9250_get_gyro(&mpu9250_data.gyro[0],&mpu9250_data.gyro[1],&mpu9250_data.gyro[
2]);
            mpu9250_get_acc (&mpu9250_data.acc[0] ,&mpu9250_data.acc[1]
,&mpu9250_data.acc[2]);
```

```
//mpu9250_get_mag(&mpu9250_data.mag[0],&mpu9250_data.mag[1],&mpu9250_data.mag[2]
);
            mpu_get_compass_reg(mpu9250_data.mag, &timestamp);
            //做单位解算 量程转换
            mpu9250_data.gyroxReal = mpu9250_data.gyro[0] *
MPU9250_GYRO_2000_SEN;
            mpu9250_data.gyroyReal = mpu9250_data.gyro[1] *
MPU9250_GYRO_2000_SEN;
            mpu9250_data.gyrozReal = mpu9250_data.gyro[2] *
MPU9250_GYRO_2000_SEN;
            mpu9250_data.accxReal  = mpu9250_data.acc[0]  *
MPU9250_ACCEL_2G_SEN;
            mpu9250_data.accyReal  = mpu9250_data.acc[1]  *
MPU9250_ACCEL_2G_SEN;
            mpu9250_data.acczReal  = mpu9250_data.acc[2]  *
MPU9250_ACCEL_2G_SEN;

            count ++;
            if(count % 100 == 0)
            {
                printf("%f, %f,
%f\r\n",mpu9250_data.anglePitch,mpu9250_data.angleRoll,mpu9250_data.angleYaw);
            }
        }
    }
}
```

调试遇到的问题

该方案调试过程中，使用的时候感觉DMP的计算很慢，稳定下来的数据也许准确，但是感觉yaw轴的数据还是不太对，这篇博客也只是成功的将代码移植了过来并且进行了运行。如果有人解决了我遇到的问题，欢迎交流。