> 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MPU6050，参考下面的文章。

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 mpu6050_write_bytes(uint8_t addr,uint8_t reg,uint8_t len,uint8_t *data);
int mpu6050_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 mpu6050_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 mpu6050_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：

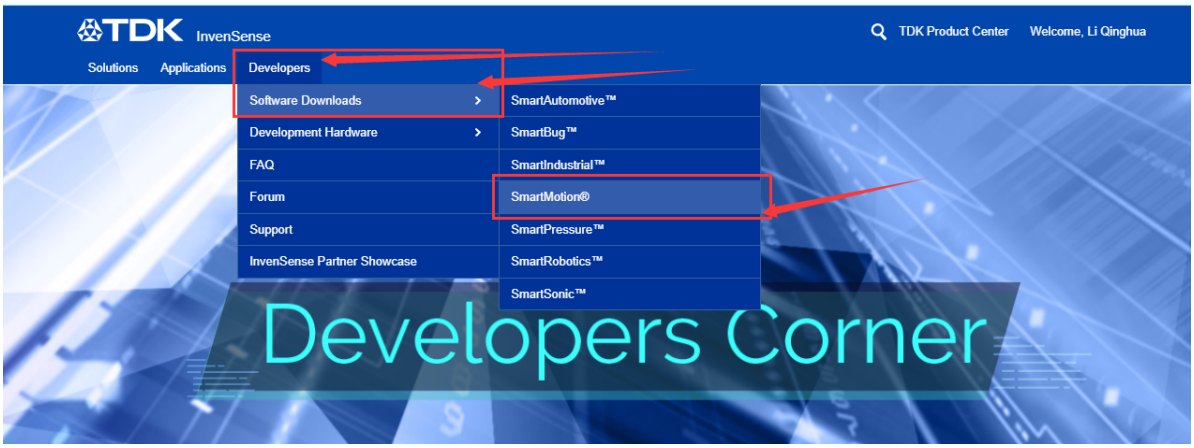
```
void u_i2c1_write_bytes(unsigned char add,unsigned char reg,unsigned char
*data,unsigned char len)
{
    HAL_I2C_Mem_Write(&hi2c1, (add<<1), reg, I2C_MEMADD_SIZE_8BIT,
data,len,HAL_MAX_DELAY);
}

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。

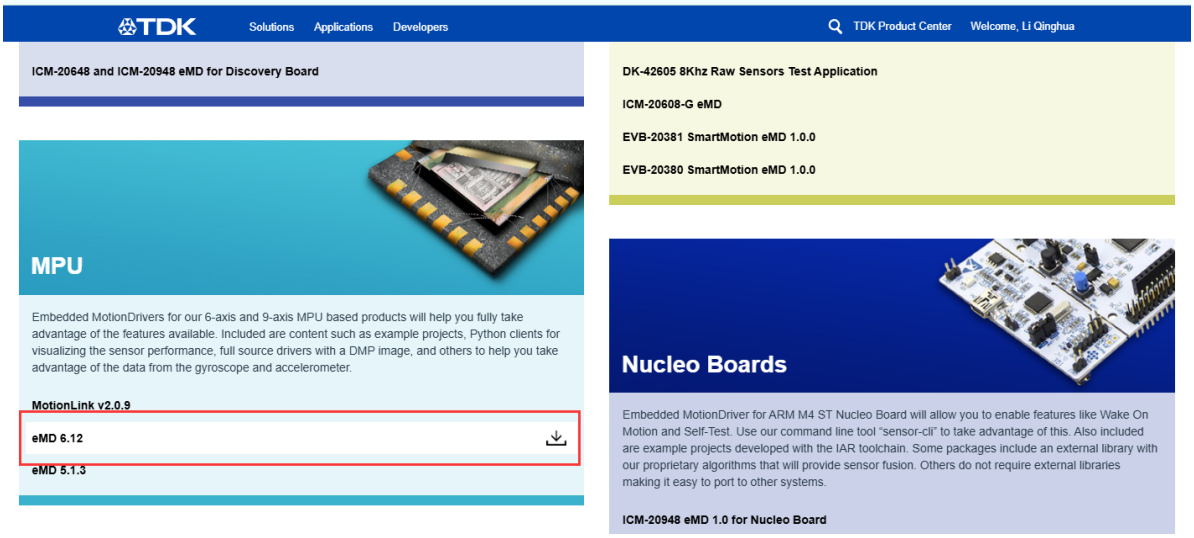
下载DMP库

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。



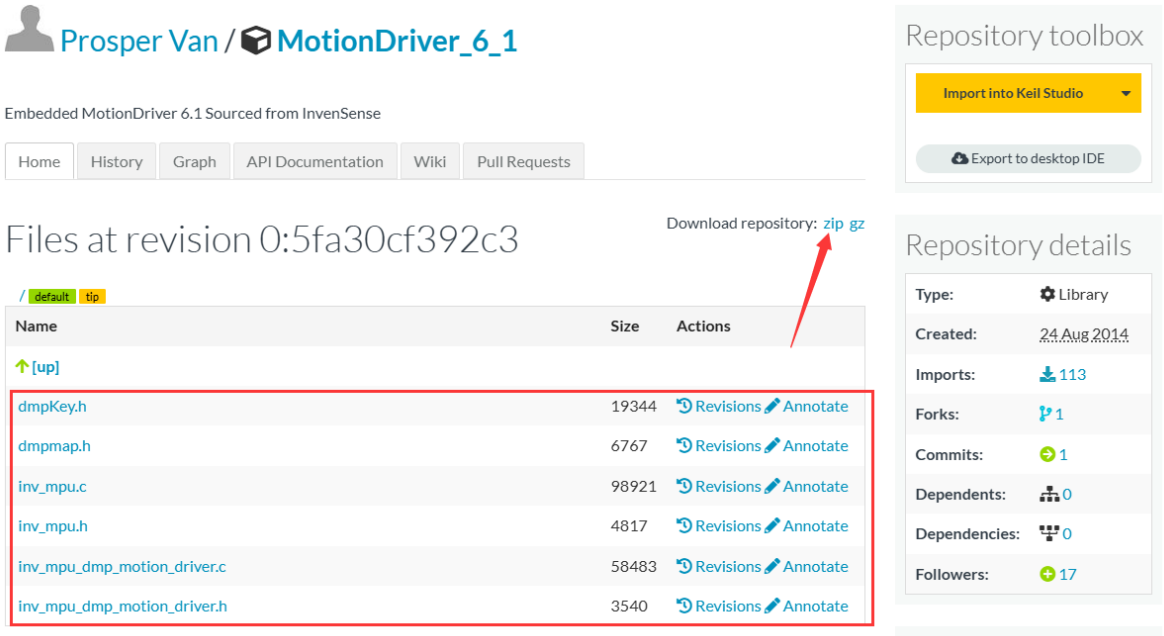
点进去后滑动到最底下，找到MPU栏目。点击压缩包下载。



或者也可以直接在下面的链接中下载仅DMP部分的文件。

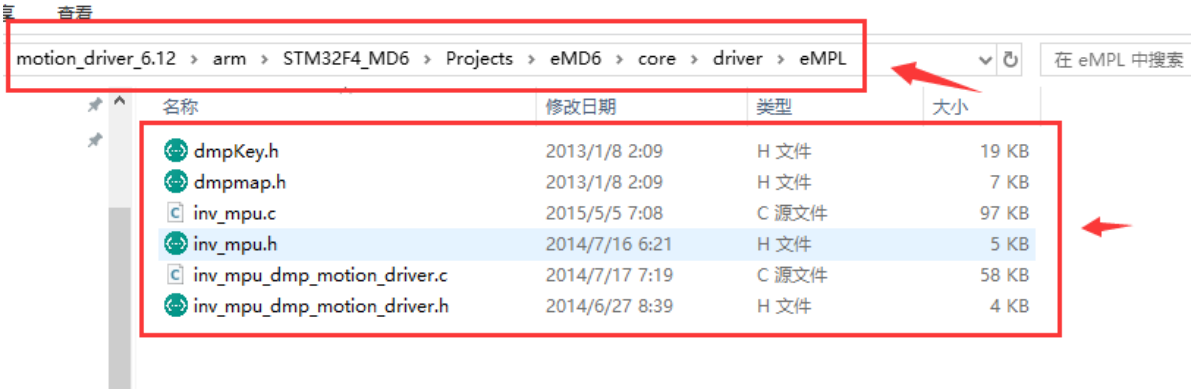
MotionDriver_V6.1下载：https://os.mbed.com/users/oprospero/code/MotionDriver_6_1

下载后会得到以下文件：



DMP移植

在下载的项目包中，找到和DMP解算相关的代码。



将所有文件放进项目工程，然后一个个处理工程文件，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。

首先是`inv_mpu.c`。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 "mpu6050.h"
#include "usart.h"

#define STM32_MPU6050
#define MPU6050
/*********************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，直接添加即可，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_MPU6050
```

```
#define i2c_write   mpu6050_write_bytes
#define i2c_read    mpu6050_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
#include "i2c.h"
#include "main.h"
#include "log.h"
#include "board-st_discovery.h"

#define i2c_write   Sensors_I2C_WriteRegister
#define i2c_read    Sensors_I2C_ReadRegister
#define delay_ms    mdelay
#define get_ms      get_tick_count
#define log_i       MPL_LOGI
#define log_e       MPL_LOGE
#define min(a,b) ((a<b)?a:b)
```

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 "mpu6050.h"

#define STM32_MPU6050
#define MPU6050
/*********************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_MPU6050

#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 "inv_mpu_dmp_motion_driver.h"
#include "stdio.h"

#define ERROR_MPU_INIT      -1
#define ERROR_SET_SENSOR    -2
#define ERROR_CONFIG_FIFO   -3
#define ERROR_SET_RATE      -4
#define ERROR_LOAD_MOTION_DRIVER    -5
#define ERROR_SET_ORIENTATION       -6
#define ERROR_ENABLE_FEATURE        -7
#define ERROR_SET_FIFO_RATE         -8
#define ERROR_SELF_TEST             -9
#define ERROR_DMP_STATE             -10

#define DEFAULT_MPU_HZ  100
#define Q30  1073741824.0f

/* 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。
*/
```

```c
static signed char gyro_orientation[9] = {-1, 0, 0,
                                           0,-1, 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);
    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)
{
    int ret;
    struct int_param_s int_param;

    ret = mpu_init(&int_param);
    if(ret != 0)return ERROR_MPU_INIT;

    //设置传感器
    ret = mpu_set_sensors(INV_XYZ_GYRO | INV_XYZ_ACCEL);
    if(ret != 0)return ERROR_SET_SENSOR;

    //设置fifo
    ret = mpu_configure_fifo(INV_XYZ_GYRO | INV_XYZ_ACCEL);
    if(ret != 0)return ERROR_CONFIG_FIFO;

    //设置采样率
    ret = mpu_set_sample_rate(DEFAULT_MPU_HZ);
    if(ret != 0)return ERROR_SET_RATE;

    //加载DMP固件
    ret = dmp_load_motion_driver_firmware();
    if(ret != 0)return ERROR_LOAD_MOTION_DRIVER;

    //设置陀螺仪方向
```

```
    ret =
dmp_set_orientation(inv_orientation_matrix_to_scalar(gyro_orientation));
    if(ret != 0)return ERROR_SET_ORIENTATION;

    //设置DMP功能
    ret = dmp_enable_feature(DMP_FEATURE_6X_LP_QUAT | DMP_FEATURE_TAP |
            DMP_FEATURE_ANDROID_ORIENT | DMP_FEATURE_SEND_RAW_ACCEL |
            DMP_FEATURE_SEND_CAL_GYRO | DMP_FEATURE_GYRO_CAL);
    if(ret != 0)return ERROR_ENABLE_FEATURE;

    //设置输出速率
    ret = dmp_set_fifo_rate(DEFAULT_MPU_HZ);
    if(ret != 0)return ERROR_SET_FIFO_RATE;

    //自检
    ret = run_self_test();
    if(ret != 0)return ERROR_SELF_TEST;

    //使能DMP
    ret = mpu_set_dmp_state(1);
    if(ret != 0)return ERROR_DMP_STATE;


    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;
}
```

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);

#endif
```

然后就可以直接使用了。

```
/**
  * @brief   使用DMP库计算角度
  * @param
  * @retval  void
 **/
static void cal_with_dmp()
{
    int count = 0;
    mpu_dmp_init();
    while(1)
    {
        //该部分更新会和初始化mpu6050时候的定义的采样率相关
        if(1 == data_ready)
        {
            data_ready = 0;

if(mpu_dmp_get_data(&mpu6050_data.anglePitch,&mpu6050_data.angleRoll,&mpu6050_da
ta.angleYaw)==0)
            {

mpu6050_get_gyro(&mpu6050_data.gyro[0],&mpu6050_data.gyro[1],&mpu6050_data.gyro[
2]);
                mpu6050_get_acc (&mpu6050_data.acc[0] ,&mpu6050_data.acc[1]
,&mpu6050_data.acc[2]);
            }
            count ++;
            if(count % 100 == 0)
            {
                printf("%f, %f,
%f\r\n",mpu6050_data.anglePitch,mpu6050_data.angleRoll,mpu6050_data.angleYaw);
            }
        }
    }
}
```

移植BUG

参考其他人的代码，发现自检的时候总会有各种问题，如果不通过，可以检查一下是不是卡在了自检，然后看看自检那部分的结果是什么，可以Debug的时候点进去，`ret = run_self_test();`这个函数，在这个函数里面打断点，用全局变量查看里面的一个函数`result = mpu_run_self_test(gyro, accel);`的输出结果，看看是什么，根据结果调整下一步中`if`判断语句中的内容，或者尝试把自检内容注释呢，我也不知道会不会有问题。