Application Note

ATK-MS6050 模块使用说明

高性能三轴加速度+三轴陀螺仪模块

使用说明

正点原子

广州市星翼电子科技有限公司

修订历史

版本	日期	原因
V1.0	2022/06/25	第一次发布
V1.1	2023/03/11	添加对阿波罗 STM32F429 开发板和阿波罗 STM32F767 开发板的支持
V1.2	2023/04/15	添加对阿波罗 STM32H743 开发板的支持

目 录

1，硬件连接

1.1 正点原子 MiniSTM32F103 开发板

ATK-MS6050 模块可直接与正点原子 MiniSTM32F103 开发板板载的 ATK 模块接口（ATK MODULE）进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS6050 模块	VCC	GND	SDA	SCL	INT	AD0
MiniSTM32F103 开发板	3.3V/5V	GND	PD2	PC12	-	PA4

表 1.1.1 ATK-MS6050 模块与 MiniSTM32F103 开发板连接关系

1.2 正点原子精英 STM32F103 开发板

ATK-MS6050 模块可直接与正点原子精英 STM32F103 开发板板载的 ATK 模块接口（ATK MODULE）进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS6050 模块	VCC	GND	SDA	SCL	INT	AD0
精英 STM32F103 开发板	3.3V/5V	GND	PB11	PB10	-	PA15

表 1.2.1 ATK-MS6050 模块与精英 STM32F103 开发板连接关系

1.3 正点原子战舰 STM32F103 开发板

ATK-MS6050 模块可直接与正点原子战舰 STM32F103 开发板板载的 ATK 模块接口（ATK MODULE）进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS6050 模块	VCC	GND	SDA	SCL	INT	AD0
战舰 STM32F103 开发板	3.3V/5V	GND	PB11	PB10	-	PA15

表 1.3.1 ATK-MS6050 模块与战舰 STM32F103 开发板连接关系

注意，若要使用正点原子战舰 STM32F103 开发板的 ATK MODULE 接口连接 ATK-MS6050 模块，需要用跳线帽将开发板板载的 P8 接线端子的 PB10(TX)和 GBC_RX 以及 PB11(RX)和 GBC_TX 用跳线帽进行短接，如下图所示：

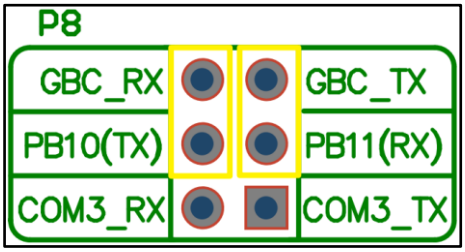

图 1.3.1 战舰 STM32F103 开发板 P8 接线端子

1.4 正点原子探索者 STM32F407 开发板

ATK-MS6050 模块可直接与正点原子探索者 STM32F407 开发板板载的 ATK 模块接口（ATK MODULE）进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS6050 模块	VCC	GND	SDA	SCL	INT	AD0
探索者 STM32F407 开发板	3.3V/5V	GND	PB11	PB10	-	PC0

表 1.4.1 ATK-MS6050 模块与探索者 STM32F407 开发板连接关系

注意，若要使用正点原子探索者 STM32F407 开发板的 ATK MODULE 接口连接 ATK-MS6050 模块，需要用跳线帽将开发板板载的 P2 接线端子的 PB10(TX)和 GBC_RX 以及 PB11(RX)和 GBC_TX 用跳线帽进行短接，如下图所示：

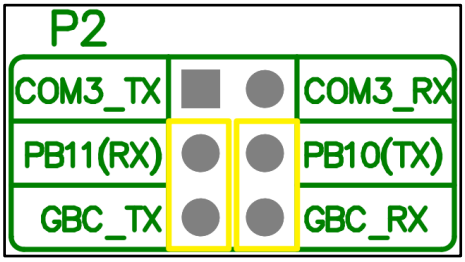

图 1.4.1 探索者 STM32F407 开发板 P2 接线端子

1.5 正点原子 F407 电机控制开发板

ATK-MS6050 模块可直接与正点原子 F407 电机控制开发板板载的 ATK 模块接口（ATK MODULE）进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS6050 模块	VCC	GND	SDA	SCL	INT	AD0
F407 电机控制开发板	3.3V/5V	GND	PC11	PC10	-	PI11

表 1.5.1 ATK-MS6050 模块与 F407 电机控制开发板连接关系

1.6 正点原子 MiniSTM32H750 开发板

ATK-MS6050 模块可直接与正点原子 MiniSTM32H750 开发板板载的 ATK 模块接口（ATK MODULE）进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS6050 模块	VCC	GND	SDA	SCL	INT	AD0
MiniSTM32H750 开发板	3.3V/5V	GND	PA3	PA2	-	PC3

表 1.6.1 ATK-MS6050 模块与 MiniSTM32H750 开发板连接关系

1.7 正点原子阿波罗 STM32F429 开发板

ATK-MS6050 模块可直接与正点原子阿波罗 STM32F429 开发板板载的 ATK 模块接口

（ATK MODULE）进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS6050 模块	VCC	GND	SDA	SCL	INT	AD0
阿波罗 STM32F429 开发板	3.3V/5V	GND	PB11	PB10	-	PA4

表 1.7.1 ATK-MS6050 模块与阿波罗 STM32F429 开发板连接关系

注意，若要使用正点原子阿波罗 STM32F429 开发板的 ATK MODULE 接口连接 ATK-MS6050 模块，需要用跳线帽将开发板板载的 P9 接线端子的 PB10(TX)和 GBC_RX 以及 PB11(RX)和 GBC_TX 用跳线帽进行短接，如下图所示：

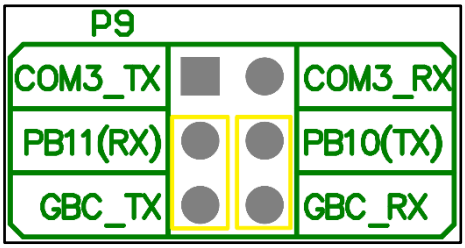

图 1.7.1 阿波罗 STM32F429 开发板 P9 接线端子

1.8 正点原子阿波罗 STM32F767 开发板

ATK-MS6050 模块可直接与正点原子阿波罗 STM32F767 开发板板载的 ATK 模块接口（ATK MODULE）进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS6050 模块	VCC	GND	SDA	SCL	INT	AD0
阿波罗 STM32F767 开发板	3.3V/5V	GND	PB11	PB10	-	PA4

表 1.8.1 ATK-MS6050 模块与阿波罗 STM32F767 开发板连接关系

注意，若要使用正点原子阿波罗 STM32F767 开发板的 ATK MODULE 接口连接 ATK-MS6050 模块，需要用跳线帽将开发板板载的 P9 接线端子的 PB10(TX)和 GBC_RX 以及 PB11(RX)和 GBC_TX 用跳线帽进行短接，如下图所示：

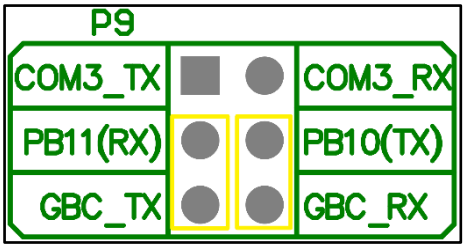

图 1.8.1 阿波罗 STM32F767 开发板 P9 接线端子

1.9 正点原子阿波罗 STM32H743 开发板

ATK-MS6050 模块可直接与正点原子阿波罗 STM32H743 开发板板载的 ATK 模块接口（ATK MODULE）进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系					
ATK-MS6050 模块	VCC	GND	SDA	SCL	INT	AD0
阿波罗 STM32H743 开发板	3.3V/5V	GND	PB11	PB10	-	PA4

表 1.9.1 ATK-MS6050 模块与阿波罗 STM32H743 开发板连接关系

注意，若要使用正点原子阿波罗 STM32H743 开发板的 ATK MODULE 接口连接 ATK-MS6050 模块，需要用跳线帽将开发板板载的 P9 接线端子的 PB10(TX)和 GBC_RX 以及 PB11(RX)和 GBC_TX 用跳线帽进行短接，如下图所示：

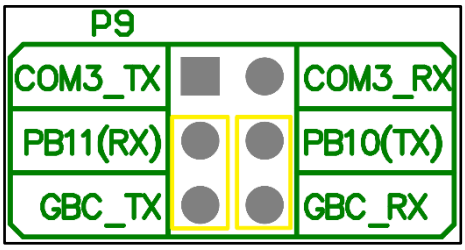

图 1.9.1 阿波罗 STM32H743 开发板 P9 接线端子

2，实验功能

2.1 ATK-MS6050 模块测试实验

2.1.1 功能说明

在本实验中，开发板主控芯片通过模拟 IIC 与 ATK-MS6050 模块进行通讯，从而获取 ATK-MS6050 模块的三轴加速度、三轴陀螺仪和温度的采样值以及 DMP 姿态解算后的欧拉角等信息，并实时在 LCD 上显示这些数据，同时可通过开发板上的按键，选择性地将获取到的数据通过串口发送到串口调试助手或通过串口发送到匿名地面站 V4 进行查看。

2.1.2 源码解读

打开本实验的工程文件夹，能够在./Drivers/BSP 目录下看到 ATK_MS6050 子文件夹，该文件夹中就包含了 ATK-MS6050 模块的驱动文件和 MPU-6050 官方的 DMP 驱动库（已移植到 STM32 上），如下图所示：

```
./Drivers/BSP/ATK_MS6050/
|-- atk_ms6050.c
|-- atk_ms6050.h
|-- atk_ms6050_iic.c
|-- atk_ms6050_iic.h
`-- eMPL
    |-- dmpKey.h
    |-- dmpmap.h
    |-- inv_mpu.c
    |-- inv_mpu.h
    |-- inv_mpu_dmp_motion_driver.c
    `-- inv_mpu_dmp_motion_driver.h
```

图 2.1.2.1 ATK-MS6050 模块驱动代码

2.1.2.1 ATK-MS6050 模块接口驱动

在图 2.1.2.1 中，atk_ms6050_iic.c 和 atk_ms6050_iic.h 是开发板与 ATK-MS6050 模块通讯而使用的模拟 IIC 驱动文件，关于模拟 IIC 的驱动介绍，请查看正点原子各个开发板对应的开发指南中模拟 IIC 对应的章节。

2.1.2.2 ATK-MS6050 模块驱动

在图 2.1.2.1 中，atk_ms6050.c 和 atk_ms6050.h 是 ATK-MS6050 模块的驱动文件，包含了 ATK-MS6050 模块初始化、读写各个寄存器的相关 API 函数。函数比较多，下面仅介绍几个重要的 API 函数。

1. 函数 atk_ms6050_init()

该函数用于初始化 ATK-MS6050 模块，具体的代码，如下所示：

```
/**
 * @brief   ATK-MS6050 初始化
 * @param   无
 * @retval  ATK_MS6050_EOK: 函数执行成功
```

```
 *          ATK_MS6050_EID: 获取 ID 错误，函数执行失败
 */
uint8_t atk_ms6050_init(void)
{
    uint8_t id;

    atk_ms6050_hw_init();           /* ATK-MS6050 硬件初始化 */
    atk_ms6050_iic_init();          /* 初始化 IIC 接口 */
    atk_ms6050_sw_reset();          /* ATK-MS050 软件复位 */
    atk_ms6050_set_gyro_fsr(3);     /* 陀螺仪传感器，±2000dps */
    atk_ms6050_set_accel_fsr(0);    /* 加速度传感器，±2g */
    atk_ms6050_set_rate(50);        /* 采样率，50Hz */
    /* 关闭所有中断 */
    atk_ms6050_write_byte(ATK_MS6050_IIC_ADDR, MPU_INT_EN_REG, 0X00);
    /* 关闭 IIC 主模式 */
    atk_ms6050_write_byte(ATK_MS6050_IIC_ADDR, MPU_USER_CTRL_REG, 0X00);
    /* 关闭 FIFO */
    atk_ms6050_write_byte(ATK_MS6050_IIC_ADDR, MPU_FIFO_EN_REG, 0X00);
    /* INT 引脚低电平有效 */
    atk_ms6050_write_byte(ATK_MS6050_IIC_ADDR, MPU_INTBP_CFG_REG, 0X80);
    /* 读取设备 ID */
    atk_ms6050_read_byte(ATK_MS6050_IIC_ADDR, MPU_DEVICE_ID_REG, &id);
    if (id != ATK_MS6050_IIC_ADDR)
    {
        return ATK_MS6050_EID;
    }
    /* 设置 CLKSEL，PLL X 轴为参考 */
    atk_ms6050_write_byte(ATK_MS6050_IIC_ADDR, MPU_PWR_MGMT1_REG, 0x01);
    /* 加速度与陀螺仪都工作 */
    atk_ms6050_write_byte(ATK_MS6050_IIC_ADDR, MPU_PWR_MGMT2_REG, 0x00);
    atk_ms6050_set_rate(50);        /* 采样率，50Hz */

    return ATK_MS6050_EOK;
}
```

从上面的代码中可以看出，函数 atk_ms6050_init()就是通过模拟 IIC 与 ATK-MS6050 模块进行通讯，向其寄存器写入对应的配置值，以此完成对 ATK-MS6050 模块的初始化，初始化完 ATK-MS6050 模块后就能够读取该模块的各种传感器的数据了。

2. 函数 atk_ms6050_get_temperature/gyroscope/accelerometer()

这三个函数分别用户获取 ATK-MS6050 模块中温度传感器、加速度传感器和陀螺仪传感器的数据，具体的代码，如下所示：

```
/**
 * @brief   ATK-MS6050 获取温度值
 * @param   temperature: 获取到的温度值（扩大了 100 倍）
 * @retval  ATK_MS6050_EOK : 函数执行成功
```

```
 *          ATK_MS6050_EACK: IIC 通讯 ACK 错误，函数执行失败
 */
uint8_t atk_ms6050_get_temperature(int16_t *temp)
{
    uint8_t dat[2];
    uint8_t ret;
    int16_t raw = 0;

    ret = atk_ms6050_read(ATK_MS6050_IIC_ADDR, MPU_TEMP_OUTH_REG, 2, dat);
    if (ret == ATK_MS6050_EOK)
    {
        raw = ((uint16_t)dat[0] << 8) | dat[1];
        *temp = (int16_t)((36.53f + ((float)raw / 340)) * 100);
    }

    return ret;
}

/**
 * @brief   ATK-MS6050 获取陀螺仪值
 * @param   gx，gy，gz: 陀螺仪 x、y、z 轴的原始度数（带符号）
 * @retval  ATK_MS6050_EOK : 函数执行成功
 *          ATK_MS6050_EACK: IIC 通讯 ACK 错误，函数执行失败
 */
uint8_t atk_ms6050_get_gyroscope(int16_t *gx, int16_t *gy, int16_t *gz)
{
    uint8_t dat[6];
    uint8_t ret;

    ret =  atk_ms6050_read(ATK_MS6050_IIC_ADDR, MPU_GYRO_XOUTH_REG, 6, dat);
    if (ret == ATK_MS6050_EOK)
    {
        *gx = ((uint16_t)dat[0] << 8) | dat[1];
        *gy = ((uint16_t)dat[2] << 8) | dat[3];
        *gz = ((uint16_t)dat[4] << 8) | dat[5];
    }

    return ret;
}

/**
 * @brief   ATK-MS6050 获取加速度值
 * @param   ax，ay，az: 加速度 x、y、z 轴的原始度数（带符号）
 * @retval  ATK_MS6050_EOK : 函数执行成功
```

```
 *          ATK_MS6050_EACK: IIC 通讯 ACK 错误，函数执行失败
 */
uint8_t atk_ms6050_get_accelerometer(int16_t *ax, int16_t *ay, int16_t *az)
{
    uint8_t dat[6];
    uint8_t ret;

    ret =  atk_ms6050_read(ATK_MS6050_IIC_ADDR, MPU_ACCEL_XOUTH_REG, 6, dat);
    if (ret == ATK_MS6050_EOK)
    {
        *ax = ((uint16_t)dat[0] << 8) | dat[1];
        *ay = ((uint16_t)dat[2] << 8) | dat[3];
        *az = ((uint16_t)dat[4] << 8) | dat[5];
    }

    return ret;
}
```

从上面的代码中可以看出，获取 ATK-MS6050 模块中温度传感器、加速度传感器和陀螺仪传感器的数据就是通过相应的寄存器地址，从寄存器中读取相应的数据即可，当然，这三个函数还对获取到的数据值做了初步处理，使得数据更加直观。

2.1.2.3 ATK-MS6050 模块 DMP 驱动

在图 2.1.1 中，子文件夹 eMPL 下包含的是 MPU-6050 官方的 DMP 驱动库，该驱动库原本是基于 MSP430 开发的，这里为了将其应用到 STM32 上，已经做了相应的移植。下面介绍几个移植后添加的 API 函数。

1. 函数 atk_ms6050_dmp_init()

该函数用于初始化 ATK-MS6050 模块的 DMP，具体的代码，如下所示：

```
/**
 * @brief   ATK-MS6050 DMP 初始化
 * @param   无
 * @retval  0: 函数执行成功
 *          1: 函数执行失败
 */
uint8_t atk_ms6050_dmp_init(void)
{
    uint8_t ret;

    ret  = mpu_init(NULL);                                      /* 硬件初始化 */
    ret += mpu_set_sensors(INV_XYZ_GYRO | INV_XYZ_ACCEL);       /* 开启指定传感器 */
    ret += mpu_configure_fifo(INV_XYZ_GYRO | INV_XYZ_ACCEL);    /* 设置 FIFO */
    ret += mpu_set_sample_rate(DEFAULT_MPU_HZ);                 /* 设置采样率 */
    ret += dmp_load_motion_driver_firmware();                   /* 加载 DMP 镜像 */
    ret += dmp_set_orientation(                                 /* 设置陀螺仪方向 */
    inv_orientation_matrix_to_scalar(gyro_orientation));
    ret += dmp_enable_feature(  DMP_FEATURE_6X_LP_QUAT      |/* 设置 DMP 功能 */
```

```
                                DMP_FEATURE_TAP             |
                                DMP_FEATURE_ANDROID_ORIENT  |
                                DMP_FEATURE_SEND_RAW_ACCEL  |
                                DMP_FEATURE_SEND_CAL_GYRO   |
                                DMP_FEATURE_GYRO_CAL);
    ret += dmp_set_fifo_rate(DEFAULT_MPU_HZ);           /* 设置 DMP 输出速率 */
    ret += mpu_set_dmp_state(1);                            /* 使能 DMP */
    ret += atk_ms6050_run_self_test();                      /* 传感器自测试 */

    return ((ret == 0) ? 0 : 1);
}
```

从上面的代码中可以看出，函数 atk_ms6050_dmp_init()会调用 MPU-6050 官方 DMP 驱动库中的函数来对 ATK-MS6050 模块的 DMP 进行初始化配置，在初始化 ATK-MS6050 模块的 DMP 后，就能够读取 DMP 输出的数据了。

2. 函数 atk_ms6050_dmp_get_data()

该函数用于获取 ATK-MS6050 模块中 DMP 输出的数据，具体的代码，如下所示：

```
/**
 * @brief   获取 ATK-MS6050 DMP 处理后的数据
 * @note    获取数据的频率需与宏 DEFAULT_MPU_HZ 定义的频率一致，
 *          获取太快，可能因 ATK-MS6050 还未进行数据采样，导致 FIFO 中无数据，从而获取失败，
 *          获取太慢，可能无法及时读出 ATK-MS6050 FIFO 中的数据，导致 FIFO 溢出，
 *          从而获取失败
 * @param   pitch: 俯仰角（精度: 0.1° 范围: -90.0° <---> +90.0°）
 *          roll : 横滚角（精度: 0.1° 范围: -180.0° <---> +180.0°）
 *          yaw  : 航向角（精度: 0.1° 范围: -180.0° <---> +180.0°）
 * @retval  0: 函数执行成功
 *          1: 函数执行失败
 */
uint8_t atk_ms6050_dmp_get_data(float *pitch, float *roll, float *yaw)
{
    float q0 = 0.0f;
    float q1 = 0.0f;
    float q2 = 0.0f;
    float q3 = 0.0f;
    short gyro[3], accel[3], sensors;
    unsigned long sensor_timestamp;
    unsigned char more;
    long quat[4];

    /* 读取 ATK-MS6050 FIFO 中数据的频率需与宏 DEFAULT_MPU_HZ 定义的频率一直
     * 读取得太快或太慢都可能导致读取失败
     * 读取太快：ATK-MS6050 还未采样，FIFO 中无数据，读取失败
     * 读取太慢：ATK-MS6050 的 FIFO 溢出，读取失败
     */
```

```
    if (dmp_read_fifo(gyro, accel, quat, &sensor_timestamp, &sensors, &more)
        != 0)
    {
        return 1;
    }

    if (sensors & INV_WXYZ_QUAT)
    {
        /* ATK-MS6050 的 DMP 输出的是姿态解算后的四元数，
         * 采用 q30 格式，即结果被放大了 2 的 30 次方倍，
         * 因为四元数并不是角度信号，因此为了得到欧拉角，
         * 就需要对 ATK-MS6050 的 DMP 输出结果进行转换
         */
        q0 = quat[0] / q30;
        q1 = quat[1] / q30;
        q2 = quat[2] / q30;
        q3 = quat[3] / q30;

        /* 计算俯仰角、横滚角、航向角
         * 57.3 为弧度转角度的转换系数，即 180/PI
         */
        *pitch  = asin(-2*q1*q3+2*q0*q2)*57.3;
        *roll   = atan2(2*q2*q3+2*q0*q1, -2*q1*q1-2*q2*q2+1)*57.3;
        *yaw    = atan2(2*(q1*q2+q0*q3), q0*q0+q1*q1-q2*q2-q3*q3)*57.3;
    }
    else
    {
        return 1;
    }

    return 0;
}
```

从上面的代码中可以看出，ATK-MS6050 模块的 DMP 输出的是姿态解算后的四元数，并且采用的是 q30 的格式进行表示的，以此该函数将获取到的 DMP 输出数据，转换成以角度表示的俯仰角、横滚角和航向角的数据信息，这样方便使用。因为转换的过程涉及较多的数学理论，因此在这里不分析具体的转换原理，感兴趣的读者可以自行通过搜索引擎了解。

2.1.2.4 实验测试代码

实验的测试代码为文件 demo.c，在工程目录下的 User 子目录中。测试代码的入口函数为 demo_run()，具体的代码，如下所示：

```
/**
 * @brief   例程演示入口函数
 * @param   无
 * @retval  无
 */
```

```
void demo_run(void)
{
    uint8_t  ret;
    uint8_t  key;
    uint8_t  niming_report = 0;
    float    pit, rol, yaw;
    int16_t  acc_x, acc_y, acc_z;
    int16_t  gyr_x, gyr_y, gyr_z;
    int16_t  temp;

    /* 初始化 ATK-MS6050 */
    ret = atk_ms6050_init();
    if (ret != 0)
    {
        printf("ATK-MS6050 init failed!\r\n");
        while (1)
        {
            LED0_TOGGLE();
            delay_ms(200);
        }
    }

    /* 初始化 ATK-MS6050 DMP */
    ret = atk_ms6050_dmp_init();
    if (ret != 0)
    {
        printf("ATK-MS6050 DMP init failed!\r\n");
        while (1)
        {
            LED0_TOGGLE();
            delay_ms(200);
        }
    }

    /* LCD UI 初始化 */
    demo_lcd_ui_init();

    while (1)
    {
        key = key_scan(0);
        if (key == KEY0_PRES)
        {
            /* 当按键 0 按下后，切换串口的上传状态
             * 当 niming_report 为 0 时，上传相关信息至串口调试助手
```

```
         * 当 niming_report 为 1 时，上传相关信息至匿名地面站 V4
         */
        niming_report = 1 - niming_report;
        if (niming_report == 0)
        {
            /* 串口调试助手的串口通讯波特率为 115200 */
            usart_init(115200);
        }
        else
        {
            /* 匿名地面站 V4 的串口通讯波特率为 500000 */
            usart_init(500000);
        }
    }

    /* 获取 ATK-MS6050 DMP 处理后的数据 */
    ret  = atk_ms6050_dmp_get_data(&pit, &rol, &yaw);
    /* 获取 ATK-MS6050 加速度值 */
    ret += atk_ms6050_get_accelerometer(&acc_x, &acc_y, &acc_z);
    /* 获取 ATK-MS6050 陀螺仪值 */
    ret += atk_ms6050_get_gyroscope(&gyr_x, &gyr_y, &gyr_z);
    /* 获取 ATK-MS6050 温度值 */
    ret += atk_ms6050_get_temperature(&temp);
    if (ret == 0)
    {
        if (niming_report == 0)
        {
            /* 上传相关数据信息至串口调试助手 */
            printf("pit: %.2f, rol: %.2f, yaw: %.2f, ", pit, rol, yaw);
            printf("acc_x: %d, acc_y: %d, acc_z: %d, ", acc_x, acc_y, acc_z);
            printf("gyr_x: %d, gyr_y: %d, gyr_z: %d, ", gyr_x, gyr_y, gyr_z);
            printf("temp: %d\r\n", temp);
        }
        else
        {
            /* 上传状态帧和传感器帧至匿名地面站 V4 */
            demo_niming_report_status( (int16_t)(rol * 100),
                                       (int16_t)((pit) * 100),
                                       (int16_t)(yaw * 100),
                                       0, 0, 0);
            demo_niming_report_senser( acc_x, acc_y, acc_z,
                                       gyr_x, gyr_y, gyr_z,
                                       0, 0, 0);
        }
```

```
            /* 在 LCD 上显示相关数据信息 */
            demo_lcd_show_msg(pit, rol, yaw, temp);
        }
    }
}
```

从上面的代码中可以看出，整个测试代码的逻辑还是比较简单的，就是要注意的是，因为与匿名地面站 V4 通讯的时候，需要传输大量的数据，因此串口的通讯波特率设置为了 500000，而与串口调试助手通讯的时候，串口的波特率依然是默认的 115200，因此在进行实验的时候，需要就串口调试助手和匿名地面站做想用的串口配置，才能够正常通讯。

2.1.3 实验现象

将 ATK-MS6050 模块按照第一节“硬件连接”中介绍的连接方式与开发板连接，并将实验代码编译烧录至开发板中，如果此时开发板连接 LCD，那么 LCD 显示的内容，如下图所示：

图 2.1.3.1 LCD 显示内容一

同时，通过串口调试助手输出实验信息，如下图所示：

图 2.1.3.2 串口调试助手显示内容一

接下来，如果 ATK-MS6050 模块初始化成功，则会分别在 LCD 和串口调试助手上显示相应的数据信息，如下图所示：

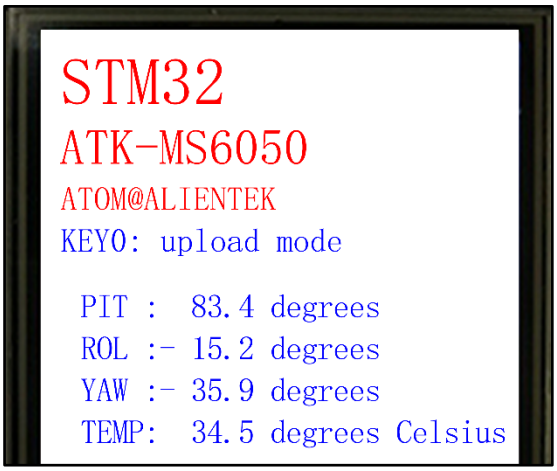

图 2.1.3.3 LCD 显示内容二

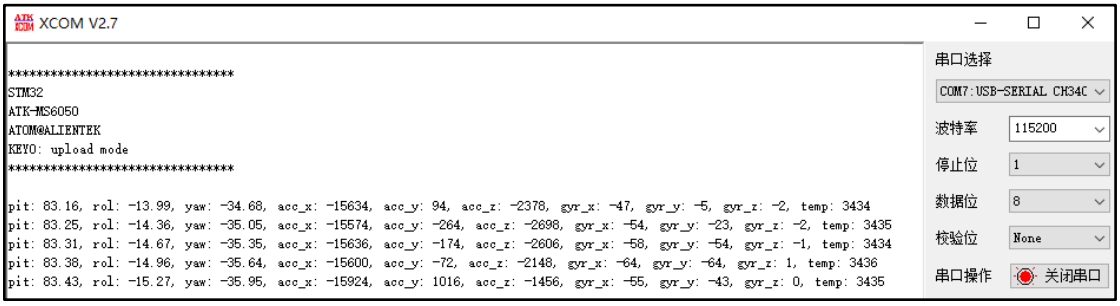

图 2.1.3.4 串口调试助手显示内容二

以上就是通过串口调试助手显示 ATK-MS6050 中各传感器采集的数据信息。同时，本实验也支持通过匿名地面站 V4，以图形化的形式，展示传感器的数据和 3D 姿态，只需按下按键 0，就能够更改串口上传数据的模式，如下图所示：

图 2.1.3.5 匿名地面站 V4 数据波形显示

上图为匿名地面站 V4 展示的数据波形图，在软件中，可以通过勾选下方的复选框，选

择想要在图表中展示的波形，本实验中展示了 X、Y、Z 轴方向上的加速度和角度值。

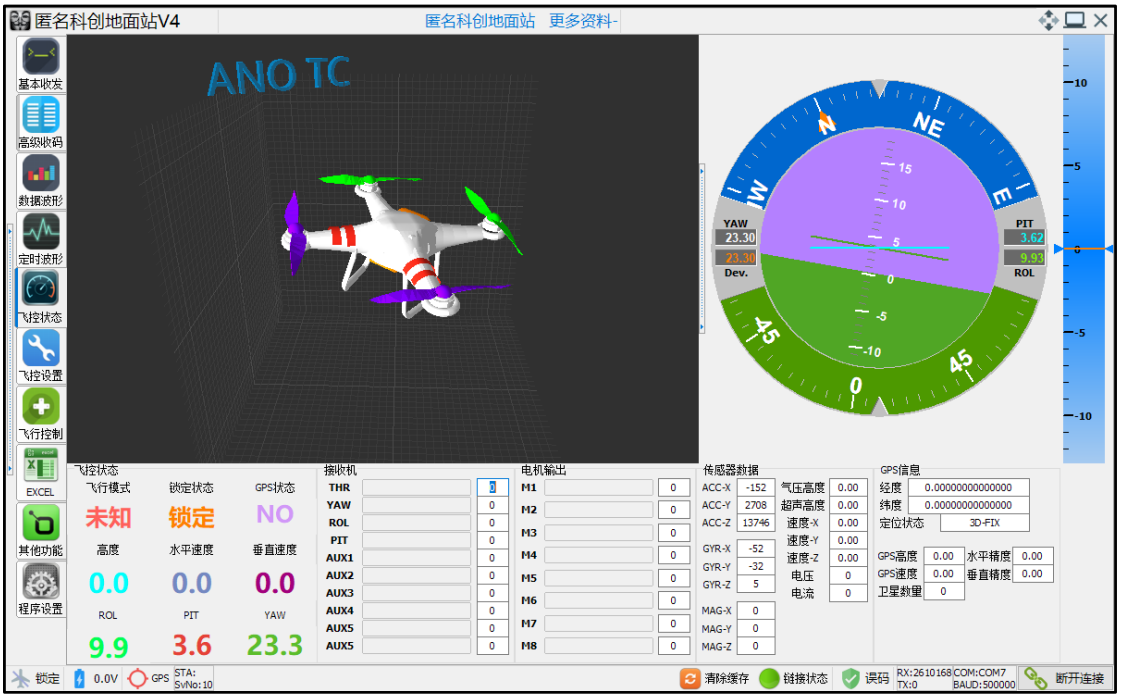

图 2.1.3.6 匿名地面站 V4 3D 姿态显示

上图为匿名地面站 V4 展示的 3D 姿态显示图，左上角的 3D 模型，能够根据开发板上报的角度数据，实时地展示开发板的 3D 姿态，并且也可通过右上角的罗盘和下方的数据，实时观察开发板上报的各种姿态数据。

3，其他

1、 购买地址：

天猫：https://zhengdianyuanzi.tmall.com

淘宝：https://openedv.taobao.com

2、 资料下载

模块资料下载地址：http://www.openedv.com/docs/modules/other/ATK-MPU6050.html

3、 技术支持

公司网址：www.alientek.com

技术论坛：http://www.openedv.com/forum.php

在线教学：www.yuanzige.com

B 站视频：https://space.bilibili.com/394620890

传真：020-36773971

电话：020-38271790

正点原子