Application Note

ATK-MS901M 模块使用说明

高性能角度传感器模块（十轴）

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 开发板的阿波罗 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-MS901M 模块通过串口直接与正点原子 MiniSTM32F103 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系			
ATK-MS901M 模块	VCC	GND	TX	RX
MiniSTM32F103 开发板	3.3V/5V	GND	PD2	PC12

表 1.1.1 ATK-MS901M 模块与 MiniSTM32F103 开发板连接关系

1.2 正点原子精英 STM32F103 开发板

ATK-MS901M 模块通过串口直接与正点原子精英 STM32F103 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系			
ATK-MS901M 模块	VCC	GND	TX	RX
精英 STM32F103 开发板	3.3V/5V	GND	PB11	PB10

表 1.2.1 ATK-MS901M 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-MS901M 模块通过串口直接与正点原子战舰 STM32F103 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系			
ATK-MS901M 模块	VCC	GND	TX	RX
战舰 STM32F103 开发板	3.3V/5V	GND	PB11	PB10

表 1.3.1 ATK-MS901M 模块与战舰 STM32F103 开发板连接关系

1.4 正点原子探索者 STM32F407 开发板

ATK-MS901M 模块通过串口直接与正点原子探索者 STM32F407 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系			
ATK-MS901M 模块	VCC	GND	TX	RX
探索者 STM32F407 开发板	3.3V/5V	GND	PB11	PB10

表 1.4.1 ATK-MS901M 模块与探索者 STM32F407 开发板连接关系

1.5 正点原子 F407 电机控制开发板

ATK-MS901M 模块通过串口直接与正点原子 F407 电机控制开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系			
ATK-MS901M 模块	VCC	GND	TX	RX
F407 电机控制开发板	3.3V/5V	GND	PC11	PC10

表 1.5.1 ATK-MS901M 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-MS901M 模块通过串口直接与正点原子 MiniSTM32H750 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系			
ATK-MS901M 模块	VCC	GND	TX	RX
MiniSTM32H750 开发板	3.3V/5V	GND	PA3	PA2

表 1.6.1 ATK-MS901M 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-MS901M 模块通过串口直接与正点原子阿波罗 STM32F429 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系			
ATK-MS901M 模块	VCC	GND	TX	RX
阿波罗 STM32F429 开发板	3.3V/5V	GND	PB11	PB10

表 1.7.1 ATK-MS901M 模块与阿波罗 STM32F429 开发板连接关系

1.8 正点原子阿波罗 STM32F767 开发板

ATK-MS901M 模块通过串口直接与正点原子阿波罗 STM32F767 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系			
ATK-MS901M 模块	VCC	GND	TX	RX
阿波罗 STM32F767 开发板	3.3V/5V	GND	PB11	PB10

表 1.8.1 ATK-MS901M 模块与阿波罗 STM32F767 开发板连接关系

1.9 正点原子阿波罗 STM32H743 开发板

ATK-MS901M 模块通过串口直接与正点原子阿波罗 STM32H743 开发板进行连接，具体的连接关系，如下表所示：

模块对应开发板	连接关系			
ATK-MS901M 模块	VCC	GND	TX	RX
阿波罗 STM32H743 开发板	3.3V/5V	GND	PB11	PB10

表 1.9.1 ATK-MS901M 模块与阿波罗 STM32H743 开发板连接关系

2，实验功能

2.1 ATK-MS901M 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 UART 与 ATK-MS901M 模块进行通讯，从而获取 ATK-MS901M 模块部分寄存器的数据，以及 ATK-MS901M 模块主动上报的各项数据，其中就包括姿态角、陀螺仪、加速度计、磁力计、气压计等传感器的数据，并将这些数据显示到串口调试助手和 LCD 上。

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_MS901M 子文件夹，该文件夹中就包含了 ATK-MS901M 模块的驱动文件，如下图所示：

```
./Drivers/BSP/ATK_MS901M/
|-- atk_ms901m.c
|-- atk_ms901m.h
|-- atk_ms901m_uart.c
`-- atk_ms901m_uart.h
```

图 2.1.2.1 ATK-MS901M 模块驱动代码

2.1.2.1 ATK-MS901M 模块接口驱动

在图 2.1.2.1 中，atk_ms901m_uart.c 和 atk_ms901m_uart.h 是开发板与 ATK-MS901M 模块通讯而使用的 UART 驱动文件，关于 UART 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 UART 对应的章节。

2.1.2.2 ATK-MS901M 模块驱动

在图 2.1.2.1 中，atk_ms901m.c 和 atk_ms901m.h 是 ATK-MS901M 模块的驱动文件，包含了 ATK-MS901M 模块初始化、操作、读写寄存器的相关 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_ms901m_init()

该函数用于初始化 ATK-MS901M 模块，具体的代码，如下所示：

```
/**
 * @brief   ATK-MS901M 初始化
 * @param   buadrate: ATK-MS901M UART 通讯波特率
 * @retval  ATK_MS901M_EOK      : ATK-MS901M 初始化成功
 *          ATK_MS901M_ERROR    : ATK-MS901M 初始化失败
 */
uint8_t atk_ms901m_init(uint32_t baudrate)
{
    uint8_t ret;

    /* ATK-MS901M UART 初始化 */
    atk_ms901m_uart_init(baudrate);
```

```
    /* 获取 ATK-MS901M 陀螺仪满量程 */
    ret = atk_ms901m_read_reg_by_id(    ATK_MS901M_FRAME_ID_REG_GYROFSR,
                                        &g_atk_ms901m_fsr.gyro,
                                        100);
    if (ret == 0)
    {
        return ATK_MS901M_ERROR;
    }

    /* 获取 ATK-MS901M 加速度计满量程 */
    ret = atk_ms901m_read_reg_by_id(    ATK_MS901M_FRAME_ID_REG_ACCFSR,
                                        &g_atk_ms901m_fsr.accelerometer,
                                        100);
    if (ret == 0)
    {
        return ATK_MS901M_ERROR;
    }

    return ATK_MS901M_EOK;
}
```

从上面的代码中可以看出，函数 atk_ms901m_init()会先初始化与 ATK-MS901M 模块通讯的 UART，然后再获取 ATK-MS901M 模块陀螺仪以及加速度计的满量程数据，这两个数据在计算 ATK-MS901M 陀螺仪以及加速度计数据的时候需要使用到，并且，通过这两个数据是否能够成功获取，来判断与 ATK-MS901M 模块的通讯是否有误。

2. 函数 atk_ms901m_read_reg_by_id()

该函数用于通过 ATK-MS901M 模块 UART 通讯的指定通讯帧 ID 来获取 ATK-MS901M 模块的寄存器值，具体的代码，如下所示：

```
/**
 * @brief   通过帧 ID 读取 ATK-MS901M 寄存器
 * @param   id      : 寄存器对应的通讯帧 ID
 *          dat     : 读取到的数据
 *          timeout : 等待数据的最大超时时间，单位：毫秒（ms）
 * @retval  0       : 读取失败
 *          其他值   : 读取到数据的长度
 */
uint8_t atk_ms901m_read_reg_by_id(uint8_t id, uint8_t *dat, uint32_t timeout)
{
    uint8_t buf[7];
    uint8_t ret;
    atk_ms901m_frame_t frame = {0};
    uint8_t dat_index;

    buf[0] = ATK_MS901M_FRAME_HEAD_L;
```

```
    buf[1] = ATK_MS901M_FRAME_HEAD_ACK_H;
    buf[2] = ATK_MS901M_READ_REG_ID(id);
    buf[3] = 1;
    buf[4] = 0;
    buf[5] = buf[0] + buf[1] + buf[2] + buf[3] + buf[4];
    atk_ms901m_uart_send(buf, 6);
    ret = atk_ms901m_get_frame_by_id(   &frame,
                                        id,
                                        ATK_MS901M_FRAME_ID_TYPE_ACK,
                                        timeout);
    if (ret != ATK_MS901M_EOK)
    {
        return 0;
    }

    for (dat_index=0; dat_index<frame.len; dat_index++)
    {
        dat[dat_index] = frame.dat[dat_index];
    }

    return frame.len;
}
```

从以上的代码中可以看出，该函数会先以据 ATK-MS901M 模块 UART 通讯的帧格式，将待发送的数据打包成帧，然后将通讯帧通过 UART 发送至 ATK-MS901M 模块，接着再调用函数 atk_ms901m_get_frame_by_id()获取 ATK-MS901M 模块回传的指定帧 ID 帧，最后就是解析接收到的数据帧，从而获取到需要的数据。

3. 函数 atk_ms901m_write_reg_by_id()

该函数用于通过 ATK-MS901M 模块 UART 通讯的指定通讯帧 ID 来写 ATK-MS901M 模块的寄存器，具体的代码，如下所示：

```
/**
 * @brief   通过帧 ID 写入 ATK-MS901M 寄存器
 * @param   id  : 寄存器对应的通讯帧 ID
 *          len : 待写入数据长度（1 或 2）
 *          dat : 待写入的数据
 * @retval  ATK_MS901M_EOK      : 寄存器写入成功
 *          ATK_MS901M_EINVAL   : 函数参数 len 有误
 */
uint8_t atk_ms901m_write_reg_by_id(uint8_t id, uint8_t len, uint8_t *dat)
{
    uint8_t buf[7];

    buf[0] = ATK_MS901M_FRAME_HEAD_L;
    buf[1] = ATK_MS901M_FRAME_HEAD_ACK_H;
    buf[2] = ATK_MS901M_WRITE_REG_ID(id);
```

```
    buf[3] = len;
    if (len == 1)
    {
        buf[4] = dat[0];
        buf[5] = buf[0] + buf[1] + buf[2] + buf[3] + buf[4];
        atk_ms901m_uart_send(buf, 6);
    }
    else if (len == 2)
    {
        buf[4] = dat[0];
        buf[5] = dat[1];
        buf[6] = buf[0] + buf[1] + buf[2] + buf[3] + buf[4] + buf[5];
        atk_ms901m_uart_send(buf, 7);
    }
    else
    {
        return ATK_MS901M_EINVAL;
    }

    return ATK_MS901M_EOK;
}
```

从以上的代码中可以看出，该函数会先以据 ATK-MS901M 模块 UART 通讯的帧格式，将待发送的数据打包成帧，然后将通讯帧通过 UART 发送至 ATK-MS901M 模块。

4. 函数 atk_ms901m_get_attitude()

该函数用于获取 ATK-MS901M 模块主动上报的姿态角数据，具体的代码，如下所示：

```
/**
 * @brief   获取 ATK-MS901M 姿态角数据
 * @param   attitude_dat    : 姿态角数据结构体
 *          timeout         : 获取数据最大等待超时时间，单位：毫秒（ms）
 * @retval  ATK_MS901M_EOK      : 获取 ATK-MS901M 姿态角数据成功
 *          ATK_MS901M_ERROR    : 获取 ATK-MS901M 姿态角数据失败
 */
uint8_t atk_ms901m_get_attitude(    atk_ms901m_attitude_data_t *attitude_dat,
                                    uint32_t timeout)
{
    uint8_t ret;
    atk_ms901m_frame_t frame = {0};

    if (attitude_dat == NULL)
    {
        return ATK_MS901M_ERROR;
    }

    ret = atk_ms901m_get_frame_by_id(   &frame,
```

```
                                        ATK_MS901M_FRAME_ID_ATTITUDE,
                                        ATK_MS901M_FRAME_ID_TYPE_UPLOAD,
                                        timeout);
    if (ret != ATK_MS901M_EOK)
    {
        return ATK_MS901M_ERROR;
    }

    attitude_dat->roll = (float)((int16_t)(frame.dat[1] << 8) | frame.dat[0])
                         / 32768 * 180;
    attitude_dat->pitch = (float)((int16_t)(frame.dat[3] << 8) | frame.dat[2])
                         / 32768 * 180;
    attitude_dat->yaw = (float)((int16_t)(frame.dat[5] << 8) | frame.dat[4])
                         / 32768 * 180;

    return ATK_MS901M_EOK;
}
```

从上面的代码中可以看出，该函数同 ATK-MS901M 模块主动上报的数据帧中获取姿态角数据的数据帧，然后将获取到的原始数据进行解析，从而获取实际的姿态角数据。

5. 函数 atk_ms901m_get_gyro_accelerometer()

该函数用于获取 ATK-MS901M 模块主动上报的陀螺仪数据和加速度计数据，具体的代码，如下所示：

```
/**
 * @brief   获取 ATK-MS901M 陀螺仪、加速度计数据
 * @param   gyro_dat            : 陀螺仪数据结构体
 *          accelerometer_dat   : 加速度计数据结构体
 *          timeout             : 获取数据最大等待超时时间，单位：毫秒（ms）
 * @retval  ATK_MS901M_EOK      : 获取 ATK-MS901M 陀螺仪、加速度计数据成功
 *          ATK_MS901M_ERROR    : 获取 ATK-MS901M 陀螺仪、加速度计数据失败
 */
uint8_t atk_ms901m_get_gyro_accelerometer(
    atk_ms901m_gyro_data_t *gyro_dat,
    atk_ms901m_accelerometer_data_t *accelerometer_dat,
    uint32_t timeout)
{
    uint8_t ret;
    atk_ms901m_frame_t frame = {0};

    if ((gyro_dat == NULL) && (accelerometer_dat == NULL))
    {
        return ATK_MS901M_ERROR;
    }

    ret = atk_ms901m_get_frame_by_id(   &frame,
```

```
                                        ATK_MS901M_FRAME_ID_GYRO_ACCE,
                                        ATK_MS901M_FRAME_ID_TYPE_UPLOAD,
                                        timeout);
    if (ret != ATK_MS901M_EOK)
    {
        return ATK_MS901M_ERROR;
    }

    if (gyro_dat != NULL)
    {
        gyro_dat->raw.x = (int16_t)(frame.dat[7] << 8) | frame.dat[6];
        gyro_dat->raw.y = (int16_t)(frame.dat[9] << 8) | frame.dat[8];
        gyro_dat->raw.z = (int16_t)(frame.dat[11] << 8) | frame.dat[10];

        gyro_dat->x = (float)gyro_dat->raw.x / 32768 *
                      g_atk_ms901m_gyro_fsr_table[g_atk_ms901m_fsr.gyro];
        gyro_dat->y = (float)gyro_dat->raw.y / 32768 *
                      g_atk_ms901m_gyro_fsr_table[g_atk_ms901m_fsr.gyro];
        gyro_dat->z = (float)gyro_dat->raw.z / 32768 *
                      g_atk_ms901m_gyro_fsr_table[g_atk_ms901m_fsr.gyro];
    }

    if (accelerometer_dat != NULL)
    {
        accelerometer_dat->raw.x = (int16_t)(frame.dat[1] << 8) | frame.dat[0];
        accelerometer_dat->raw.y = (int16_t)(frame.dat[3] << 8) | frame.dat[2];
        accelerometer_dat->raw.z = (int16_t)(frame.dat[5] << 8) | frame.dat[4];

        accelerometer_dat->x = (float)accelerometer_dat->raw.x / 32768 *
        g_atk_ms901m_accelerometer_fsr_table[g_atk_ms901m_fsr.accelerometer];
        accelerometer_dat->y = (float)accelerometer_dat->raw.y / 32768 *
        g_atk_ms901m_accelerometer_fsr_table[g_atk_ms901m_fsr.accelerometer];
        accelerometer_dat->z = (float)accelerometer_dat->raw.z / 32768 *
        g_atk_ms901m_accelerometer_fsr_table[g_atk_ms901m_fsr.accelerometer];
    }

    return ATK_MS901M_EOK;
}
```

从上面的代码中可以看出，该函数同 ATK-MS901M 模块主动上报的数据帧中获取陀螺仪和加速度计的数据帧，然后将获取到的原始数据进行解析，从而获取实际的陀螺仪和加速度计数据。

6. 函数 atk_ms901m_get_magnetometer()

该函数用于获取 ATK-MS901M 模块主动上报的磁力计数据，具体的代码，如下所示：

```
/**
```

```
 * @brief   获取 ATK-MS901M 磁力计数据
 * @param   magnetometer_dat    : 磁力计数据结构体
 *          timeout             : 获取数据最大等待超时时间，单位：毫秒（ms）
 * @retval  ATK_MS901M_EOK      : 获取 ATK-MS901M 磁力计数据成功
 *          ATK_MS901M_ERROR    : 获取 ATK-MS901M 磁力计数据失败
 */
uint8_t atk_ms901m_get_magnetometer
    atk_ms901m_magnetometer_data_t *magnetometer_dat,
    uint32_t timeout)
{
    uint8_t ret;
    atk_ms901m_frame_t frame = {0};

    if (magnetometer_dat == NULL)
    {
        return ATK_MS901M_ERROR;
    }

    ret = atk_ms901m_get_frame_by_id(   &frame,
                                        ATK_MS901M_FRAME_ID_MAG,
                                        ATK_MS901M_FRAME_ID_TYPE_UPLOAD,
                                        timeout);
    if (ret != ATK_MS901M_EOK)
    {
        return ATK_MS901M_ERROR;
    }

    magnetometer_dat->x = (int16_t)(frame.dat[1] << 8) | frame.dat[0];
    magnetometer_dat->y = (int16_t)(frame.dat[3] << 8) | frame.dat[2];
    magnetometer_dat->z = (int16_t)(frame.dat[5] << 8) | frame.dat[4];
    magnetometer_dat->temperature =
                (float)((int16_t)(frame.dat[7] << 8) | frame.dat[6]) / 100;

    return ATK_MS901M_EOK;
}
```

从上面的代码中可以看出，该函数同 ATK-MS901M 模块主动上报的数据帧中获取磁力计的数据帧，然后将获取到的原始数据进行解析，从而获取实际的磁力计数据。

7. 函数 atk_ms901m_get_barometer()

该函数用于获取 ATK-MS901M 模块主动上报的气压计数据，具体的代码，如下所示：

```
/**
 * @brief   获取 ATK-MS901M 气压计数据
 * @param   barometer_dat   : 气压计数据结构体
 *          timeout         : 获取数据最大等待超时时间，单位：毫秒（ms）
 * @retval  ATK_MS901M_EOK      : 获取 ATK-MS901M 气压计数据成功
```

```
 *          ATK_MS901M_ERROR    : 获取 ATK-MS901M 气压计数据失败
 */
uint8_t atk_ms901m_get_barometer(
    atk_ms901m_barometer_data_t *barometer_dat,
    uint32_t timeout)
{
    uint8_t ret;
    atk_ms901m_frame_t frame = {0};

    if (barometer_dat == NULL)
    {
        return ATK_MS901M_ERROR;
    }

    ret = atk_ms901m_get_frame_by_id(   &frame,
                                        ATK_MS901M_FRAME_ID_BARO,
                                        ATK_MS901M_FRAME_ID_TYPE_UPLOAD,
                                        timeout);
    if (ret != ATK_MS901M_EOK)
    {
        return ATK_MS901M_ERROR;
    }

    barometer_dat->pressure = (int32_t) (frame.dat[3] << 24) |
                                        (frame.dat[2] << 16) |
                                        (frame.dat[1] << 8) |
                                        frame.dat[0];
    barometer_dat->altitude = (int32_t) (frame.dat[7] << 24) |
                                        (frame.dat[6] << 16) |
                                        (frame.dat[5] << 8) |
                                        frame.dat[4];
    barometer_dat->temperature =
                (float)((int16_t)(frame.dat[9] << 8) | frame.dat[8]) / 100;

    return ATK_MS901M_EOK;
}
```

从上面的代码中可以看出，该函数同 ATK-MS901M 模块主动上报的数据帧中获取气压计的数据帧，然后将获取到的原始数据进行解析，从而获取实际的气压计数据。

2.1.2.3 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;

    /* 初始化 ATK-MS901M */
    ret = atk_ms901m_init(115200);
    if (ret != 0)
    {
        printf("ATK-MS901M init failed!\r\n");
        while (1)
        {
            LED0_TOGGLE();
            delay_ms(200);
        }
    }

    printf("ATK-MS901M init success!\r\n\n");

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

    while (1)
    {
        key = key_scan(0);

        switch (key)
        {
            case KEY0_PRES:
            {
                /* 获取并显示 ATK-MS901 数据 */
                demo_key0_fun();
                break;
            }
            default:
            {
                break;
            }
        }

        delay_ms(10);
    }
```

```
}
```

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，主要就是先调用函数 atk_ms901m_init()对 ATK-MS901M 模块及其相关组件进行初始化，然后就是每当检测到按键 0 被按下时，就调用一次函数 demo_key0_fun()获取 ATK-MS901M 模块的部分数据并将其显示到串口小时助手和 LCD 上。

2.1.3 实验现象

将 ATK-MS901M 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 ATK-MS901M 模块初始化成功，并且此时开发板连接 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

加下来按下按键 0，来获取 ATK-MS901M 模块主动上报的数据，并将其显示到串口调试助手和 LCD 上，LCD 上显示的内容，如下图所示：

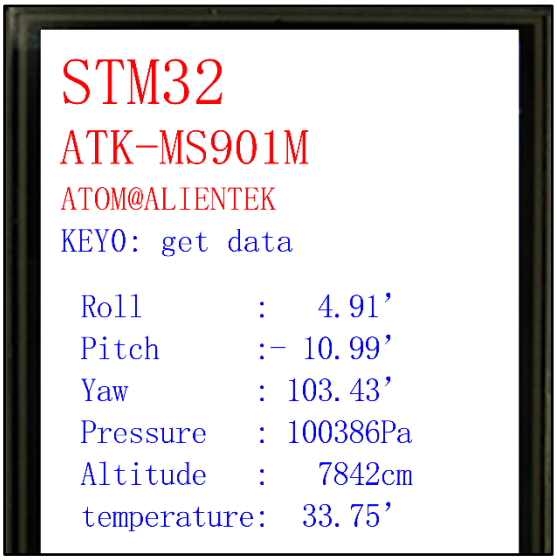

图 2.1.3.3 LCD 显示内容二

同时，串口调试助手上显示的内容，如下图所示：

图 2.1.3.4 串口调试助手显示内容二

可以看到，当按下按键 0 之后，串口调试助手和 LCD 上显示了 ATK-MS901M 模块传感器采集到的部分数据信息，其中就包括了姿态角、陀螺仪、加速度计、磁力计、气压计等传感器的数据。

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-IMU901.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