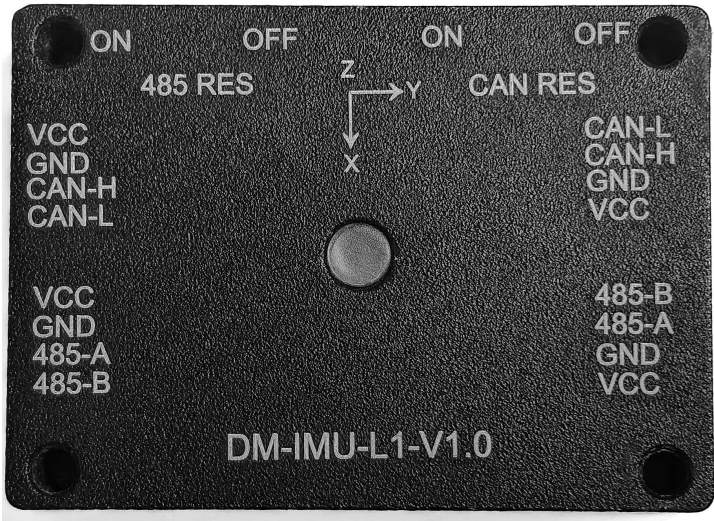


DAMIAO | 达妙科技

DM-IMU-L1 六轴 IMU 模块

使用说明书 V1.0 2025.03.16



日期	版本	变更内容
2025.03.16	V1.0	初版创建

目 录

免责声明	2
注意事项	2
简 介	2
参数指标	3
1、姿态角度指标	3
2、加速度计指标	3
3、陀螺仪指标	3
4、物理与电气指标	3
尺寸说明	4
坐标系定义及安装说明	4
1、坐标系定义	4
2、安装注意事项	5
通信及指令	5
1、USB 通信	5
2、RS485 通信	7
3、CAN 通信	7
固件升级	9
IMU 校准	10
1、陀螺仪校准	10
2、加速度计校准	11
附录一 IMU 灯语	14
附录二 线性映射表及转换函数	15
附录三 CAN 解析例程	17
附录四 CRC16 校验程序	20

免责声明

感谢您购买达妙科技 DAMIAO DM-IMU-L1 六轴 IMU（以下简称“IMU”）。在使用本产品之前，请仔细阅读并遵循本文及达妙科技提供的所有安全指引，否则可能会给您和周围的人带来伤害，损坏本产品或其他周围物品。一旦使用本产品，即视为您已经仔细阅读本文档，理解、认可和接受本文档及本产品所有相关文档的全部条款和内容。您承诺仅出于正当目的使用本产品。您承诺对使用本产品以及可能带来的后果负全部责任。达妙科技对于直接或间接使用本产品而造成的损坏、伤害以及任何法律责任不予负责。

DAMIAO 是深圳市达妙科技有限公司的商标。本文出现的产品名称、品牌等，均为其所属公司的商标。本产品及手册为深圳市达妙科技有限公司版权所有。未经许可，不得以任何形式复制翻印。本文档及本产品所有相关的文档最终解释权归深圳市达妙科技有限公司所有。如有更新，恕不另行通知。

注意事项

1. 请按照说明书将连接线正确连接后再使用，避免造成接口以及电路板损坏。
2. 请按照说明书建议的电压、电流、温度等工作环境使用，以免损坏，影响产品的使用寿命。
3. 使用前请检查各零部件是否完好。如有部件缺失、老化、损坏等，请停止使用。
4. 使用时做好防护，不要擅自拆开外壳暴露 PCBA，防止静电、物理损坏等；保持电路板干净整洁，避免异物等导致的短路或性能降低。
5. 电路板上电或使用过程中，出现打火、冒烟、烧焦味等异常情况时，请立即关掉电源。

简介

DM-IMU-L1 是一款内置 BMI088 三轴加速度计+三轴陀螺仪自带 EKF 四元数姿态解算 IMU 产品。主控频率高达 200MHz，支持 USB、RS485、CAN 协议输出，支持输出加速度、角速度、欧拉角、四元数等数据，数据输出频率高达 1kHz。传感器芯片底部采用孤岛结构设计，提升 IMU 抗震性能。传感器芯片周围带有加热电路，可实现恒温控制，有效减少陀螺仪温漂，无全温标定。

主要特性：

- ◆ 支持 CAN、USB、RS485 协议输出
- ◆ 支持 100-1000 输出频率可调
- ◆ 自带加热电路可实现恒温控制
- ◆ 内部 EKF 算法姿态解算

参数指标

1、姿态角度指标

参数		典型值	(校准后)一小时零漂
横滚角	量程	$\pm 180^\circ$	0.02°
航向角	量程	$\pm 180^\circ$	12.686°
俯仰角	量程	$\pm 90^\circ$	0.064°

备注：测试条件均为室温 25°C

2、加速度计指标

参数	典型值
量程	$\pm 6\text{G}$
自由度	3
非线性度	$\pm 0.5\% \text{FS}$
零点偏移	20mg
带宽	230Hz

3、陀螺仪指标

参数	典型值
量程	$\pm 2000^\circ/\text{s}$
自由度	3
非线性度	$\pm 0.05\% \text{FS}$
零点偏移	$\pm 1^\circ/\text{s}$
带宽	116Hz

4、物理与电气指标

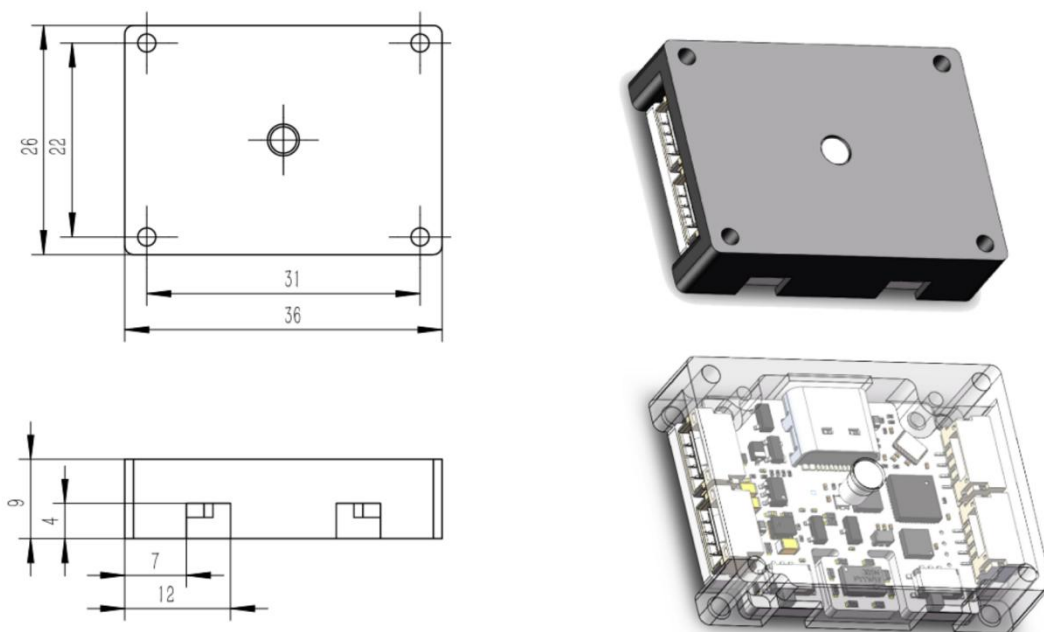
参数	典型值
输入电压	5-28V
功耗	@24V 温升时 2.5W 稳定时 0.47W
通信接口	USB、RS485、CAN
输出频率	100-1000Hz 可调
尺寸	36x26x9mm
重量	约 11.5g

*Tips:

1、加速度角速度指标来自 BMI088 数据手册,本 IMU 只做数据校准不做滤波处理,直接输出校准后的值,更具体的参数请移至 MEMS 芯片手册。

2、输入电压为 5V 时,功耗约为 0.34W。

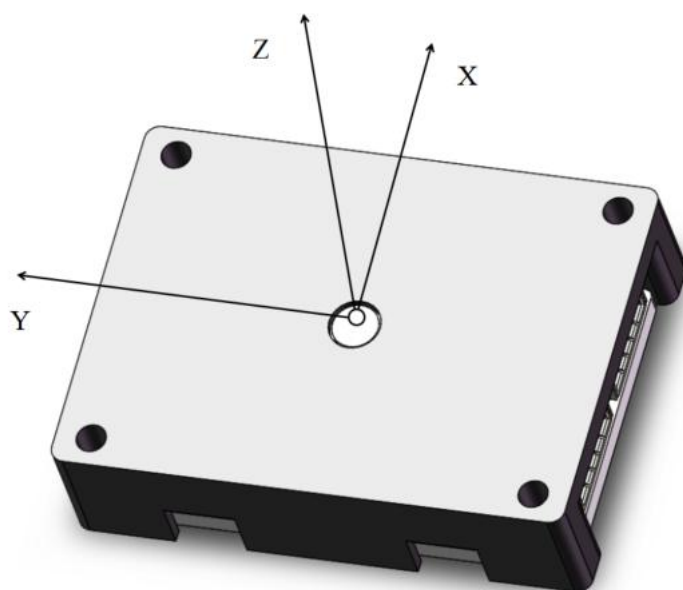
尺寸说明



坐标系定义及安装说明

1、坐标系定义

DM-IMU-L1 的坐标系定义如下图。绕 X 轴旋转为横滚角，绕 Y 轴旋转为俯仰角，绕 Z 轴旋转为航向角。



2、安装注意事项

- (1) 注意螺纹孔尺寸以及螺纹深度。
- (2) 尽量远离震动源。
- (3) 避免导光柱上方有挤压物，防止压坏 PCB 板。

通信及指令

1、USB 通信

IMU 数据会通过 USB 接口持续以用户设定的发送频率发送。默认会发送角速度、加速度以及欧拉角数据，暂时无法发送四元数数据，后续可通过固件升级来解锁相应功能。

■ USB 控制指令

指令	指令说明
0xAA 0x00 0x00 0x0D	重启 IMU
0xAA 0x01 0x04 0x0D	关闭加速度数据
0xAA 0x01 0x05 0x0D	关闭角速度数据
0xAA 0x01 0x06 0x0D	关闭欧拉角数据
0xAA 0x01 0x07 0x0D	关闭四元数数据
0xAA 0x01 0x14 0x0D	开启加速度数据
0xAA 0x01 0x15 0x0D	开启角速度数据
0xAA 0x01 0x16 0x0D	开启欧拉角数据
0xAA 0x01 0x17 0x0D	开启四元数数据
0xAA 0x02 0x01 0x0D	设置输出频率为 100Hz
0xAA 0x02 0x02 0x0D	设置输出频率为 125Hz
0xAA 0x02 0x03 0x0D	设置输出频率为 200Hz
0xAA 0x02 0x04 0x0D	设置输出频率为 250Hz
0xAA 0x02 0x05 0x0D	设置输出频率为 500Hz
0xAA 0x02 0x06 0x0D	设置输出频率为 1000Hz
0xAA 0x04 0x00 0x0D	关闭恒温控制
0xAA 0x04 0x01 0x0D	开启恒温控制
0xAA 0x05 温度值 0x0D	设置恒温温度(最高 60°)
0xAA 0x06 0x00 0x0D	退出设置模式
0xAA 0x06 0x01 0x0D	进入设置模式
0xAA 0x08 CANID 0x0D	设置 CANID (从机 ID)
0xAA 0x08 MSTID 0x0D	设置 MSTID (主机 ID)
0xAA 0x0A 通信类型 0x0D	设置通信类型
0xAA 0x03 0x01 0x0D	保存参数

*Tips:

- 1、设置恒温温度值需进行 10 进制到 16 进制转换后发送；
- 2、切换通信类型 0x00:USB 0x01:RS485 0x01:CAN；

■ USB 数据帧格式

①加速度数据包格式

数据域	数据内容	备注
DATA[0]	0x55	帧头
DATA[1]	0xAA	标志号
DATA[2]	从机 ID	指令设置的 CANID
DATA[3]	0x01	寄存器 ID
DATA[4]	Ax_L1	X 轴加速度低 8 位
DATA[5]	Ax_L2	X 轴加速度低 16 位
DATA[6]	Ax_H1	X 轴加速度高 24 位
DATA[7]	Ax_H2	X 轴加速度高 32 位
DATA[8]	Ay_L1	Y 轴加速度低 8 位
DATA[9]	Ay_L2	Y 轴加速度低 16 位
DATA[10]	Ay_H1	Y 轴加速度高 24 位
DATA[11]	Ay_H2	Y 轴加速度高 32 位
DATA[12]	Az_L1	Z 轴加速度低 8 位
DATA[13]	Az_L2	Z 轴加速度低 16 位
DATA[14]	Az_H1	Z 轴加速度高 24 位
DATA[15]	Az_H2	Z 轴加速度高 32 位
DATA[16]	CRC16_L	CRC16 校验低八位
DATA[17]	CRC16_H	CRC16 校验高八位
DATA[18]	0x0A	帧尾

②角速度数据包格式

数据域	数据内容	备注
DATA[0]	0x55	帧头
DATA[1]	0xAA	标志号
DATA[2]	从机 ID	指令设置的 CANID
DATA[3]	0x02	寄存器 ID
DATA[4]	Gx_L1	X 轴角速度低 8 位
DATA[5]	Gx_L2	X 轴角速度低 16 位
DATA[6]	Gx_H1	X 轴角速度高 24 位
DATA[7]	Gx_H2	X 轴角速度高 32 位
DATA[8]	Gy_L1	Y 轴角速度低 8 位
DATA[9]	Gy_L2	Y 轴角速度低 16 位
DATA[10]	Gy_H1	Y 轴角速度高 24 位
DATA[11]	Gy_H2	Y 轴角速度高 32 位
DATA[12]	Gz_L1	Z 轴角速度低 8 位
DATA[13]	Gz_L2	Z 轴角速度低 16 位
DATA[14]	Gz_H1	Z 轴角速度高 24 位
DATA[15]	Gz_H2	Z 轴角速度高 32 位
DATA[16]	CRC16_L	CRC16 校验低八位
DATA[17]	CRC16_H	CRC16 校验高八位
DATA[18]	0x0A	帧尾

③欧拉角数据包格式

数据域	数据内容	备注
DATA[0]	0x55	帧头
DATA[1]	0xAA	标志号
DATA[2]	从机 ID	指令设置的 CANID
DATA[3]	0x03	寄存器 ID
DATA[4]	Ax_L1	X 轴加速度低 8 位
DATA[5]	Ax_L2	X 轴加速度低 16 位
DATA[6]	Ax_H1	X 轴加速度高 24 位
DATA[7]	Ax_H2	X 轴加速度高 32 位
DATA[8]	Ay_L1	Y 轴加速度低 8 位
DATA[9]	Ay_L2	Y 轴加速度低 16 位
DATA[10]	Ay_H1	Y 轴加速度高 24 位
DATA[11]	Ay_H2	Y 轴加速度高 32 位
DATA[12]	Az_L1	Z 轴加速度低 8 位
DATA[13]	Az_L2	Z 轴加速度低 16 位
DATA[14]	Az_H1	Z 轴加速度高 24 位
DATA[15]	Az_H2	Z 轴加速度高 32 位
DATA[16]	CRC16_L	CRC16 校验低八位
DATA[17]	CRC16_H	CRC16 校验高八位
DATA[18]	0x0A	帧尾

*Tips:

- 1、读取到的拼接数据要做指针类型数据转换，不然无法得到真实的浮点数。
- 2、角速度单位为 rad/s，加速度单位为 m/s^2 ，角度单位为 $^{\circ}$ ，使用时请注意单位的转化。
- 3、CRC16 的校验数据域为 DATA[0]至 DATA[15]，CRC 校验程序请务必使用本说明书附录里的代码。

2、RS485 通信

485 通信协议的控制指令和数据帧格式参考上文 USB 通信内容。485 波特率默认为 921600（暂时无法修改，后续通过更新固件开放更改），私有协议不支持 ModBus。

3、CAN 通信

CAN 通信波特率默认为 1Mbps（当前固件版本无法调整 CAN 总线波特率）。数据获取使用一发一收模式，用户通过使用特定的数据报文去获取对应数据寄存器里的内容。本模式可输出四元数数据，由于 CAN 协议的特殊性，IMU 数据都做线性映射处理，精度有所降低。

■ CAN 数据获取说明

CANID	0x6FF
数据帧类型	标准帧
DLC	4
DATA[0]	CANID_L
DATA[1]	CANID_H
DATA[2]	RID
DATA[3]	0xCC

寄存器列表

RID	寄存器说明
0x01	加速度数据
0x02	角速度数据
0x03	欧拉角数据
0x04	四元数数据

■ 3、CAN 反馈报文格式

以下数据帧反馈报文无特殊说明均为标准帧、数据帧。

①加速度数据

报文 ID	MST ID
DLC	8
DATA[0]	REG=1
DATA[1]	温度
DATA[2]	Ax_L
DATA[3]	Ax_H
DATA[4]	Ay_L
DATA[5]	Ay_H
DATA[6]	Az_L
DATA[7]	Az_H

②角速度数据

报文 ID	MST ID
DLC	8
DATA[0]	REG=2
DATA[1]	默认为 0
DATA[2]	Gx_L
DATA[3]	Gx_H
DATA[4]	Gy_L
DATA[5]	Gy_H
DATA[6]	Gz_L
DATA[7]	Gz_H

③欧拉角数据

报文 ID	MST ID
DLC	8
DATA[0]	REG=3
DATA[1]	默认为 0
DATA[2]	Pitch_L
DATA[3]	Pitch_H
DATA[4]	Yaw_L
DATA[5]	Yaw_H
DATA[6]	Roll_L
DATA[7]	Roll_H

④四元数数据

四元数每项数据都为 14 位线性映射值

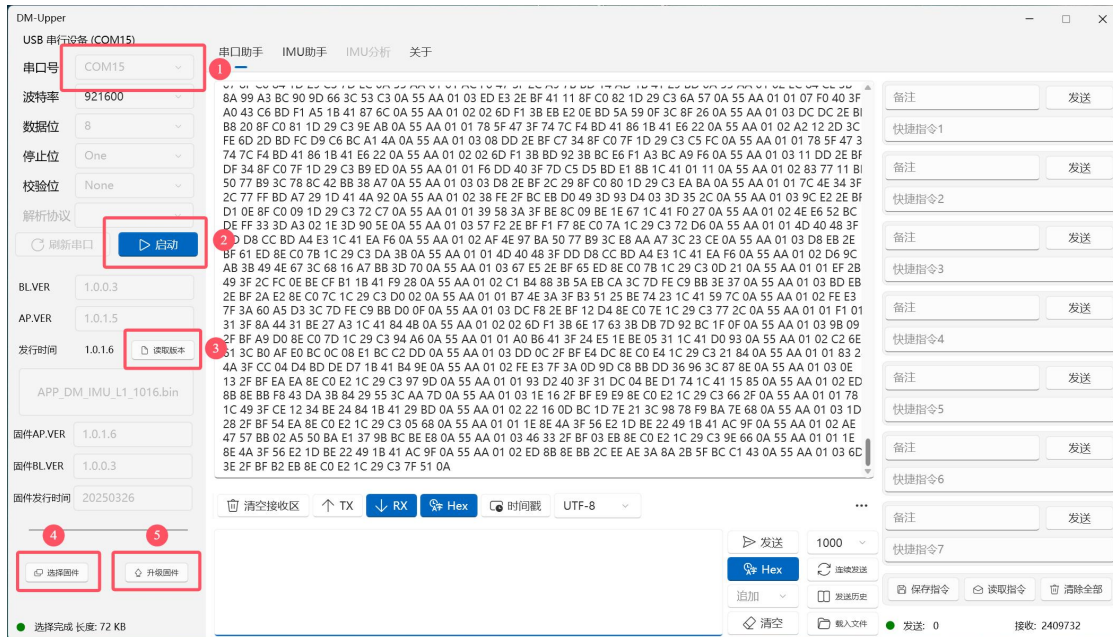
报文 ID	MST ID
DLC	8
DATA[0]	Reg=4
DATA[1]	W[13:6]
DATA[2]	W[5:0]&X[13:12]
DATA[3]	X[11:4]
DATA[4]	X[3:0]&Y[13:10]
DATA[5]	Y[9:2]
DATA[6]	Y[1:0]&Z[13:8]
DATA[7]	Z[7:0]

*Tips:

除了四元数数据，其他数据均为 16 位映射值（范围为[-32768,32768]），四元数数据为 14 位映射值（范围为[-8192,8192]）。可以注意到的是，加速度数据中带有有一个温度值，此值为真实值并非映射值。具体映射函数及转换例程可参考附录内容。

固件升级

IMU 固件升级通过使用 DM-IMU-Upper 上位机软件进行升级，下面所示流程为上位机升级固件步骤。



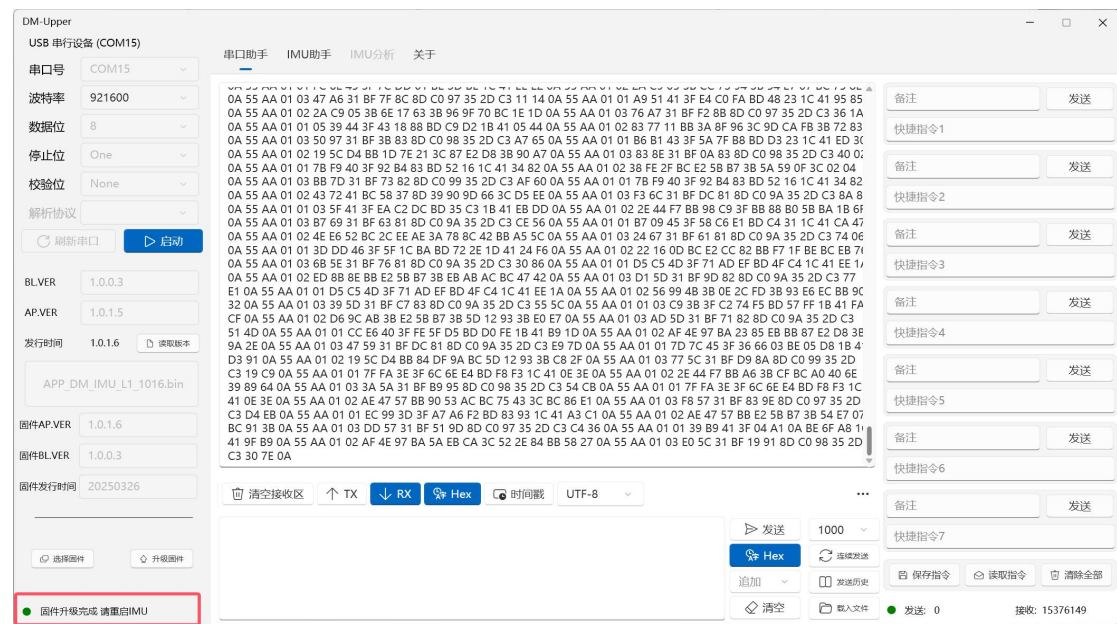
第一步：选中合适的串口；

第二步：点击启动按钮即可打开串口；

第三步：点击读取版本按钮可以读取到当前的固件版本信息；

第四步：选择固件.bin 后缀的文件；

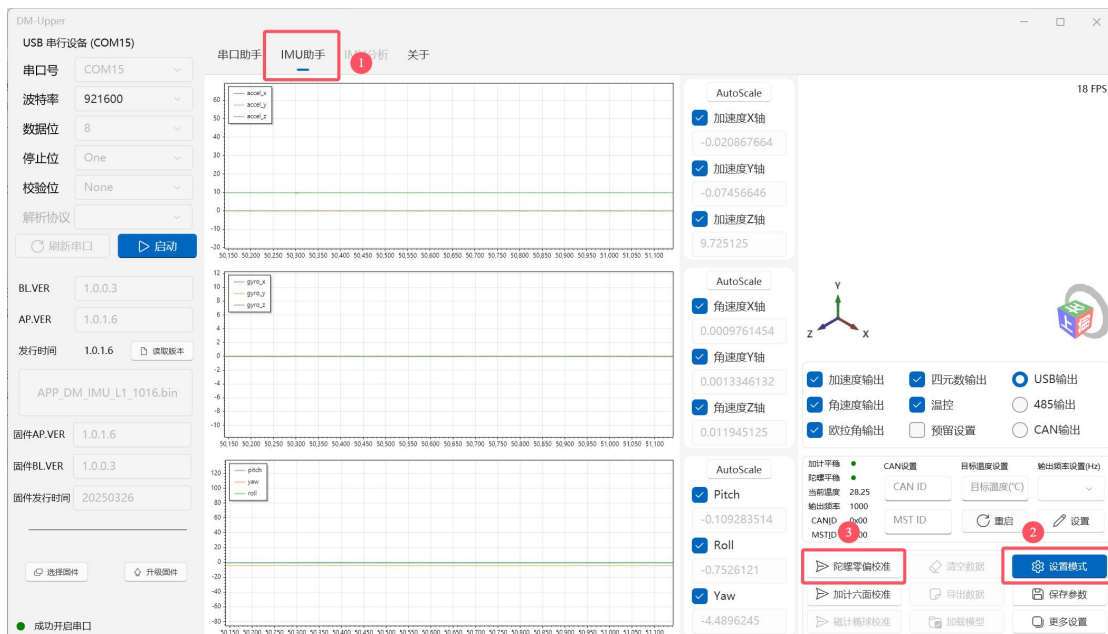
第五步：点击升级固件，等待进度条读取完毕；



当左下角出现“固件升级完成 请重启 IMU”字样后，重新拔插 TypeC 数据线进行 IMU 重启。重启后现象：IMU 绿灯闪烁三下后进入绿灯呼吸状态即完成升级。可点击读取版本确认版本是否升级完成。

IMU 校准

1、陀螺仪校准



第一步：点击标签页进入“IMU 助手”。

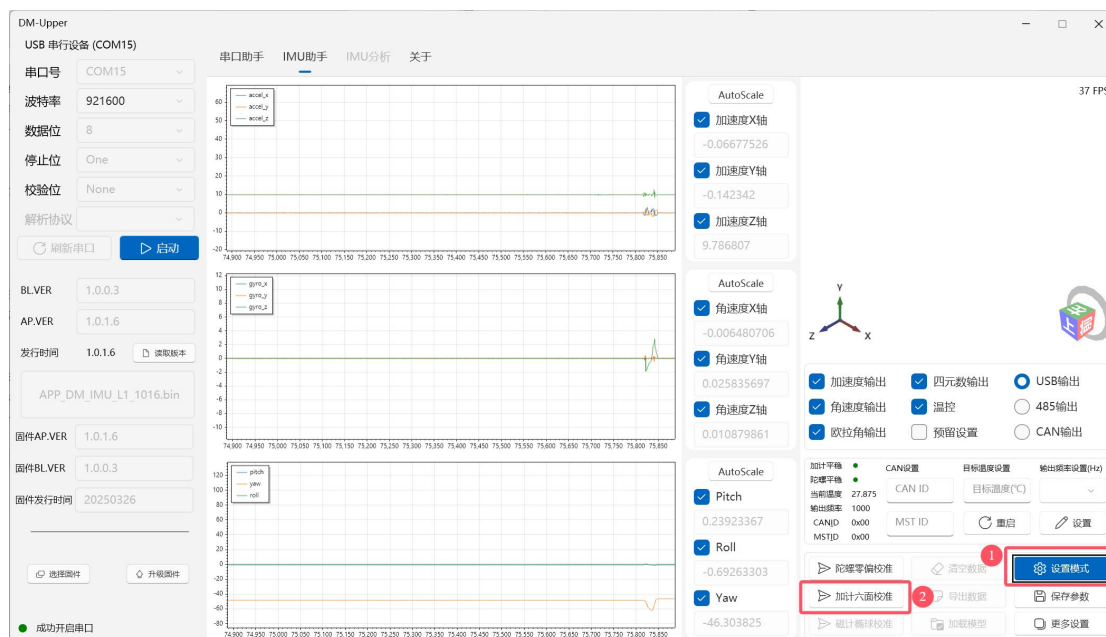
第二步：点击右下角进入设置模式，此时 IMU 指示灯变成呼吸黄灯。

第三步：点击陀螺零偏校准。

此时观察 IMU 指示灯会进行按间隔闪烁大概六次，校准完成后 IMU 会自动重启，然后指示灯变成呼吸绿灯，此时需要重新打开上位机串口。

2、加速度计校准

如果您刚刚校准过陀螺仪，请重新打开一下串口，以便进行加速度计的校准。

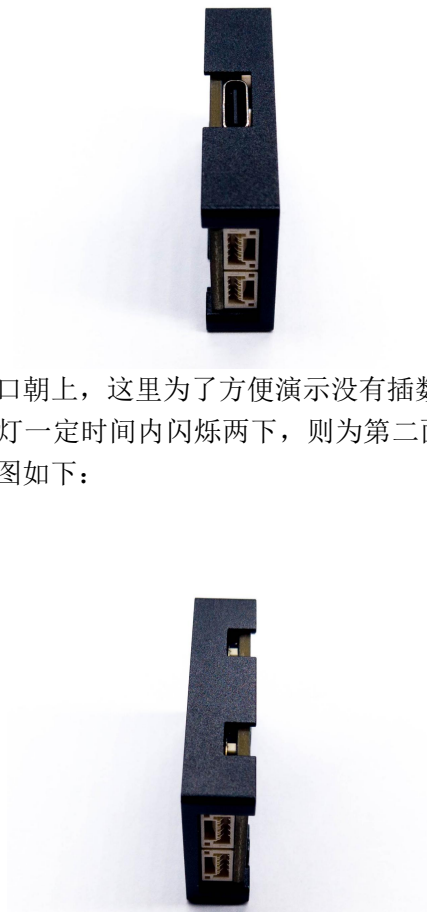


第一步：点击设置模式按钮，IMU 会进入设置模式。

第二步：点击加速计六面校准按钮。

第三步：观测 IMU 指示灯，如果在一段时间内闪烁一次则为第一面数据采样。

此时需要将 IMU 翻到第一面，第一面示例图如下：



可以观测到此时 C 口朝上，这里为了方便演示没有插数据线，实际校准时需要插线。

第四步：观测指示灯一定时间内闪烁两下，则为第二面数据采集，此时需要将 IMU 翻到第二面，第二面示例图如下：

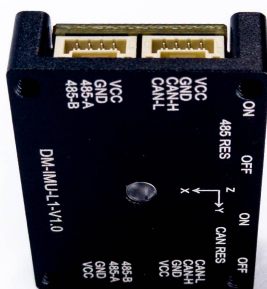
等待指示灯进行三下闪烁，则第二面数据采集完成。

第五步：观测指示灯一定时间内闪烁三下，则为第三面数据采集，此时需要将 IMU 翻到第三面，第三面示例图如下：



等待指示灯进行四下闪烁，则第三面数据采集完成。

第六步：观测指示灯一定时间内闪烁四下，则为第四面数据采集，此时需要将 IMU 翻到第四面，第四面示例图如下：



等待指示灯进行五下闪烁，则第四面数据采集完成。

第七步：观测指示灯一定时间内闪烁五下，则为第五面数据采集，此时需要将 IMU 翻到第五面，第五面示例图如下：



等待指示灯进行六下闪烁，则第五面数据采集完成。

第八步：观测指示灯一定时间内闪烁六下，则为第六面数据采集，此时需要将 IMU 翻面到第六面，第六面示例图如下：



等待第六面数据采集完成后，IMU 会进行重启，如需继续使用上位机请重新打开串口。

附录一 IMU 灯语

指示灯状态	状态描述
绿灯呼吸	正常模式
黄灯呼吸	设置模式
红灯常亮	错误状态
黄灯 1s 亮灭一次	陀螺仪校准中
黄灯间隔亮灭 1 次	加计六面校准一面数据采集中
黄灯间隔亮灭 2 次	加计六面校准二面数据采集中
黄灯间隔亮灭 3 次	加计六面校准三面数据采集中
黄灯间隔亮灭 4 次	加计六面校准四面数据采集中
黄灯间隔亮灭 5 次	加计六面校准五面数据采集中
黄灯间隔亮灭 6 次	加计六面校准六面数据采集中

附录二 线性映射表及转换函数

```

#define ACCEL_CAN_MAX (58.8f)
#define ACCEL_CAN_MIN (-58.8f)
#define GYRO_CAN_MAX (34.88f)
#define GYRO_CAN_MIN (-34.88f)
#define PITCH_CAN_MAX (90.0f)
#define PITCH_CAN_MIN (-90.0f)
#define ROLL_CAN_MAX (180.0f)
#define ROLL_CAN_MIN (-180.0f)
#define YAW_CAN_MAX (180.0f)
#define YAW_CAN_MIN (-180.0f)
#define TEMP_MIN (0.0f)
#define TEMP_MAX (60.0f)
#define Quaternion_MIN (-1.0f)
#define Quaternion_MAX (1.0f)

/**
*****
* @brief:      float_to_uint: 浮点数转换为无符号整数函数
* @param[in]:  x_float:      待转换的浮点数
* @param[in]:  x_min:        范围最小值
* @param[in]:  x_max:        范围最大值
* @param[in]:  bits:         目标无符号整数的位数
* @retval:      无符号整数结果
* @details:      将给定的浮点数 x 在指定范围 [x_min, x_max] 内进行线性映射, 映射结果为一个
指定位数的无符号整数
*****
**/
int float_to_uint(float x_float, float x_min, float x_max, int bits)
{
    /* Converts a float to an unsigned int, given range and number of bits */
    float span = x_max - x_min;
    float offset = x_min;
    return (int) ((x_float-offset)*((float)((1<<bits)-1))/span);
}

/**
*****
* @brief:      uint_to_float: 无符号整数转换为浮点数函数
* @param[in]:  x_int: 待转换的无符号整数
* @param[in]:  x_min: 范围最小值
* @param[in]:  x_max: 范围最大值
* @param[in]:  bits:  无符号整数的位数

```


* @retval: 浮点数结果
* @details: 将给定的无符号整数 `x_int` 在指定范围 `[x_min, x_max]` 内进行线性映射，映射结果为一个浮点数

*/

```
float uint_to_float(int x_int, float x_min, float x_max, int bits)
{
    /* converts unsigned int to float, given range and number of bits */
    float span = x_max - x_min;
    float offset = x_min;
    return ((float)x_int)*span/((float)((1<<bits)-1)) + offset;
}
```

附录三 CAN 解析例程

```

void IMU_RequestData(uint16_t can_id,uint8_t reg)
{
    CAN_TxHeaderTypeDef tx_header;
    uint8_t cmd[4]={(uint8_t)can_id,(uint8_t)(can_id>>8),reg,0xCC};
    uint32_t returnBox;
    tx_header.DLC=4;
    tx_header.IDE=CAN_ID_STD;
    tx_header.RTR=CAN_RTR_DATA;
    tx_header.StdId=0x6FF;

    if(HAL_CAN_GetTxMailboxesFreeLevel(&hcan2)>1)
    {
        HAL_CAN_AddTxMessage(&hcan2,&tx_header,cmd,&returnBox);
    }
}

void IMU_UpdateAccel(uint8_t* pData)
{
    uint16_t accel[3];

    accel[0]=pData[3]<<8|pData[2];
    accel[1]=pData[5]<<8|pData[4];
    accel[2]=pData[7]<<8|pData[6];

    imu.accel[0]=uint_to_float(accel[0],ACCEL_CAN_MIN,ACCEL_CAN_MAX,16);
    imu.accel[1]=uint_to_float(accel[1],ACCEL_CAN_MIN,ACCEL_CAN_MAX,16);
    imu.accel[2]=uint_to_float(accel[2],ACCEL_CAN_MIN,ACCEL_CAN_MAX,16);

}

void IMU_UpdateGyro(uint8_t* pData)
{
    uint16_t gyro[3];

    gyro[0]=pData[3]<<8|pData[2];
    gyro[1]=pData[5]<<8|pData[4];
    gyro[2]=pData[7]<<8|pData[6];

    imu.gyro[0]=uint_to_float(gyro[0],GYRO_CAN_MIN,GYRO_CAN_MAX,16);
    imu.gyro[1]=uint_to_float(gyro[1],GYRO_CAN_MIN,GYRO_CAN_MAX,16);
}

```

```

    imu.gyro[2]=uint_to_float(gyro[2],GYRO_CAN_MIN,GYRO_CAN_MAX,16);
}

```

```

void IMU_UpdateEuler(uint8_t* pData)
{
    int euler[3];

    euler[0]=pData[3]<<8|pData[2];
    euler[1]=pData[5]<<8|pData[4];
    euler[2]=pData[7]<<8|pData[6];

    imu.pitch=uint_to_float(euler[0],PITCH_CAN_MIN,PITCH_CAN_MAX,16);
    imu.yaw=uint_to_float(euler[1],YAW_CAN_MIN,YAW_CAN_MAX,16);
    imu.roll=uint_to_float(euler[2],ROLL_CAN_MIN,ROLL_CAN_MAX,16);
}

```

```

void IMU_UpdateQuaternion(uint8_t* pData)
{
    int w = pData[1]<<6| ((pData[2]&0xF8)>>2);
    int x = (pData[2]&0x03)<<12|(pData[3]<<4)|((pData[4]&0xF0)>>4);
    int y = (pData[4]&0x0F)<<10|(pData[5]<<2)|(pData[6]&0xC0)>>6;
    int z = (pData[6]&0x3F)<<8|pData[7];

    imu.q[0] = uint_to_float(w,Quaternion_MIN,Quaternion_MAX,14);
    imu.q[1] = uint_to_float(x,Quaternion_MIN,Quaternion_MAX,14);
    imu.q[2] = uint_to_float(y,Quaternion_MIN,Quaternion_MAX,14);
    imu.q[3] = uint_to_float(z,Quaternion_MIN,Quaternion_MAX,14);
}

```

```

void IMU_UpdateData(uint8_t* pData)
{
    switch(pData[0])
    {
        case 1:
            IMU_UpdateAccel(pData);
            break;
        case 2:
            IMU_UpdateGyro(pData);
            break;
        case 3:
            IMU_UpdateEuler(pData);

```

```
        break;
    case 4:
        IMU_UpdateQuaternion(pData);
        break;
    }
}
```

附录四 CRC16 校验程序

```
const uint16_t CRC16_table[256] =
{
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7, 0x8108, 0x9129,
    0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF,
    0x1231, 0x0210, 0x3273, 0x2252, 0x52B5, 0x4294, 0x72F7, 0x62D6, 0x9339, 0x8318,
    0xB37B, 0xA35A, 0xD3BD, 0xC39C, 0xF3FF, 0xE3DE,
    0x2462, 0x3443, 0x0420, 0x1401, 0x64E6, 0x74C7, 0x44A4, 0x5485, 0xA56A, 0xB54B,
    0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D,
    0x3653, 0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4, 0xB75B, 0xA77A,
    0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D, 0xC7BC,
    0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861, 0x2802, 0x3823, 0xC9CC, 0xD9ED,
    0xE98E, 0xF9AF, 0x8948, 0x9969, 0xA90A, 0xB92B,
    0x5AF5, 0x4AD4, 0x7AB7, 0x6A96, 0x1A71, 0x0A50, 0x3A33, 0x2A12, 0xDBFD, 0xCBDC,
    0xFBBF, 0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A,
    0x6CA6, 0x7C87, 0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41, 0xEDAE, 0xFD8F,
    0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B, 0x8D68, 0x9D49,
    0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32, 0x1E51, 0x0E70, 0xFF9F, 0xEFBE,
    0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A, 0x9F59, 0x8F78,
    0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C, 0xC12D, 0xF14E, 0xE16F, 0x1080, 0x00A1,
    0x30C2, 0x20E3, 0x5004, 0x4025, 0x7046, 0x6067,
    0x83B9, 0x9398, 0xA3FB, 0xB3DA, 0xC33D, 0xD31C, 0xE37F, 0xF35E, 0x02B1, 0x1290,
    0x22F3, 0x32D2, 0x4235, 0x5214, 0x6277, 0x7256,
    0xB5EA, 0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F, 0xD52C, 0xC50D, 0x34E2, 0x24C3,
    0x14A0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
    0xA7DB, 0xB7FA, 0x8799, 0x97B8, 0xE75F, 0xF77E, 0xC71D, 0xD73C, 0x26D3, 0x36F2,
    0x0691, 0x16B0, 0x6657, 0x7676, 0x4615, 0x5634,
    0xD94C, 0xC96D, 0xF90E, 0xE92F, 0x99C8, 0x89E9, 0xB98A, 0xA9AB, 0x5844, 0x4865,
    0x7806, 0x6827, 0x18C0, 0x08E1, 0x3882, 0x28A3,
    0xCB7D, 0xDB5C, 0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8, 0xABBB, 0xBB9A, 0x4A75, 0x5A54,
    0x6A37, 0x7A16, 0x0AF1, 0x1AD0, 0x2AB3, 0x3A92,
    0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B, 0x9DE8, 0x8DC9, 0x7C26, 0x6C07,
    0x5C64, 0x4C45, 0x3CA2, 0x2C83, 0x1CE0, 0x0CC1,
    0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B, 0xBFBA, 0x8FD9, 0x9FF8, 0x6E17, 0x7E36,
    0x4E55, 0x5E74, 0x2E93, 0x3EB2, 0x0ED1, 0x1EF0
};
```

```
uint16_t Get_CRC16(uint8_t *ptr, uint16_t len)
{
    uint16_t crc = 0xFFFF;
    for (size_t i = 0; i < len; ++i)
    {
        uint8_t index = (crc >> 8 ^ ptr[i]);
        crc = ((crc << 1) ^ CRC16_table[index]);
    }
    return crc;
}
```