



1 产品概述

MIM02 模块内置 TDK 高精度 6 轴 IMU(惯性测量单元) 传感器芯片, 通过处理器读取传感器数据, 并经过内部复杂运算后通过串口输出加速度, 角速度, 角度等数据, 大大减轻了用户开发难度和工作量。同时精心的 PCB 布局和工艺保证了模块收到外接的干扰最小, 测量的精度最高。

采用先进的数字滤波技术, 能有效降低测量噪声, 提高测量精度。

模块内部集成了姿态解算器, 配合动态卡尔曼滤波算法, 能够在动态环境下准确输出模块的当前姿态, 姿态测量精度 0.01 度, 稳定性极高, 性能甚至优于某些专业的倾角仪!

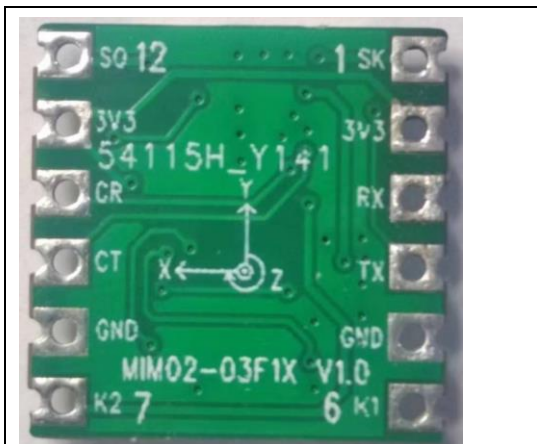
MIM02 模块适用于机器人、机械臂、智能小车、无人机、扫地机器人、游戏手柄、智能头盔、虚拟直播、空鼠、翻页笔、电动牙刷、美容仪、惯性导航、智能乐器及角度检测等领域。

2 性能参数

- 1) 供电电压 VDD: $3.3 \pm 0.2V$ I/O 高电平/低电平: $>0.7VDD / <0.3VDD$
- 2) 电流: $<3.5mA$
- 3) 体积: $15mm \times 15.0mm \times 1.0mm$
- 4) 焊盘孔间距: $2.54mm$
- 5) 测量维度: 加速度: 3 维, 角速度: 3 维, 姿态角: 3 维
- 6) 量程: 加速度: $\pm 16g$, 角速度: $\pm 2000^{\circ} / s$
- 7) 分辨率: 加速度: $6.1 \times 10^{-5}g$, 角速度: $7.6 \times 10^{-3}^{\circ} / s$
- 8) 稳定性: 加速度: $0.01g$, 角速度 $0.05^{\circ} / s$
- 9) 姿态测量稳定度: 0.01°
- 10) 数据输出频率 $100Hz/20Hz$
- 11) 数据接口: 串口 (TTL 电平) 波特率 $115200bps/9600bps$



3 引脚说明



名称	功能
TX	串行数据发送，TTL 电平
RX	串行数据接收，TTL 电平
GND	地线
3V3	模块电源，接 3.3V
SK	厂家保留，悬空
SD	厂家保留，悬空
CT	厂家保留，悬空
CR	厂家保留，悬空

4 轴向说明

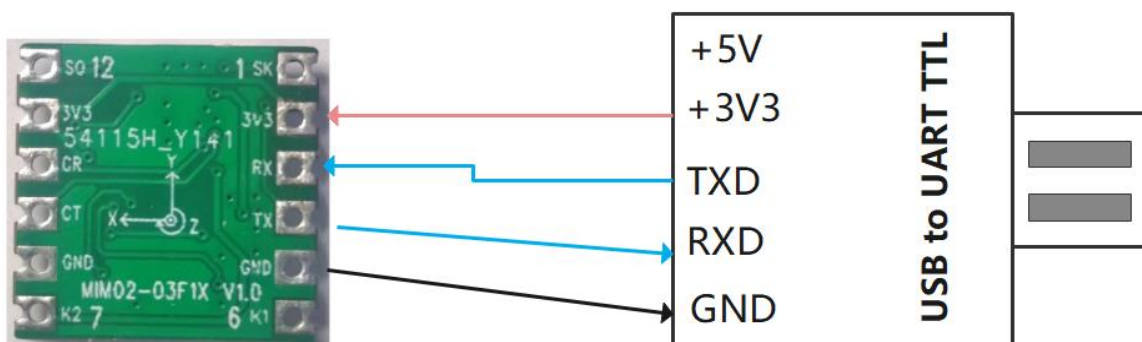
如上表格中图所示，模块的轴向在上图的右上角标示出来，向右为 X 轴，向上为 Y 轴，垂直与纸面向外为 Z 轴。旋转的方向按右手法则定义，即右手大拇指指向轴向，四指弯曲的方向即为绕该轴旋转的方向。



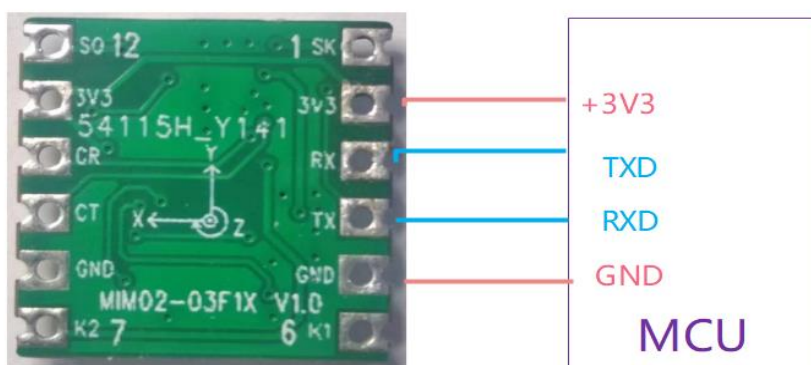
5 硬件连接方法

5.1 连计算机

与计算机连接，需要 USB 转 TTL 电平的串口模块。USB 串口模块连接 MIM02 模块的方法是：USB 串口模块的+3V3，TXD，RXD，GND 接 MIM02 模块的 3V3，RX，TX，GND，注意 TXD 和 RXD 的交叉。

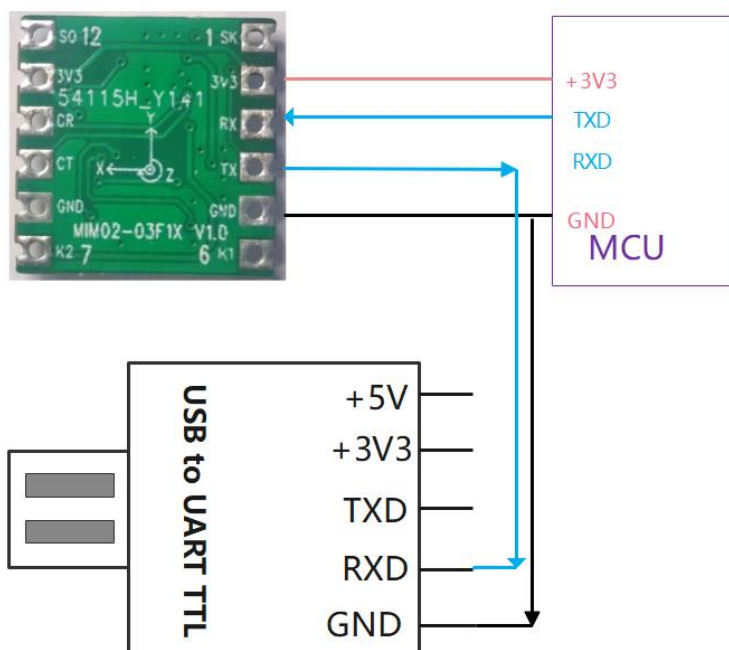


5.2 连单片机



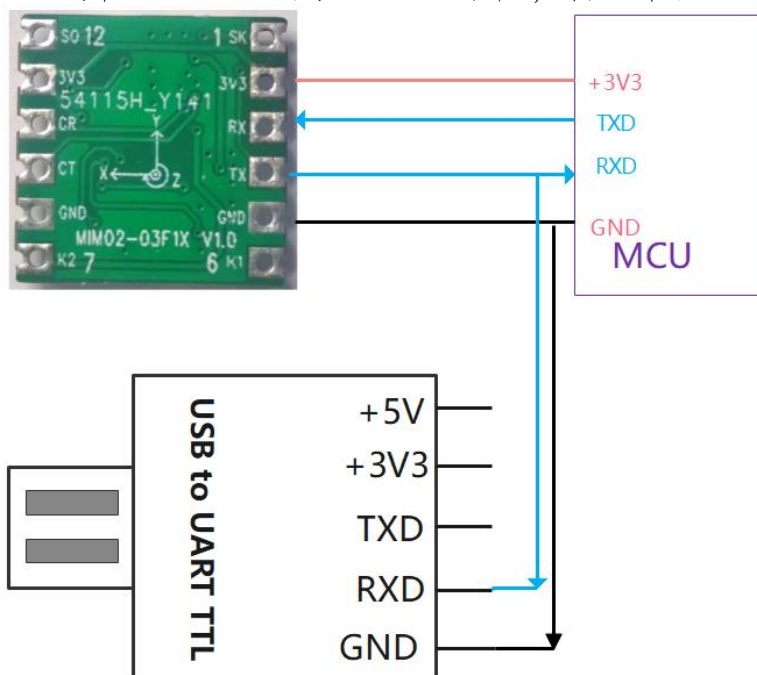


5.3 MCU 连单片机并输出调试信息。



5.4 用上位机监视模块与单片机的通信。

如果需要在 MCU 接受 MIM02 模块的输出数据的同时，用上位机监视当前的数据，可以将 USB 转串口模块的 RX 接到模块的 TX 引脚上，并共地即可。





6 通信协议

电平: TTL 电平(非 RS232 电平, 若将模块错接到 RS232 电平可能造成模块损坏)波特率:
115200bps/9600bps, 停止位 1, 校验位 0。

6.1 上位机至模块

指令内容	功能	备注
0xFF 0xAA 0x52	航向角清零	使Z轴角度归零, 掉电不保存
0xFF 0xAA 0x63	波特率115200bps, 帧率100Hz	缺省值, 掉电保存
0xFF 0xAA 0x64	波特率9600bps, 帧率20Hz	掉电保存
0xFF 0xAA 0x65	模块 X 轴为机头方向	掉电保存
0xFF 0xAA 0x66	模块 Y 轴为机头方向	缺省值, 掉电保存
0xFF 0xAA 0x67	零偏校准	掉电保存

说明:

出厂默认设置使用串口, 波特率115200bps, 帧率100Hz。配置可通过上位机软件配置, 因为所有配置都是掉电保存的, 所以只需配置一次就行。

6.2 模块至上位机:

模块发送至上位机每帧数据分为 3 个数据包, 分别为加速度包、角速度包和角度包, 3 个数据包顺序输出。波特率 115200bps 时每隔 10ms 输出 1 帧数据, 波特率 9600bps 时每隔 50ms 输出一帧数据。

6.2.1 加速度输出:

数据编号	数据内容	含义
0	0x55	包头
1	0x51	标识这个包是加速度包
2	AxL	x 轴加速度低字节
3	AxH	x 轴加速度高字节
4	AyL	y 轴加速度低字节
5	AyH	y 轴加速度高字节
6	AzL	z 轴加速度低字节
7	AzH	z 轴加速度高字节
8	TL	温度低字节
9	TH	温度高字节
10	Sum	校验和



加速度计算公式：

$$a_x = ((AxH < 8) | AxL) / 32768 * 16g \text{ (g 为重力加速度, 可取 } 9.8m/s^2)$$

$$a_y = ((AyH < 8) | AyL) / 32768 * 16g \text{ (g 为重力加速度, 可取 } 9.8m/s^2)$$

$$a_z = ((AzH < 8) | AzL) / 32768 * 16g \text{ (g 为重力加速度, 可取 } 9.8m/s^2)$$

温度计算公式：

$$T = ((TH < 8) | TL) / 128 + 25 \text{ } ^\circ C$$

校验和：

$$Sum = 0x55 + 0x51 + AxH + AxL + AyH + AyL + AzH + AzL + TH + TL$$

6.2.2 角速度输出：

数据编号	数据内容	含义
0	0x55	包头
1	0x52	标识这个包是角速度包
2	wxL	x 轴角速度低字节
3	wxH	x 轴角速度高字节
4	wyL	y 轴角速度低字节
5	wyH	y 轴角速度高字节
6	wzL	z 轴角速度低字节
7	wzH	z 轴角速度高字节
8	InftypeL	产品信息类型低字节
9	InftypeH	产品信息类型高字节
10	Sum	校验和

角速度计算公

式：

$$w_x = ((wxH < 8) | wxL) / 32768 * 2000 \text{ (} ^\circ / s)$$

$$w_y = ((wyH < 8) | wyL) / 32768 * 2000 \text{ (} ^\circ / s)$$

$$w_z = ((wzH < 8) | wzL) / 32768 * 2000 \text{ (} ^\circ / s)$$

产品信息类型计算公式：Inftype = ((InftypeH < 8) | InftypeL)

Inftype 取值范围为 0--3，具体取值含义如下：

0. 当前输出信息类型为型号
1. 当前输出信息类型为软件版本号
2. 当前输出信息类型为配置信息
3. 当前输出信息类型校准状态信息

校验和：

$$Sum = 0x55 + 0x52 + wxH + wxL + wyH + wyL + wzH + wzL + InftypeH + InftypeL$$



6.2.3 角度输出：

数据编号	数据内容	含义
0	0x55	包头
1	0x53	标识这个包是角度包
2	RollL	x 轴角度低字节
3	RollH	x 轴角度高字节
4	PitchL	y 轴角度低字节
5	PitchH	y 轴角度高字节
6	YawL	z 轴角度低字节
7	YawH	z 轴角度高字节
8	InfvalueL	产品信息值低字节
9	InfvalueH	产品信息值高字节
10	Sum	校验和

角度计算公式：

滚转角(x 轴) $Roll = ((RollH \ll 8) | RollL) / 32768 * 180(^{\circ})$

俯仰角(y 轴) $Pitch = ((PitchH \ll 8) | PitchL) / 32768 * 180(^{\circ})$

偏航角(z 轴) $Yaw = ((YawH \ll 8) | YawL) / 32768 * 180(^{\circ})$

产品信息值计算公式：

$Infvalue = ((InfvalueH \ll 8) | InfvalueL)$

当 $Inftype == 0$ 时，即当前输出信息类型为型号时， $Infvalue$ 为模块型号代码。

当 $Inftype == 1$ 时，即当前输出信息类型为软件版本号， $Infvalue$ 为模块软件版本号。

当 $Inftype == 2$ 时，即当前输出信息类型为配置信息，

$InfvalueL$ 为串口配置信息；

$InfvalueH$ 为机体坐标系配置信息。

$InfvalueL$ 取值范围如下：

$InfvalueL == 2$ ：串口波特率为 115200，

$InfvalueL == 10$ ：串口波特率为 9600；

$InfvalueH$ 取值范围如下：

$InfvalueH == 0$ ：X 轴正方向为机头方向，

$InfvalueH == 1$ ：Y 轴正方向为机头方向；



当 $\text{InfType}=3$ 时,即当前输出信息类型校准信息 InfValue 为陀螺仪零偏校准状态信息。
取值范围如下:

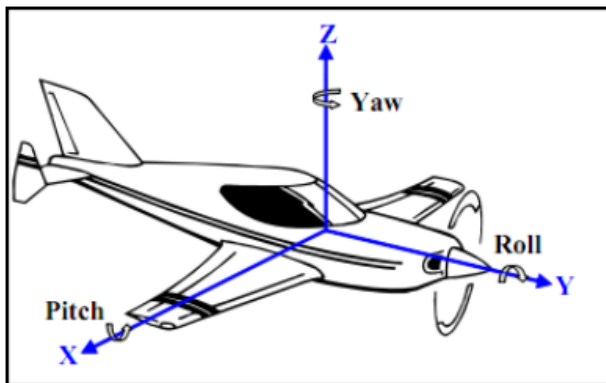
$\text{InfValue}=0$: 无校准操作;
 $\text{InfValue}=1$: 正在校准;
 $\text{InfValue}=2$: 校准完成;

校验和:

$\text{Sum}=0x55+0x53+\text{RollH}+\text{RollL}+\text{PitchH}+\text{PitchL}+\text{YawH}+\text{YawL}+\text{InfValueH}+\text{InfValueL}$

注:

1. 姿态角解算时所使用的坐标系为东北天坐标系, 正方向放置模块, 如下图所示向左为 X 轴, 向前为 Y 轴, 向上为 Z 轴。欧拉角表示姿态时的坐标系旋转顺序定义为 Z-X-Y, 即先绕 Z 轴转, 再绕 X 轴转, 再绕 Y 轴转。



2. 滚转角范围是 $\pm 180^\circ$ 度, 由于坐标旋转顺序是 Z-X-Y, 在表示姿态的时候, 俯仰角(上位机的 Y 轴角度)的范围只有 $\pm 90^\circ$ 度, 超过 90° 度后会变换到小于 90° 度, 同时让翻滚角(上位机的 X 轴角度)的角度变成负角度。详细原理请大家自行百度欧拉角及姿态表示的相关信息。
3. 由于三轴是耦合的, 只有在小角度的时候会表现出独立变化, 在大角度的时候姿态角度会耦合变化, 比如当俯仰角(上位机的 Y 轴角度)接近 90° 度时, 即使姿态只绕 X 轴转动, 即只改变俯仰角, 翻滚角(上位机的 X 轴角度)和偏航角(上位机的 Z 轴角度)也会跟着发生较大变化, 这是欧拉角表示姿态的固有问题。



6.3 数据解析示例代码

```
double a[3],w[3],Angle[3],T;

void DecodeIMUData(unsigned char chrTemp[])
{
    switch(chrTemp[1])
    {
        case 0x51:
            a[0] = (short(chrTemp[3]<<8|chrTemp[2]))/32768.0*16;
            a[1] = (short(chrTemp[5]<<8|chrTemp[4]))/32768.0*16;
            a[2] = (short(chrTemp[7]<<8|chrTemp[6]))/32768.0*16;
            break;
        case 0x52:
            w[0] = (short(chrTemp[3]<<8|chrTemp[2]))/32768.0*2000;
            w[1] = (short(chrTemp[5]<<8|chrTemp[4]))/32768.0*2000;
            w[2] = (short(chrTemp[7]<<8|chrTemp[6]))/32768.0*2000;
            break;
        case 0x53:
            Angle[0] = (short(chrTemp[3]<<8|chrTemp[2]))/32768.0*180;
            Angle[1] = (short(chrTemp[5]<<8|chrTemp[4]))/32768.0*180;
            Angle[2] = (short(chrTemp[7]<<8|chrTemp[6]))/32768.0*180;
            T = (short(chrTemp[9]<<8|chrTemp[8]))/128.0+25.0;
            printf("a = %4.3ft%4.3ft%4.3ft\r\n",a[0],a[1],a[2]);
            printf("w = %4.3ft%4.3ft%4.3ft\r\n",w[0],w[1],w[2]);
            printf("Angle = %4.2ft%4.2ft%4.2ftT=%4.2f\r\n",Angle[0],Angle[1],Angle[2],T);
            break;
    }
}
```

6.4 嵌入式环境下解析数据实例

分成两个部分，一个是中断接收，找到数据的头，然后把数据包放入数组中；另一个是数据解析，放在主程序中。中断部分（以下为 AVR 单片机代码，不同单片机读取寄存器略有差异，需根据实际情况调整）：

```
unsigned char Re_buf[11],counter=0;
unsigned char sign;

interrupt [USART_RXC] void usart_rx_isr(void) //USART 串行接收中断
{
    Re_buf[counter]=UDR;//不同单片机略有差异
```



```
if(counter==0&&Re_buf[0]!=0x55) return; //第 0 号数据不是帧头，跳过
```

```
counter++;
```

```
if(counter==11) //接收到 11 个数据
```

```
{
```

```
    counter=0; //重新赋值，准备下一帧数据的接收
```

```
    sign=1;
```

```
}
```

```
}
```

主程序部分：

```
float a[3],w[3],angle[3],T;
```

```
extern unsigned char Re_buf[11],counter;
```

```
extern unsigned char sign;
```

```
while(1)
```

```
{
```

```
    if(sign)
```

```
    {
```

```
        sign=0;
```

```
        if(Re_buf[0]==0x55) //检查帧头
```

```
        {
```

```
            switch(Re_buf [1])
```

```
            {
```

```
                case 0x51:
```

```
                    a[0] = (short(Re_buf [3]<<8| Re_buf [2]))/32768.0*16;
```

```
                    a[1] = (short(Re_buf [5]<<8| Re_buf [4]))/32768.0*16;
```

```
                    a[2] = (short(Re_buf [7]<<8| Re_buf [6]))/32768.0*16;
```

```
                    break;
```

```
                case 0x52:
```

```
                    w[0] = (short(Re_buf [3]<<8| Re_buf [2]))/32768.0*2000;
```

```
                    w[1] = (short(Re_buf [5]<<8| Re_buf [4]))/32768.0*2000;
```

```
                    w[2] = (short(Re_buf [7]<<8| Re_buf [6]))/32768.0*2000;
```

```
                    break;
```

```
                case 0x53:
```

```
                    angle[0] = (short(Re_buf [3]<<8| Re_buf [2]))/32768.0*180;
```

```
                    angle[1] = (short(Re_buf [5]<<8| Re_buf [4]))/32768.0*180;
```

```
                    angle[2] = (short(Re_buf [7]<<8| Re_buf [6]))/32768.0*180;
```

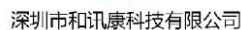
```
                    T = (short(Re_buf [9]<<8| Re_buf [8]))/128.0+25.0;
```

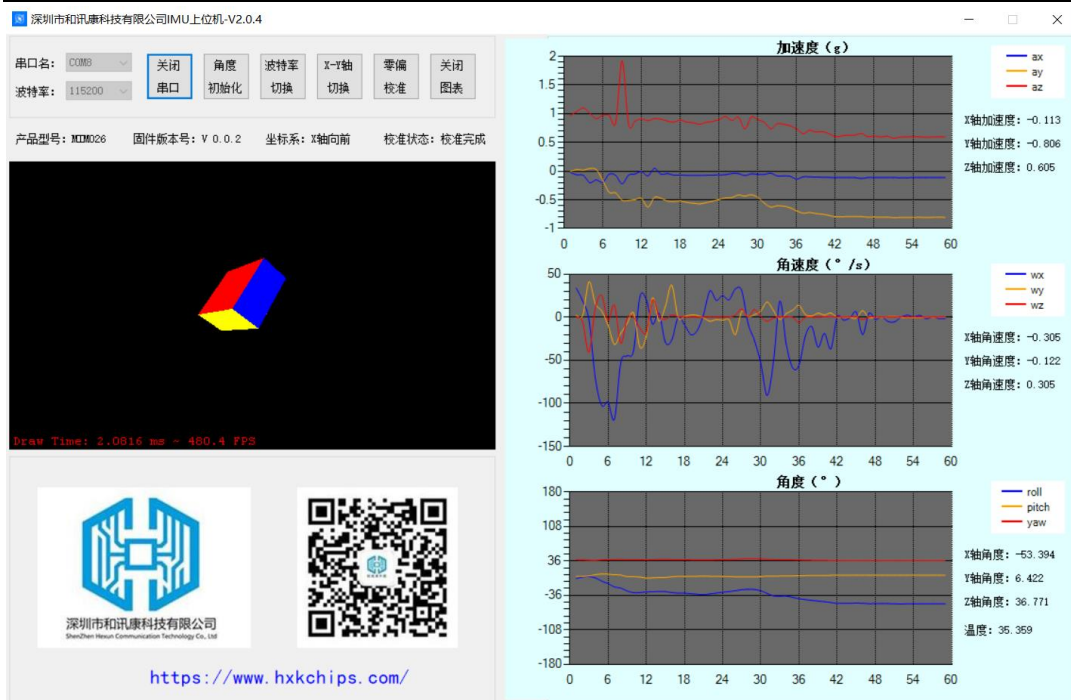
```
                    break;
```

```
            }
```

```
        }
```

```
    }
```





(4) 若发现上位机 3D 模型运动方向与 MIM02 模块不一致时, 可将 MIM02 模块的机头方向 (X 轴或 Y 轴的正方向) 对准屏幕并水平放置在桌面, 点击“角度初始化”按钮将航向角归零。

(5) 点击“X-Y 轴切换”按钮可将 MIM02 模块的机头方向在 X 轴正方向和 Y 轴正方向之间切换。

(6) 点击“零偏校准”按钮可对 MIM02 模块进行零偏校准, 模块会在按下按钮后 2 秒内进行校准, 校准过程中请将模块水平放置在桌面并保持静止。校准后的零偏数据存储在模块中, 只应用于模块姿态角的计算, 模块输出的加速度数据和角速度数据不会修正零偏, 还是 IMU 芯片输出的原始数据。

(7) 点击“关闭图表”按钮可将右侧数据曲线图表关闭, 关闭图表后上位机的 3D 模型跟随模块运动的显示会更加流畅。

8. 注意事项

注意: 本模块不含磁场计, 没有磁场的观测量对偏航角进行滤波, 所以偏航角度是通过纯积分计算出来的, 不可避免地会有漂移现象, 只能实现短时间内的旋转角度测量。而 X, Y 轴角度可以通过重力场进行滤波修正, 不会出现漂移现象。

