

## 高精度 6 轴惯性导航模块说明书

### 1 产品概述

此六轴模块采用高精度的陀螺加速度计 MPU6050，通过处理器读取 MPU6050 的测量数据然后通过串口输出，免去了用户自己去开发 MPU6050 复杂的 I2C 协议，同时精心的 PCB 布局和工艺保证了 MPU6050 收到外接的干扰最小，测量的精度最高。

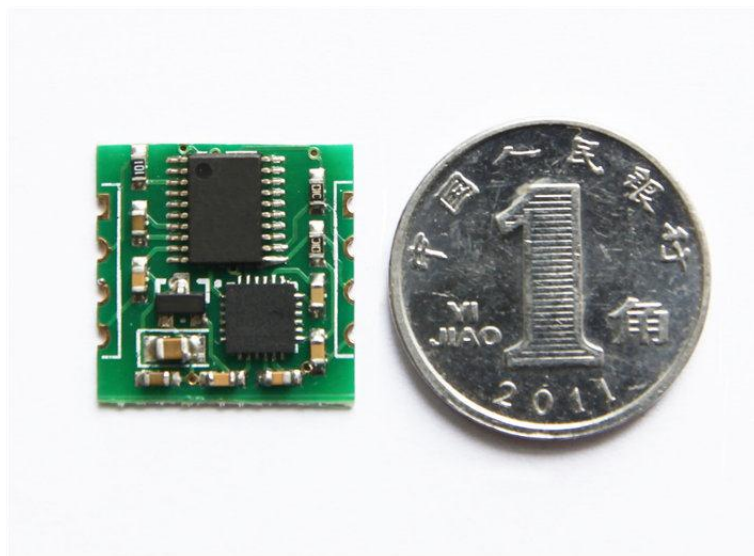
模块内部自带电压稳定电路，可以兼容 3.3V/5V 的嵌入式系统，连接方便。

模块保留了 MPU6050 的 I2C 接口，以满足高级用户希望访问底层测量数据的需求。

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！

采用邮票孔镀金工艺，品质保证，可嵌入用户的 PCB 板中。

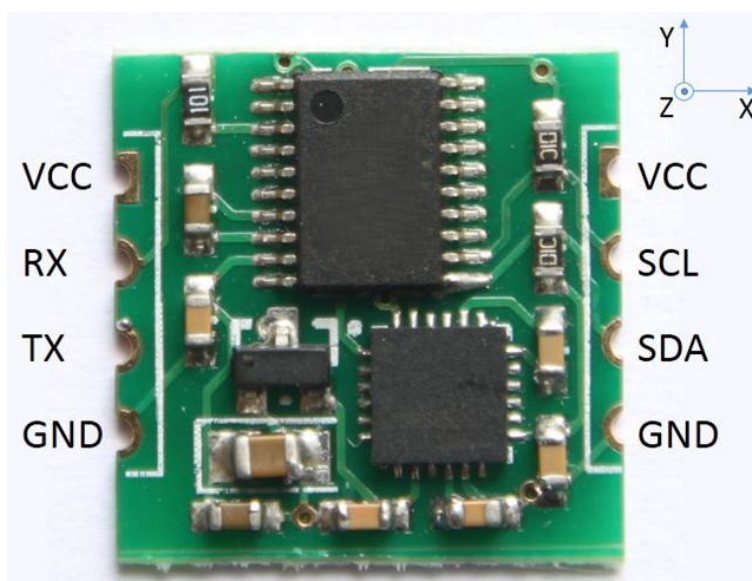


### 2 性能参数

- 1、电压：3V~6V
- 2、电流：<10mA
- 3、体积：15.24mm X 15.24mm X 2mm

- 4、焊盘间距：上下 100mil(2.54mm)，左右 600mil(15.24mm)
- 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： $6.1e-5g$ ，角速度： $7.6e-3^\circ/s$ 。
- 8、稳定性：加速度：0.01g，角速度  $0.05^\circ/s$ 。
- 9、姿态测量稳定度： $0.01^\circ$ 。
- 10、数据输出频率 100Hz(波特率 115200)/20Hz(波特率 9600)。
- 11、数据接口：串口（TTL 电平），I2C（直接连 MPU6050，无姿态输出）
- 10、波特率 115200kps/9600kps。

### 3 引脚说明：

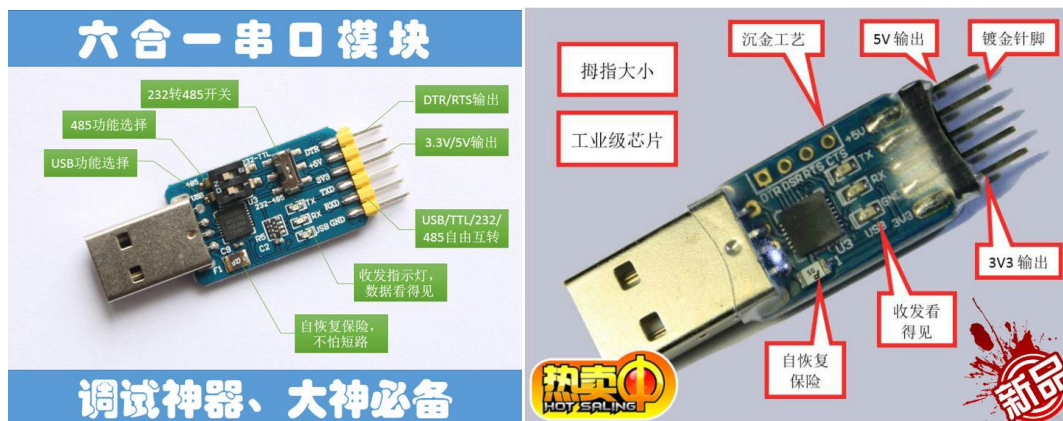


名称	功能
VCC	模块电源，3.3V 或 5V 输入
RX	串行数据输入，TTL 电平
TX	串行数据输出，TTL 电平
GND	地线
SCL	I2C 时钟线
SDA	I2C 数据线

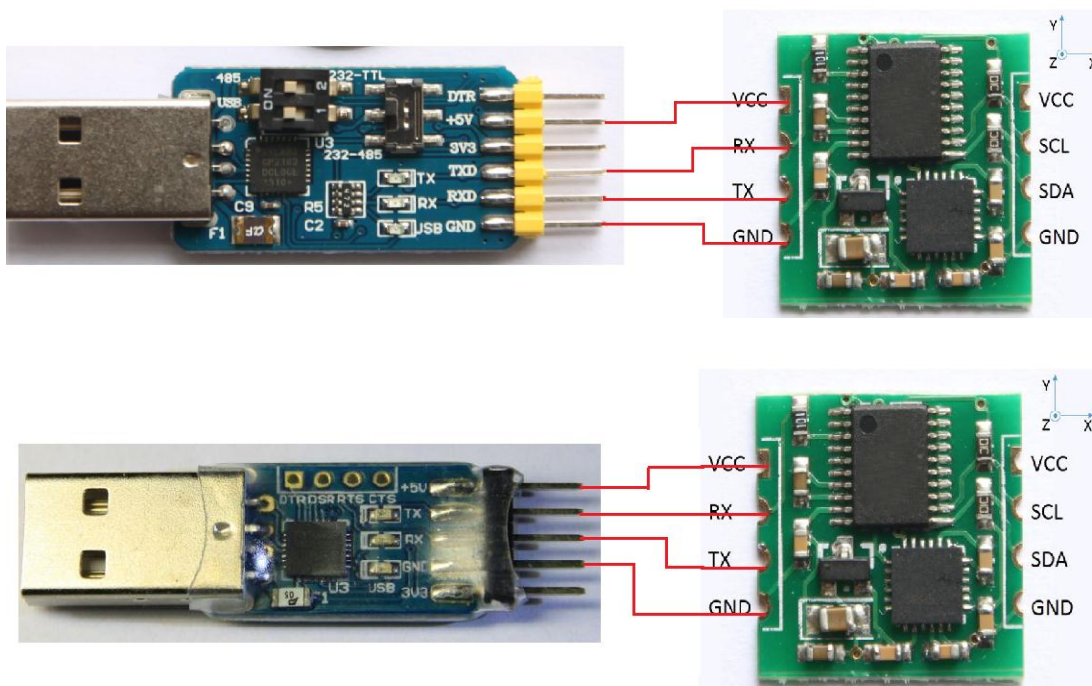
## 4 硬件连接方法

### 4.1 与计算机

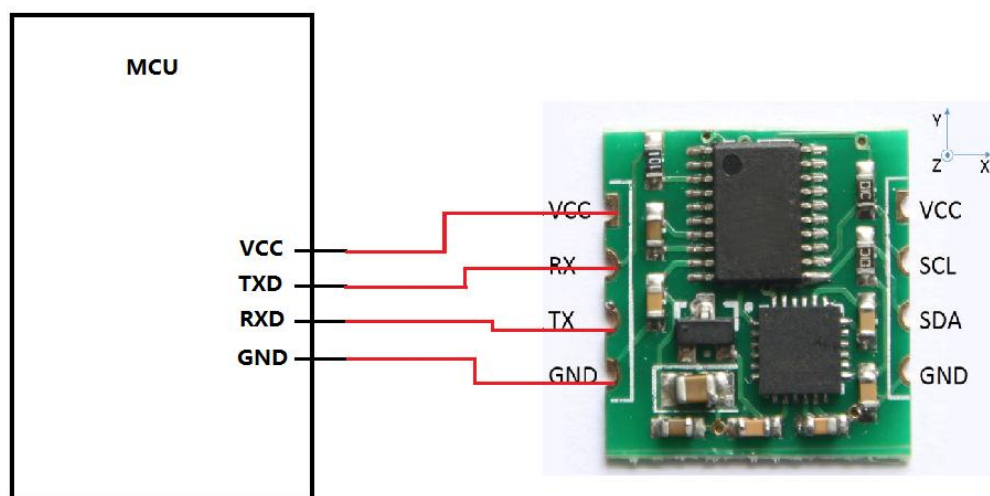
与计算机连接，需要 USB 转 TTL 电平的串口模块。推荐以下两款 USB 转串口模块。



USB 串口模块连接 6050 模块的方法是：USB 串口模块的+5V，TXD，RXD，GND 接 6050 模块的 VCC，RX，TX，GND。注意 TXD 和 RXD 的交叉。

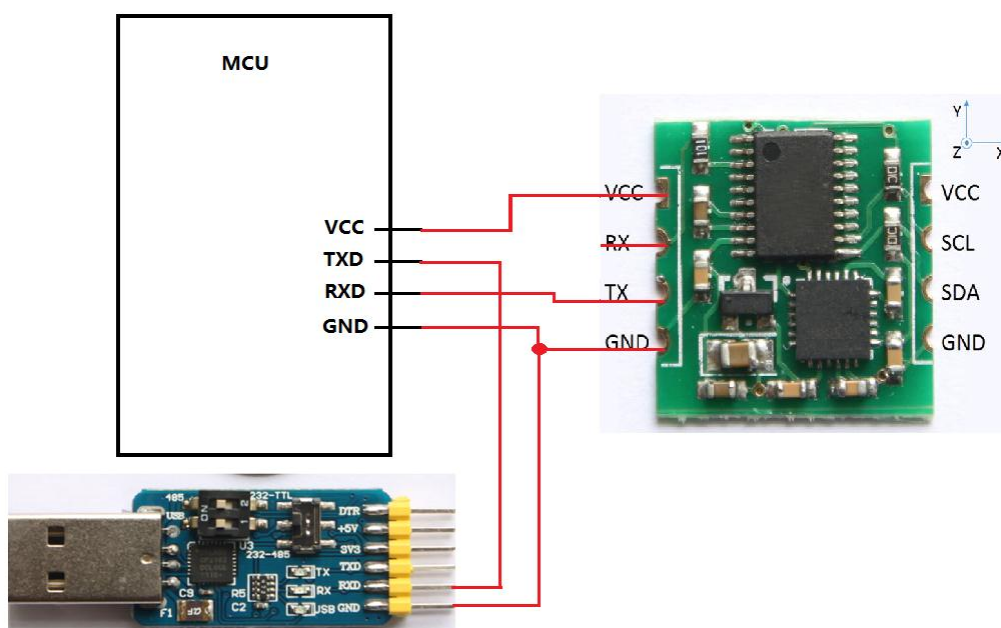


## 4.2 连单片机



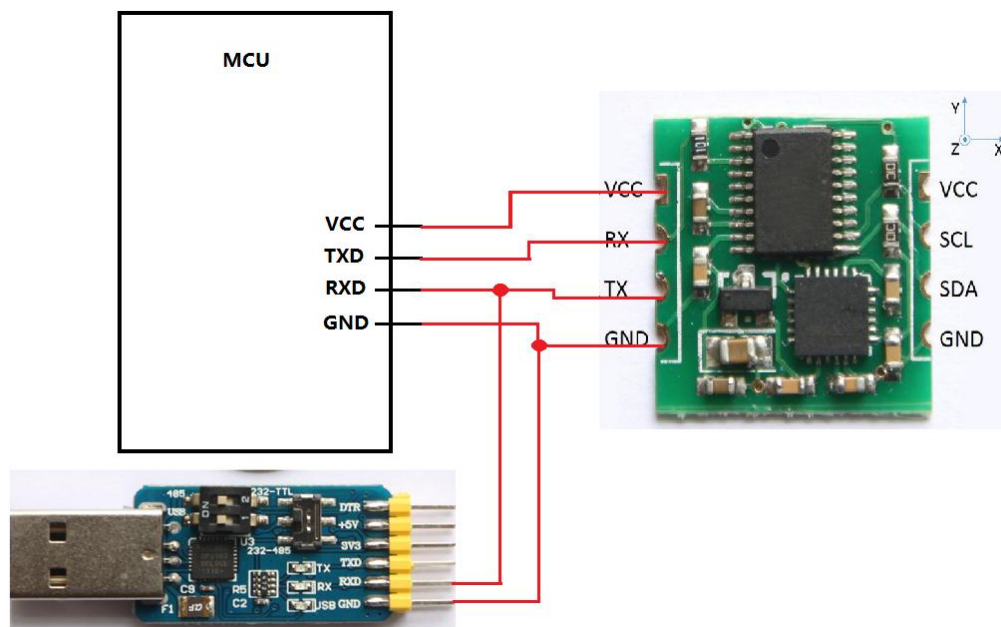
## 4.3 MCU 连单片机并输出调试信息。

通常情况下，MCU 的串口资源比较紧张，有的单片机只有一个串口，而且调试的时候需要通过串口输出调试信息，这时可以将 MCU 的 TX 引脚连接到 USB 转串口模块的 RX 上，6050 模块的 TX 接到 MCU 的 RX 引脚上，这样 MCU 既可以收到 6050 模块的数据，又可以输出调试信息了。只是 MCU 无法输出串口指令给 6050 模块了，不过模块的配置都是可以掉电保存的，而且校准可以再上电后第三秒钟自动执行，通常情况下不用发送任何指令即可工作。



#### 4.4 用上位机监视模块与单片机的通信。

如果需要在 MCU 接受 6050 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。



## 5 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：115200/9600，停止位 1，校验位 0。

### 5.1 上位机至模块

指令内容	功能	备注
0x52	角度初始化	使 Z 轴角度归零
0x61	使用串口，禁用 I2C	掉电保存，建议使用上位机修改
0x62	禁用串口，使用 I2C 接口	掉电保存，建议使用上位机修改
0x63	波特率 115200，帧率 100Hz	掉电保存，建议使用上位机修改
0x64	波特率 9600，帧率 20Hz	掉电保存，建议使用上位机修改

说明：

1. 模块上电以后需先保持静止，模块内部的 MCU 会在模块静止的时候进行自动校准（消除陀螺零漂），校准以后 Z 轴的角度会重新初始化为 0，Z 轴角度输出为 0 时，可视为自动校准完成的信号。

2. 出厂默认设置使用串口，波特率 115200，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。

## 5.2 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 115200 时每隔 10ms 输出 1 帧数据，波特率 9600 时每隔 50ms 输出一帧数据。

### 5.2.1 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:

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

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

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

温度计算公式:

$$T = ((TH \ll 8) | TL) / 340 + 36.53 \quad ^\circ C$$

校验和:

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

### 5.2.2 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	TL	温度低字节
9	TH	温度高字节
10	Sum	校验和

角速度计算公式:

$$w_x = ((w_xH \ll 8) | w_xL) / 32768 * 2000 (\%s)$$

$$w_y = ((w_yH \ll 8) | w_yL) / 32768 * 2000 (\%s)$$

$$w_z = ((w_zH \ll 8) | w_zL) / 32768 * 2000 (\%s)$$

温度计算公式:

$$T = ((TH \ll 8) | TL) / 340 + 36.53 \text{ } ^\circ\text{C}$$

校验和:

$$\text{Sum} = 0x55 + 0x52 + w_xH + w_xL + w_yH + w_yL + w_zH + w_zL + TH + TL$$

### 5.2.3 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	TL	温度低字节
9	TH	温度高字节
10	Sum	校验和

角速度计算公式:

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

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

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

温度计算公式:

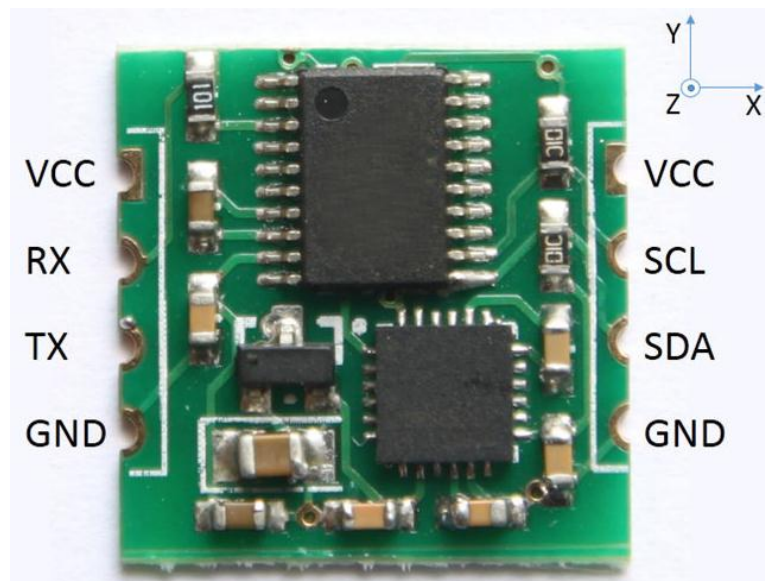
$$T = ((TH \ll 8) | TL) / 340 + 36.53 \text{ } ^\circ\text{C}$$

校验和:

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

注:

1. 姿态角结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-y-x，即先绕 z 轴转，再绕 y 轴转，再绕 x 轴转。
2. 滚转角的范围虽然是  $\pm 180$  度，但实际上由于坐标旋转顺序是 Z-Y-X，在表示姿态的时候，俯仰角(Y 轴)的范围只有  $\pm 90$  度，超过 90 度后会变换到小于 90 度，同时让 X 轴的角度大于 180 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 X 轴接近 90 度时，即使姿态只绕 X 轴转动，Y 轴的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。



### 5.3 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;
            T = (short(chrTemp[9]<<8|chrTemp[8]))/340.0+36.25;
            printf("a = %4.3f\t%4.3f\t%4.3f\t\r\n",a[0],a[1],a[2]);
            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;
            T = (short(chrTemp[9]<<8|chrTemp[8]))/340.0+36.25;
            printf("w = %4.3f\t%4.3f\t%4.3f\t\r\n",w[0],w[1],w[2]);
            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]))/340.0+36.25;
            printf("Angle = %4.2f\t%4.2f\t%4.2f\tT=%4.2f\r\n",Angle[0],Angle[1],Angle[2],T);
            break;
    }
}
```

### 5.4 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(一下为 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
```



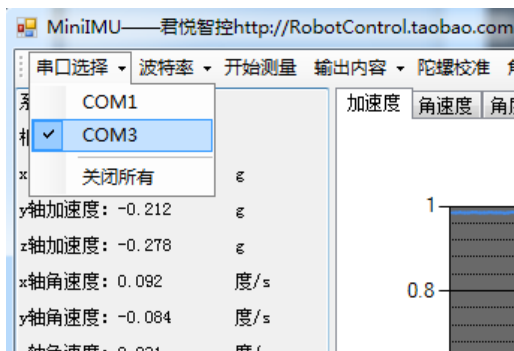
```

{
    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;
                    T = (short(Re_buf [9]<<8| Re_buf [8]))/340.0+36.25;
                    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;
                    T = (short(Re_buf [9]<<8| Re_buf [8]))/340.0+36.25;
                    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]))/340.0+36.25;
                    break;
            }
        }
    }
}

```

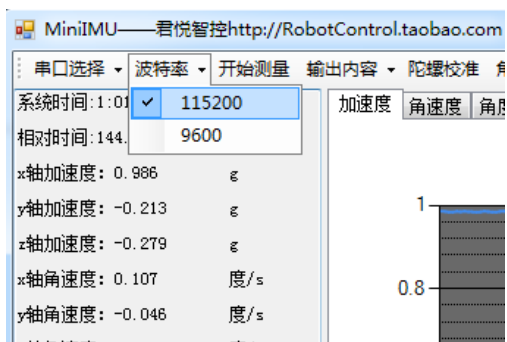
## 6 上位机使用方法

选择正确的串口

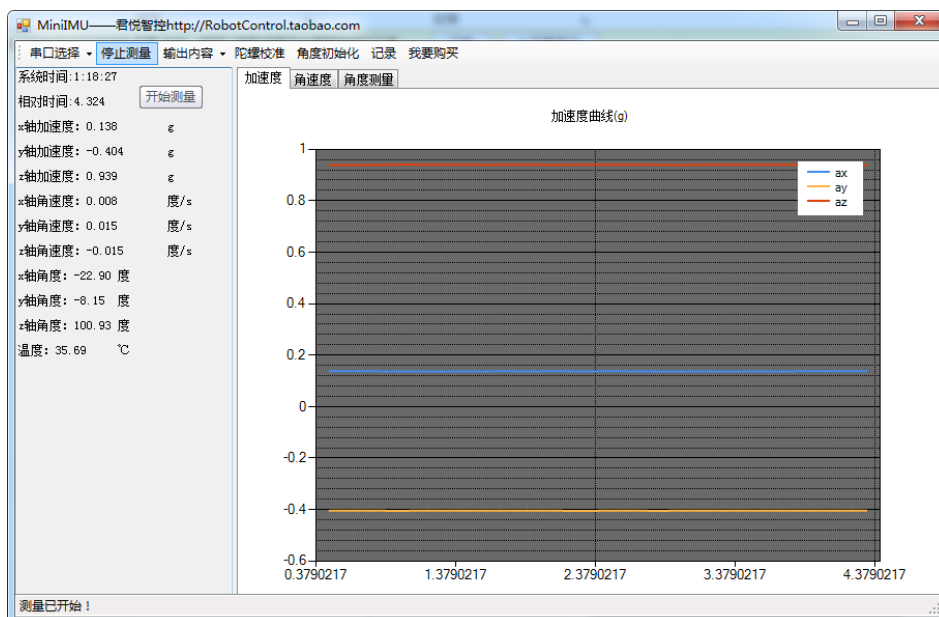


正常情况下，选择好正确的串口就可以看到数据了。

如果需配置波特率，请点击



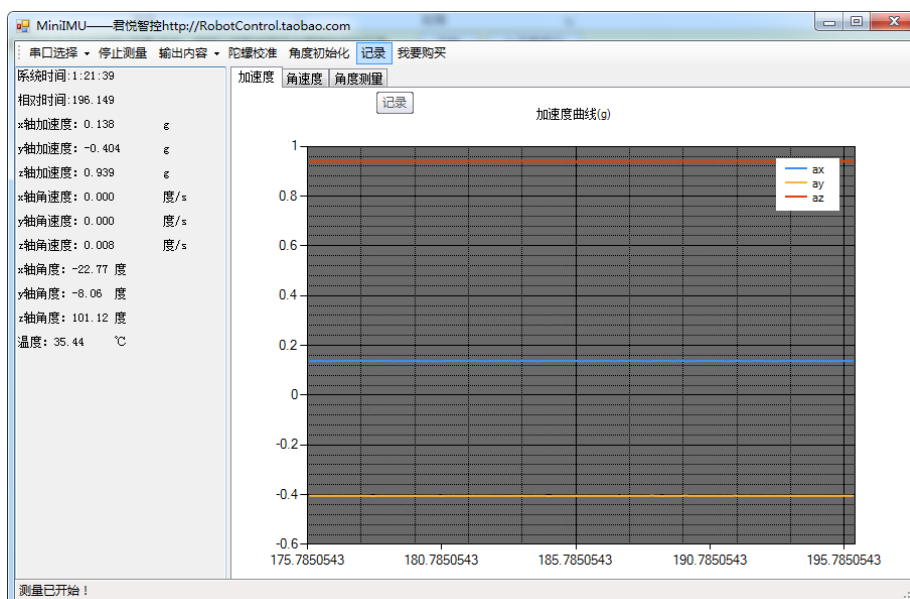
点开始测量、停止测量，选择是否使用串口输出数据。



陀螺仪校准按钮用于校准陀螺零位，校准时需保持静止。

角度初始化用于让 Z 轴的角度数据归零。

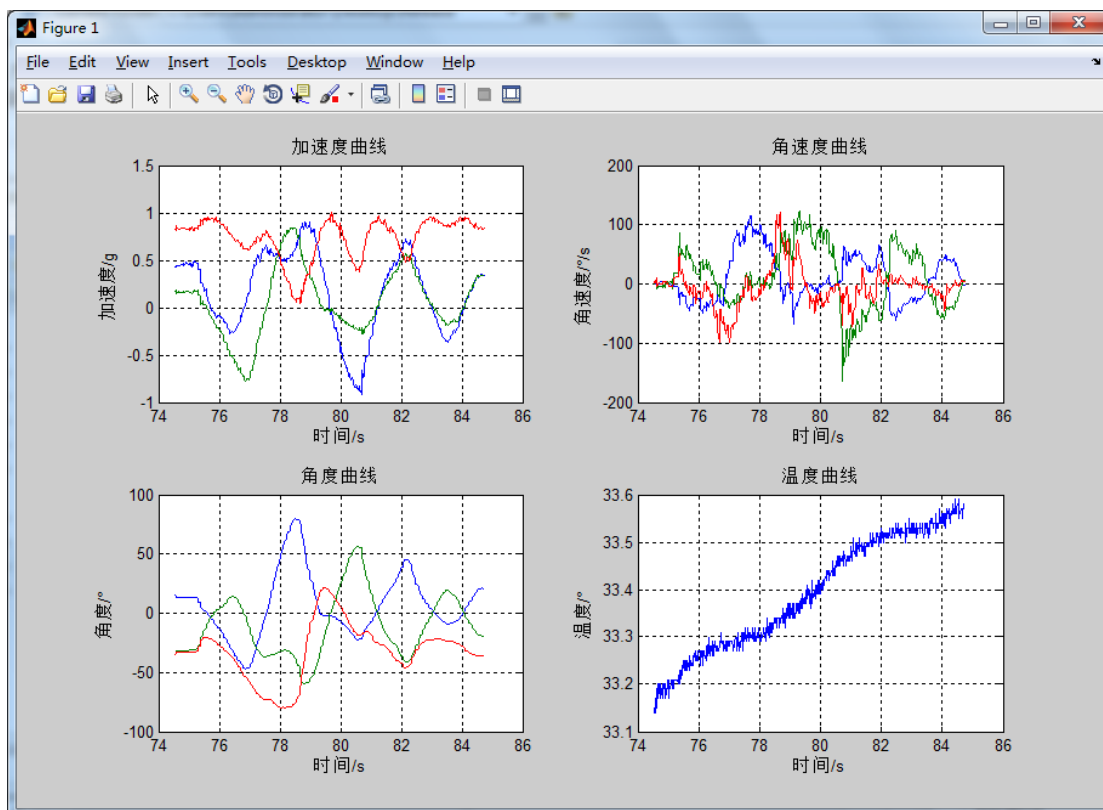
点记录按钮可以将数据保存为文件



保存的文件在上位机程序的目录下：



数据可以导入到 Exel 或者 Matlab 中进行分析。在 Matlab 环境下运行上位机根目录下的“Matlab 绘图.m”文件，可以绘制数据曲线图。



## 7 机械尺寸

