FSS-NAV680-AG

高精度组合导航系统

使用手册

原极（上海）科技有限公司

在使用本产品之前，请务必先仔细阅读本使用手册

目录

1. 产品简介

1.1 产品概述

针对低速自动驾驶在农田、果园、景区、户外等轻遮挡场景下的精密定位与连续定位需求，原极基于自研的 IMU 模组和组合算法，集成和芯星通全系统全频点高精度定位定向模块 UM982，设计了这款卫星惯性组合产品 NAV680-AG。它无需接入轮式里程计，内置原极针对车载场景的多模型智能位置融合算法，可在空天或轻遮挡应用场景下为农机，巡检车，景区游览车等自动驾驶客户提供高精度车载姿态、航向、连续定位和测速等，助力低速自动驾驶的全场景可靠运行，提升自动驾驶设备的终端应用体验。

为了满足农机，矿车等场景测前轮转角的需求，NAV680-AG 可接入我司前轮转角传感器 G200，可直接输出前轮转向角度。

1.2 技术指标

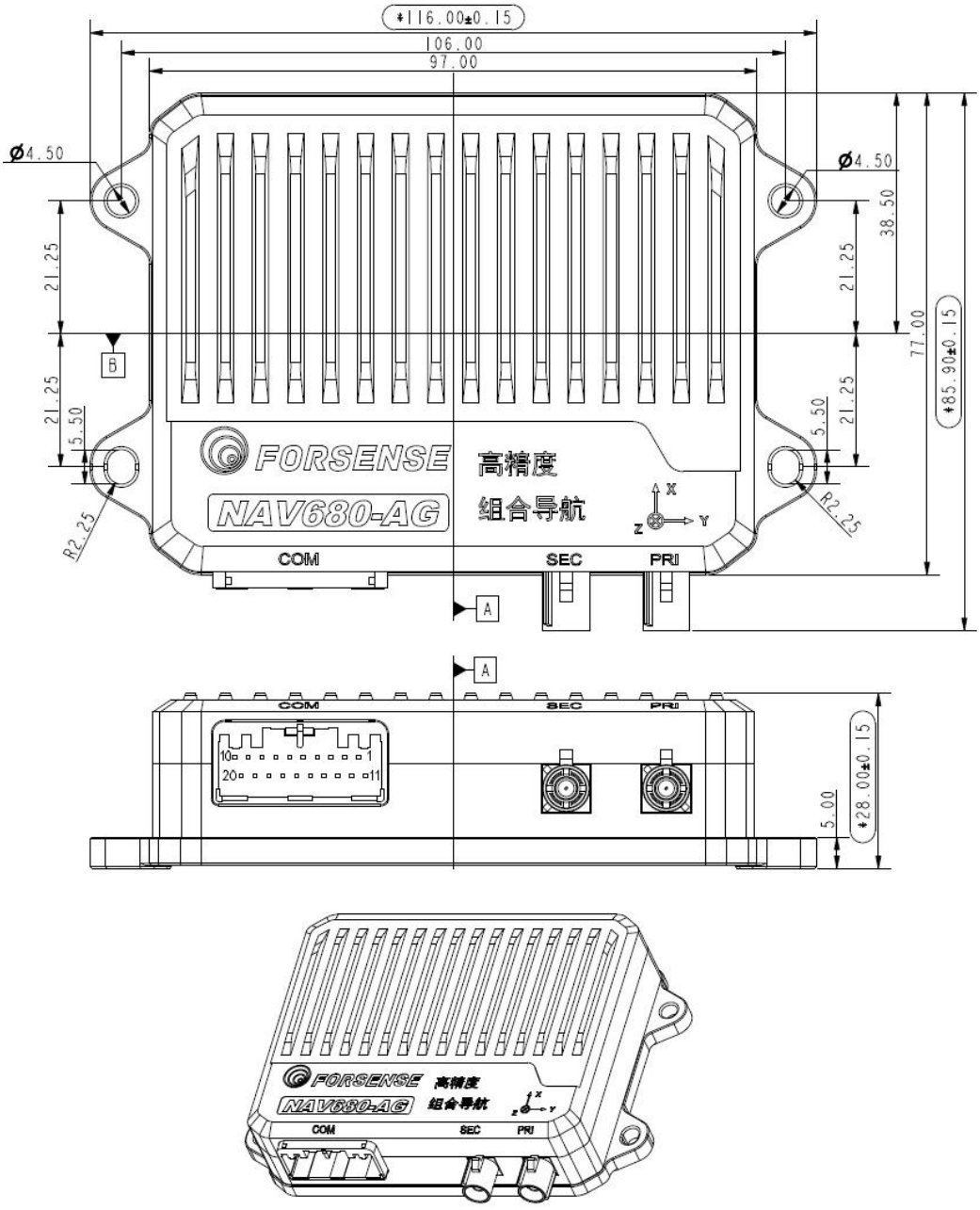
表 1

姿态精度	Roll/Pitch :<0.2° rms Heading:<0.3° rms	
更新率	100hz	
陀螺量程	±500°/s	
陀螺零偏不稳定性	4deg/h @1σ	
加速度计量程	±6g	
加速度计零偏稳定性	0.04mg @1σ	
位置推算精度	<2% @1σ （车载场景，丢星 30s，无轮速计组合）	
RTK 指标	定位精度(RMS)	单点：1.5m RTK:2cm+1ppm
	定向精度(RMS)	0.2°/1m 基线
	速度精度(RMS)	0.05m/s
	PPS 精度（RMS）	20ns
	更新率	20hz
	RTK 初始化时间	<5s

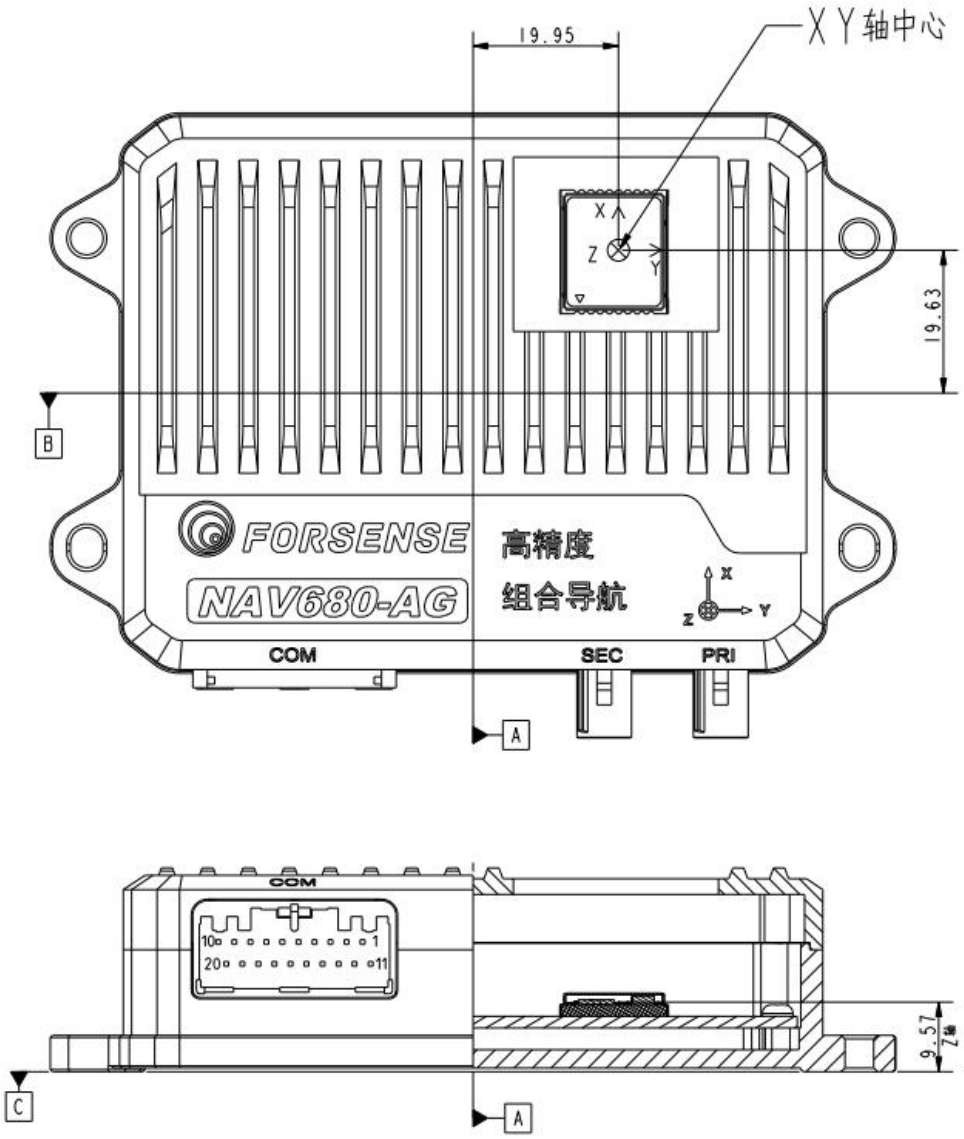
2. 硬件组成

2.1 机械尺寸

图 1 机械尺寸（单位：mm）

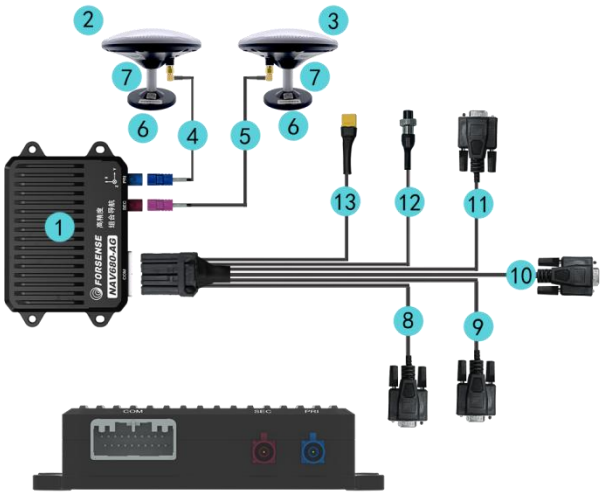


IMU 机械中心尺寸


19.95
X Y 轴中心
X
Z
Y
19.63
B
FORSENSE
NAV680-AG
高精度
组合导航
X
Z
Y
COM
SEC
PRI
A
COM
10
1
20
11
9.57
C
A

2.2 接口说明

NAV680-AG 包含 COM 总线接口，引出电源线和信号线，PRI 主天线和 SEC 副天线。



	名称	数量
1	NAV680-AG组合导航系统	1个
2	主天线（定位天线）	1个
3	副天线（定位天线）	1个
4	主天线连接线	1个
5	副天线连接线	1个
6	天线吸盘	2个
7	天线柱	2个
8	集线束-RS232-A接口	1个
9	集线束-RS232-B接口	1个
10	集线束-CAN A接口	1个
11	集线束-CAN B接口	1个
12	集线束-I/O接口	1个
13	集线束-电源接口	1个

2.3 数据线接口定义

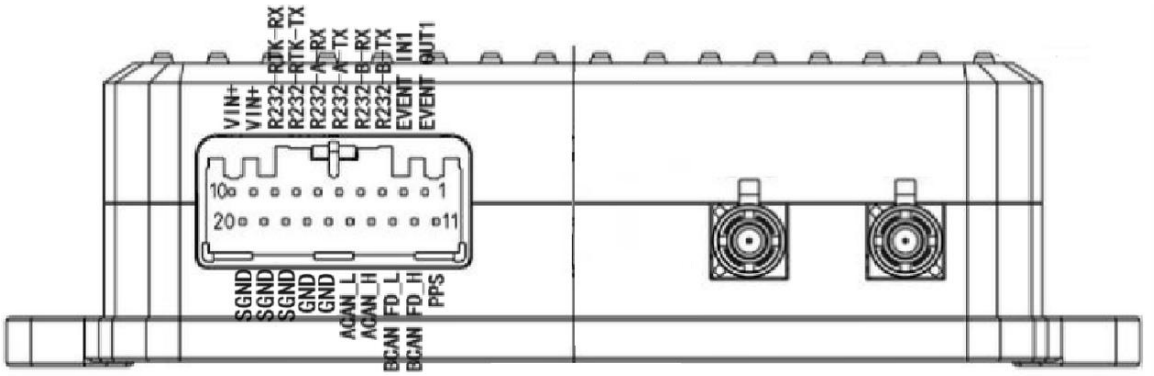
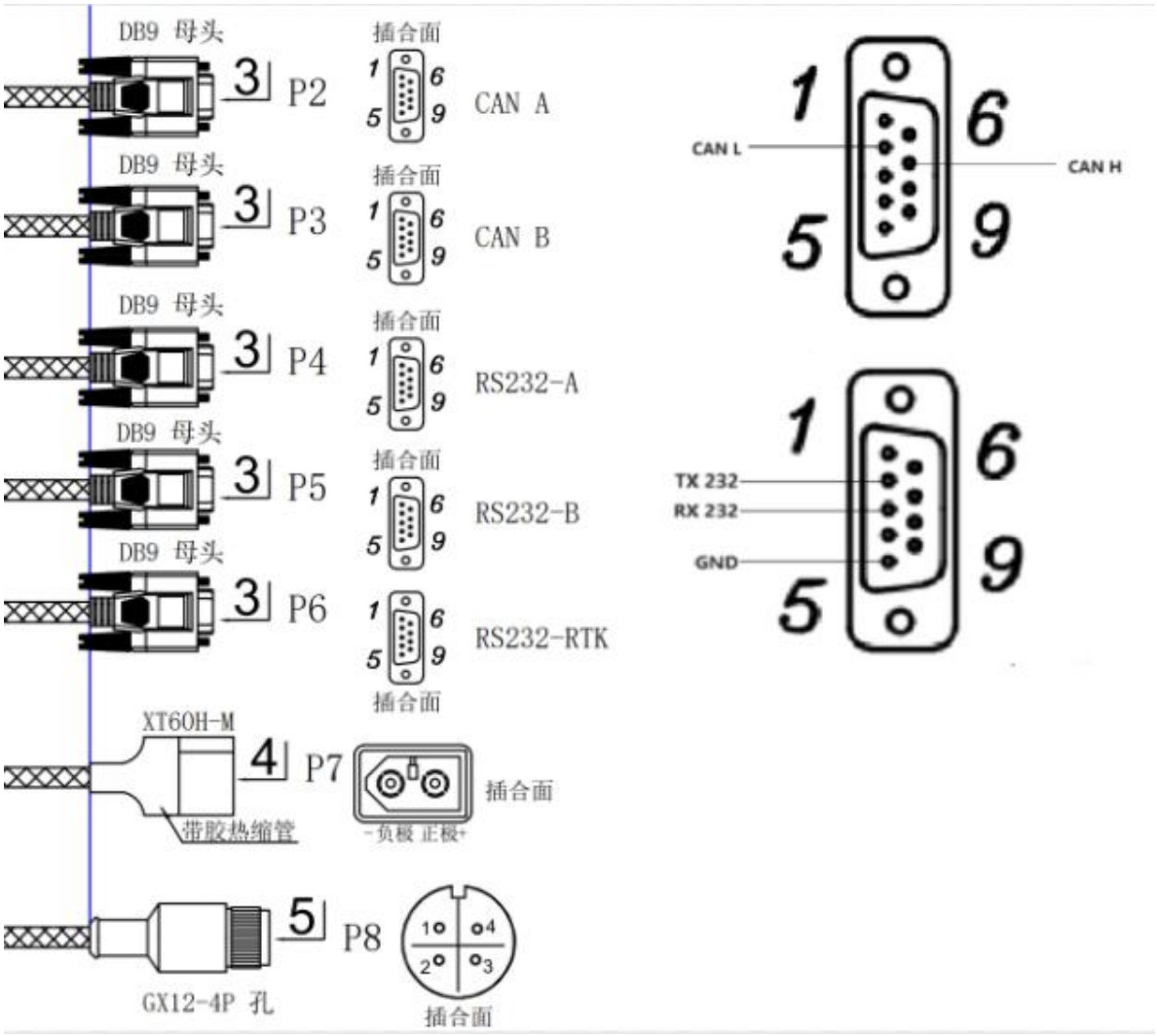


图 2 数据线接口定义图

对应引脚定义如下

PIN 序	定义	端口
1	EVENT OUT1	IO 口 （预留）
2	EVENT IN1	
3	MCU TX3 232	R232-B
4	MCU RX3 232	
5	MCU TX2 232	R232-A
6	MCU RX2 232	
7	RTK TX1 232	R232-RTK
8	RTK RX1 232	
9	VIN+	电源正极 9-24V
10	VIN+	电源正极 9-24V
11	PPS	RTK 秒脉冲信号
12	BCAN FD_H	CAN_FD B（预留）
13	BCAN FD_L	
14	ACAN_H	CAN_ A （默认波特率为 500k，模式 Motorola，输入车辆（轮速、档位等）信号）
15	ACAN_L	

PIN 序	定义	端口
16	GND	电源地（电源负极）
17	GND	
18	SGND	信号地
19	SGND	
20	SGND	



说明：RS232-A 串口 A，RS232-B 串口 B，RS232-RTK 串口差分导入，CAN A，CAN B（未定义），XT60H-M 电源，IO（未定义）。

2.4 电气特性

表 2 最大额定绝对值

参数	符号	最小值	典型值	最大值	单位
电源输入	VIN	9	12	24	V
电源地	GND				
功耗	P	0		1.2	W
使用温度	T	-40	\	85	℃

2.5 物理特性

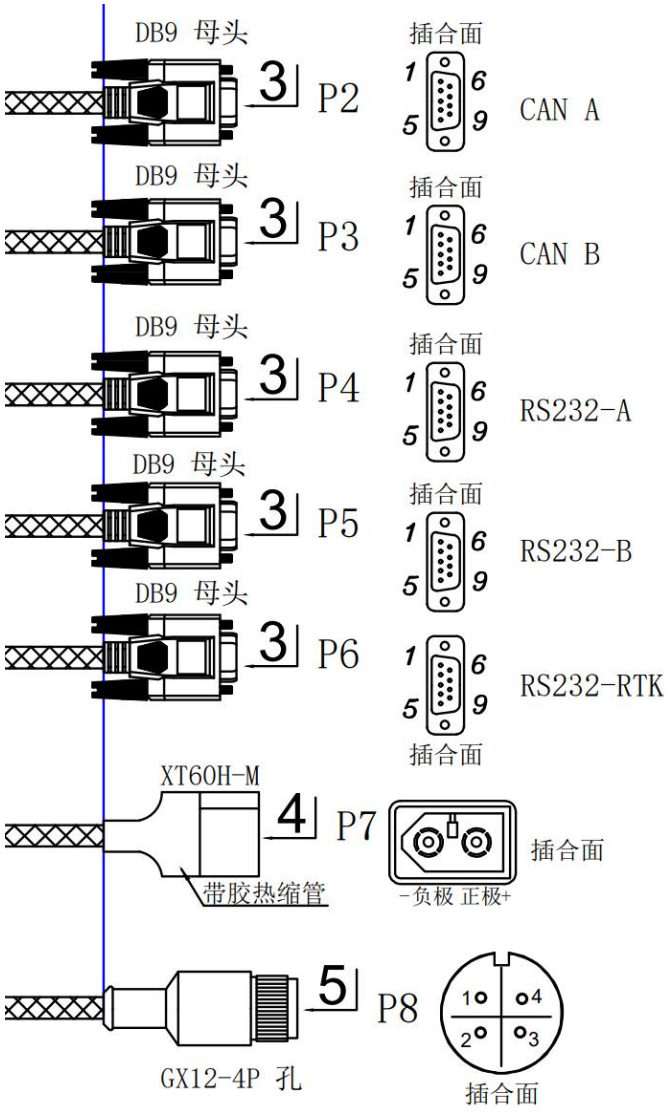
表 3 物理特性

工作温度	-40℃~+85℃
存储温度	-55℃~+95℃
湿度	95%非凝露

3. 硬件接线方式

3.1 硬件接线方式

图 2 硬件接线方式



说明：RS232-A 串口 A，RS232-B 串口 B，RS232-RTK 串口差分导入，CAN A，CAN B（未定义），XT60H-M 电源，IO（未定义）。

3.2 天线

NAV680-AG 引出 PRI 和 SEC 主副天线接口，不接天线，使用万用表测试，即空载时提供电压为 DC4.8～5.4V；模块射频口接天线时，常温下，工作电流为 30～100mA 时测试，能对外提供 DC4.6V±0.2V 的天线馈电。NAV680-AG 采用有源天线时注意与天线间的 50 欧姆阻抗匹配。

4. 坐标系定义

图 3 坐标系示意图

本产品坐标系使用 前-右-下（FRD）坐标系，欧拉角范围如下：

绕 Z 轴方向旋转： 航向角 Yaw 范围： 0° ～360°；

绕 X 轴方向旋转： 横滚角 Roll 范围： -180°～180°；

绕 Y 轴方向旋转： 俯仰角 Pitch 范围： -90°～90°。

横滚、俯仰、航向角度示意图如下：

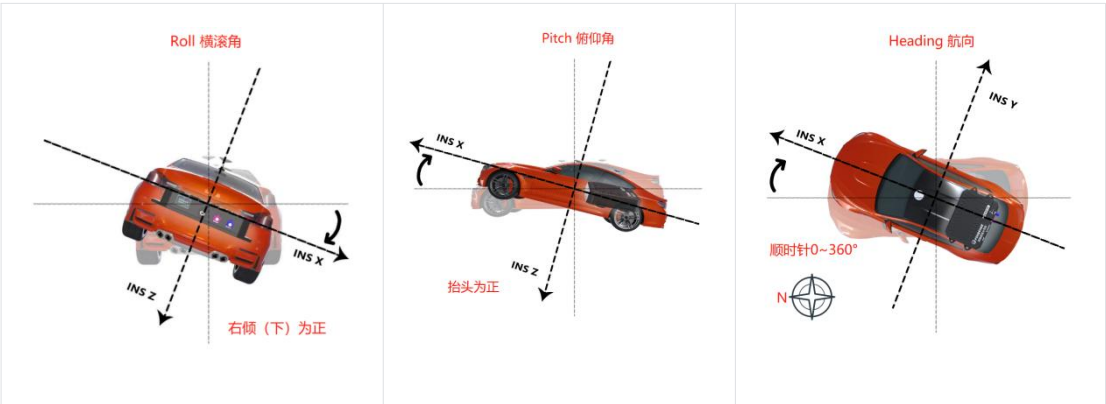


图 4 横滚、俯仰、航向角示意图

5. 组合导航输出协议

5.1 二进制协议

注意：

- CRC 校验为从帧头开始，不包含 CRC 校验位本身，该帧所有字节的 CRC 校验，校验计算方式和例程见附录。
- 帧长为除去帧头，帧 ID，帧长和校验位之外的所有数据字节总数。
- 小端模式，先发送低字节。

内容	类型	相对位置
帧头 1：0xAA	Uint8	0
帧头 2：0x55	Uint8	1
帧 ID：0x0166	Uint16	2
帧长：0x005E	Uint16	4
GPS 周内秒（ms）	Uint32	6
GPS 周计数	Uint16	10
纬度（度×10000000）	Int32	12
经度（度×10000000）	Int32	16
高度（毫米）	Int32	20
北向速度（m/s）	Float	24
东向速度（m/s	Float	28
地向速度（m/s）	Float	32
横滚角（度）	Float	36

俯仰角（度）	Float	40
航向角（度）	Float	44
前轮陀螺 Z 轴角速度（度/S）	Float	48
RTK 单天线航向（度） （有接入前轮陀螺时输出为前轮转角）	Float	52
加速度计 X 轴（g）	Float	56
加速度计 Y 轴（g）	Float	60
加速度计 Z 轴（g）	Float	64
陀螺仪 X 轴（deg/s）	Float	68
陀螺仪 Y 轴（deg/s）	Float	72
陀螺仪 Z 轴（deg/s）	Float	76
IMU 温度（℃）	Float	80
RTK 定位状态（同 GGA 中定位状态） 0:未定位 1:单点定位 2:伪距差分定位 4:固定解 5:浮点解	Uint8	84
卫星数量	Uint8	85
差分延时	Uint8	86
双天线定向状态 50 表示已定向 其他表示未定向	Uint8	87

位置精度因子（cm） 组合导航初始化后有效	Uint16	88
状态位： bit0:1 表示 RTK 数据有效，0 表示无效 Bit1:1 表示 PPS 信号有效，0 表示无效 Bit2:1 表示组合导航已初始化，0 表示未初始化 （连接单天线车辆起步后速度大于 0.5m/s 可初始化，双天线车辆无需起步定向状态即可初始化） Bit3:1 表示前轮转角有效，0 无效 Bit4:1 表示组合导航已收敛，0 未收敛 Bit5:1 表示前轮陀螺数据有效，0 无效（未外接陀螺可忽略） Bit6:1 表示方向盘电机数据有效，0 无效 Bit7、Bit8: 01 表示车辆前进 10 表示车辆后退 00 表示无效	Uint16	90
预留 1	Uint32	92
预留 2	Uint32	96
CRC 校验	Uint32	100

5.2 nmea 协议

- 支持按 nmea 格式输出组合后数据，
- 与二进制数据流无法同时输出，输出 nmea 数据流则不能输出二进制数据流，切换数据流前需先按 7.2.2 指令停止当前数据流输出。
- 目前支持以下语句。配置方式间 7.3 章节

GPGGA

GPRMC

GPHDT （航向信息）

GPVTG （地面速度信息）

GPZDA （UTC 时间及日期）

GPATT （原极自定义报文）

GPATT 格式如下表

Name	*Unit*	*Format*	*Example*	*Description*
Sentence Identifier		String	$GNATT	
Time		hhmmss.sss	170834.000	17:08:34 UTC
Status		Character	1	0: invalid 1: valid
Roll Angle	degree	3 decimal places	-4.891	range ±90, right side down defined as positive
Indicator for roll		character	R	Roll indicator
Pitch Angle	degree	3 decimal places	3.122	range ±90, head up defined as positive
Indicator for Pitch		character	P	Pitch indicator
Heading Angle	degree	3 decimal places	124.005	range 0~360, to true North, counter clockwise defined as positive
Roll Angle uncertainty	degree	3 decimal places	0.432	range 0~360
Pitch Angle uncertainty	degree	3 decimal places	0.811	range 0~360
Heading Angle uncertainty	degree	3 decimal places	1.202	range 0~360
Checksum		Hex	*68	Used by program to check for transmission errors

5.3 CAN 协议

表 1 CAN 扩展帧格式 0x19FF CC9A （注：单位为 deg*100 ， int16 类型）

扩展帧 ID	1	2	3	4	5	6	7	8
0x19FF CC9A	横滚角		俯仰角		艏向角		航迹角	

表 2 CAN 扩展帧格式 0x19FF CD9A（注：单位为 deg/s*52.0127 ， int16 类型）

扩展帧 ID	1	2	3	4	5	6	7	8
0x19FF CD9A	gyro_x		gyro_y		gyro_z		预留	

表 3 CAN 扩展帧格式 0x19FF CE9A（注：单位为 g*3276.8 ， int16 类型）

扩展帧 ID	1	2	3	4	5	6	7	8
0x19FF CE9A	accel_x		accel_y		accel_z		状态位	

状态位：

Bit0:RTK 板卡数据有效标志位，1 表示有效，0 表示无效

Bit1:PPS 有效标志位，1 表示有效，0 表示无效

Bit2:前轮转角有效标志位，1 表示有效，0 表示无效

Bit3:前轮陀螺仪有效标志位，1 表示有效，0 表示无效

Bit4:RTK 定位固定解标志位，1 表示固定解，0 表示未固定解

Bit5:RTK 定向固定解标志位，1 表示固定解，0 表示未固定解

Bit6:航向初始化标志位，1 表示已初始化，0 表示未初始化（仅针对单天线车身陀螺）

5.4 RTK 定位状态表

二进制	ASCII	描述
0	NONE	无解
1	FIXEDPOS	位置由 FIX POSITION 命令指定
2	FIXEDHEIGHT	暂不支持
8	DOPPLER_VELOCITY	速度由即时多普勒信息导出

16	SINGLE	单点定位
17	PSRDIFF	伪距差分解
18	WAAS	SBAS 定位
32	L1_FLOAT	L1 浮点解
33	IONOFREE_FLOAT	消电离层浮点解
34	NARROW_FLOAT	窄巷浮点解
48	L1_INT	L1 固定解
49	WIDE_INT	宽巷固定解
50	NARROW_INT	窄巷固定解

6. 参数配置

6.1 通过串口配置

6.1.1 配置杆臂

例如配置杆臂向量为 X=1.2m, Y=0.2m, Z=-1.0m

指令 ：AT+CLUB_VECTOR=1.2,0.2,-1.0\r\n

应答 ：GPS_POS_X=1.2,GPS_POS_Y=0.2,GPS_POS_Z=-1.0/r/n

说明：杆臂向量为主天线相位中心相对设备中心位置的三维矢量（X,Y,Z），单位为米。其中，

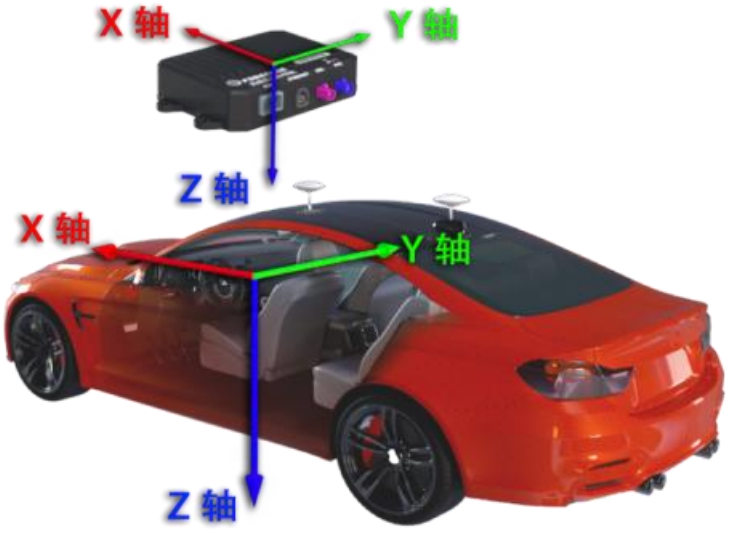
在前右下车体坐标系下

若天线在设备中心位置 X 轴的前方，则为正数，否则为负数；

若天线在设备中心位置 Y 轴的右方，则为正数，否则为负数；

若天线在设备中心位置的**上方**为**负数**，否则为**正数（通常在上方）**

坐标系示意图如图 5 所示



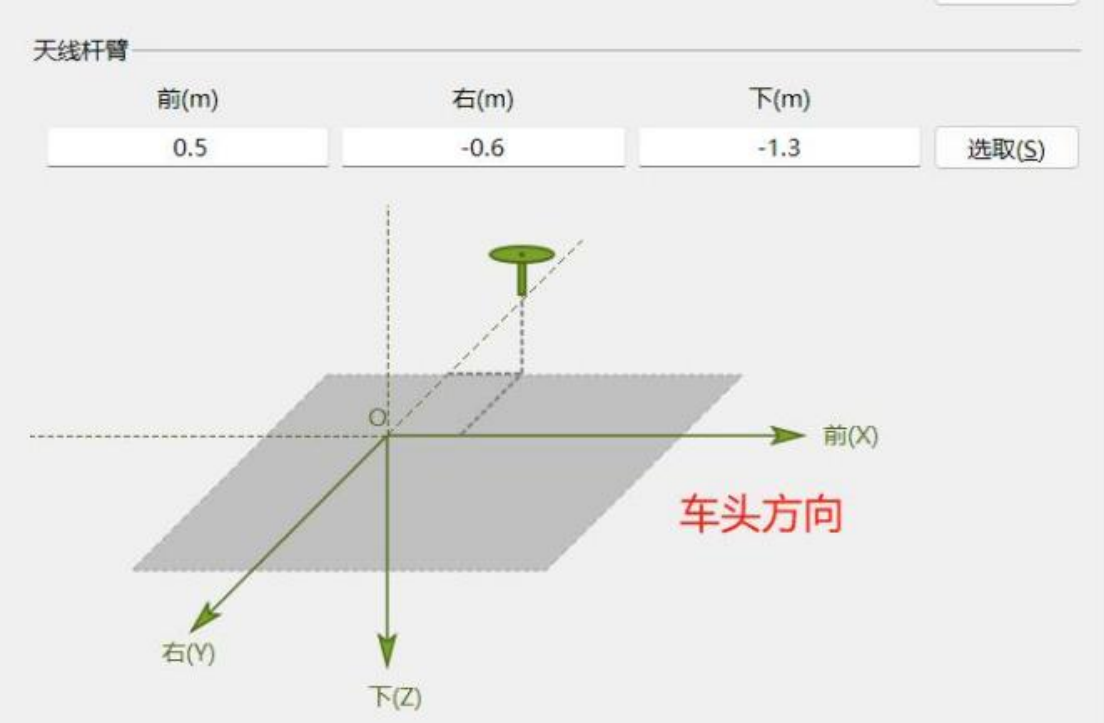


图 5 坐标系示意图

6.1.2 配置输出二进制组合导航数据流

指令 ：AT+SETNAV\r\n

应答 ：OK\r\n

若配置停止输出

指令 ：AT+SETNO\r\n

应答 ：OK\r\n

6.1.3 配置输出 NMEA 格式数据流

若配置 NEMA 语句输出，则 6.1.2 组合导航数据流不输出

如需切换为 6.1.2 组合导航数据流输出，需先停止当前数据流输出

配置指令如下

GPGGA

例：以 5hz 频率输出 GPGGA 语句： AT+GPGGA=5\r\n

应答：OK\r\n

GPRMC

例：以 1hz 频率输出 GPRMC 语句： AT+GPRMC=1\r\n

应答：OK\r\n

GPHDT（航向信息）

例：以 1hz 频率输出 GPHDT 语句： AT+GPHDT= 1\r\n

应答：OK\r\n

GPVTG（地面速度信息）

例：以 1hz 频率输出 GPVTG 语句： AT+GPVTG= 1\r\n

应答：OK\r\n

GPZDA（UTC 时间及日期）

例：以 1hz 频率输出 GPZDA 语句： AT+GPZDA= 1\r\n

应答：OK\r\n

GPATT（自定义报文）

例：以 1hz 频率输出 GPATT 语句： AT+GPATT= 1\r\n

应答：OK\r\n

若配置停止输出

指令 ：AT+SETNO\r\n

应答 ：OK\r\n

6.1.4 配置当前数据流停止输出

指令 ：AT+SETNO\r\n

应答 ：OK\r\n

6.1.5 配置组合导航输出的位置、速度投影点

若配置输出组合导航设定的投影点结果，则配置指令为：

指令 ：AT+PROJ_VECTOR=1.0,2.0,3.0\r\n

应答 ：PROJ_VECTOR_X=1.0, PROJ_VECTOR_Y=2.0, PROJ_VECTOR_Z=3.0/r/n

说明 ：组合导航输出默认为天线相位中心投影点结果，若需输出其他位置结果，则需配置此位置的杆臂向量，配置方法同 7.1 杆臂配置

6.1.6 配置 RTK 双天线安装角

若配置 RTK 双天线安装角为 0 度，则配置指令为：

指令 ：AT+RTK_ANGLE=0\r\n

应答 ：ANGLE=0\r\n

说明 ：安装角为主天线指向副天线的射线与车头方向的夹角，顺时针为正，逆时针为负，角度输入范围-180°~180°

注意：配置指令保存后需断电重启生效；双天线间隔距离需大于 50cm

6.1.7 标定双天线

双天线标定流程如下

1. 连接串口，输入 AT+SETNO\r\n 关闭所有打印

2. 发送指令 AT+RTK_BIAS_EST=1\r\n 开启标定流程

3. 车辆直行，车速不低于 3km/h，串口返回 RTK_BIASE_CORRECT_DONE 后表示标定成功

4. 发送 AT+CONFIG\r\n 指令可以看到打印，RTK_BIAS_FLAG_AND_VALUE=99,XX（标定角度），检查是否合理，安装正常的情况下小于 1 度

5. AT+SAVE\r\n 保存结果，重新上电。

6.1.8 配置数据输出频率

若配置数据输出频率为 10hz，则配置指令为：

指令 ：AT+OUTRATE=10\r\n

应答 ：OK\r\n

6.1.9 配置波特率

仅支持配置波特率为 115200 或 230400，默认波特率为 115200

若配置 IMU 串口波特率为 230400，则配置指令为：

指令 ：AT+BAUD=230400\r\n

应答 ：BAUD=230400\r\n

注意：配置指令且保存后需断电重启生效

6.1.10 配置车辆轴距

若配置车辆轴距为 2m，则配置指令为：

AT+WHEEL_BASE=2\r\n

注：指令配置后需保存

6.1.11 打印所有配置信息

若查询所有配置过的信息，则配置指令为：

AT+CONFIG\r\n

6.1.12 查询版本号

AT+VERSION\r\n

6.1.13 配置安装旋转角

目前仅支持如下旋转角度

x 轴旋转 180 度

z 轴旋转 90 180 270

配置指令如下

若 安装旋转 角度为 X 轴旋转 180 度，则配置指令为：

指令 ：AT+INSTALL_ANGLE=180,0,0\r\n

应答 ：INST_ANGLE_X=180.000,INST_ANGLE_Y=0.000,INST_ANGLE_Z=0.000

若 安装旋转 角度为 Z 轴旋转 180 度，则配置指令为：

指令 ：AT+INSTALL_ANGLE=0,0,180\r\n

应答 ：INST_ANGLE_X=0.000,INST_ANGLE_Y=0.000,INST_ANGLE_Z=180.000

6.1.14 开启坡地模式

针对始终存在 5 度以上坡地场景(常见于新疆，内蒙等地)，可开启坡地模式进行补偿，

配置命令为：

AT+SLOPE=0.05\r\n 有坡度的农机场景可配置此语句，默认为 0。

配置以后输入 AT+SAVE\r\n 保存配置，重启后生效

6.1.15 保存参数

以上包括杆臂，数据流，输出频率等参数，配置完以后都需要输入次保存指令，重启后才能生效

指令 ：AT+SAVE\r\n

应答 ：OK\r\n

6.2 通过 CAN 配置

6.2.1 查询版本号

示例：

ID = 0x19FFF326

Data = 0x9A 0x07 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

应答（版本号 V1.21.2.2）

ID = 0x19FFF29A

Data = 0x26 0xC7 0x01 0x15 0x02 0x02 0xFF 0xFF

6.2.2 配置轴距和杆臂

示例

ID = 0X0DFFC126

Data = 轴距 2 字节+杆臂 2 字节*3（单位 cm）

应答

ID = 0X19FFC09A

Data = 轴距 2 字节+杆臂 2 字节*3（单位 cm）

6.2.3 配置双天线安装角度

ID = 0X0DFFC326

Data = 角度（int16，单位：度*100）+0xFF*6

6.2.4 查询轴距和杆臂，双天线安装角度

ID = 0X0DFFC226

Data =0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

应答

ID = 0X19FFC29A

Data = 轴距 2 字节+杆臂 2 字节*3（单位 cm）

安装角度查询指令：

ID = 0X0DFFC526

Data =0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

应答

ID = 0X19FFC29A

Data = 角度*100 2 字节

6.2.5 配置滤波器截止频率

ID = 0X0DFFC626

Data = 截止频率（uint8，定义见下方）+0xFF*7

应答

ID =**0x19FFF69A**

Data =截止频率（uint8，定义见下方）+0xFF*7

常用截止频率定义：

0x44：10Hz

0x66：20Hz

0xBB：47Hz

6.2.6 查询截止频率

ID = 0X0DFFC726

Data =0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

应答：

ID =0x19FFF79A

Data =截止频率（uint8，定义见下方）+0xFF*7

6.2.7 配置 CAN 输出更新率和波特率

波特率支持 1M，500K，250K（分别表示为 0x01，0x02，0x03）

更新率支持 20，50，100Hz（分别表示为 0x14，0x32，0x64）

配置完以后重启生效。

ID = 0x19FFF326

Data = 0x9A 0x4A 波特率 更新率 0xFE 0xFB 0xF9 0xFF

应答

ID =0x19FFF59A

Data =0xFF 0x4A 波特率 更新率 0xFF 0xFF 0xFF 0xFF

6.2.8 查询更新率和波特率：

ID = 0x19FFF426

Data =0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

应答

ID =0x19FFF49A

Data =0xFF 0x4A 波特率 更新率 0XFF 0XFF 0XFF 0xFF

6.2.9 保存参数指令

ID = 0X0DFFCF26

Data =0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

应答

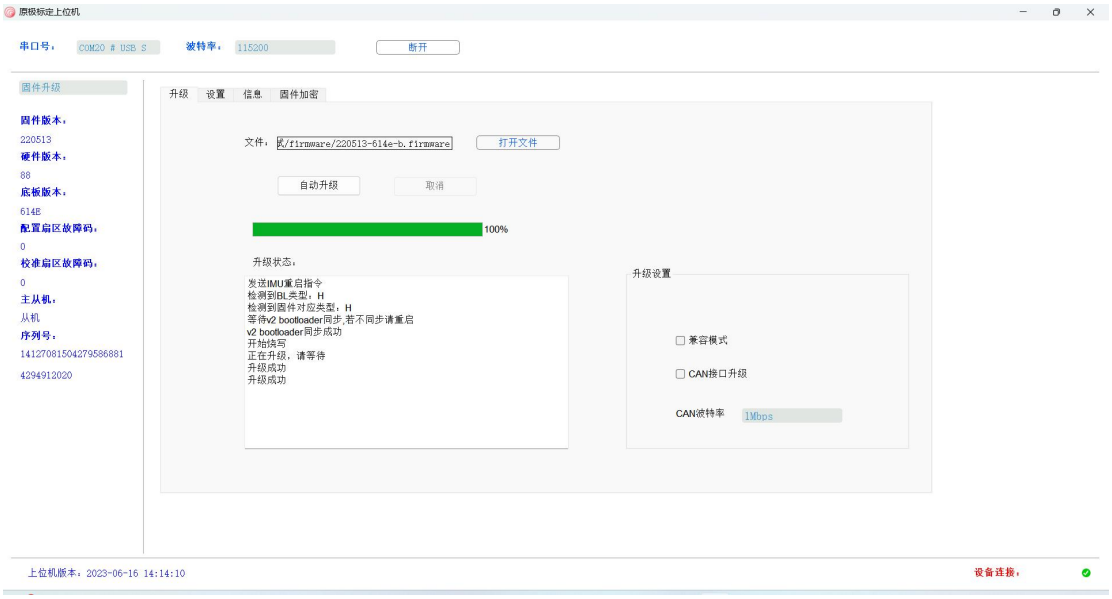
ID = 0X19FFCF9A

7. 固件升级

7.1 通过上位机

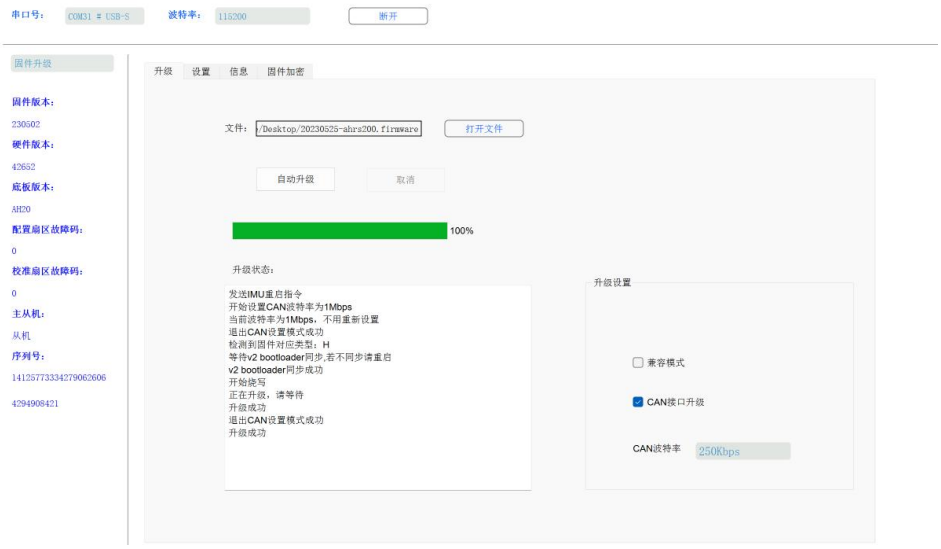
7.1.1 R232 串口直连

使用原极 IMU 测试上位机——选择固件升级——打开固件——点击自动升级。



7.1.2 使用附带 USB 转 CAN 模块

使用原极 IMU 测试上位机——选择固件升级——打开固件——选择 CAN 接口升级——设置升级后的固件波特率——点击自动升级。



7.2 串口推送

升级总共分为以下几个步骤：

第一步：发送升级指令

发送升级指令，告诉模块做好升级准备，命令发送完成后，设备会 在 flash 某个区域记录下升级标志位， 然后进行软重启， 此时 IMU 模块将进入

BOOTLOADER 中。

```
上位机发送升级指令如下：
  cmd_bl[34] ={0x55,0xaa,0xbb,0x88,0x18,0x00,0x00,0x00,0xc8,0x42,0x00,0x00,
  0x48,0x43,0x2c,0x01,0x00,0x00,0x90,0x01,0x00,0x00,0xf4,0x01,0x00,0x00,
  0x58,0x02,0x00,0x00,0x40,0x97,0x46,0x6a};
  此时 imu 模块将会重启后进入 bootloader 中
```

第二步：发送 **HC32MCU_FORSENSE** 字符串

IMU 模块进入 bootloader 中后，IMU 模块会主动发送 HC32_UPLOADER 字符串， 在此期间，上位机需要发送 HC32MCU_FORSENSE 字符串，IMU 模块一旦收到此 字符串，将不会跳转进 APP，将留在 bootload r 中处于程序待升级状态， 同时也

会停止主动发送任何消息

```
//发送完升级指令后，紧接着发送如下字符串，确保 imu 模块不会跳转到 APP 区域，同时处于待升级状态
```

```
QString str = "HC32MCU_FORSENSE" ;
for(int i=0;i<10;i++)
{
  _port_device->write(str.toLatin1()); sleep_ms(50);
}
```

第三步：发送同步指令

IMU 模块在 bootloader 中处于待升级状态时，不会主动发送信息，只会被动的响 应上位机的指令。此时上位机发送命令 Send_CMD_LONG(0X21,0,0,0,0,0,0)后，等 待 IMU 模块的响应 。 IMU 收到此命令后，立即回复 Send CMD ACK (0x64,0x10,0)

数据 。上位机一旦判断接收到此数据，就表示完成了同步。

```
上位机与 imu 模块同步过程：
1.上位机发送同步命令，命令码为 0x21，发送后等待 imu 模块响应
 Send_CMD_LONG(0x21,0,0,0,0,0,0);
2.imu 模块收到 0x21 的命令码后，发送响应数据，数据命令如下：
Send_CMD_ACK(0x64,0x10,0);
3.上位机判断是否收到了 imu 的响应数据，如收到则表示完成了同步
注：Send_CMD_LONG 函数与 Send_CMD_ACK 函数见文章末尾
```

第四步：发送擦除命令

上位机发送擦除命令 Send_CMD_LONG(0x23,0,0,0,0,0,0)，指令发送后，IMU 模块 会将 APP 区域的内容全部擦除干净，并将最终的执行结果告诉上位机 。上位机 会根据执行结果是否要再次发送擦除命令。一旦擦除成功，IMU 模块的 APP 将无 法还原。另外上位机必须等到 IMU 擦除成功的响应后，才能进行下一步，否则有

可能造成后续的升级失败。

```
1.上位机发送擦除命令，命令码为 0x23，发送完此命令后耐心等待 imu 回复执行结果
   Send_CMD_LONG(0x23,0,0,0,0,0,0);
2.imu 收到 0x23 的命令码后，将会把 APP 区域数据擦除干净，并将最终的执行结果回复给上位机
    send_CMD_ACK(0x64,0x10,0); //表示擦除成功
```

```
    send_CMD_ACK(0x64,0x11,0); //表示擦除失败，需要重新擦除
    注意：上位机必须收到擦除成功的响应，才能进行下一步
```

第五步： 发送升级数据包

擦除成功后 ， 将进入最重要的发送固件数据环节 ， 发送数据使用函 Send_Upload_Data 进行发送 。上位机将升级固件进行分包，每包固定大小 64 字节，最后一包不足 64 字节的按照实际字节数进行发送 。每一帧数据包含有效数 据长度与此包在整个固件中的偏移地址。上位机每发送一帧数据，必须等待 IMU 模块的响应，判断 IMU 成功获取到此帧数据后，再发送下一帧 。IMU 模块成功收 到上位机的数据包后，会发送响应数据，并根据偏移地址写入指定的 flash 地址。

如果写入 flash 失败，发送失败命令，写入成功则不发。

```
例：将大小 1000 字节的升级文件 uint8_t Upgrade_Data[1000]发送给 imu 模块
 //发送第一包：
  1.1 通过函数将 0~63 字节发送到 imu 模块
  Send_Upload_Data（0x27,0,0,0x40,Upgrade_Data);
  //上面函数第一参数 0x27 为固定值，第二参数 0 为固定值，第三参数 0 为偏移地址，
    第四参数 0x40 为有效字节长度，第五参数为发送数据的首地址

 1.2 imu 成功收到数据后，会发送响应数据，并根据偏移地址写入指定的 flash 地址
   send_CMD_ACK(0x753D,0x00,0);
   //上面函数第一参数 0x753D 为固定值，第二参数 0 为固定值，第三参数为偏移地址。
   imu 模块将数据写入 flash 过程中如果写入失败将发送失败命令 send_CMD_ACK(0x64,0x11,0);

   //发送第二包：
   2.1 上位机收到 imu 响应数据后，将发送第二包数据 Send_Upload_Data（0x27,0,0x40,0x40,Upgrade_Data+0x40);
   2.2 imu 成功收取到第二包数据后，发送响应： send_CMD_ACK(0x753D,0x00,0x40);
   2.3 imu 模块将数据写入 flash 过程中如果写入失败将发送失败命令 send_CMD_ACK(0x64,0x11,0);


   //发送第三包
   3.1 上位机收到 imu 响应数据后，将发送第三包数据 Send_Upload_Data（0x27,0,0x80,0x40,Upgrade_Data+0x80);
   3.2 imu 成功收取到第三包数据后，发送响应： send_CMD_ACK(0x753D,0x00,0x80);
   3.3 imu 模块将数据写入 flash 过程中如果写入失败将发送失败命令 send_CMD_ACK(0x64,0x11,0);
  ----------
  //发送第十五包：
```

```
   15.1 上位机收到 imu 响应数据后，将发送第十五包数据 Send_Upload_Data（0x27,0,0x380,0x40,Upgrade_Data+0x380);
   15.2 imu 成功收取到第十五包数据后，发送响应： send_CMD_ACK(0x753D,0x00,0x380);
   15.3 imu 模块将数据写入 flash 过程中如果写入失败将发送失败命令 send_CMD_ACK(0x64,0x11,0);
   //发送第十六包：
   15.1 上位机收到 imu 响应数据后，将发送第十六包数据 Send_Upload_Data（0x27,0,0x3C0,0x28,Upgrade_Data+0x3C0);
   15.2 imu 成功收取到第十六包数据后，发送响应： send_CMD_ACK(0x753D,0x00,0x3C0);
   15.3 imu 模块将数据写入 flash 过程中如果写入失败将发送失败命令 send_CMD_ACK(0x64,0x11,0);
  发送结束
```

第六步：获取 **CRC** 校验码

一般情况下升级固件的命令规则为 IMU614e-b#CRC1373387121.firmware ， CRC 字符串后面紧跟着的是已经计算好的 CRC 校验值 。升级文件发送完成后，上位 机需要发送校验指令， 以判断 IMU 模块收到的升级文件是否有误 。上位机发送 Send_CMD_LONG(0x29,0,0,0,0,0,0)命令后，获取 IMU 模块自身计算的 CRC 校验码，

如果上位机判断 CRC 校验值有误，应该从第四步擦除命令开始进行重新升级。

```
上位机发送获取 crc 校验码指令，等待 imu 响应
 Send_CMD_LONG(0x29,0,0,0,0,0,0);
imu 模块响应发送 crc 校验值数据：
send_CMD_ACK(0x753C,0x10,crc32_data);
 其中 crc32_data 值为 imu 模块本身计算的 crc32 数据
```

第七步：发送重启命令

升在上位机判断 crc 校验值正确后，发送重启命令，升级成功

```
判断 crc 校验值正确后，发送重启命令：
 Send_CMD_LONG(0x30,0,0,0,0,0,0);
```

固件升级完毕，断电重启后可以通过读取版本号判断是否升级成功。

函数定义：

1. Send_CMD_LONG 函数定义如下：

```
struct  MULTI_LONG_CMD_STRUCT
{
uint8_t header1;
uint8_t header2;
uint16_t id;
uint16_t length;
float  param1;
float  param2;
uint32_t param3;
uint32_t param4;
int32_t param5;
int32_t param6;
uint32_t check_crc;
}__attribute__((packed));
```

```
void  :Send_CMD_LONG(uint16_t cmd_id,float cm1,float cm2,uint32_t cm3,uint32_t cm4,int32_t cm5,int32_t cm6)
{
    uint8_t check_sum=0;
    struct MULTI_LONG_CMD_STRUCT data_cmd_long __attribute__((packed));
    data_cmd_long.header1=0x55;
    data_cmd_long.header2=0xAA;
    data_cmd_long.id=cmd_id;
    data_cmd_long.length=sizeof(data_cmd_long)-10;
    data_cmd_long.param1=cm1;
    data_cmd_long.param2=cm2;
    data_cmd_long.param3=cm3;
    data_cmd_long.param4=cm4;
    data_cmd_long.param5=cm5;
    data_cmd_long.param6=cm6;
    int len=sizeof(data_cmd_long)-4;
    uint32_t check_crc=1;
    data_cmd_long.check_crc=crc_crc32(check_crc,(uint8_t *)(&data_cmd_long), len);
    send((uint8_t *)(&data_cmd_long),sizeof(data_cmd_long));
}
```

2. Send_CMD_ACK 函数定义如下：

```
struct  CMD_ACK_STRUCT
{
uint8_t header1;
uint8_t header2;
uint16_t id;
```

```
uint16_t length;
uint32_t command;/*<  Command ID (of acknowledged command).*/
uint32_t result;/*<  Result of command.*/
uint32_t check_crc;
}__attribute__((packed));

void Send_CMD_ACK(uint16_t cmd_id,uint16_t ack_id,uint32_t result)
{
    uint32_t check_crc=0;
    struct CMD_ACK_STRUCT data_cmd_ack __attribute__((packed));
    data_cmd_ack.header1=0xAA;
    data_cmd_ack.header2=0x55;
    data_cmd_ack.id=cmd_id;
    data_cmd_ack.length=sizeof(data_cmd_ack)-10;
    data_cmd_ack.command=ack_id;
    data_cmd_ack.result=result;
    int len=sizeof(data_cmd_ack)-4;
    check_crc=1;
    data_cmd_ack.check_crc=crc_crc32(check_crc,(uint8_t *)(&data_cmd_ack), len);
    Cout((uint8_t *)(&data_cmd_ack),sizeof(data_cmd_ack));
}
```

Send_Upload_Data 函数定义如下：

```
struct  UPLOAD_DATA
{
uint8_t header1;
uint8_t header2;
uint16_t id;
uint16_t length;
uint8_t param[64];
uint32_t offset;
uint16_t size;
uint8_t cmd;
uint32_t check_crc;
}__attribute__((packed));

struct UPLOAD_DATA upload_data;
void Send_Upload_Data(uint8_t cmd_id,uint8_t cmd,uint32_t offset,uint16_t size,uint8_t* param)
{
    upload_data.header1=0x55;
```

```
    upload_data.header2=0xAA;
    upload_data.id=cmd_id;
    upload_data.length=sizeof(UPLOAD_DATA)-10;
    upload_data.cmd=cmd;
    for(int i=0;i<size;i++)
    upload_data.param[i] = *(param+i);
    upload_data.offset=offset;
    upload_data.size=size;
    int len=sizeof(UPLOAD_DATA)-4;
    uint32_t check_crc=1;
    upload_data.check_crc=crc_crc32(check_crc,(uint8_t *)(&upload_data), len);
    send((uint8_t *)(&upload_data),sizeof(UPLOAD_DATA));
}
```

4. CRC32 校验函数如下：

```
static const uint32_t crc32_tab[] = {
    0x00000000, 0x77073096, 0xee0e612c, 0x990951ba, 0x076dc419, 0x706af48f,
    0xe963a535, 0x9e6495a3, 0x0edb8832, 0x79dcb8a4, 0xe0d5e91e, 0x97d2d988,
    0x09b64c2b, 0x7eb17cbd, 0xe7b82d07, 0x90bf1d91, 0x1db71064, 0x6ab020f2,
    0xf3b97148, 0x84be41de, 0x1adad47d, 0x6ddde4eb, 0xf4d4b551, 0x83d385c7,
    0x136c9856, 0x646ba8c0, 0xfd62f97a, 0x8a65c9ec, 0x14015c4f, 0x63066cd9,
    0xfa0f3d63, 0x8d080df5, 0x3b6e20c8, 0x4c69105e, 0xd56041e4, 0xa2677172,
    0x3c03e4d1, 0x4b04d447, 0xd20d85fd, 0xa50ab56b, 0x35b5a8fa, 0x42b2986c,
    0xdbbbc9d6, 0xacbcf940, 0x32d86ce3, 0x45df5c75, 0xdcd60dcf, 0xabd13d59,
    0x26d930ac, 0x51de003a, 0xc8d75180, 0xbfd06116, 0x21b4f4b5, 0x56b3c423,
    0xcfba9599, 0xb8bda50f, 0x2802b89e, 0x5f058808, 0xc60cd9b2, 0xb10be924,
    0x2f6f7c87, 0x58684c11, 0xc1611dab, 0xb6662d3d, 0x76dc4190, 0x01db7106,
    0x98d220bc, 0xefd5102a, 0x71b18589, 0x06b6b51f, 0x9fbfe4a5, 0xe8b8d433,
    0x7807c9a2, 0x0f00f934, 0x9609a88e, 0xe10e9818, 0x7f6a0dbb, 0x086d3d2d,
    0x91646c97, 0xe6635c01, 0x6b6b51f4, 0x1c6c6162, 0x856530d8, 0xf262004e,
    0x6c0695ed, 0x1b01a57b, 0x8208f4c1, 0xf50fc457, 0x65b0d9c6, 0x12b7e950,
    0x8bbeb8ea, 0xfcb9887c, 0x62dd1ddf, 0x15da2d49, 0x8cd37cf3, 0xfbd44c65,
    0x4db26158, 0x3ab551ce, 0xa3bc0074, 0xd4bb30e2, 0x4adfa541, 0x3dd895d7,
    0xa4d1c46d, 0xd3d6f4fb, 0x4369e96a, 0x346ed9fc, 0xad678846, 0xda60b8d0,
    0x44042d73, 0x33031de5, 0xaa0a4c5f, 0xdd0d7cc9, 0x5005713c, 0x270241aa,
    0xbe0b1010, 0xc90c2086, 0x5768b525, 0x206f85b3, 0xb966d409, 0xce61e49f,
    0x5edef90e, 0x29d9c998, 0xb0d09822, 0xc7d7a8b4, 0x59b33d17, 0x2eb40d81,
    0xb7bd5c3b, 0xc0ba6cad, 0xedb88320, 0x9abfb3b6, 0x03b6e20c, 0x74b1d29a,
    0xead54739, 0x9dd277af, 0x04db2615, 0x73dc1683, 0xe3630b12, 0x94643b84,
    0x0d6d6a3e, 0x7a6a5aa8, 0xe40ecf0b, 0x9309ff9d, 0x0a00ae27, 0x7d079eb1,
    0xf00f9344, 0x8708a3d2, 0x1e01f268, 0x6906c2fe, 0xf762575d, 0x806567cb,
    0x196c3671, 0x6e6b06e7, 0xfed41b76, 0x89d32be0, 0x10da7a5a, 0x67dd4acc,
```

8. 使用范例

8.1 设备安装

1. 模块应牢靠固定在刚性平面上，避免安装在震动大的位置。

2. 模块安装朝向应与车头方向保持下图所述关系。

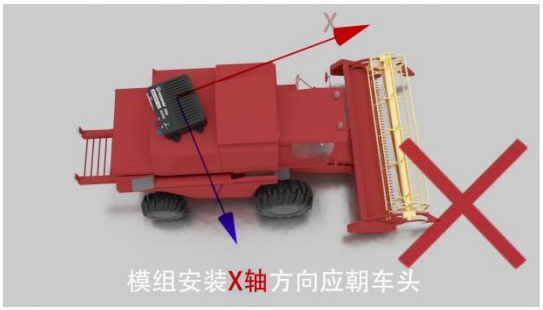
3. PRI 为主天线（定位天线）射频接头，SEC 为副天线（定向天线）射频接头。RTK 的双天线定向结果为主天线指向副天线的射线与地理真北方向的夹角。

正确安装示意图如下

X 轴朝向车头

图 8　正确安装示意图

以下安装方式均是错误安装



8.2 RTK 差分数据导入

用户须通过 RS232-RTK 接口获取 RTK 的 GPGGA 消息，登录千寻或六分的网络 CORS 账号并导入差分数据，进而使 RTK 进入固定解状态。

8.3 配置杆臂参数

杆臂向量为 RTK 主天线相位中心相对 IMU 安装位置的三维矢量（X，Y，Z），单位为米。其中，

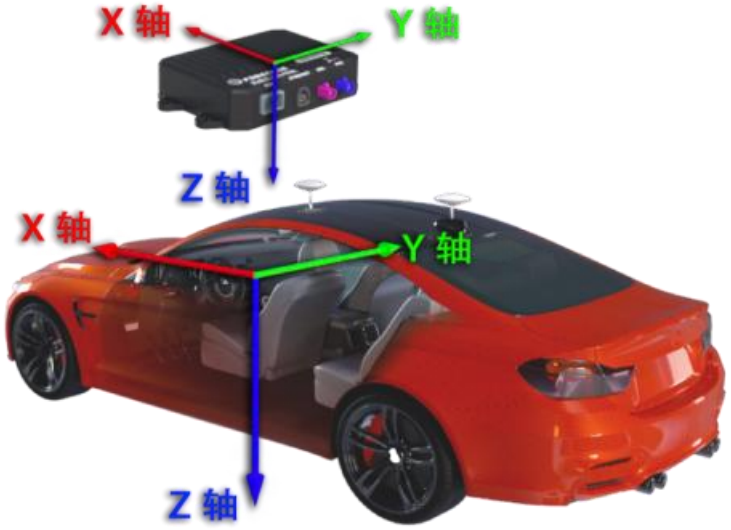
- 在前右下车体坐标系下

若天线在设备中心位置 X 轴的前方，则为正数，否则为负数；

若天线在设备中心位置 Y 轴的右方，则为正数，否则为负数；

若天线在设备中心位置的**上方**为**负数**，否则为**正数（通常在上方）**

坐标系示意图如图 5 所示



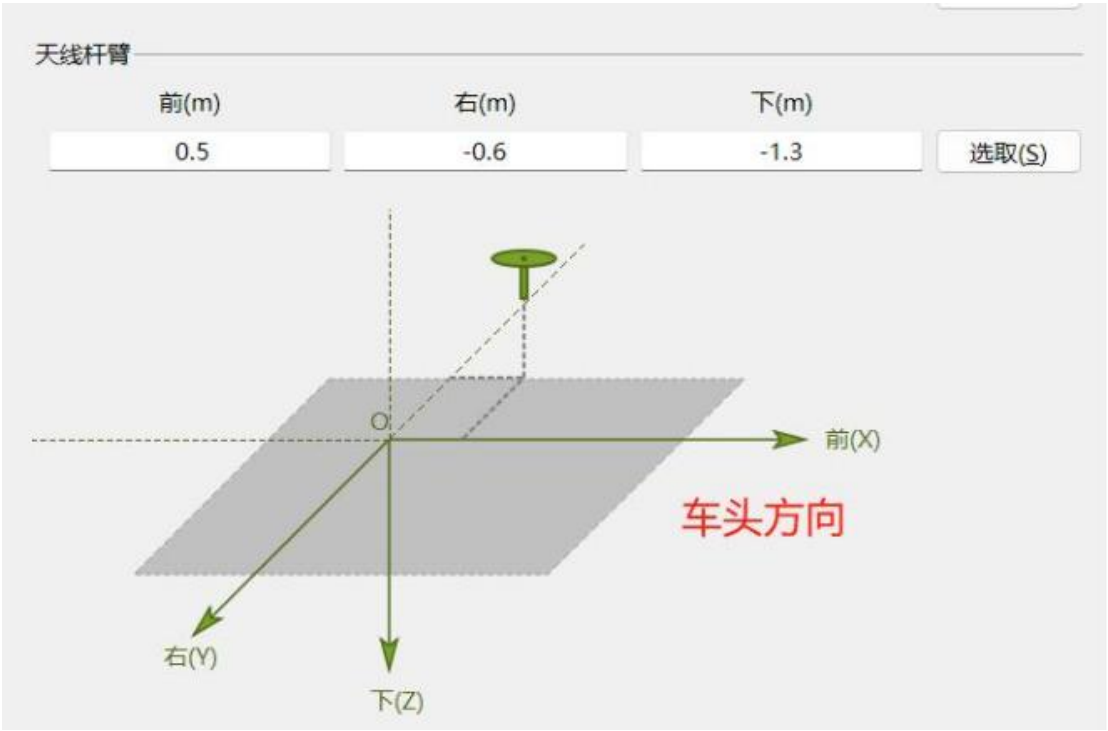


图 9 坐标系示意图

8.4 保存参数

所有配置指令配置完成后，需发送保存参数指令“AT+SAVE\r\n"

9. ROS 驱动

ROS 驱动下载地址:https://www.forsense.cn/download/

9.1 安装 ROS serial

安装 ROS serial 软件包，本例程依赖 ROS 提供的 serial 包实现串口通信。

首先执行如下命令，下载安装 serial 软件包：

```
sudo apt-get install ros -melodic-serial
```

然后输入 `roscd serial` 命令，进入 serial 下载位置，如果安装成功，就会出现如下信息：

```
/opt/ros/melodic/share/serial
```

9.2 编译代码

```
cd FS982_ros/
```

```
catkin_make
```

```
[ 10%] Generating Javascript code from forsense_ins/forsense_insData.msg
Scanning dependencies of target forsense_ins_generate_messages_eus
Scanning dependencies of target forsense_ins_generate_messages_py
[ 20%] Generating Lisp code from forsense_ins/forsense_insData.msg
[ 30%] Generating EusLisp code from forsense_ins/forsense_insData.msg
[ 40%] Generating Python from MSG forsense_ins/forsense_insData
[ 40%] Built target forsense_ins_generate_messages_nodejs
[ 50%] Generating EusLisp manifest code for forsense_ins
[ 50%] Built target forsense_ins_generate_messages_lisp
Scanning dependencies of target forsense_ins_generate_messages_cpp
[ 60%] Generating C++ code from forsense_ins/forsense_insData.msg
[ 70%] Generating Python msg __init__.py for forsense_ins
[ 70%] Built target forsense_ins_generate_messages_cpp
[ 70%] Built target forsense_ins_generate_messages_py
[ 70%] Built target forsense_ins_generate_messages_eus
Scanning dependencies of target forsense_ins_generate_messages
Scanning dependencies of target forsense_ins
[ 70%] Built target forsense_ins_generate_messages
[ 90%] Building CXX object CMakeFiles/forsense_ins.dir/serial_parse.cpp.o
[ 90%] Building CXX object CMakeFiles/forsense_ins.dir/forsense_ins.cpp.o
[100%] Linking CXX executable /home/wenfeng/nav619_ros1/devel/lib/forsense_ins/f
orsense_ins
[100%] Built target forsense_ins
wenfeng@ubuntu:~/nav619_ros1$
```

编译完成

9.3 将 IMU 通过 USB 接入系统

查看是否接入：

lsusb

```
wenfeng@ubuntu:~$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 005: ID 0403:6001 Future Technology Devices International, Ltd FT
232 USB-Serial (UART) IC
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
wenfeng@ubuntu:~$
```

查看 USB 端口号：

ls /dev/ttyU*

```
wenfeng@ubuntu:~$ ls /dev/ttyU*
/dev/ttyUSB0
wenfeng@ubuntu:~$
```

配置打开 USB 转串口权限：

sudo chmod 777 /dev/ttyUSB0

9.4 查看数据

执行 roscore 开启 ROS

回到 serial_imu_ws 文件夹下 执行

source devel/setup.bash

执行启动 rosrun

rosrun forsense_ins forsense_ins

```
wenfeng@ubuntu:~/nav619_ros1$ rosrun forsense_ins forsense_ins
[ INFO] [1695457979.128623440]: /dev/ttyUSB0 is opened.
```

打开新窗口

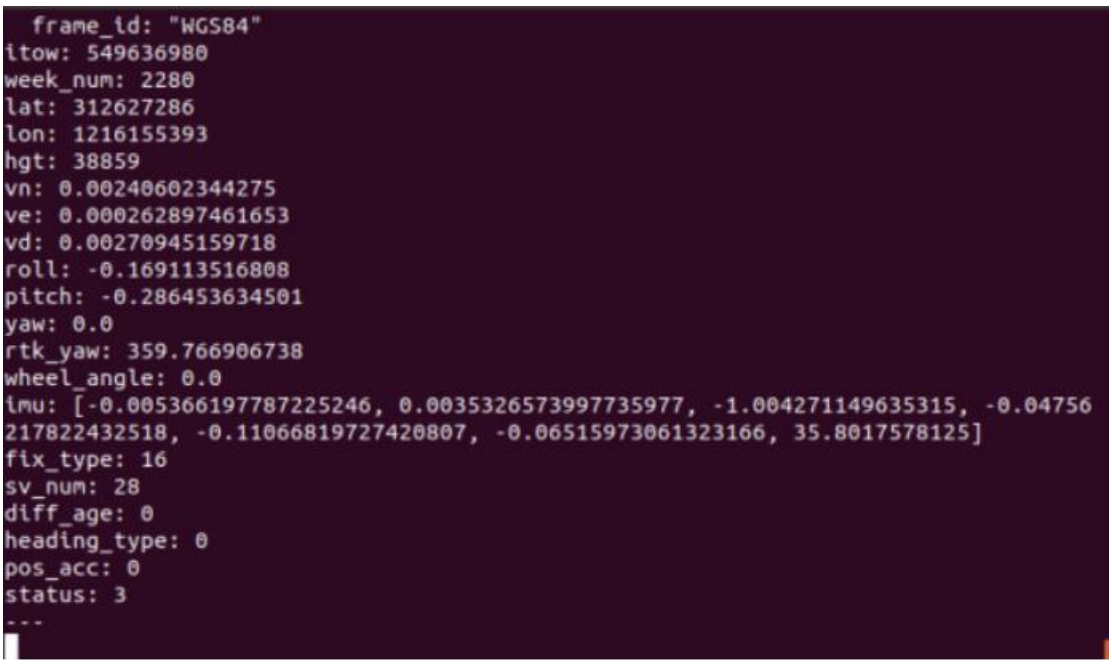
source devel/setup.bash

rostopic list

```
wenfeng@ubuntu:~/nav619_ros1$ rostopic list
/nav619Data
/rosout
/rosout_agg
```

输入命令查看 IMU 数据

```
rostopic echo /FS982Data
```



10. 搭配 G200 使用

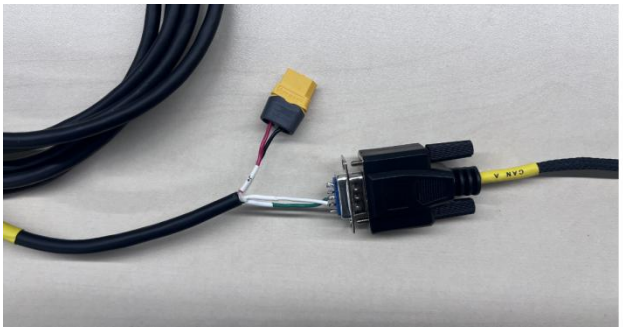
1. G200 安装要求：

A. G200 必须安装在前轮的前桥上(能随前轮一起转动的水平刚性位置)

B. G200 出线口指向车尾安装

2. G200 的 CAN 接口连接到集线束的 CAN A 上，电源是 5V 供电

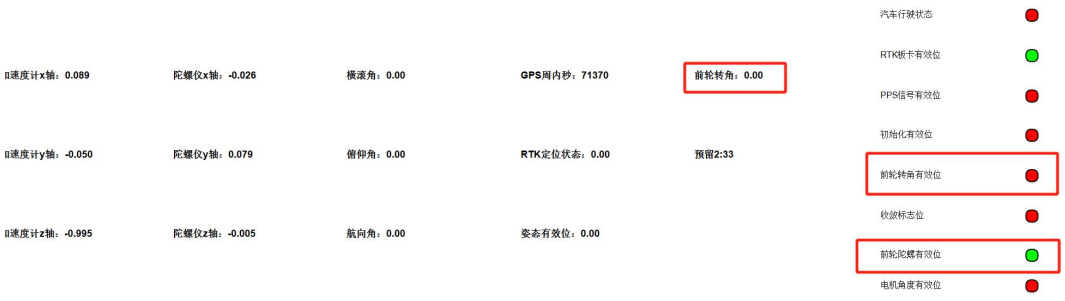
图 10 接线示意图



3. 先测得车辆的轴距，以车辆的轴距 2m 为例，主机的串口连接串口助手，发送 AT+WHEEL_BASE=2,收到应答 WHEEL_BASE=2,再发送保存指令 AT+SAVE,收到应答 OK,设备需重新上电

4. 设备连接 FS982-AG 解码上位机 V3.3.4,进入上位机的首页，选择串口号，波特率(默认 115200)，点击连接，左边的设备信息显示出来表示连接成功，点击左下角的模式启动，正确接入 G200，前轮陀螺有效位显示绿色，否侧为红色，车辆初始化成功之后，前轮转角有效位显示位绿色，否侧为红色，车辆走动后会实时输出前轮转角数据。

图 11 G200 正确接入示意图



11. 附件

标配件：

NAV680-AG 主机

集线束

RS232 串口线

XT60

主天线（定位天线）

副天线（定向天线）

主天线连接线

副天线连接线

选配件：

G200

12. 更新记录

最新手册版本的地址：NAV680-AG_Datasheet_产品手册

版本	日期	状态/注释
版本 1.0	2024.01.09	首次发行
版本 2.0	2024.01.12	增加 RTK 性能指标
版本 3.0	2024.01.15	增加更新记录
版本 4.0	2024.01.18	调整主副天线附件图片
版本 5.0	2024.01.24	去掉 CAN B
版本 6.0	2024.01.26	1.增加 CAN 协议相关内容， 2.增加 RTK 定位状态标
版本 7.0	2024.02.20	修正杆臂配置描述
版本 8.0	2024.02.22	修正旋转角配置应答描述
版本 9.0	2024.03.26	增加坡地模式/搭配 G200 使用描述
版本 10.0	2024.04.09	增加标定双天线