FORSENSE
原极科技

双天线高精度组合导航系统

NAV680-AG

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 2
1.1 产品清单 2
1.2 产品概述 2
1.3 技术指标 3
2. 硬件组成 5
2.1 机械尺寸 5
2.2 接口说明 7
2.3 数据线接口定义 8
2.4 电气物理特性 10
3. 硬件接线方式 11
3.1 硬件接线方式 11
3.2 天线 11
4. 固件升级 12
4.1 通过上位机 12
4.1.1 R232串口直连 12
4.1.2 使用附带USB转CAN模块 12
4.2 串口推送 14
5. ROS驱动 21
5.1 安装 ROS serial 21
5.2 编译代码 22
5.3 将 IMU 通过 USB 接入系统 23
5.4 查看数据 24
6. 搭配G200使用 26
7. 常见问题列表 29
8. 更新记录 31

1. 产品简介

1.1 产品清单

在您打开包装箱时，请确认包装箱中产品：

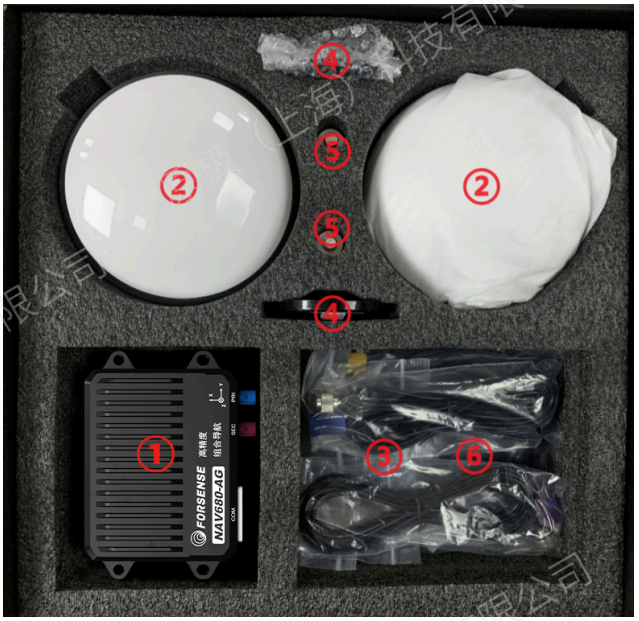
名称	数量	示意图
①NAV680-AG组合导航系统	1	
②卫星天线	2	
③天线线缆 主天线型号:TNC公头转fakra母头(C型)(蓝色) 副天线型号:TNC公头转fakra母头(D型)(紫色)	2	
④天线吸盘	2	
⑤天线柱	2	
⑥集线束(含3个RS232、2个CAN、1个PPS、1个电源接口(9-24V).	1	

图 1 产品实物图

若有缺失，请及时与销售人员联系

1.2 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，集成全星座、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，支持BDS B1I、B2I、B3I；GPS L1、L2、L5；GLONASS G1、G2；Galileo E1、E5a、E5b；QZSS L1、L2、L5。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。

1.3 技术指标

表 1

姿态精度	**Roll/Pitch :<0.2° rms** **Heading:<0.3° rms** **(车速大于1km/h，阿克曼转向结构车载场景)**	
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： 水平1.5m/高程2.5m RTK: 水平1cm+1ppm 高程2cm+1ppm
	双天线定向精度(RMS)	0.1°/1m基线
	速度精度(RMS)	0.03m/s
	PPS精度（RMS）	20ns
	更新率	10hz
	RTK初始化时间	<5s（典型值）

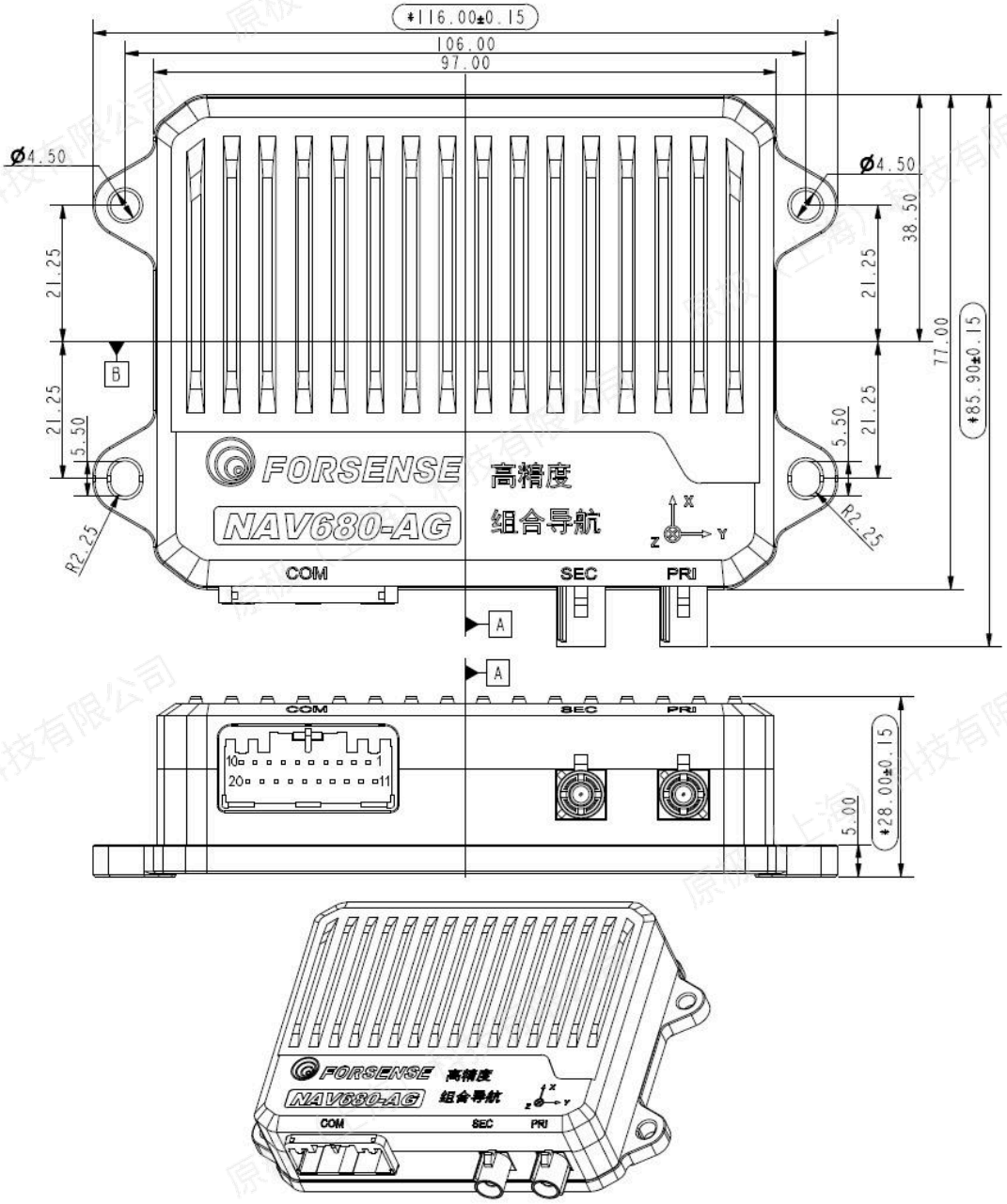
接口	3×RS232 2×CAN 2×GNSS 天线接口 1×电源接口 1×PPS接口

原极（上海）科技有限公司

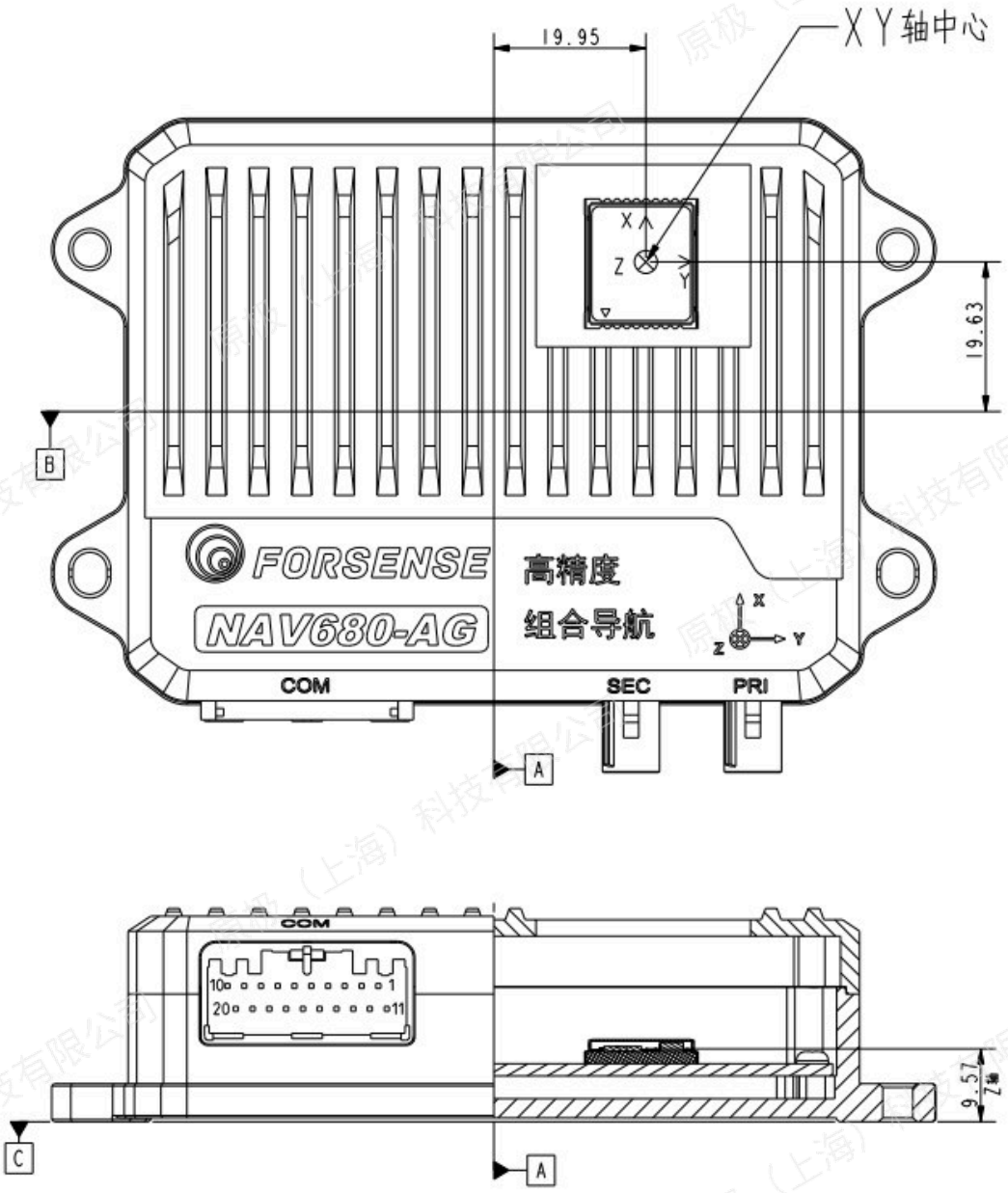
2. 硬件组成

2.1 机械尺寸

图 2 机械尺寸（单位：mm）



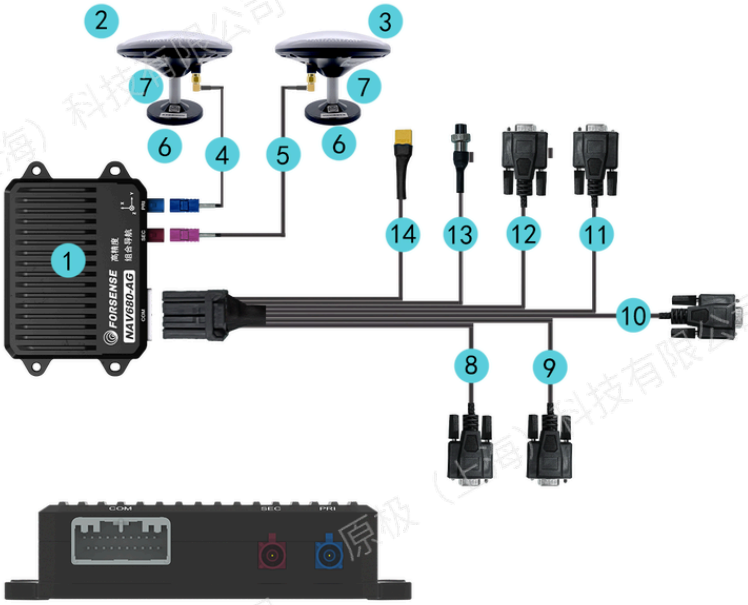
IMU机械中心尺寸


19.95
X Y轴中心
19.63
X
Z
Y
B
FORSENSE
NAV680-AG
高精度
组合导航
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-RS232-RTK接口	1个
13	集线束-IO接口	1个
14	集线束-电源接口	1个

2.3 数据线接口定义

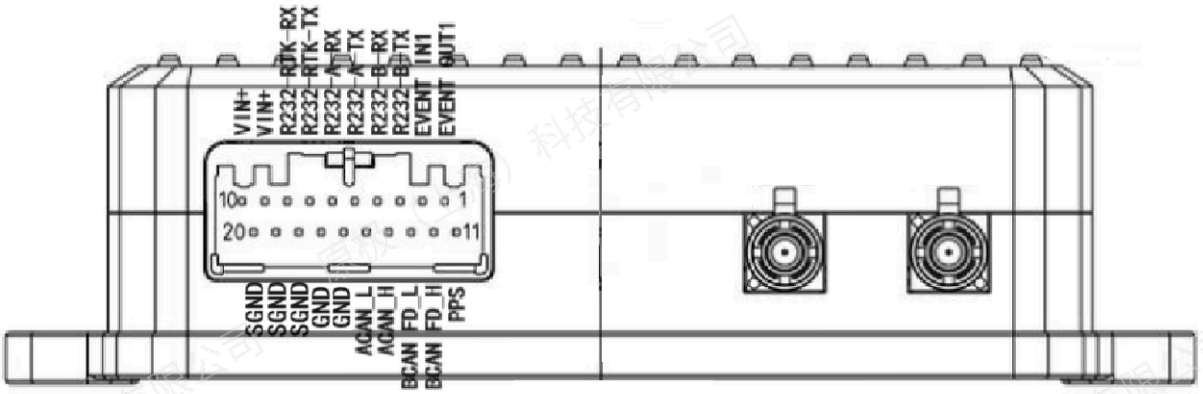
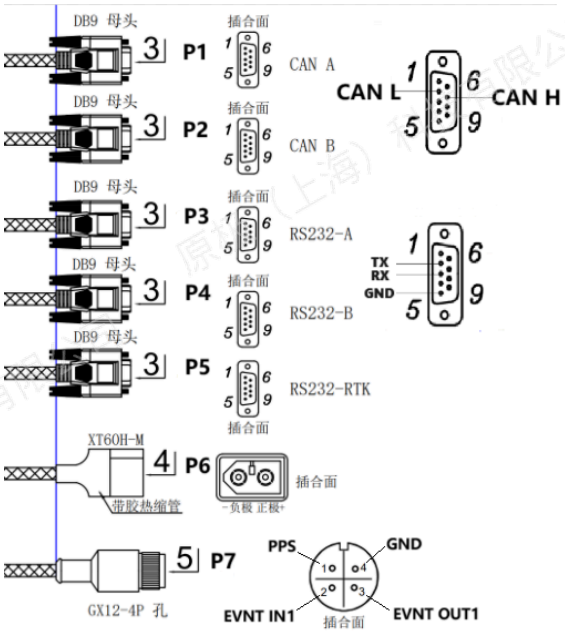


图 3 数据线接口定义图

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	RS232-B
4	MCU RX3 232	
5	MCU TX2 232	RS232-A
6	MCU RX2 232	
7	RTK TX1 232	RS232-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）
15	ACAN_L	
16	GND	电源地
17	GND	
18	SGND	信号地
19	SGND	
20	PGND	屏蔽地



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）。

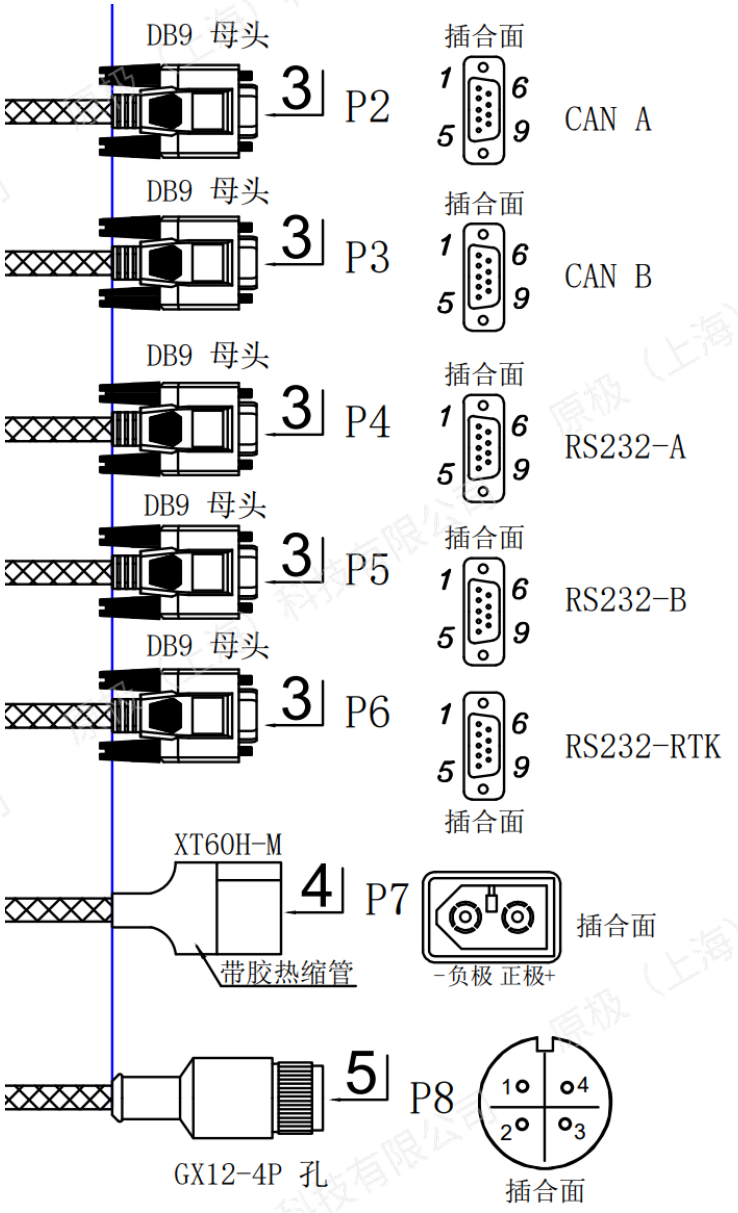
2.4 电气物理特性

表 2

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		1.8		W
使用温度	T	-40	\	85	℃
存储温度	T	-40	\	95	℃
湿度	95%非凝露				
尺寸	116*85.9*28mm				
重量	203g				

3. 硬件接线方式

3.1 硬件接线方式



3.2 天线

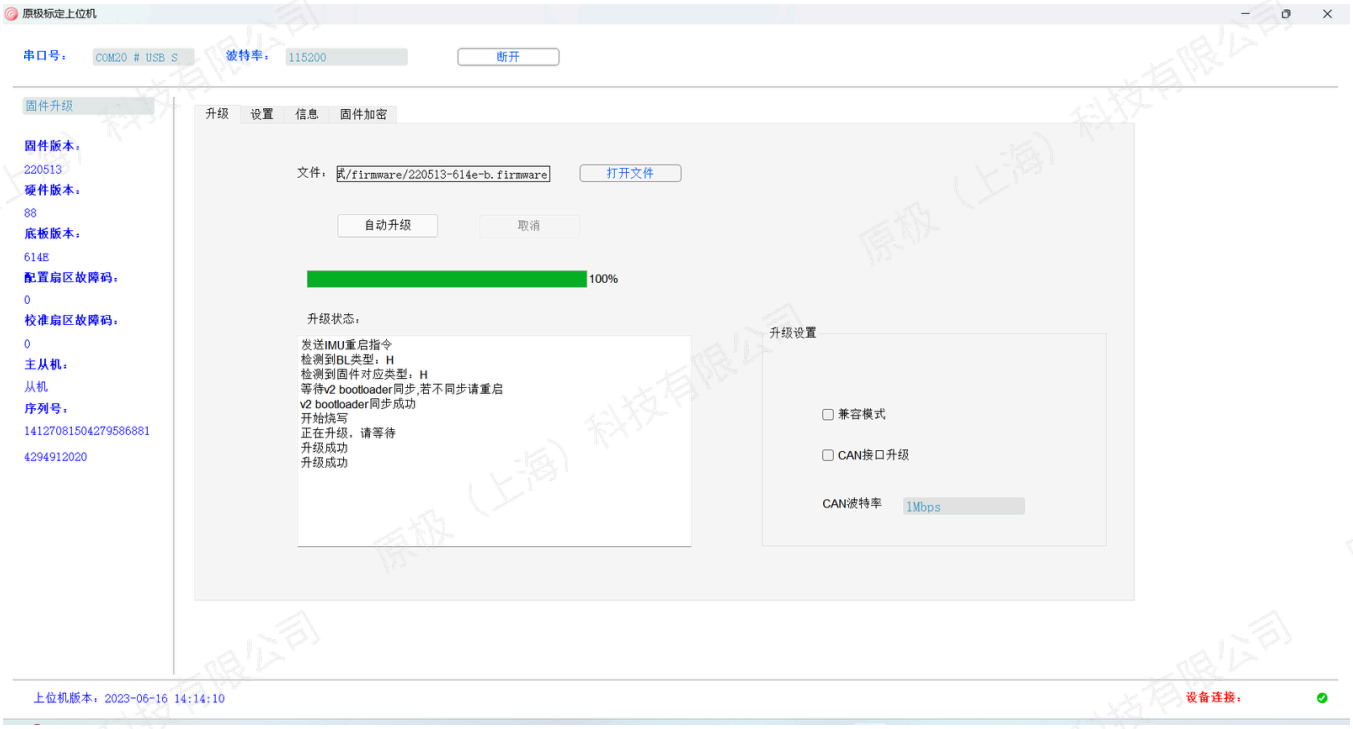
NAV680-AG 引出 PRI 和 SEC 主副天线接口，不接天线，使用万用表测试，即空载时提供电压为 DC4.8～5.4V；模块射频口接天线时，常温下，工作电流为 30～100mA 时测试，能对外提供 DC4.6V±0.2V 的天线馈电。NAV680-AG 采用有源天线时注意与天线间的 50 欧姆阻抗匹配。

4. 固件升级

4.1 通过上位机

4.1.1 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。



4.1.2 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。


串口号：
COM31 # USB-S
波特率：
115200
断开
固件升级
固件版本：
230502
硬件版本：
42652
底板版本：
AH20
配置扇区故障码：
0
校准扇区故障码：
0
主从机：
从机
序列号：
14125773334279062606
4294908421
升级
设置
信息
固件加密
文件：
y/Desktop/20230525-ahrs200.firmware
打开文件
自动升级
取消
100%
升级状态：
发送IMU重启指令
开始设置CAN波特率为1Mbps
当前波特率为1Mbps，不用重新设置
退出CAN设置模式成功
检测到固件对应类型：H
等待v2 bootloader同步,若不同步请重启
v2 bootloader同步成功
开始烧写
正在升级，请等待
升级成功
退出CAN设置模式成功
升级成功
升级设置
兼容模式
CAN接口升级
CAN波特率
250Kbps

4.2 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,
```

5. ROS驱动

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



5.1 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

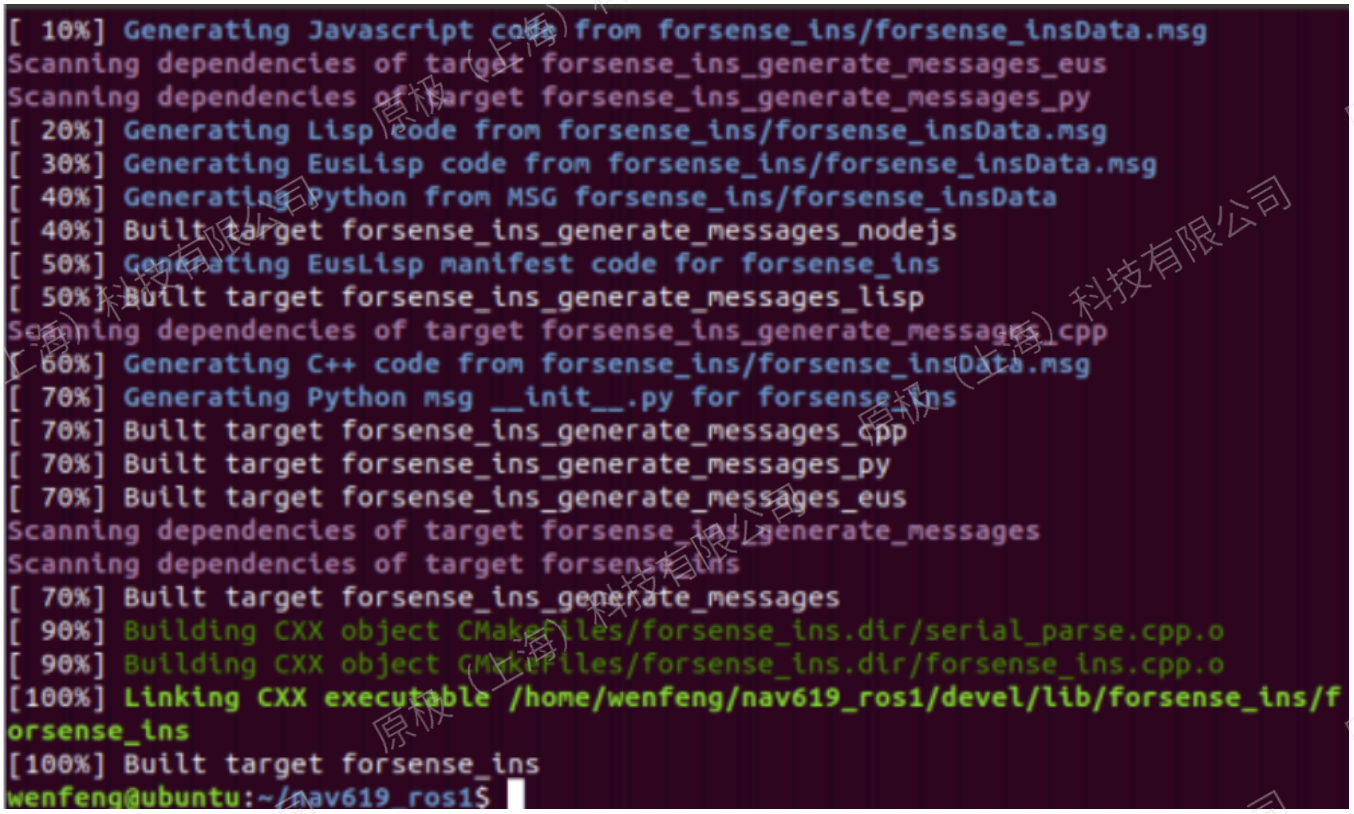
下信息：

/opt/ros/melodic/share/serial

5.2 编译代码

cd FS982_ros/

catkin_make

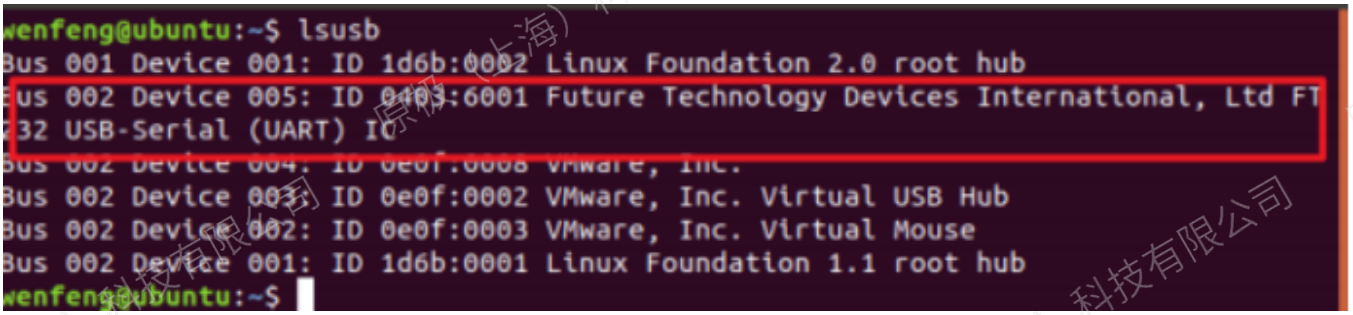


编译完成

5.3 将 IMU 通过 USB 接入系统

查看是否接入：

lsusb



查看 USB 端口号：

ls /dev/ttyU*



配置打开 USB 转串口权限：

sudo chmod 777 /dev/ttyUSB0

5.4 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

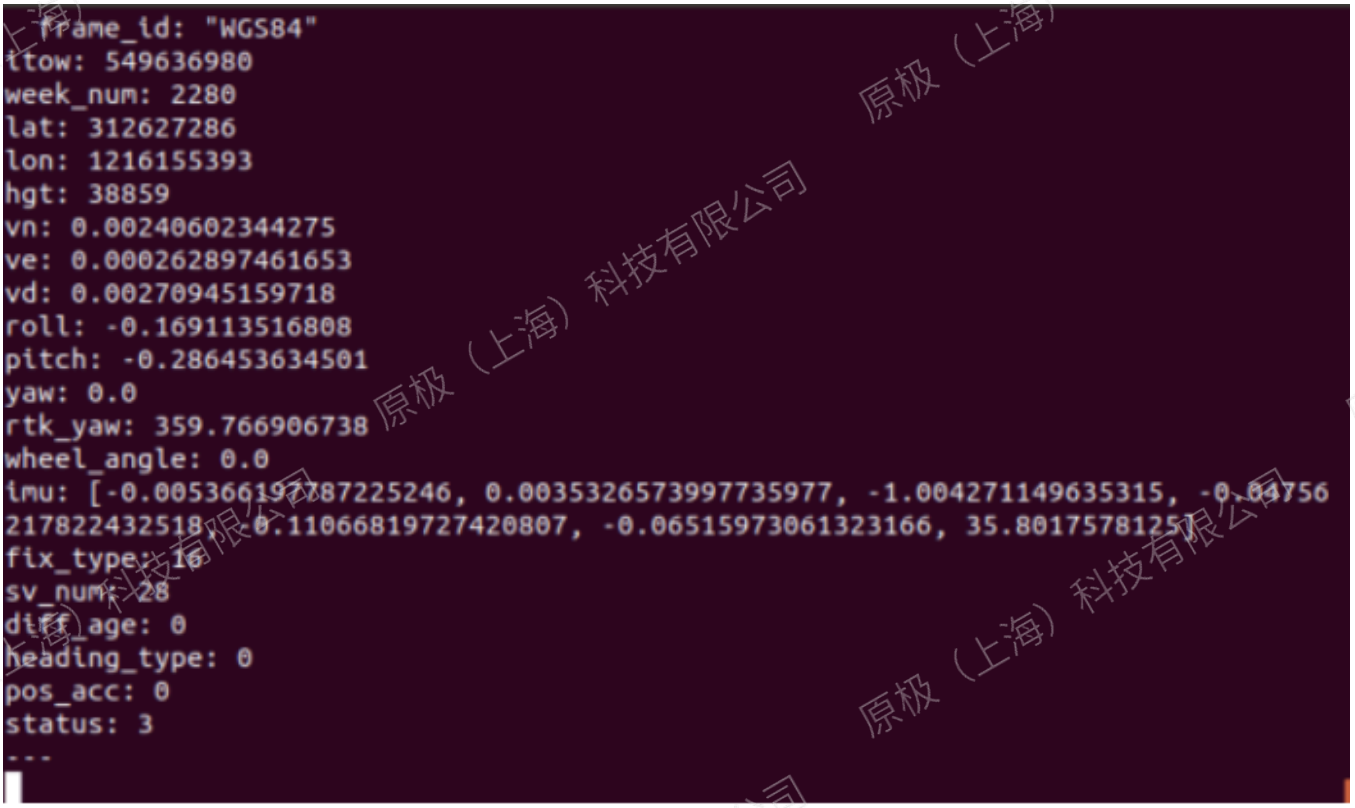
source devel/setup.bash

rostopic list

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

输入命令查看 IMU 数据

rostopic echo /FS982Data


frame_id: "WGS84"
itow: 549636980
week_num: 2280
lat: 312627286
lon: 1216155393
hgt: 38859
vn: 0.00240602344275
ve: 0.000262897461653
vd: 0.00270945159718
roll: -0.169113516808
pitch: -0.286453634501
yaw: 0.0
rtk_yaw: 359.766906738
wheel_angle: 0.0
imu: [-0.005366197787225246, 0.0035326573997735977, -1.004271149635315, -0.04756
217822432518, -0.11066819727420807, -0.06515973061323166, 35.80175781251]
fix_type: 16
sv_num: 28
diff_age: 0
heading_type: 0
pos_acc: 0
status: 3
---

6. 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，输入电源最大支持15V，建议5V或12V供电。

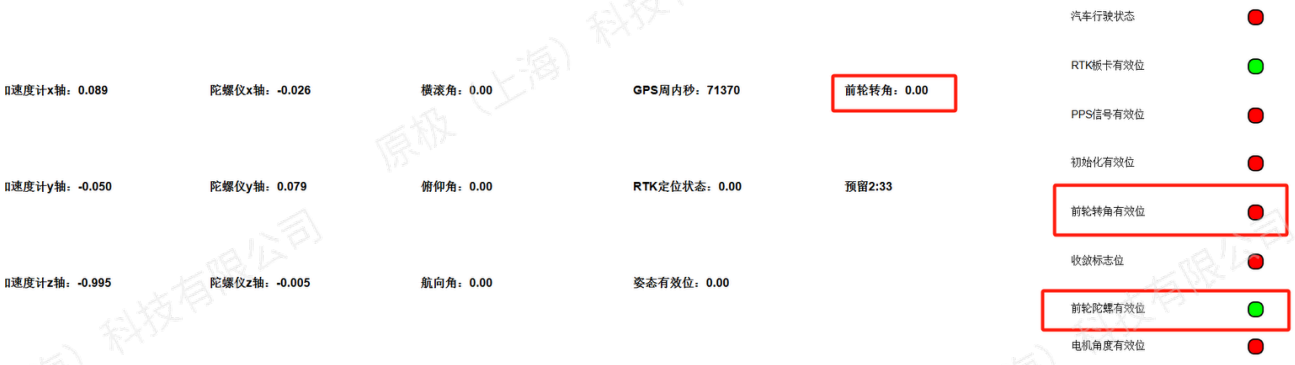
图10 接线示意图



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



标配件：



NAV680-AG 主机



集线束



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





RS232 串口线 XT60



G200

7. 常见问题列表

问题	解答
上位机无法连接	请检查串口是否被占用、产品是否正常通电，若在连接过程中上位机断连，可能是串口松动，可插拔串口线后重新打开上位机。
定位状态始终是1	确认差分数据是否正常接入，排查波特率，差分账号，挂载点等信息是否正确
双天线定向状态一直达不到50	请排查以下因素： 1.如定向状态始终为0，请排查副天线是否正确认接入，天线馈线是否损坏都因素。 2.如定向状态不是0但是一直到不了50，请确认测试环境是否开阔，双天线距离是否在50CM以上。
接有源天线情况下信号正常， 接无源天线情况下无信号	无源天线馈线不能超过1.5M。
天线接有源功分器情况下信号正常，使用设备给天线供电的情况下无信号	确认天线馈线是否存在短路或者上电情况下插拔天线导致静电等原因使得天线供电电路异常，目前电路设计中已有保险丝设计，重新上电后可以恢复。
无法初始化	初始化需满足以下条件， 单天线：定位状态固定解+行驶速度0.5m/s以上+直线行驶带加减速 双天线：定位状态固定解+定向状态50
串口丢包	请排查以下因素：1.串口线需要支持至少115200波特率 2.电脑串口延时需要配成2ms
怎么判断G200物理连接是否正确	通过解析状态位的数字，看bit5是否有效
连接G200情况下，前轮陀螺有效位无效	使用CAN分析仪检查G200的CAN波特率与AG的CAN波特率是否一致，如不一致，请修改为一致，具体修改指令参考G200与AG手册参数配置页面
连接G200情况下，前轮陀螺有效位有效，前轮角度异常	请排查以下因素：1.前轮角度输出正确的前提是，AG设备要完成初始化之后，才能输出正确的前轮角度 2.轴距是否配置，未配置轴距会导致前轮转角无效
CAN配置杆臂，双天线角度等信息配置后结果与配置不符	排查是否按照以下方式配置： 小端模式，低字节在前，单位为CM，负数需要补码
nmea协议与二进制协议是否可以同步输出?	不支持，只能同时输出其中一种协议
连接G200情况下，前轮转角左打死右打死角度不对称	请排查以下因素： 1.G200安装是否水平安装 2.如果仅在静止状态上电，需要在0度的时候上电，或者人工驾驶直行一段即可修正回来
拔掉天线的情况为何还有卫星数?	UM982信号跟踪能力较强，如果周围有信号源，拔掉天线的情况会通过耦合的方式继续跟踪一段时间，实际卫星失锁场景下天线仍然处于连接状态，外部信号无法进入，不影响使用

使用金属锅盖遮挡天线模拟失锁，状态仍然固定解	错误测试方法： 金属锅盖只能遮挡正上方的信号，侧向信号通过车顶反射到锅盖底部再反射到天线上，会造成假固定现象
ROS驱动读取不到数据	设备未配置上电默认输出组合数据流，配置上电默认输出组合数据可解决
航向误差大	可能存在以下几种原因： 1.杆臂,轴距配置错误，单位被放大了10倍， 2.确认是否刚性固定 3.是否远离强振动源
与参考基准姿态误差超标	确认统计时是否扣除安装误差（一般以与基准之间的误差均值为安装误差）
小端配置CAN指令时如何补码操作	补码是针对负数进行的，-20，X-65536=-20 X=65516 65516对应的十六进制是FFEC,65535对应的十六进制是FFFF,因为是从0开始计算的，最大值是65536，当前结果减去最大值为补码值

8. 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1.1	2024.01.12	增加RTK性能指标
版本 1.2	2024.01.15	增加更新记录
版本1.3	2024.01.18	调整主副天线附件图片
版本1.4	2024.01.24	去掉CAN B
版本1.5	2024.01.26	1.增加CAN协议相关内容， 2.增加RTK定位状态标
版本1.6	2024.02.20	修正杆臂配置描述
版本1.7	2024.02.22	修正旋转角配置应答描述
版本1.8	2024.03.26	增加坡地模式/搭配G200使用描述
版本1.9	2024.06.12	1.增加部分新增AT指令 2.增加CAN协议经纬度，高程报文 3.增加常见问题清单
版本1.10	2025.01.23	调整表图序