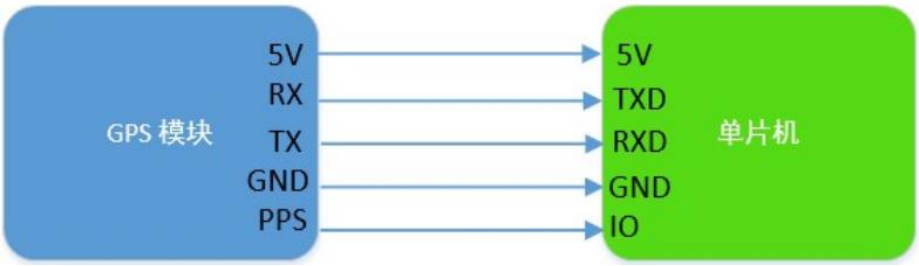
简介

这是一款可以实时了解当前所在位置的小尺寸定位导航模块，基于一款六合一多模卫星导航定位芯片 AT6558 研发而成，支持多种卫星导航系统，包含 32 个跟踪通道，可以同时接收六个卫星导航系统的 GNSS 信号，包括中国的 BDS（北斗卫星 导航系统），美国的 GPS，俄罗斯的 GLONASS，欧盟的 GALILEO，日本的 QZSS 以及卫星增强系统 SBAS（WAAS，EGNOS，GAGAN，MSAS），并且实现联合定位、导航与授时。

该模块兼容性强，串口输出模式可兼容各类配置了串口输出的主板，arduino、树莓派、STM32 的各类主板均能使用；定位精度误差实测为 3m 左右，和各类智能手机定位基本一致；功耗低至 0.1W，搭配一个小电源就能长时间持续稳定工作。



连线图

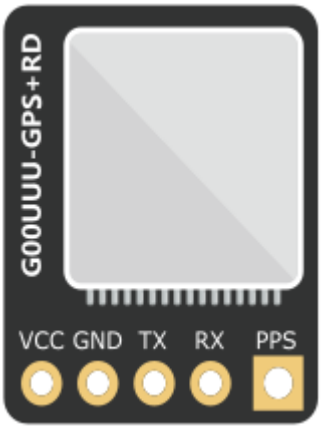
小巧的体积能让它嵌入到各类无人机，智能小车中，可适用于车载导航、手持定位、可穿戴设备等领域，还能直接替换 Ublox MAX 系列模块。

技术规格

- 工作电压：3.3-5V
- 具备 SMA 天线接口和 IPEX 天线接口
- 板载 E2PROM 可设置保存波特率等信息
- 板载 XH414 充电电子，加速热启动搜星
- 支持 A-GNSS
- 冷启动捕获灵敏度：-148dBm
- 跟踪灵敏度：-162dBm
- 定位精度：2.5 米（CEP50，开阔地）
- 首次定位时间：32 秒（也有可能是几分钟，要看具体环境而定）
- 低功耗：连续运行<25mA
- 内置天线检测及天线短路保护功能

- 尺寸：13.1mm x 15.7mm

引脚说明

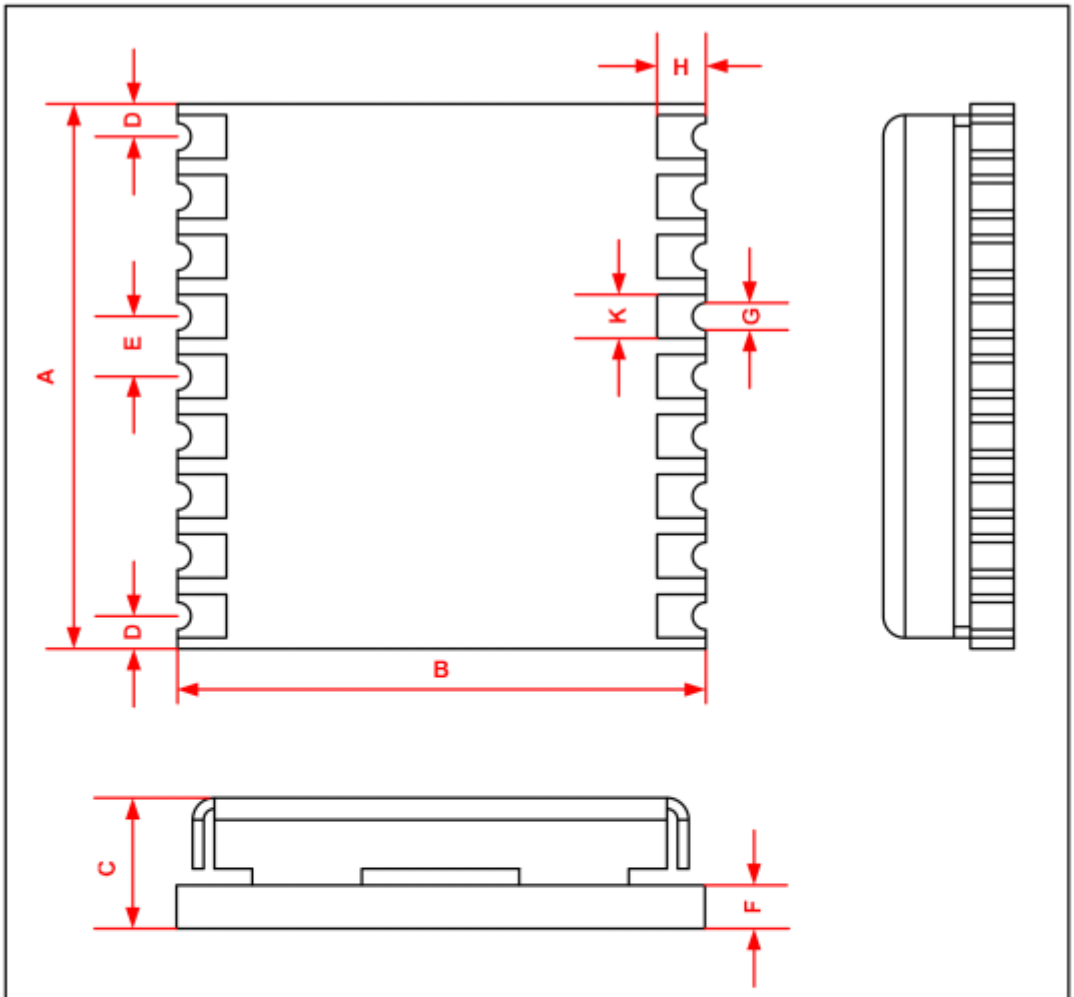


名称	功能
VCC	电源输入（5V）
GND	地线
TX	串口发射
RX	串口接收
PPS	秒脉冲输出

模块介绍

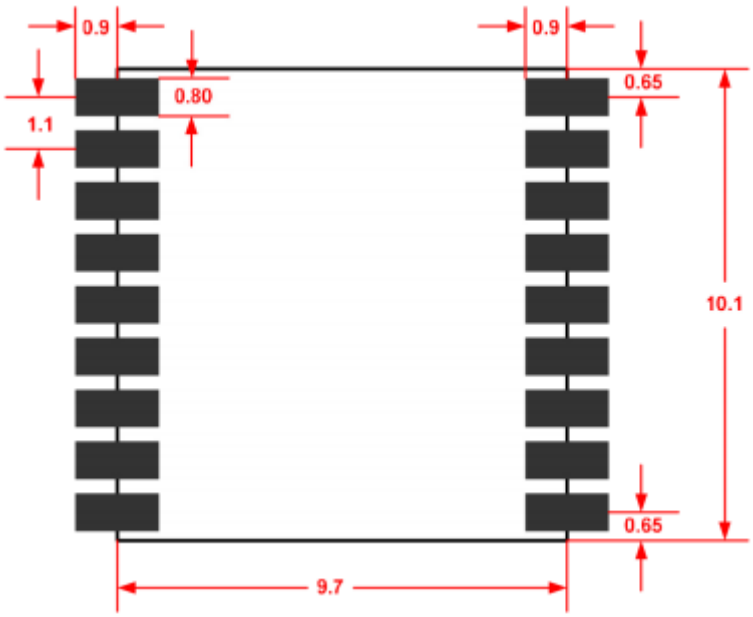
技术描述

- **外观尺寸（单位：mm）**

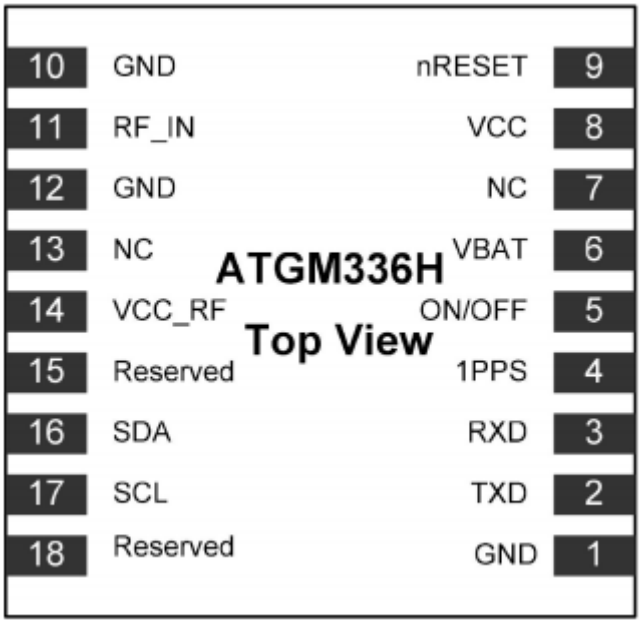


Symbol	Min.(mm)	Typ.(mm)	Max.(mm)
A	10.0	10.1	10.7
B	9.6	9.7	9.8
C	2.2	2.4	2.6
D	0.55	0.65	0.95
E	1.0	1.1	1.2
F		0.8	
G	0.4	0.5	0.6
H	0.8	0.9	1.0
K	0.7	0.8	0.9

- **PCB layout (单位：mm)**



- **PIN 排列图**



- **管脚定义**

引脚编号	名称	I/O	描述	电气特性
1	GND	I	地	
2	TXD	O	导航数据输出	NMEA0183 协议
3	RXD	I	交互命令输入	配置命令输入
4	1PPS	O	秒脉冲输出	
5	ON/OFF	I	模块关断控制，低电平有效	

6	VBAT	I	RTC 及 SRAM 后备电源	提供 1.5~3.6V 电源以保证模块热启动
7	NC	I		
8	VCC	I	模块电源输入	直流 3.3V±10%，100mA
9	nRESET	I	模块复位输入，低电平有效	不用时悬空
10	GND	I	地	
11	RF_IN	I	天线信号输入	
12	GND	I	地	
13	NC			
14	VCC_RF	O	输出电源	+3.3V，可给天线供电
15	保留			悬空
16	SDA	I/O	I²C 数据接口	悬空
17	SCL	O	I²C 时钟接口	悬空
18	保留			悬空

- **电气参数**

极限参数

参数	符号	最小值	最大值	单位
模块供电电压（VCC）	Vcc	-0.3	3.6	V
备份电池电压（VBAT）	Vbat	-0.3	3.6	V
数字输入引脚电压	Vin	-0.3	Vcc+0.2	V
最大可承受 ESD 水平	VESD（HBM）		2000	V

运行条件

参数	符号	最小值	典型值	最大值	单位
供电电压	VCC	2.7	3.3	3.6	V
Vcc 峰值电流（不包含天线）	Ipeak			100	mA

备份电源	Vbat	1.5	3.0	3.6	V
备份电源（Vbat 电流）	Ibat		10		uA
输入引脚	Vil			0.2*Vcc	V
	Vih	0.7*Vcc			V
输出引脚	Vol Io=-12mA			0.4	V
	Voh Io=12mA	Vcc-0.5			V
有源天线输出电压	VCC_RF		3.3		V
天线短路保护电流，电源来自 VCC_RF （=3.3V）	Lant Short		50		mA
天线开路电流，电源来自 VCC_RF （=3.3V）	Lant Open		3		mA
天线增益	Gant	15		30	dB

- **技术规范**

指标	技术参数
信号接收	BDS/GPS/GLONASS/GALILEO/QZSS/SBAS
射频通道数目	三通道射频，支持全星座 BDS、GPS 和 GLONASS 同时接收
冷启动 TTFF	≤35S
热启动 TTFF	≤1S
重捕获 TTFF	≤1S
冷启动捕获灵敏度	-148dBm
热启动捕获灵敏度	-156dBm
重捕获灵敏度	-160dBm
跟踪灵敏度	-162dBm

定位精度	< 2m（1σ）
测速精度	< 0.1m/s（1σ）
授时精度	< 30ns（1σ）
定位更新率	1Hz（默认），最大 10Hz
串口特性	波特率范围：4800~115200bps，默认 9600bps，8 个数据位，无校验，1 个停止位
协议	NMEA0183
最大高度	18000m
最大速度	515m/s
最大加速度	4g
后备电池	1.5-3.6V
电源供电	2.7-3.6V
GPS&BD 典型功耗	< 25mA@3.3V
工作温度	-40~+85℃
存储温度	-45~+125℃
尺寸	10.1mm*9.7mm*2.4mm
重量	0.6g

数据解析

测试最好是带电脑到户外空旷地进行，若是把天线放在阳台外面的话，有一定几 率定位失败，这个受楼距，遮挡物等因素影响。空旷地首次定位一般是一分钟以 内。板载 LED 保持一定的频率闪烁证明定位成功了，我们用串口来看下数据，默认波特率是 9600.

定位成功后 用串口显示数据，串口显示数据如下

$GNGGA,084852.000,2236.9453,N,11408.4790,E,1,05,3.1,89.7,M,0.0,M,,*48
$GNGLL,2236.9453,N,11408.4790,E,084852.000,A,A*4C
$GPGSA,A,3,10,18,31,,,,,,,,,,6.3,3.1,5.4*3E
$BDGSA,A,3,06,07,,,,,,,,,,6.3,3.1,5.4*24
$GPGSV,3,1,09,10,78,325,24,12,36,064,,14,26,307,,18,67,146,27*71
$GPGSV,3,2,09,21,15,188,,24,13,043,,25,55,119,,31,36,247,30*7F
$GPGSV,3,3,09,32,42,334,*43

$BDGSV,1,1,02,06,68,055,27,07,82,211,31*6A

$GNRMC,084852.000,A,2236.9453,N,11408.4790,E,0.53,292.44,1412

16,,,A*7 5 $GNVTG,292.44,T,,M,0.53,N,0.98,K,A*2D

$GNZDA,084852.000,14,12,2016,00,00*48

$GPTXT,01,01,01,ANTENNA OK*35

数据里面我们能看到三种数据类型

GN、GP、BD 分别代表 双模模式、GPS 模式、北斗模式

NMEA0183 协议 帧格式内容可以参考以下几个表格

(1) $GPGGA （GPS 定位信息）

序号	名称	样例数据	单位	描述
	消息 ID	$GPGGA		GGA 协议的数据头
<1>	定位点的 UTC 时间	161229.487		格式：hhmmss.sss
<2>	纬度	3723.2475		格式：ddmm.mmmm
<3>	纬度方向	N		N：北纬；S：南纬
<4>	经度	12158.3416		格式：dddmm.mmmm
<5>	经度方向	W		W：西经；E：东经
<6>	GPS 定位状态指示	1		0：未定位 1：无差分，SPS 模式，定位有效 2：无差分，SPS 模式，定位有效 3：PPS 模式，定位有效
<7>	使用卫星数量	07		从 00 到 12（不足 10 的前面补 0）
<8>	水平精度衰减因子	1.0		范围：0.5-99.9
<9>	海平面高度	9.0	米	范围：-9999.9-9999.9
<10>	高度单位	M		M 表示高度单位为米
<11>	大地椭球面相对于海平面的高度		米	范围：-999.9-9999.9
<12>	高度单位	M		M 表示高度单位为米
<13>	差分修订时间		秒	从最近一次接收到差分信号开始的秒数，如果不是差分定位，此项为空

<14>	差分参考基站 ID 号	0000		范围：0000-1023，如果不是差分定位，此项为空
hh	校验和	18		$与*之间所有字符 ASCII 码的校验和（各字节做异或运算，得到校验和后，再转换成 16 进制格式的 ASCII 字符）
	回车和换行	<CR><LF>		代表协议帧结束

(2) $GPGLL （地理定位信息）

序号	名称	样例数据	单位	描述
	消息 ID	$GPGLL		GLL 协议的数据头
<1>	纬度	3723.2475		格式：ddmm.mmmm
<2>	纬度方向	N		N：北纬；S：南纬
<3>	经度	12158.3416		格式：dddmm.mmmm
<4>	经度方向	W		W：西经；E：东经
<5>	定位点的 UTC 时间	161229.487		格式：hhmmss.sss
<6>	数据状态	A		A：定位数据有效；V：定位数据无效
hh	校验和	2C		

(3) $GPGSA （当前卫星信息）

序号	名称	样例数据	单位	描述
	消息 ID	$GPGSA		GSA 协议的数据头
<1>	定位模式	A		M：手动；A：自动
<2>	定位类型	3		1：无定位信息 2：二维定位 3：三维定位
<3>	第 1 信道正在使用的卫星 PRN 编码编号	07		PRN 码：（Pseudo Random Noise，伪随机噪声码）范围是 01 至 32，最多可接收 12 颗卫星信息
<4>	第 2 信道正在使用的卫星 PRN 编码编号	02		同上
<5>	第 3 信道正在使用的卫星 PRN 编码编号	26		同上

<6>	第 4 信道正在使用的卫星 PRN 编码编号	27		同上
<7>	第 5 信道正在使用的卫星 PRN 编码编号	09		同上
<8>	第 6 信道正在使用的卫星 PRN 编码编号	04		同上
<9>	第 7 信道正在使用的卫星 PRN 编码编号	15		同上
<10>	第 8 信道正在使用的卫星 PRN 编码编号			同上
<11>	第 9 信道正在使用的卫星 PRN 编码编号			同上
<12>	第 10 信道正在使用的卫星 PRN 编码编号			同上
<13>	第 11 信道正在使用的卫星 PRN 编码编号			同上
<14>	第 12 信道正在使用的卫星 PRN 编码编号			同上
<15>	PDOP 综合位置精度因子	1.8		范围：0.5-99.9
<16>	HDOP 综合位置精度因子	1.0		范围：0.5-99.9
<17>	VDOP 综合位置精度因子	1.5		范围：0.5-99.9
hh	校验和	2C		
	回车和换行	<CR><LF>		代表协议帧结束

(4) $GPGSV（可见卫星信息）

序号	名称	样例数据	单位	描述
	消息 ID	$GPGSV		GSV 协议的数据头
<1>	本次 GSV 语句的总数目	2		范围：1-3
<2>	当前 GSV 语句序号	1		范围：1-3
<3>	当前可见卫星总数	07		范围：00-12
<4>	卫星 PRN 码编号	07		范围：01-32
<5>	卫星仰角	79	度	范围：00-90
<6>	卫星方位角	048	度	范围：000-359
<7>	信噪比	42	dBHz	范围：00-99
<4>	卫星 PRN 码编号	02		范围：01-32
<5>	卫星仰角	51	度	范围：00-90
<6>	卫星方位角	062	度	范围：000-359
<7>	信噪比	43	dBHz	范围：00-99
<4>	卫星 PRN 码编号	26		范围：01-32
<5>	卫星仰角	36	度	范围：00-90
<6>	卫星方位角	256	度	范围：000-359
<7>	信噪比	42	dBHz	范围：00-99
<4>	卫星 PRN 码编号	27		范围：01-32
<5>	卫星仰角	27	度	范围：00-90
<6>	卫星方位角	138	度	范围：000-359
<7>	信噪比	42	dBHz	范围：00-99
hh	校验和	71		
	回车和换行	<CR><LF>		代表协议帧结束

(5) $GPRMC（最简定位信息）

序号	名称	样例数据	单位	描述
	消息 ID	$GPRMC		RMC 协议的数据头
<1>	定位点的 UTC 时间	161229.487		格式：hhmmss.sss
<2>	定位状态	A		A：定位；V：导航
<3>	纬度	3723.2475		格式：ddmm.mmmm
<4>	纬度方向	N		N：北纬；S：南纬
<5>	经度	12158.3416		格式：dddmm.mmmm
<6>	经度方向	W		W：西经；E：东经

<7>	对地航速	0.13	Knots	范围：000.0-999.9
<8>	对地航向	309.62	度	以真北为参考基准，二维方向指向，相当于二维罗盘
<9>	数据状态	A		A：定位数据有效；V：定位数据无效
<10>	磁偏角		度	范围：000-180
<11>	磁偏角方向			E:东，W:西
hh	校验和	10		
	回车和换行	<CR><LF>		代表协议帧结束

(6) $GPVTG（地面速度信息）

序号	名称	样例数据	单位	描述
	消息 ID	$GPVTG		VTG 协议的数据头
<1>	对地航向	309.62	度	以真北为参考基准，二维方向指向，相当于二维罗盘
<2>		T		真北参照系
<3>	磁偏角		度	
<4>		M		磁北参照系
<5>	对地航速	0.13	Knots	范围：000.0-999.9
<6>		N		表示：节，Knots
<7>	水平运动速度	0.2		
<8>		K		表示：公里/时，km/h
hh	校验和	6E		
	回车和换行	<CR><LF>		代表协议帧结束

（7）天线状态输出

$GPTXT，01,01,01，ANTENNA OK*35

ok 表示天线已经检测到，open 代表天线断开

（8）关于 UTC 时间和当前北京时间的计算

$GNGGA,084852.000,2236.9453,N,11408.4790,E,1,05,3.1,89.7,M,0.0,M,,*48

所看到的就是 UTC 时间，格式是 hhmmss.sss ，小数点后三位秒忽略，那就 08 点 48 分 52 秒。

UTC + 时区差 = 本地时间

时区差东为正，西为负。在此，把东八区时区差记为 +08， 所以北京时间是 16 点 48 分 5 秒

（9）关于经纬度的换算

$GNRMC,084852.000,A,2236.9453,N,11408.4790,E,0.53,292.44,141216,,,A*7 5

数据格式：度分格式 换算成百度，谷歌地图的格式

纬度：ddmm.mmmm，北纬 2236.9453，22+(36.9453/60)= 22.615755

经度：dddmm.mmmm，东经 11408.4790，114+(08.4790/60)=114.141317

（10）关于热启动 温启动 冷启动 的阐述

冷启动是指在一个陌生的环境下启动 GPS 直到 GPS 和周围卫星联系并且计算出 坐标的启动过程。以下几种情况开机均属冷启动：

1、初次使用时；

2、电池耗尽导致星历信息丢失时；

3、关机状态下将接收机移动 1000 公里以上距离。也就是说冷启动是通过硬件方 式的强制性启动，因为距离上次操作 GPS 已经把内部的定位信息清除掉，GPS 接 收机失去卫星参数，或者已经存在的参数和实际接收到卫星参数相差太多，导致 导航仪无法工作，必须从新获得卫星提供的坐标数据，所以说车辆从地库里启动 导航百分百算冷启动，这也是从地库出来搜星时间长的原因。

温启动是指距离上次定位时间超过 2 个小时的启动，搜星定位时间介于冷启动和 热启动之间。如果您前一日使用过 GPS 定位，那么次日的第一次启动就属于温启 动，启动后会显示上次的位置信息。因为上次关机前的经纬度和高度已知，但由 于关机时间过长，星历发生了变化，以前的卫星接受不到了，参数中的若干颗卫 星已经和 GPS 接收机失去了联系，需要继续搜星补充位置信息，所以搜星的时间 要长于热启动，短于冷启动。

热启动是指在上次关机的地方没有过多移动启动 GPS，但距离上次定位时间必须 小于 2 个小时，通过软件的方式，进行一些启动前的保存和关闭等准备工作后的 启动。

使用教程

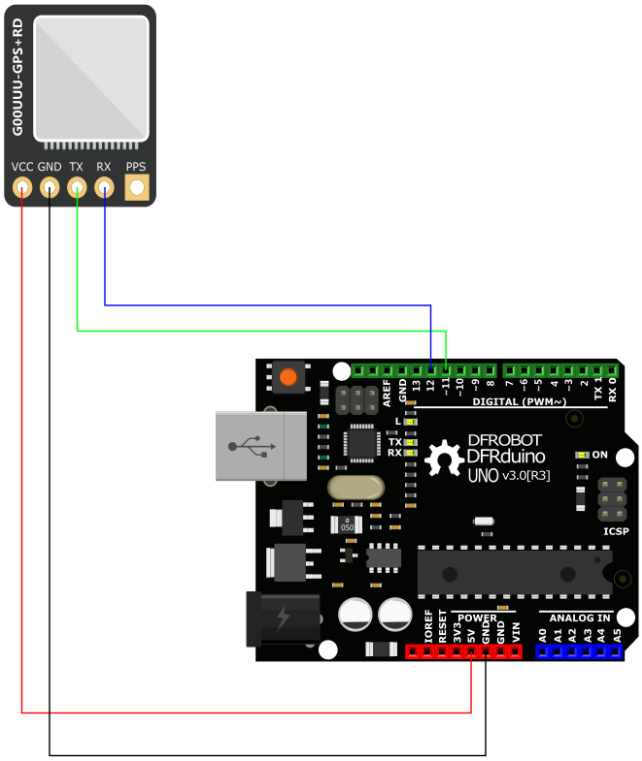
准备

- **硬件**

- Arduino UNO　x1
- GPS+BDS 北斗双模模块　x1

- 软件
  - Arduino IDE，点击下载 Arduino IDE

接线图



示例代码 1（读取 GPS 返回的数据）

```
#include <SoftwareSerial.h>
SoftwareSerial GpsSerial(12, 11); //RX,TX

void setup()
{
  Serial.begin(115200);  //Debug Serial
  GpsSerial.begin(9600);  //Gps Serial
}

void loop()
{  while (GpsSerial.available() > 0)
  {
    byte gpsData = GpsSerial.read();
   Serial.write(gpsData);
  }
```

```
}
```

显示结果 1（读取 GPS 返回的数据）

```
14:49:53.504 -> $BDGSV,2,1,07,02,,,14,03,,,03,06,,,18,07,40,150,20*51
14:49:53.545 -> $BDGSV,2,2,07,09,60,331,26,10,,,24,16,60,021,20*62
14:49:53.587 -> $GNRMC,064953.000,A,3040.04847,N,10348.56476,E,5.18,148.47,291119,,,A*7C
14:49:53.668 -> $GNVTG,148.47,T,,M,5.18,N,9.58,K,A*25
14:49:53.712 -> $GNZDA,064953.000,29,11,2019,00,00*44
14:49:53.753 -> $GPTXT,01,01,01,ANTENNA OK*35
14:49:54.078 -> $GNGGA,064954.000,3040.04684,N,10348.56480,E,1,09,1.7,491.2,M,0.0,M,,*71
14:49:54.170 -> $GNGLL,3040.04684,N,10348.56480,E,064954.000,A,A*45
14:49:54.221 -> $GPGSA,A,3,10,14,20,31,32,34,,,,,,,4.5,1.7,4.1*32
14:49:54.269 -> $BDGSA,A,3,07,09,16,,,,,,,,,,4.5,1.7,4.1*28
14:49:54.314 -> $GPGSV,3,1,09,10,83,085,24,12,26,070,15,14,37,295,28,20,52,141,29*78
14:49:54.397 -> $GPGSV,3,2,09,25,,,22,31,37,226,20,32,54,329,32,33,,,24*7B
14:49:54.442 -> $GPGSV,3,3,09,34,56,093,22*4E
14:49:54.482 -> $BDGSV,2,1,07,02,,,14,03,,,03,06,,,18,07,40,150,20*51
14:49:54.561 -> $BDGSV,2,2,07,09,60,331,26,10,,,24,16,60,021,20*62
14:49:54.610 -> $GNRMC,064954.000,A,3040.04684,N,10348.56480,E,5.44,146.67,291119,,,A*76
14:49:54.690 -> $GNVTG,146.67,T,,M,5.44,N,10.08,K,A*1D
14:49:54.731 -> $GNZDA,064954.000,29,11,2019,00,00*43
14:49:54.773 -> $GPTXT,01,01,01,ANTENNA OK*35
14:49:55.075 -> $GNGGA,064955.000,3040.04529,N,10348.56511,E,1,10,1.0,491.5,M,0.0,M,,*75
14:49:55.174 -> $GNGLL,3040.04529,N,10348.56511,E,064955.000,A,A*49
14:49:55.212 -> $GPGSA,A,3,10,12,14,20,31,32,34,,,,,2.3,1.0,2.1*30
14:49:55.297 -> $BDGSA,A,3,07,09,16,,,,,,,,,,2.3,1.0,2.1*29
14:49:55.337 -> $GPGSV,3,1,09,10,83,085,24,12,26,070,30,14,37,295,28,20,52,141,29*7F
14:49:55.378 -> $GPGSV,3,2,09,25,,,22,31,37,226,20,32,54,329,33,33,,,24*7A
14:49:55.470 -> $GPGSV,3,3,09,34,56,093,22*4E
```

示例代码 2（解析 GPS 数据；该例程解析了$GNRMC）

```
#include <SoftwareSerial.h>
SoftwareSerial GpsSerial(12, 11);  //RX,TX

struct
{
  char GPS_DATA[80];
  bool GetData_Flag;      //获取到 GPS 数据标志位
  bool ParseData_Flag;    //解析完成标志位
  char UTCTime[11];       //UTC 时间
  char latitude[11];      //纬度
  char N_S[2];            //N/S
  char longitude[12];     //经度
  char E_W[2];            //E/W
  bool Usefull_Flag;      //定位信息是否有效标志位
```

```
} Save_Data;

const unsigned int gpsRxBufferLength = 600;
char gpsRxBuffer[gpsRxBufferLength];
unsigned int gpsRxLength = 0;

void setup()
{
  Serial.begin(115200);   //Debug Serial
  GpsSerial.begin(9600);  //Gps Serial

  Serial.println("DFRobot Gps");
  Serial.println("Wating...");

  Save_Data.GetData_Flag = false;
  Save_Data.ParseData_Flag = false;
  Save_Data.Usefull_Flag = false;
}

void loop()
{
  Read_Gps();         //获取 GPS 数据
  parse_GpsDATA();    //解析 GPS 数据
  print_GpsDATA();    //输出解析后的数据
}

void Error_Flag(int num)
{
  Serial.print("ERROR");
  Serial.println(num);
  while (1)
  {
    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
    delay(500);
  }
}

void print_GpsDATA()
```

```
{
  if (Save_Data.ParseData_Flag)
  {
    Save_Data.ParseData_Flag = false;

    Serial.print("Save_Data.UTCTime = ");
    Serial.println(Save_Data.UTCTime);

    if(Save_Data.Usefull_Flag)
    {
      Save_Data.Usefull_Flag = false;
      Serial.print("Save_Data.latitude = ");
      Serial.println(Save_Data.latitude);
      Serial.print("Save_Data.N_S = ");
      Serial.println(Save_Data.N_S);
      Serial.print("Save_Data.longitude = ");
      Serial.println(Save_Data.longitude);
      Serial.print("Save_Data.E_W = ");
      Serial.println(Save_Data.E_W);
    }
    else
    {
      Serial.println("GPS DATA is not usefull!");
    }
  }
}

void parse_GpsDATA()
{
  char *subString;
  char *subStringNext;
  if (Save_Data.GetData_Flag)
  {
    Save_Data.GetData_Flag = false;
    Serial.println("**********************");
    Serial.println(Save_Data.GPS_DATA);

    for (int i = 0 ; i <= 6 ; i++)
    {
      if (i == 0)
```

```
    {
      if ((subString = strstr(Save_Data.GPS_DATA, ",")) == NULL)
        Error_Flag(1);    //解析错误
    }
    else
    {
      subString++;
      if ((subStringNext = strstr(subString, ",")) != NULL)
      {
        char usefullBuffer[2];
        switch(i)
        {
          case 1:memcpy(Save_Data.UTCTime, subString, subStringNext - subString);break;    //获取 UTC 时间
          case 2:memcpy(usefullBuffer, subString, subStringNext - subString);break;    //获取定位状态
          case 3:memcpy(Save_Data.latitude, subString, subStringNext - subString);break;   //获取纬度信息
          case 4:memcpy(Save_Data.N_S, subString, subStringNext - subString);break;    //获取 N/S
          case 5:memcpy(Save_Data.longitude, subString, subStringNext - subString);break;  //获取纬度信息
          case 6:memcpy(Save_Data.E_W, subString, subStringNext - subString);break;    //获取 E/W

          default:break;
        }
        subString = subStringNext;
        Save_Data.ParseData_Flag = true;
        if(usefullBuffer[0] == 'A')
          Save_Data.Usefull_Flag = true;
        else if(usefullBuffer[0] == 'V')
          Save_Data.Usefull_Flag = false;
      }
      else
      {
        Error_Flag(2);    //解析错误
      }
    }
  }
```

```
  }
}

void Read_Gps()
{
  while (GpsSerial.available())
  {
    gpsRxBuffer[gpsRxLength++] = GpsSerial.read();
    if (gpsRxLength == gpsRxBufferLength)RST_GpsRxBuffer();
  }

  char* GPS_DATAHead;
  char* GPS_DATATail;
  if ((GPS_DATAHead = strstr(gpsRxBuffer, "$GPRMC,")) != NULL || (GPS_DATAHead =
strstr(gpsRxBuffer, "$GNRMC,")) != NULL )
  {
    if (((GPS_DATATail = strstr(GPS_DATAHead, "\r\n")) != NULL) && (GPS_DATATail >
GPS_DATAHead))
    {
      memcpy(Save_Data.GPS_DATA, GPS_DATAHead, GPS_DATATail - GPS_DATAHead);
      Save_Data.GetData_Flag = true;

      RST_GpsRxBuffer();
    }
  }
}

void RST_GpsRxBuffer(void)
{
  memset(gpsRxBuffer, 0, gpsRxBufferLength);      //清空
  gpsRxLength = 0;
}
```

显示结果 2（解析数据）

```
13:54:53.641 -> Save_Data.UTCTime = 055453.000
13:54:53.690 -> Save_Data.latitude = 3040.09146
13:54:53.724 -> Save_Data.N_S = N
13:54:53.773 -> Save_Data.longitude = 10348.55584
13:54:53.773 -> Save_Data.E_W = E
13:54:54.574 -> **************
13:54:54.574 -> $GNRMC,055454.000,A,3040.09175,N,10348.55593,E,0.00,0.00,251119,,,A*76
13:54:54.641 -> Save_Data.UTCTime = 055454.000
13:54:54.692 -> Save_Data.latitude = 3040.09175
13:54:54.741 -> Save_Data.N_S = N
13:54:54.741 -> Save_Data.longitude = 10348.55593
13:54:54.791 -> Save_Data.E_W = E
13:54:55.577 -> **************
13:54:55.577 -> $GNRMC,055455.000,A,3040.09204,N,10348.55601,E,0.00,0.00,251119,,,A*7A
13:54:55.671 -> Save_Data.UTCTime = 055455.000
13:54:55.671 -> Save_Data.latitude = 3040.09204
13:54:55.717 -> Save_Data.N_S = N
13:54:55.754 -> Save_Data.longitude = 10348.55601
13:54:55.790 -> Save_Data.E_W = E
13:54:56.570 -> **************
13:54:56.618 -> $GNRMC,055456.000,A,3040.09228,N,10348.55606,E,0.00,0.00,251119,,,A*70
13:54:56.664 -> Save_Data.UTCTime = 055456.000
13:54:56.705 -> Save_Data.latitude = 3040.09228
13:54:56.738 -> Save_Data.N_S = N
13:54:56.773 -> Save_Data.longitude = 10348.55606
13:54:56.822 -> Save_Data.E_W = E
13:54:57.582 -> **************
13:54:57.582 -> $GNRMC,055457.000,A,3040.09259,N,10348.55615,E,0.00,0.00,251119,,,A*75
13:54:57.687 -> Save_Data.UTCTime = 055457.000
13:54:57.687 -> Save_Data.latitude = 3040.09259
13:54:57.721 -> Save_Data.N_S = N
13:54:57.772 -> Save_Data.longitude = 10348.55615
13:54:57.819 -> Save_Data.E_W = E
```

常见问题

还没有客户对此产品有任何问题，欢迎通过 qq 或者论坛联系我们！

更多问题及有趣的应用，可以 访问论坛 进行查阅或发帖

更多

DFRobot 商城购买链接