



北京理工大学
Beijing Institute of Technology

《计算机原理与应用 B》

第二次课程大作业

任课教师	李海	学生姓名	胡森康
作业日期	2021 年 1 月	学 号	1120183150
作业类型	<input type="checkbox"/> 原理验证	班 级	05961808
	<input checked="" type="checkbox"/> 综合设计	学 院	信息与电子学院
	<input type="checkbox"/> 自主创新	专 业	电子信息工程
选 题	B	开发环境	Python 3.8



信息与电子学院

SCHOOL OF INFORMATION AND ELECTRONICS

基于 Python 实现 NMEA-0183 协议下 GPS 数据的串口通讯程序设计及可视化

胡森康 1120183150

摘 要

本项目搭建了虚拟串口环境，同时利用 GPS 模拟器产生虚拟 GPS 数据。并利用 Python 从虚拟串口读入此数据，提取虚拟 GPS 数据中的经纬度，并进行可视化显示。

本文详细阐述了搭建虚拟串口环境的流程，以及调试串口的方法；另外详细介绍了 NEMA-0183 协议，以及利用 Python 从串口导入 GPS 模拟器产生的基于 NEMA-0183 协议的 GPS 数据的思路和程序；

关键词：GPS Python 串口 可视化 NEMA-0183 协议

目 录

1	项目背景	1
2	基本理论	1
2.1	NEMA-0183 协议简介	1
2.2	NEMA-0183 协议举例说明	2
3	搭建虚拟环境	2
3.1	产生虚拟串口	2
3.2	产生虚拟 GPS 数据	4
3.3	调试串口	5
4	程序设计	5
4.1	程序设计框图	5
4.2	程序设计过程	6
5	结果分析	7
6	结论	8
7	参考文献	9

1 项目背景

倾覆沉没的钻井平台，顺流直冲的运油车头，直坠入海的满载客机。交通海上应急反应特勤队第一时间抵达，站在水火咆哮的最前面，守在危急撤离的最后面，用生命对抗天灾人祸。但在自然面前，特勤员毕竟没有超能力，血肉之躯踩在死亡边缘，真实的恐惧无数次让这些斗士颤抖、无助和气馁。而海上救援的字典里没有“退缩”。特勤队有一个 GPS 接收机报告当前的位置，但是 GPS 接收机的屏幕被打破了，只能从 GPS 接收机的串口读取位置数据。

GPS 接收机有一个 RS232 串口用来输出数据，接口参数为：波特率为 9600，8 位数据位，1 位停止位，没有校验位。

在此背景下，本项目致力于开发出一款 APP，可以从 GPS 接收机串口读入数据，并进行可视化显示，以解决特勤队的困难处境。

2 基本理论

2.1 NEMA-0183 协议简介

NMEA-0183 是美国国家海洋电子协会 (National Marine Electronics Association) 为统一海洋导航规范而制定的标准，该格式标准已经成为国际通用的一种格式，协议的内容在兼容 NMEA-0180 和 NMEA-0182 的基础上，增加了 GPS、测深仪、罗经方位系统等多种设备接口和通讯协议定义，同时还允许一些特定厂商对其设备通信自定协议 (如 Garmin GPS, Deso 20 等)。NMEA-0183 格式数据串的所有数据都采用 ASCII 文本字符表示，数据传输以“\$”开头，后面是语句头。语句头由五个字母组成，分为两部分，前两个字母表示“系统 ID”，即表示该语句是属于何种系统或设备，后三个字母表示“语句 ID”，表示该语句是关于何方面的数据。语句头后是数据体，包含不同的数据体字段，语句末尾为校验码 (可选)，以回车换行符 <CR><LF> 结束，也就是 ASCII 字符“回车”(十六进制的 0DH) 和“换行”(十六进制的 0AH)。每行语句最多包含 82 个字符 (包括回车换行符和“\$”符号)。数据字段以逗号分隔识别，空字段保留逗号。以 GPS 的 GPRMC 语句为：

```
$GPRMC,<1>,<2>,<3>,<4>,<5>,<6>,<7>,
<8>,<9>,<10>,<11>,<12> *hh <CR><LF>
```

其中 GP 表示该语句是 GPS 定位系统的，RMC 表示该语句输出的是 GPS 定位信

息，后面是数据体。最后校验码 *hh 是用做校验的数据。在通常使用时，它并不是必须的，但是当周围环境中有较强的电磁干扰时，则推荐使用。hh 代表了“\$”和“*”的所有字符的按位异或值（不包括这两个字符）。个别厂商自己定义语句格式以“\$P”开头，其后是 3 个字符的厂家 ID 识别号，后接自定义的数据体。

2.2 NEMA-0183 协议举例说明

以下面的 GPRMC 语句为例：

\$GPRMC,024813.640,A,3158.4608,N,11848.3737,E,10.05,324.27,150706,,,A*50

字段 0: \$GPRMC，语句 ID，表明该语句为 Recommended Minimum Specific GP-S/TRANSIT Data (RMC) 推荐最小定位信息

字段 1: UTC 时间，hhmmss.ss 格式

字段 2: 状态，A= 定位，V= 未定位

字段 3: 纬度 dddmm.mmmm，度分格式（前导位数不足则补 0）

字段 4: 纬度 N (北纬) 或 S (南纬)

字段 5: 经度 dddmm.mmmm，度分格式（前导位数不足则补 0）

字段 6: 经度 E (东经) 或 W (西经)

字段 7: 速度，节，Knots

字段 8: 方位角，度

字段 9: UTC 日期，DDMMYY 格式

字段 10: 磁偏角，(000 - 180) 度（前导位数不足则补 0）

字段 11: 磁偏角方向，E= 东，W= 西

字段 16: 校验值^[1]

3 搭建虚拟环境

搭建虚拟环境的流程图如图 (3-1) 所示。

3.1 产生虚拟串口

下载安装 Configure Virtual Serial Port Driver (VSPD) 软件，VSPD 界面如图 (3-2) 所示。在此处产生一对端口，分别为 COM10 和 COM11。

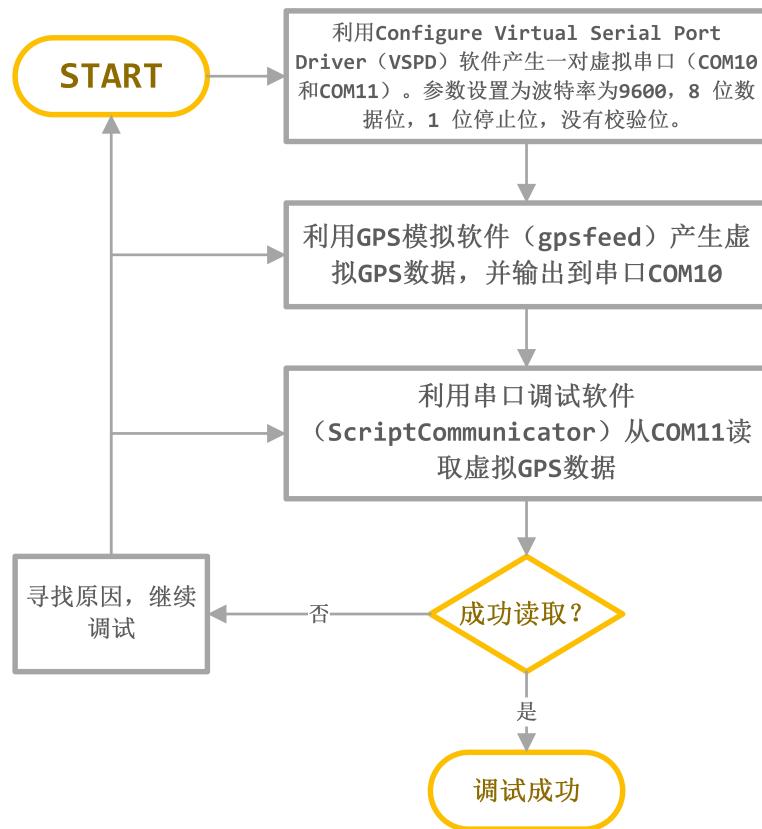


图 3-1: 搭建虚拟环境流程图

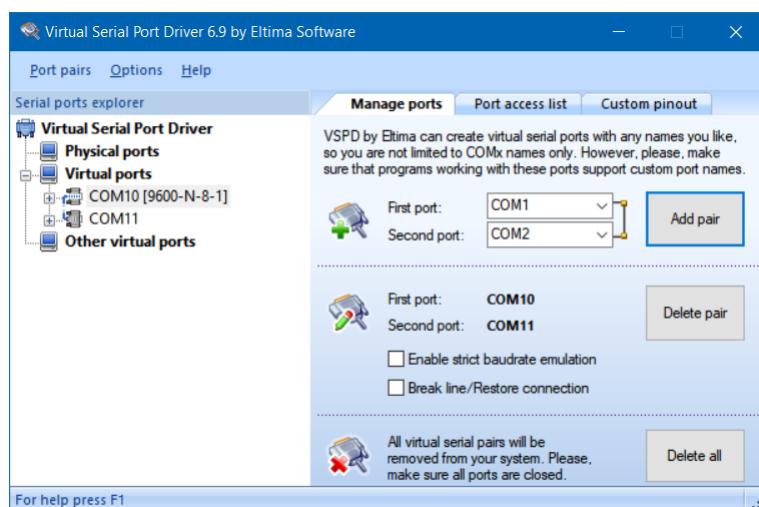


图 3-2: VSPD 界面

并检查是否配置成功，打开电脑的设备管理器，如下图(3-3)所示。可以看到 COM10 和 COM11 被成功配置。

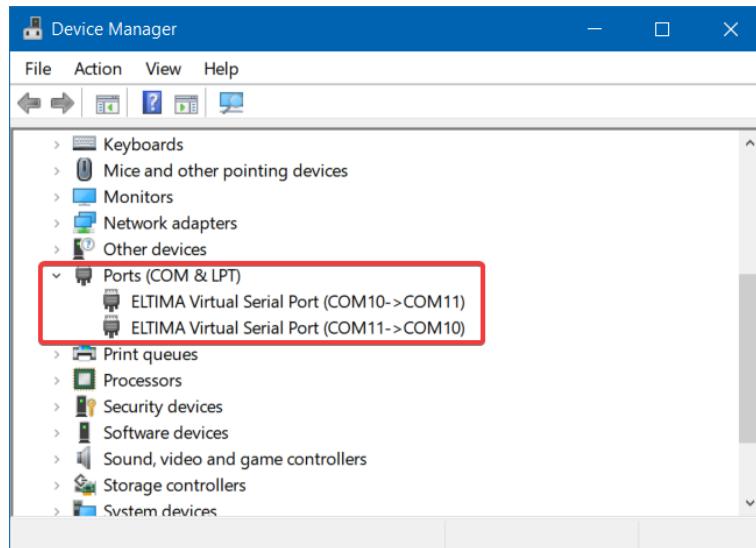


图 3-3: COM10 和 COM11 被成功配置

3.2 产生虚拟 GPS 数据

下载 GPS 模拟器 gpsfeed+，并配置好参数，如下图(3-4)所示。

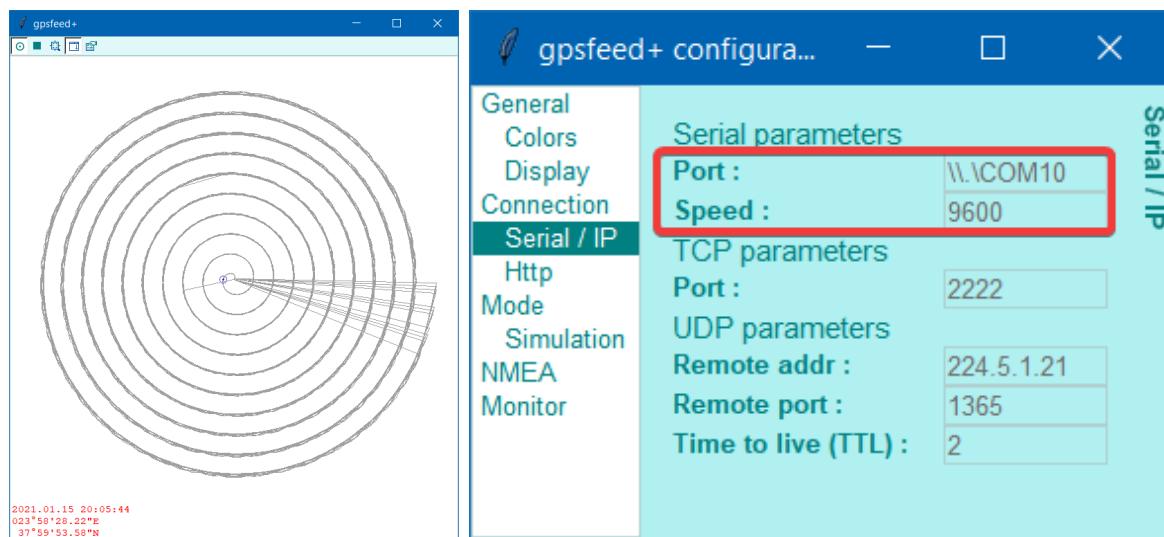


图 3-4: gpsfeed+ 界面图和参数配置

配置 gpsfeed+ 将其产生的虚拟 GPS 数据发送到 COM10 串口，并配置波特率为 9600。

在填写串口时，应写为\\.\COM??。因为 gpsfeed+ 是用 TCL 语言开发的，它打开串口用的是 puts 命令，而 puts 命令是通过 Windows API 的 CreateFile 函数实现的。CreateFile 函数只能打开 COM1~COM9，而对于大于 9 的 COM 口，需要采用“\\.\COM??”的格式打开^[2]。

3.3 调试串口

下载串口调试工具 ScriptCommunicator，配置其接收串口为 COM11，波特率为 9600，可以正常接收到数据，如下图（3-5）所示。

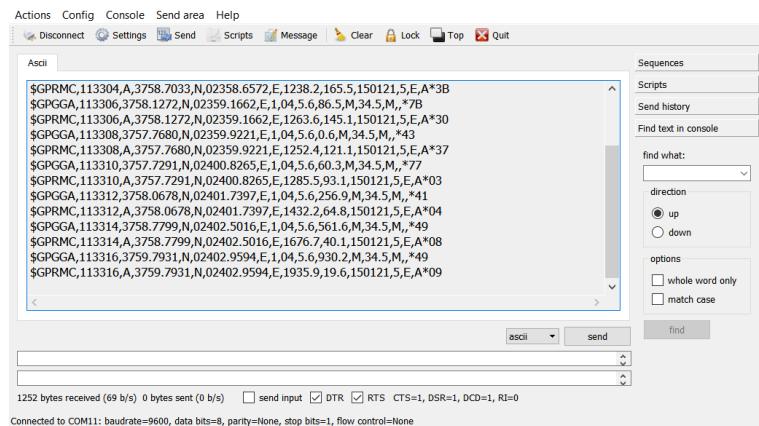


图 3-5: ScriptCommunicator 从串口 COM11 读取数据

4 程序设计

4.1 程序设计框图

程序设计框图如下图（4-6）所示。

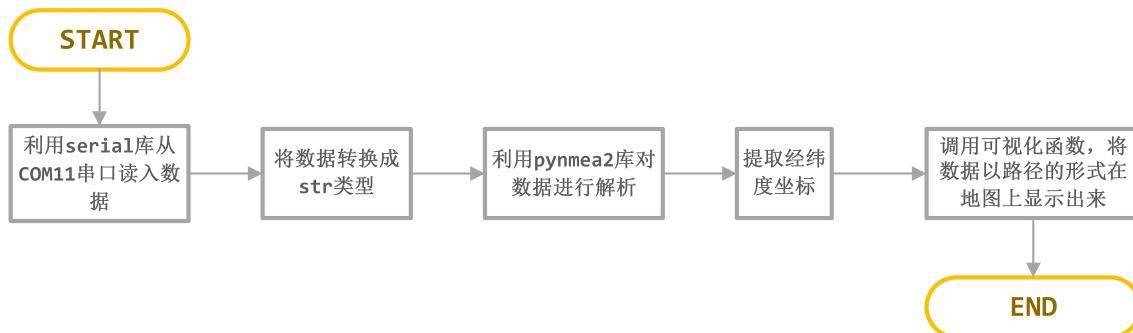


图 4-6: Python 程序设计框图

在此程序中，主要利用了两个 Python 库，分别为 serial 和 pynmea2 库。通过 serial 库可以从串口读取数据；通过 pynmea2 库可以对 GPS 数据进行解析，进而得到经纬度坐标。

4.2 程序设计过程

1. 利用 serial 库打开串口

```
1 ser = serial.Serial(  
2     port='COM11',  
3     baudrate=9600,  
4     bytesize=8,  
5     parity='N',  
6     stopbits=1)
```

2. 读取数据并将数据转换为 str 类型

```
1 data=ser.readline()  
2 data_str=str(data,encoding="utf-8")
```

3. 利用 pynmea2 库对数据进行解析

```
1 msg=pynmea2.parse(data_str)
```

4. 提取经纬度坐标

```
1 location0=[msg.latitude,msg.longitude]  
2 location.append(location0)
```

5. 将数据显示在地图上

```
1 draw_gps(location, path,'f3.html')
```

5 结果分析

1. 设置程序从串口读入 10 个 GPS 数据，得到纬度和经度坐标，并将此数据显示在地图上，得到的路径如下图所示（5-7），在希腊雅典附近。

```
[38.028265, 23.67692833333333]
[38.028265, 23.67692833333333]
[37.90428833333334, 23.68694333333332]
[37.90428833333334, 23.68694333333332]
[37.681396666666664, 23.89648]
[37.681396666666664, 23.89648]
[37.6627683333333, 24.023581666666665]
[37.6627683333333, 24.023581666666665]
[37.69341166666667, 24.14953333333334]
[37.69341166666667, 24.14953333333334]
```

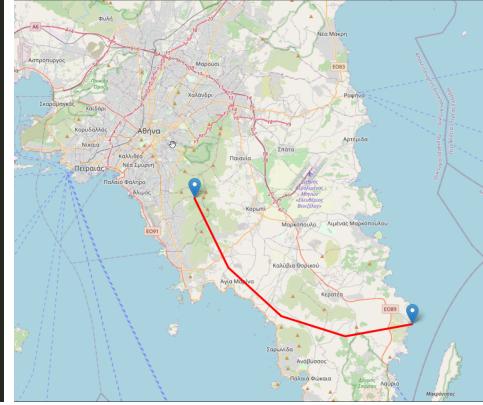


图 5-7: 从串口读取的 10 个纬度经度坐标及路径

2. 设置程序从串口读入 100 个 GPS 数据，得到纬度和经度坐标，并将此数据显示在地图上，得到的路径如下图（5-8）所示。

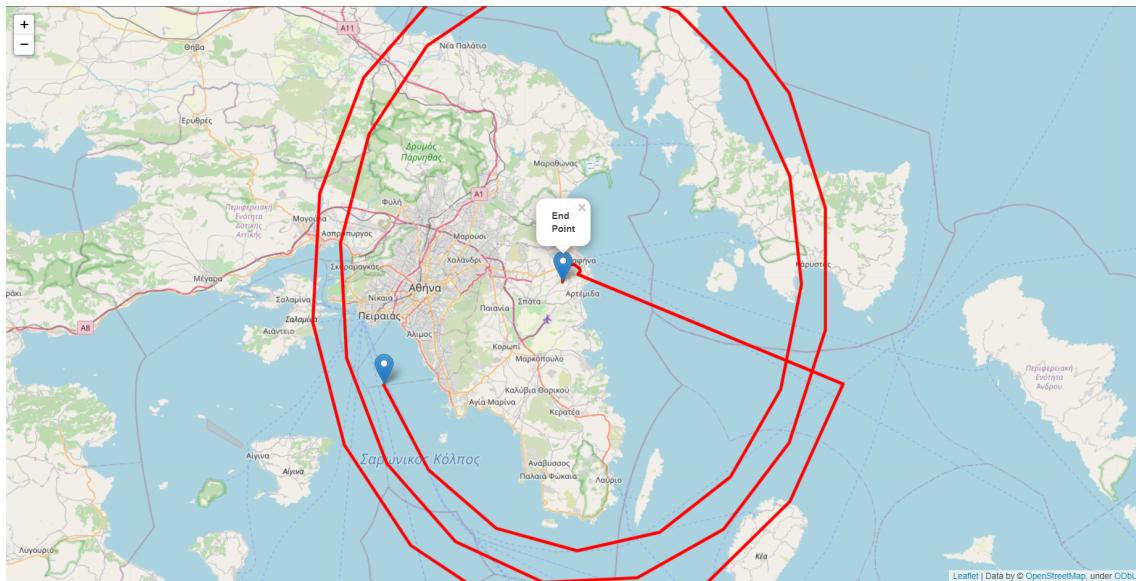


图 5-8: 从串口读取的 100 个纬度经度所绘制的路径

3. 设置程序从串口读入 1000 个 GPS 数据，得到纬度和经度坐标，并将此数据显示在地图上，得到的路径如下图（5-9）所示。

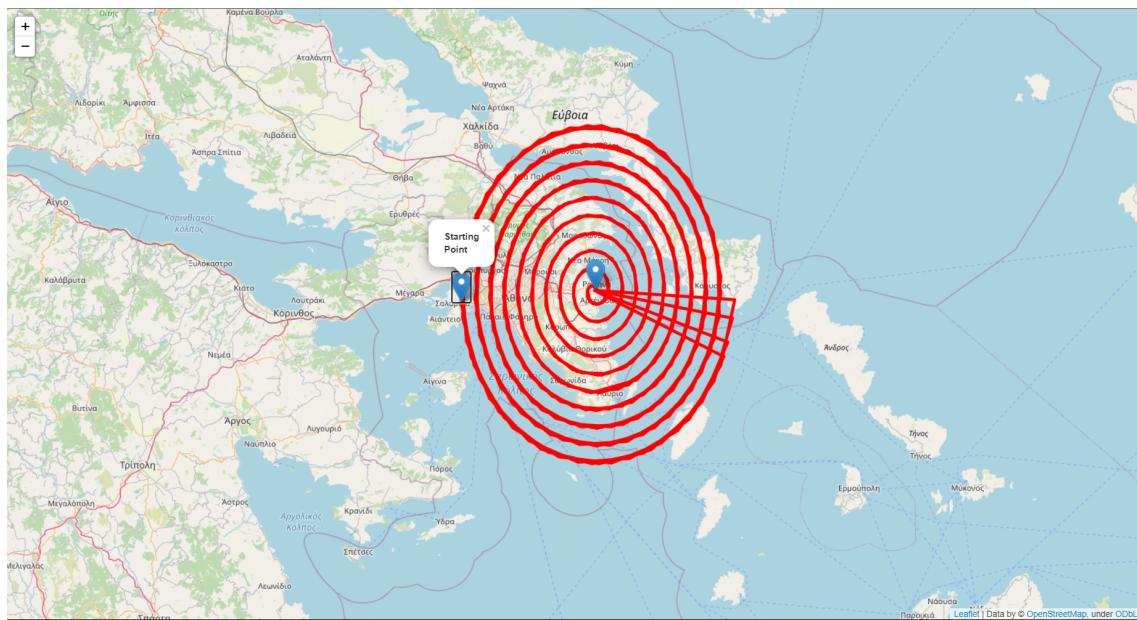


图 5-9: 从串口读取的 1000 个纬度经度所绘制的路径

6 结论

本文利用了 Python 实现了 NMEA-0183 协议下 GPS 数据的串口通讯程序设计，并对数据进行了可视化显示，将路径显示在了地图上。

同时搭建了虚拟串口并利用 GPS 模拟器产生了虚拟的 GPS 数据，便于对本程序进行验证和测试。

在编写程序的过程中，遇到了一些困难，在此一一列出并说明如何解决。

1. 在产生虚拟串口时，运用老师推荐的 com0com 虚拟串口软件时，虽然设备管理器已经检测到了串口，但因为一些不可知的原因，设备管理器中显示串口不可用。无奈之下重新下载了虚拟串口软件 Configure Virtual Serial Port Driver (VSPD)，使此问题得到解决
2. 在 gpsfeed+ 产生虚拟 GPS 数据时，发现软件并没有将数据输出到串口，而是输出到了本地文件，其他同学也遇到了此问题，但通过按照李老师所撰写的推送（[gpsfeed+ 如何使用编号大于 9 的串口](#)）中的步骤操作，解决了此问题。
3. 在进行 Python 程序编写的过程中，发现接收到的串口数据并不是 str 的数据类型，而是 bytes 类型，不能直接通过 pynmea2 库进行处理，因此利用 str() 语句将 bytes 型转换为 str 型，问题得到解决。

在完成此项目的过程中，更加意识到了 Python 语言的强大，利用 C 语言需要几十

行的代码，但利用 Python 可能只需要几行即可解决，因此以后要在数据分析，深度学习等重要领域加强对 Python 语言的掌握。

7 参考文献

- [1] 韩友美, 钟政, 桑逢云. NEMA—0183 协议下 GPS 数据的实时串口通讯程序设计 [A]. 山东土地学会、山东测绘学会. 山东省“数字国土”学术交流会论文集 [C]. 山东土地学会、山东测绘学会: 山东省科学技术协会, 2007:4.
- [2] [gpsfeed+ 如何使用编号大于 9 的串口](#)