

# 无人船通讯

- mqtt

- 系统安装 <https://blog.csdn.net/ypbsyy/article/details/80528979>
- 官网 <https://www.eclipse.org/paho/>
- mosquitto常用指令 <https://www.cnblogs.com/xcsn/p/11360413.html>
- python下使用
  - 安装
    - `sudo apt install python3-pip`
    - `python3 -m pip install paho-mqtt`

- python安装包

- `mosquitto -c /usr/etc/mosquitto/mosquitto.conf`

- `mosquitto_sub -h localhost -p 1883 -t "topic" -v | hexdump`

- 7 /mode/set 控制模式 (1B)

- 8 /nav/start [开始点 (2B)]

- 9 10无参数

- 15. /status 参数格式为: | 前往点 (2B) | 控制模式 (1B) | 任务类型 (1B) | 任务状态 (1B) | 工作模式 (1B) |

- 8B 4B

- 16. 命令-0x010D (Client <-- USV), 对应的mqtt消息为 /gps

- 无人船的当前位置, 发送频率由传感器设备的频率决定, 一般不低于5Hz, 参数格式为: | WGS84纬度 (8B) |

- WGS84经度 (8B) |

- WGS84纬度: 双精度浮点数

- WGS84经度: 双精度浮点数

- 17. 命令-0x010E (Client <-- USV), 对应的mqtt消息为 /pose

- 无人船的姿态数据, 发送频率同传感器设备的频率, 一般不低于5Hz, 参数格式为: | 艏向 (4B) | 纵摇 (4B) | 横摇 (4B) |

- 艏向: 单精度浮点数, 单位为度, 正北为0, 顺时针为正方向

- 纵摇: 单精度浮点数, 单位为度

- 横摇: 单精度浮点数, 单位为度

- 18. 命令-0x010F (Client <-- USV), 对应的mqtt消息为 /vtg

- 无人船的速度和航向, 发送频率同传感器设备的频率, 一般不低于5Hz, 参数格式为: | 速度 (4B) | 航向 (4B) |




























- 速度: 单精度浮点数, 单位为米每秒

- 航向: 单精度浮点数, 单位为度, 正北为0, 顺时针为正方向

- 19. 命令-0x0110 (Client <-- USV), 对应的mqtt消息为 /vel

- 无人船的速度, 发送频率同传感器设备的频率, 一般不低于5Hz, 参数格式为: | 速度 (4B) |

- advanced ip scanner 结果

	LAPTOP-TL04TFBJ	192.168.43.209	Hon Hai Precision Ind. Co...	54:13:79:A3:6C:F1
	LAPTOP-TL04TFBJ	192.168.56.1		0A:00:27:00:00:0C
	192.168.66.3	192.168.66.3		
	192.168.66.1	192.168.66.1		
	192.168.56.255	192.168.56.255		0A:00:27:00:00:0C
	192.168.43.255	192.168.43.255	Hon Hai Precision Ind. Co...	54:13:79:A3:6C:F1
	192.168.43.197	192.168.43.197		88:40:3B:D0:AA:C2
	192.168.216.3	192.168.216.3		
	192.168.216.1	192.168.216.1		
	192.168.124.9	192.168.124.9		
	192.168.124.7	192.168.124.7		
	192.168.124.5	192.168.124.5		
	192.168.124.3	192.168.124.3		
	192.168.124.15	192.168.124.15		
	192.168.124.13	192.168.124.13		
	192.168.124.11	192.168.124.11		
	192.168.124.1	192.168.124.1		
	192.168.123.27	192.168.123.27		
	192.168.123.25	192.168.123.25		
	192.168.123.23	192.168.123.23		
	192.168.123.21	192.168.123.21		
	192.168.123.19	192.168.123.19		
	192.168.123.17	192.168.123.17		
	192.168.111.9	192.168.111.9		
	192.168.111.7	192.168.111.7		
	192.168.111.11	192.168.111.11		
	192.168.111.1	192.168.111.1		

- 更改ip地址为 192.168.1.xxx即可 目的：设置为在同一局域网下
- 控制指令无反应原因
  - 循环函数出错
  - 怀疑 client.loop()需要一直执行

## 测试指令

- 消息类 无人船->客户端
  - mosquitto\_sub -h localhost -p 1883 -t "/status" -v

15. 命令-0x010C (Client ← USV)，对应的mqtt消息为 /status

无人船状态反馈数据，发送频率为1Hz，参数格式为：| 前往点 (2B) | 控制模式 (1B) | 任务类型 (1B) | 任务状态 (1B) | 工作模式 (1B) |

前往点序号：16位整数，表示无人船正前往这个点

控制模式：0表示自动，1表示手动

任务类型：扩展用，忽略

任务状态：0表示停止，1表示暂停，2表示正在运行，3表示空闲

工作模式：与0x04进行与操作，不为0表示正在测绘（比如第3个点是测绘点，那么往第4个点的过程中，工作模式的值就为4）

```
oscar@ubuntu:~/python_ws$ mosquitto_sub -h 192.168.1.230 -p 1883 -t "/status"
```

```
oscar@ubuntu:~/python_ws$ mosquitto_sub -h 192.168.1.230 -p 1883 -t "/status" lhexdump
00000000 ffff 0001 0000 ff0a 01ff 0000 0a00 ffff
00000010 0001 0000 ff0a 01ff 0000 0a00 ffff 0001
00000020 0000 ff0a 01ff 0000 0a00 ffff 0001 0000
```

- `mosquitto_sub -h localhost -p 1883 -t "/gps" -v`

- `mosquitto_sub -h localhost -p 1883 -t "/pose" -v 姿态`

```

oscar@ubuntu:~/python_ws$ mosquitto_sub -h 192.168.1.230 -p 1883 -t "/pose" -v | hexdump
00000000 702f 736f 2065 ea42 9a99 993f 9a99 00bf
00000010 0000 2f0a 6f70 6573 4220 cce9 3fcd cc8c
00000020 becd cc4c 0acd 702f 736f 2065 ea42 cdcc
00000030 8c3f cdcc 0000 0000 2f0a 6f70 6573 4220
00000040 33ea 3f33 9999 be9a cc4c 0acd 702f 736f
00000050 2065 ea42 0000 993f 9a99 4cbe cdcc 2f0a
00000060 6f70 6573 4220 33ea 3f33 9999 bd9a cccc
00000070 0acd 702f 736f 2065 ea42 6666 8c3f cdcc
00000080 00bf 0000 2f0a 6f70 6573 4220 ccea 3fcd
00000090 0080 be00 cc4c 0acd 702f 736f 2065 ea42
000000a0 3333 a63f 6666 4cbe cdcc 2f0a 6f70 6573
000000b0 4220 00eb 3f00 9999 be9a 9999 0a9a 702f
000000c0 736f 2065 eb42 3333 993f 9a99 4cbe cdcc
000000d0 2f0a 6f70 6573 4220 00ea 3f00 cc8c becd
000000e0 cc4c 0acd 702f 736f 2065 ea42 cdcc 993f

```

- `mosquitto_sub -h localhost -p 1883 -t "/vtg" -v 速度与航向`

```

osc@ubuntu:~$ mosquitto_sub -h 192.168.1.230 -p 1883 -t "/vtg" -v
/vtg ?◆◆B◆◆H
/vtg ?◆◆B◆◆H
/vtg ?◆◆B◆◆
/vtg ?◆◆B◆◆
/vtg ?◆◆B◆◆
/vtg ?◆◆B◆◆
/vtg ?◆◆B◆◆
/vtg ?◆E
      Bf
/vtg ?◆E
      Bf
/vtg ?◆\◆ByG◆
/vtg ?◆\◆ByG◆
/vtg ?Y◆B◆=q
/vtg ?Y◆B◆=q
/vtg ?X◆WC◆◆
/vtg ?X◆WC◆◆
/vtg ?P◆6C◆
/vtg ?P◆6C◆

```

- mosquitto\_sub -h localhost -p 1883 -t "/vel" -v 仅速度

19. 命令-0x0110 (Client ← USV)，对应的mqtt消息为 /vel

无人船的速度，发送频率同传感器设备的频率，一般不低于5Hz，参数格式为：|速度（4B）|

速度：单精度浮点数，单位为米每秒

```

osc@ubuntu:~$ mosquitto_sub -h 192.168.1.230 -p 1883 -t "/vel" -v
/vel ?=◆/
/vel ??◆◆
/vel ?=◆
/vel >◆
/vel >m◆d
/vel =◆`
/vel =◆{◆
/vel >◆G
/vel >◆◆
/vel ?◆◆
/vel ?%◆l
/vel ? ◆Z
/vel ?
◆5
/vel >◆S*
^Z

```

- mosquitto\_sub -h localhost -p 1883 -t "/hdt" -v 艏向

20. 命令-0x0111 (Client ← USV)，对应的mqtt消息为 /hdt

无人船的艏向，发送频率同传感器设备的频率，一般不低于5Hz，参数格式为：|艏向（4B）|

艏向：航向：单精度浮点数，单位为度，正北为0，顺时针为正方向

```

osc@ubuntu:~/python_ws$ mosquitto_sub -h 192.168.1.230 -p 1883 -t "/hdt" -v lhexdump
00000000 682f 7464 4220 66eb 0a66 682f 7464 4220
00000010 99e9 0a9a 682f 7464 4220 99ea 0a9a 682f
00000020 7464 4220 00eb 0a00 682f 7464 4220 33ea
00000030 0a33 682f 7464 4220 cce9 0acd 682f 7464
00000040 4220 99eb 0a9a 682f 7464 4220 99ea 0a9a
00000050 682f 7464 4220 00eb 0a00 682f 7464 4220
00000060 cce9 0acd 682f 7464 4220 33ea 0a33 682f
00000070 7464 4220 33eb 0a33 682f 7464 4220 33eb
00000080 0a33 682f 7464 4220 00ea 0a00 682f 7464
00000090 4220 00eb 0a00 682f 7464 4220 66eb 0a66
000000a0 682f 7464 4220 00eb 0a00 682f 7464 4220
000000b0 33ea 0a33 682f 7464 4220 00ec 0a00 682f
000000c0 7464 4220 99eb 0a9a 682f 7464 4220 99ea
000000d0 0a9a 682f 7464 4220 cceb 0acd 682f 7464
000000e0 4220 33ea 0a33 682f 7464 4220 66ea 0a66
000000f0 682f 7464 4220 cce9 0acd 682f 7464 4220
00001000 99ea 0a9a 682f 7464 4220 66ea 0a66 682f
00001100 7464 4220 66eb 0a66 682f 7464 4220 99ea
00001200 0a9a 682f 7464 4220 66ea 0a66 682f 7464

```

- `mosquitto_sub -h localhost -p 1883 -t "/bat" -v` 剩余电量

21. 命令-0x0112 (Client ← USV)，对应的mqtt消息为 /bat

无人船剩余电量的百分比，发送频率0.2Hz，参数格式为：| 电量百分比 (1B) |  
电量百分比：0 ~ 100，其他值表示没有采集到剩余电量数据。

```

osc@ubuntu:~/python_ws$ mosquitto_sub -h 192.168.1.230 -p 1883 -t "/bat"
W
W
W
W
W
_

```

- `mosquitto_sub -h localhost -p 1883 -t "/radar/object" -v` 雷达探测信息

22. 命令-0x0113 (Client ← USV)，对应的mqtt消息为 /radar/object

雷达探测到的目标信息，参数格式为：| 编号 (1B) | 距离 (4B) | 方位 (4B) |  
编号：0 ~ 255，雷达能同时探测多个障碍物，每个障碍物都有编号  
距离：单精度浮点数，障碍物相对船的距离，单位为米  
方位：单精度浮点数，障碍物相对船头指向的方位，顺时针为正，单位为度

- `mosquitto_sub -h localhost -p 1883 -t "/radar/status" -v` 避障状态与避障行为

23. 命令-0x0114 (Client ← USV)，对应的mqtt消息为 /radar/status

无人船当前的避障状态以及避障行为，发送频率0.5Hz，参数格式为：| 避障使能 (1B) | 避障行为 (1B) |  
避障使能：0表示已经禁用避障，1表示已经启用避障  
避障行为：0表示无避障，1表示向左避障，2表示向右避障，3表示向后避障

```

osc@ubuntu:~/python_ws$ mosquitto_sub -h 192.168.1.230 -p 1883 -t "/radar/status" -v
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status
/radar/status

```

- mosquitto\_sub -h localhost -p 1883 -t "/bat/info" -v 实时电池参数

30. 命令-0x011B (Client ← USV)，对应的mqtt消息为 /bat/info

无人船返回实时的电池参数，参数如格式为：| 功率 (4B) | 电压 (4B) | 电流 (4B) | 温度 (2B) |  
 功率：单精度浮点数，单位为瓦  
 电压：单精度浮点数，单位为伏  
 电流：单精度浮点数，单位为安  
 温度：16位整数，单位为摄氏度

```

osc@ubuntu:~/python_ws$ mosquitto_sub -h 192.168.1.230 -p 1883 -t "/bat/info" -v
/bat/info A+1B+>+>+>
/bat/info B+>
/bat/info B+>
^Z
[29]+  Stopped                  mosquitto_sub -h 192.168.1.230 -p 1883 -t "/bat/info" -v
osc@ubuntu:~/python_ws$ mosquitto_sub -h 192.168.1.230 -p 1883 -t "/bat/info" -v lhexdump
00000000 622f 7461 692f 666e 206f 0000 0000 0042
00000010 cdcc 0000 0000 1d00 2f0a 6162 2f74 6e69
00000020 6f66 0020 0000 4200 cc00 00cd 0000 0000
00000030 0a1d 622f 7461 692f 666e 206f 0000 0000
00000040 0042 cdcc 0000 0000 1d00 2f0a 6162 2f74
00000050 6e69 6f66 0020 0000 4200 cc00 00cd 0000

```

- 控制类 客户端->无人船

- 速度方向

6. 命令-0x0102 (Client → USV)，对应的mqtt消息为 /ctrl

控制船的速度和方向，通用方式，适用于带舵的船和差速控制的船，要求控制频率不小于2Hz,建议频率10Hz。参数格式为：

速度 (4字节) | 方向 (4字节) | 优先级 (1字节) |

速度：单精度浮点数，取值范围为-1 ~ 1，-1表示全速后退，1表示全速前进。

方向：单精度浮点数，取值范围为-1 ~ 1，-1表示满舵左转，1表示满舵右转。

优先级：0~100，遥控器具有最高的优先级100，建议其他终端不大于50。

- 船控模式

7. 命令-0x0103 (Client → USV)，对应的mqtt消息为 /mode/set

改变船的控制模式，参数格式为：| 控制模式 (1B)

控制模式：0表示自动，1表示手动

- 开始点.

8. 命令-0x0104 (Client → USV)，对应的mqtt消息为 /nav/start

控制无人船，开始任务，参数可选，不带参数时默认为0，格式为：

[| 开始点 (2B) |]

开始点：16位整数，表示这个点和这个点之前的任务已经完成，无人船开始往下一个点航行。

- 暂停任务



9. 命令-0x0105 (Client → USV), 对应的mqtt消息为 /nav/pause

控制无人船, 暂停任务, 暂停后收到/nav/start命令, 会继续从当前位置开始航行, 无参数。

- 终止任务

10. 命令-0x0106 (Client → USV), 对应的mqtt消息为 /nav/stop

控制无人船, 停止任务, 停止后收到/nav/start命令, 会从第一个点开始航行, 无参数。

- 避障开关

24. 命令-0x0115 (Client → USV), 对应的mqtt消息为 /radar/set

启用或关闭避障功能, 参数格式为: | 避障使能 (1B) |

避障使能: 0表示禁用避障, 1表示启用避障

- 发送信号并获取无人船状态

25. 命令-0x0116 (Client → USV), 对应的mqtt消息为 /status/get

获取无人船当前的状态, 无人船接收到这个指令后, 会马上给客户端返回状态数据 /status, 该命令无参数

- 设置无人船航行任务

26. 命令-0x0117 (Client → USV), 对应的mqtt消息为 /wp/set

设置无人船的航行任务, 参数格式为: | 任务类型 (1B) | 任务点数 (2B) | 任务点数据0... 任务点数据n | 采样点数据0...

采样点数据x | 监测点数据0... 监测点数据y |

任务类型: 0 工作任务, 1 返航任务, 2 伴航任务

任务点数: 无符号整数, 表示任务点的个数

任务数据: 任务数据的总长度由任务点数决定, 每个任务点数据的长度是17个字节, 依次为 | 类型 (1B) | 纬度 (8B) |

经度 (8B) |, 类型的第0位表示当前点是否为采样点, 第1位表示当前点是否为监测点, 第2位表示当前点是否为测绘点, 纬度为双精度浮点数, 精度为双精度浮点数, 单位为度。一个点可以为采样监测和测绘的任意组合。

采样数据: 采样数据的总长度由任务点中的采样点数决定, 每个采样点的数据长度是9个字节, 依次为 | 采样瓶号 (1B) |

采样容量 (4B) |

采样深度 (4B) |, 采样瓶号的范围是0到255, 采样容量为单精度浮点型, 单位为升, 采样深度为单精度浮点数, 单位为米。

监测数据: 监测数据的总长度由任务点中的监测点数决定, 每个监测点的数据长度是2个字节, 为16位整数, 表示在该点的监测时间长度, 单位为秒。

- 获取当前任务数据

28. 命令-0x0119 (Client → USV), 对应的mqtt消息为 /wp/get

获取无人船当前的任务数据, 无人船接收到这个指令后, 会马上给客户端返回任务数据/wp/info, 该命令无参数。

- ping (是否自动?)

29. 命令-0x011A (Client → USV), 对应的mqtt消息为 /ping

基站与无人船建立连接后, 会每隔3秒向无人船发送这个命令, 表示基站与无人船的连接正常。该命令无参数。

- 设置无人船自动航行最小速度

31. 命令-0x011C (Client → USV), 对应的mqtt消息为 /speed/set

设置无人船自动航行的最小速度, 最大速度和速度控制类型, 参数格式为: | 最小速度 (4B) | 最大速度 (4B) |

速度控制类型 (1B) |

最小速度: 单精度浮点数, 取值范围为0~1

最大速度: 单精度浮点数, 取值范围为0~1

控制类型: 0 定功率方式, 1 定速度方式

- 获取设置的自动航行速度参数

32. 命令-0x011D (Client → USV), 对应的mqtt消息为 /speed/get

获取当前设置的自动航行的速度参数, 无人船收到改命令后, 马上会发送 /speed

- 获取无人船GPS数据

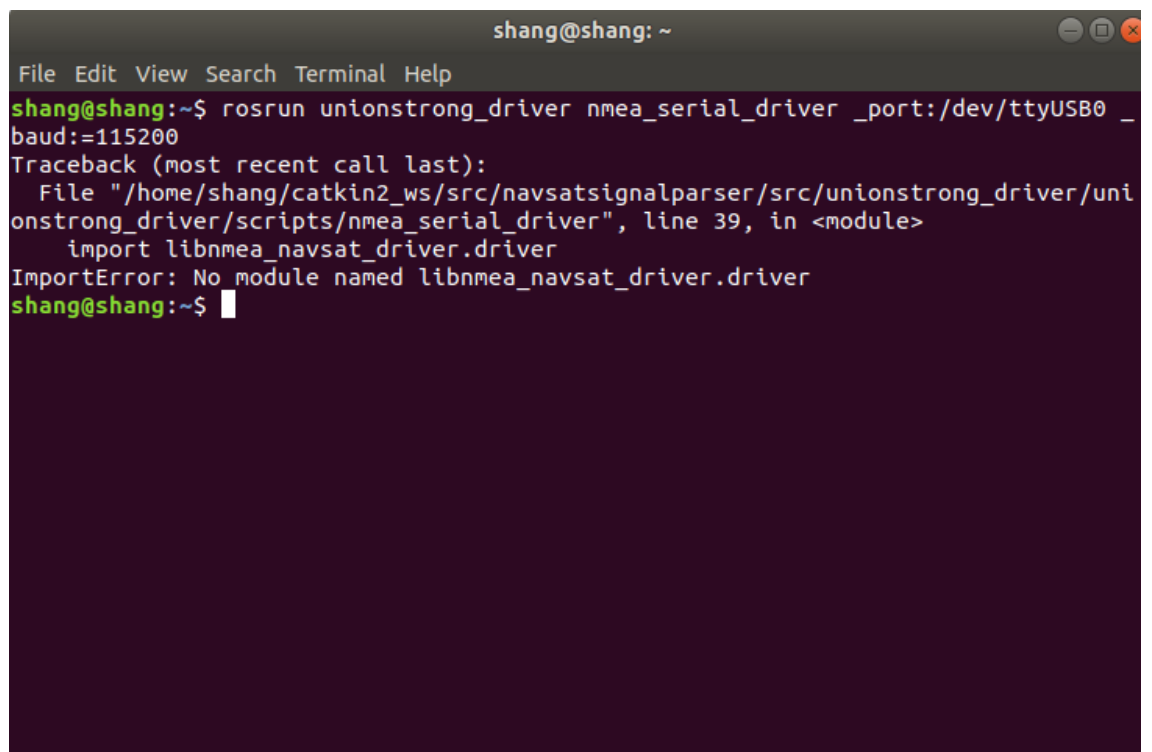
- <https://www.usilab.cn:10080/yaoxt3/NavSatSignalParser/src/master> 此软件, 串口获取数据

- 一个类似教程 [https://blog.csdn.net/m0\\_37857300/article/details/82391023](https://blog.csdn.net/m0_37857300/article/details/82391023)

- 要求

- 串口工具-cutecom

- `sudo apt-get install cutecom`
- 串口连接线
- 软件
  - ros环境搭建与包编译
    - [https://blog.csdn.net/jimson\\_zhu/article/details/81226978](https://blog.csdn.net/jimson_zhu/article/details/81226978)
      - [https://blog.csdn.net/qg\\_23348963/article/details/79585109](https://blog.csdn.net/qg_23348963/article/details/79585109) `sudo apt-get install ros-kinetic-nmea-navsat-driver` 失败, 去官网下载 [http://wiki.ros.org/nmea\\_navsat\\_driver#](http://wiki.ros.org/nmea_navsat_driver#)
      - `echo "source /opt/ros/我们的ros版本(如indigo)/setup.bash" >> ~/.bashrc` 配置系统环境
      - `$ tar zxvf`
      - `~/Downloads/NavSatSignalParser-master.tar.gz -C ~/catkin_ws/src`  
解压文件到指定文件夹
      - <https://blog.csdn.net/wallewa96/article/details/70500328>
      - `$ echo "export ROS_PACKAGE_PATH=~/catkin_ws/src:${ROS_PACKAGE_PATH}" >> ~/.bashrc`  
`$ ~/.bashrc` 将创建的文件夹添加到环境变量中
  - 出错



```

shang@shang: ~
File Edit View Search Terminal Help
shang@shang:~$ rosruntime unionstrong_driver nmea_serial_driver _port:/dev/ttyUSB0 _
baud:=115200
Traceback (most recent call last):
  File "/home/shang/catkin2_ws/src/navsat_signal_parser/src/unionstrong_driver/uni
onstrong_driver/scripts/nmea_serial_driver", line 39, in <module>
    import libnmea_navsat_driver.driver
ImportError: No module named libnmea_navsat_driver.driver
shang@shang:~$

```

- ImportError: No module named libnmea\_navsat\_driver.driver
  - 解决
  - <https://answers.ros.org/question/249490/can-the-nmea-drivers-be-used-with-kinetic/>
    - `source ~/catkin_ws/devel/setup.bash` `catkin_make` 后再 `source` 一遍



- 错误2 ImportError: No module named lcm

```
shang@shang: ~/catkin2_ws
File Edit View Search Terminal Help
[ 0%] Built target geometry_msgs_generate_messages_lisp
[ 0%] Built target std_msgs_generate_messages_eus
[ 0%] Built target geometry_msgs_generate_messages_eus
[ 0%] Built target std_msgs_generate_messages_cpp
[ 0%] Built target nav_msgs_generate_messages_cpp
[ 0%] Built target geometry_msgs_generate_messages_cpp
[ 14%] Built target unionstrong_msgs_generate_messages_nodejs
[ 42%] Built target unionstrong_msgs_generate_messages_py
[ 57%] Built target unionstrong_msgs_generate_messages_lisp
[ 85%] Built target unionstrong_msgs_generate_messages_eus
[100%] Built target unionstrong_msgs_generate_messages_cpp
[100%] Built target unionstrong_msgs_generate_messages
shang@shang:~/catkin2_ws$ source ~/catkin2_ws/devel/setup.bash
shang@shang:~/catkin2_ws$ rosrn unionstrong_driver nmea_serial_driver _port:/dev/ttyUSB0 _baud:=115200
Traceback (most recent call last):
  File "/home/shang/catkin2_ws/src/navsatsignalparser/src/unionstrong_driver/unionstrong_driver/scripts/nmea_serial_driver", line 39, in <module>
    import libnmea_navsat_driver.driver
  File "/home/shang/catkin2_ws/src/navsatsignalparser/src/unionstrong_driver/unionstrong_driver/src/libnmea_navsat_driver/driver.py", line 3, in <module>
    import lcm
ImportError: No module named lcm
shang@shang:~/catkin2_ws$ 6~
```

- 获取无人船激光雷达数据

- <https://blog.csdn.net/littlethunder/article/details/51920681>

- 错误

- 配置有线网络

```
shang@shang:~$ /etc/init.d/networking restart
[....] Restarting networking (via systemctl): networking.serviceJob for networki
ng.service failed because the control process exited with error code. See "syste
mctl status networking.service" and "journalctl -xe" for details.
failed!
```

- 网卡配置文件添加一个不可达路由就会出现这个报错。 <https://blog.csdn.net/w294954902/article/details/82628909>

- 情况分析

- 工控机使用

- gps, 激光雷达使用

- 1.硬件, 连接

- 电池

- 2.网络层

- 网络配置

- 

- 3.数据处理

- gps 位置

- 数据格式转换

- 单位, 数据使用

- 激光雷达
  - ros图像显示
  - 数据处理
- 程序编程，控制
  - python
    - 利用工控机，通过mqtt协议与主控通讯，达到控制效果
    - 控制程序

```
import paho.mqtt.client as mqtt
import struct

client = mqtt.Client()

def on_connect(client,userdata,flags,rc):
    print("connected with result code " + str(rc))
    client.subscribe('/pose')

def on_message(client,userdata,msg):
    topic = '/ctrl'
    v=0.5
    r=0.5
    p=50
    data = struct.pack('>ffB',v,r,p)
    client.publish(topic,payload=data)
    if msg.topic == '/pose':
        a,b,c = struct.unpack('>fff',msg.payload)
        print("pose",a,b,c)

client.on_connect=on_connect
client.on_message=on_message
client.connect("192.168.1.230",1883,600)
client.loop_forever()
~
~
~
~
~
```

- ros
  - 数据获取时用到，需要知道与python结合使用