Three-way_timestamps_Rpi

October 26, 2019

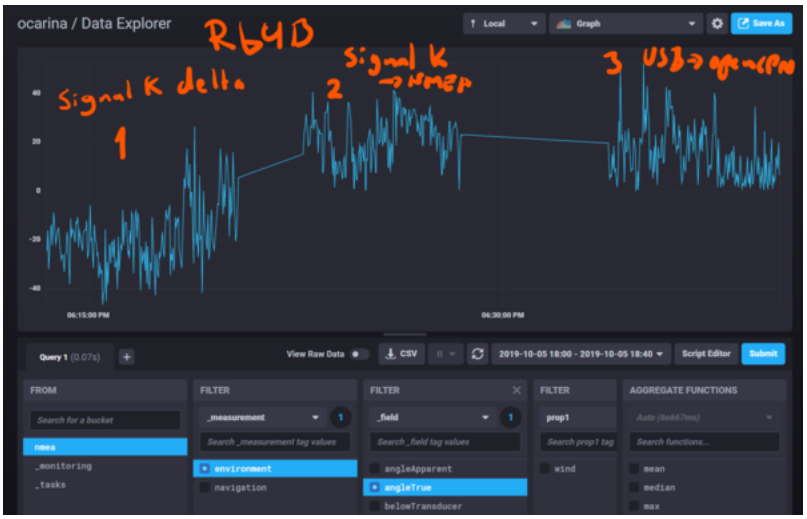
1 Comparison of timestamps in three alternative NMEA data paths

1.1 Rasbian "Buster" - OpenCPN v5.0.0 - Signal K v1.17.0 - DashT v.0.5.2

We observe a five minute sampling period stored in InfluxDB database for each of the use case for single value of Apparent Wind Angle:

1. data via Signal K delta TCP channel with Signal K timestamps at its own reception
2. data via Signal K to NMEA-0183 via TCP channel timestamps at reception at the InfluxDB instruments
3. data directly from USB to OpenCPN

In all above cases the USB is set to 115200 baud at reception on Rasbian "Buster" (Raspberry Pi 4B 4GB) running OpenCPN v5.0.0. Data is originated from Raymarine SeaTalk (4800 baud) and converted to USB in MiniPlex II multiplexer - about 40 values per second are transmitted through this channel but only True Wind Angle timestamp behaviour is observed in this test.



(zoom)

1.2 Numerical comparison

[33]:
```
import numpy as np
import pandas as pd
```

1.2.1 Data via Signal K delta TCP channel

(zoom)

[34]:
```
df = pd.read_csv("2019-10-05_224308_SignalK_delta_zoom_Rpi.csv", sep=',',␣
↪header=3)
```

[35]:
```
df.head()
```

[35]:
```
   Unnamed: 0  result  table                 _start                  _stop  \
0         NaN     NaN      0  2019-10-05T16:14:00Z  2019-10-05T16:19:00Z
1         NaN     NaN      0  2019-10-05T16:14:00Z  2019-10-05T16:19:00Z
2         NaN     NaN      0  2019-10-05T16:14:00Z  2019-10-05T16:19:00Z
3         NaN     NaN      0  2019-10-05T16:14:00Z  2019-10-05T16:19:00Z
4         NaN     NaN      0  2019-10-05T16:14:00Z  2019-10-05T16:19:00Z

                          _time  _value     _field _measurement prop1
0  2019-10-05T16:14:01.169Z   -28.1  angleTrue  environment  wind
1  2019-10-05T16:14:03.106Z   -26.6  angleTrue  environment  wind
2  2019-10-05T16:14:05.045Z   -26.1  angleTrue  environment  wind
3  2019-10-05T16:14:06.049Z   -30.1  angleTrue  environment  wind
4  2019-10-05T16:14:07.956Z   -33.1  angleTrue  environment  wind
```

[36]:
```
df._value.describe()
```

[36]:
```
count    166.000000
mean     -21.151807
std       11.759992
min      -46.600000
25%      -28.975000
50%      -22.100000
75%      -14.725000
max       26.500000
Name: _value, dtype: float64
```

[37]:
```
df1 = pd.to_datetime(df['_time'])
```

[38]:
```
df1.describe()
```

[38]:
```
count                                166
unique                               166
top       2019-10-05 16:15:40.881000+00:00
freq                                   1
first     2019-10-05 16:14:01.169000+00:00
last      2019-10-05 16:18:59.339000+00:00
Name: _time, dtype: object
```

[39]:
```
df2 = df1.astype(np.int64).div(1e6)
```

[40]:
```
df3 = df2.diff()
```

[41]:
```
df3.describe()
```

[41]:
```
count     165.000000
mean     1807.090909
std       313.462833
min      1000.000000
25%      1863.000000
50%      1931.000000
75%      1942.999756
max      2225.000000
Name: _time, dtype: float64
```

1.2.2 Data via Signal K to NMEA-0183 converter TCP channel

(zoom)

[42]:
```
nf = pd.read_csv("2019-10-05_224509_SignalK_NMEA_TCP_zoom_Rpi.csv", sep=',',␣
↪header=3)
```

[43]:
```
nf.head()
```

[43]:
```
   Unnamed: 0  result  table                 _start                  _stop  \
0         NaN     NaN      0  2019-10-05T16:23:00Z  2019-10-05T16:28:00Z
1         NaN     NaN      0  2019-10-05T16:23:00Z  2019-10-05T16:28:00Z
2         NaN     NaN      0  2019-10-05T16:23:00Z  2019-10-05T16:28:00Z
3         NaN     NaN      0  2019-10-05T16:23:00Z  2019-10-05T16:28:00Z
4         NaN     NaN      0  2019-10-05T16:23:00Z  2019-10-05T16:28:00Z

                      _time  _value     _field _measurement prop1
0  2019-10-05T16:23:00.423Z    19.6  angleTrue  environment  wind
1  2019-10-05T16:23:02.379Z    26.6  angleTrue  environment  wind
2  2019-10-05T16:23:04.295Z    26.1  angleTrue  environment  wind
3   2019-10-05T16:23:05.38Z    26.1  angleTrue  environment  wind
4  2019-10-05T16:23:07.104Z    40.6  angleTrue  environment  wind
```

[44]:
```
nf._value.describe()
```

[44]:
```
count    188.000000
mean      21.622872
std        9.844945
min        0.500000
25%       13.850000
50%       22.850000
75%       29.600000
max       41.200000
Name: _value, dtype: float64
```

[45]:
```
nf1 = pd.to_datetime(nf['_time'])
```

[46]:
```
nf1.describe()
```

[46]:
```
count                                188
unique                               188
top       2019-10-05 16:24:50.701000+00:00
freq                                   1
first     2019-10-05 16:23:00.423000+00:00
last      2019-10-05 16:27:59.431000+00:00
Name: _time, dtype: object
```

[47]:
```
nf2 = nf1.astype(np.int64).div(1e6)
```

[48]:
```
nf3 = nf2.diff()
```

[49]:
```
nf3.describe()
```

[49]:
```
count     187.000000
mean     1598.973262
std       422.940687
min      1000.000000
25%      1026.500000
50%      1877.000000
75%      1928.000000
max      2036.000000
Name: _time, dtype: float64
```

1.2.3 Data without Signal K directly from USB

(zoom)

[50]:
```
of = pd.read_csv("2019-10-05_224657_USB_to_0_zoom_Rpi.csv", sep=',', header=3)
```

[51]:
```
of.head()
```

[51]:

	Unnamed: 0	result	table	_start	_stop	_time	_value	_field	_measurement	prop1
0	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	2019-10-05T16:34:01.719Z	8.1	angleTrue	environment	wind
1	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	2019-10-05T16:34:03.658Z	9.1	angleTrue	environment	wind
2	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	2019-10-05T16:34:05.585Z	4.1	angleTrue	environment	wind
3	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	2019-10-05T16:34:07.545Z	1.6	angleTrue	environment	wind
4	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	2019-10-05T16:34:09.462Z	7.6	angleTrue	environment	wind

[52]:
```
of._value.describe()
```

```
[52]: count    172.000000
      mean      16.232558
      std       10.655020
      min        0.500000
      25%        7.900000
      50%       15.600000
      75%       22.225000
      max       52.100000
      Name: _value, dtype: float64
```

```
[53]: of1 = pd.to_datetime(of['_time'])
```

```
[54]: of1.describe()
```

```
[54]: count                                172
      unique                               172
      top       2019-10-05 16:36:01.753000+00:00
      freq                                   1
      first     2019-10-05 16:34:01.719000+00:00
      last      2019-10-05 16:38:58.943000+00:00
      Name: _time, dtype: object
```

```
[55]: of2 = of1.astype(np.int64).div(1e6)
```

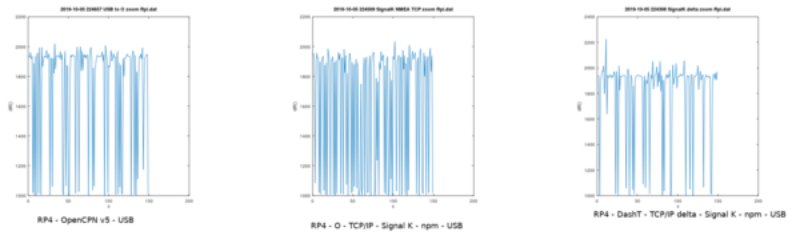
```
[56]: of3 = of2.diff()
```

```
[57]: of3.describe()
```

```
[57]: count     171.000000
      mean     1738.152047
      std       366.632300
      min      1000.000000
      25%      1825.499634
      50%      1925.000000
      75%      1942.500000
      max      2025.999756
      Name: _time, dtype: float64
```

1.3 Graphical comparison

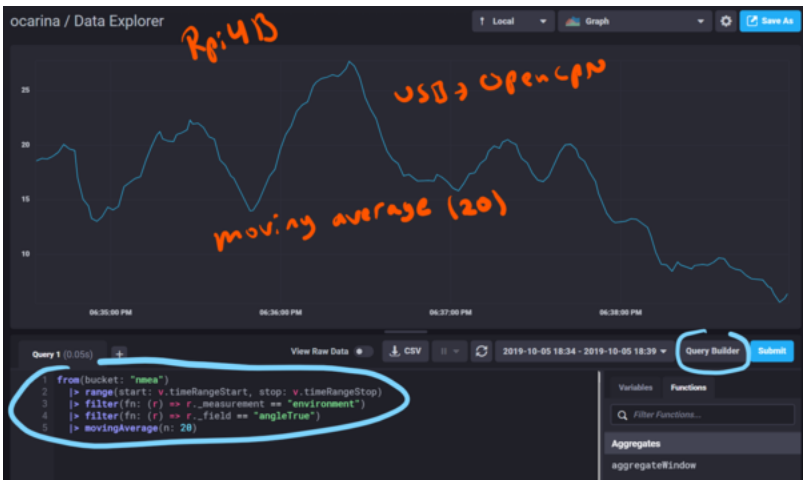
GNU Octave v5.1/MATLAB script `jitterplots.m` was developed to present the jitter in graphical plot format, including the cumulative difference. The datafiles were truncated to hold 150 measurement points each. A helper script `raspberryjitter.m` was used to produce the following plots.



(zoom)

1.4 Miscellaneous observations

1.4.1 Applying moving average



(zoom)

[58]:
```
af = pd.read_csv("2019-10-05_224657_USB_to_0_zoom_Rpi_moving_average.csv",␣
↪sep=',', header=3)
```

[59]:
```
af.head()
```

[59]:

	Unnamed: 0	result	table	_start	_stop	\
0		NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z
1		NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z
2		NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z
3		NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z
4		NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z

	_time	_value	_field	_measurement	prop1
0	2019-10-05T16:34:33.669Z	18.570	angleTrue	environment	wind

```
1  2019-10-05T16:34:35.605Z  18.795  angleTrue  environment  wind
2  2019-10-05T16:34:37.541Z  18.745  angleTrue  environment  wind
3  2019-10-05T16:34:39.468Z  18.995  angleTrue  environment  wind
4  2019-10-05T16:34:41.444Z  19.365  angleTrue  environment  wind
```

[60]:
```
af._value.describe()
```

[60]:
```
count    153.000000
mean      16.781928
std        5.240853
min        5.600000
25%       13.300000
50%       17.200000
75%       20.085000
max       27.700000
Name: _value, dtype: float64
```

1.5 Summary of results

data path	timestamp	standard deviation	maximum time difference
1 Signal K delta	at source	313 ms	2225 ms
2 Signal K NMEA TCP	at reception	423 ms	2036 ms
3 USB to OpenCPN	at reception	367 ms	2025 ms

1.6 Conclusion

Judged by a human eye there is no difference between the three methods - the needles and values are jumping back and forth as always!

The difference will come apparent when we want to eliminate that jumping by applying some statistical and continuous algorithms on the received time series data. The accuracy of the time stamps is, of course important for any time series analysis.

> Based on the graphs we make an assumption that the upstream system, the Miniplex multiplexer receives the wind data from SeaTalk with $1Hz$ $(1/s)$ frequency and that it adds some jitter into the system but that jitter is not possible to measure. However, that particular jitter source is the same to all three methods observed, allowing the comparison

On Raspberry with Raspian Linux we can observe:

1. Signal K NMEA-0183 conversion and TCP/IP feed to OpenCPN which the passes the data to the plug-in loses less packages than the other two solutions - it receives more packages in a period. However, jitter-wise it the worst of the three - about every other second a data frame is missed.
2. Somewhat surprisingly, the direct connection of OpenCPN to the USB source is performing more badly than when it receives the data indirectly through Signal K using TCP/IP. The

jitter is less important, however.

3. Jitter-wise, the direct connection to the Signal K is the best, as expected. However, on this platform it is losing more packages than the two other methods. This can be explained partly by the low performance of the CPU on this platform - there is probably a saturation either on the npm-thread on Signal K side sending the delta-values or the on the POSIX communication thread of the Dashboard-Tactics plug-in receiving them, or on both.

On this platform the Signal K TCP feed to OpenCPN is the method to use if the number of samples is preferred, wxWidgets and OpenCPN TCP/IP implementation works very well here.

The direct Signal K connection's advantage is the reduced jitter also in this platform, not to mention the functional enhancements it provides. It can be used together with the Signal K TCP/IP feed since the requirement was precisely to reduce jitter for the key parameters.

Albeit the OpenCPN direct USB connection on a modest Rasberry Pi 4B board allows similar performances than more powerful i7 CPU based processor under Window 10, a similar level of jitter with lesser performance would suggest to prefer the usage of Signal K and TCP/IP. Of course, a direct TCP/IP connection to the Miniplex multiplexer is possible but not studied. Performance-wise this can be expected to provide similar results but since there is no performance penalty by using Signal K it would be pity not to profit from the extra functionality it provides.

Finally, let's consider the following statement from *Numerical Recipes In C: The Art of Scientific Computing (ISBN 0-521-43108-5)* concerning irregularly sampled data, where the values f_i are not uniformly spaced in time:

> If the change in f across the full width of the $N = nL + nR + 1$ point window is less than $\sqrt{N/2}$ times the measurement noise on a single point, then the cheap method can be used.

The "*cheap method*" is to ignore the jitter. But as the above graphics illustrates, were are constantly switching, here for the wind data, between $1Hz$ and $0.5Hz$ for f_i!