demo2_jupyter_accessing_AESR_mongodb_and_plotting_data

September 21, 2017

This Jupyter Notebook shows examples of loading AESR data
170921 RG

It is assumed that a mongod is running (on a remote raspi in this case, but could be local) with AESR data

Looking at data from holiday run on Sept 21, 2017

1 Import Packages

```
In [1]: import matplotlib
        %matplotlib inline
        import matplotlib.pyplot as plt
        import numpy as np
        import pprint
        import time
        import datetime
        import math
        matplotlib.rcParams['figure.figsize'] = (10,7) # Make plots appropriate size
```

---

2 Attach to Mongo Database with AESR Data

3 AESR_20170921T122431 contains very useful sensor data

```
In [3]: from pymongo import MongoClient
        client = MongoClient('aesraspi.local')
        client.database_names()
        # A list of all databases should appear
```

```
Out[3]: ['AESR_20170921T122431',
         'AESR_20170921T113547',
         'AESR_20170921T113919',
         'AESR_20170921T113719',
         'AESR_20170921T122413',
```

```
'AESR_20170921T122343',
'AESR_20170921T113505',
'AESR_20170921T113826',
'local']
```

```
In [13]: plot_title = 'AESR_20170921T122431'

         db = client.AESR_20170921T122431 # This dataset has a lot of useful data
         db.collection_names() # List the colllections that are available
```

```
Out[13]: ['system.indexes', 'log', 'data', 'wps']
```

```
In [14]: c = db.data # Collection with all sensor data
         # dir(c) # Check methods that are available on the collection
```

3.1 Look at Individual Database Entries

```
In [6]: first = c.find_one() # First element
        pprint.pprint(first) # Look at detailed structure of first element

        t1 = first['_id'].generation_time # Timestamp for first element
        t1

        # If you want to print each of the entries
        # for x in c.find({'atype':'VOLT_MON'}):
        #     pprint.pprint(x['param']['volt'])
```

```
{'_id': ObjectId('59c3e7d6d6861b73e2cba15a'),
 'atype': 'VOLT_MON',
 'itype': None,
 'param': {'dir_volt': 21724, 'volt': 11.839941326334424},
 'ts': 1506011094.4221318}
```

```
Out[6]: datetime.datetime(2017, 9, 21, 16, 24, 54, tzinfo=<bson.tz_util.FixedOffset object at 0x
```

```
In [7]: first['atype'] # Access data type
```

```
Out[7]: 'VOLT_MON'
```

3.2 Example loading Battery Voltage Data

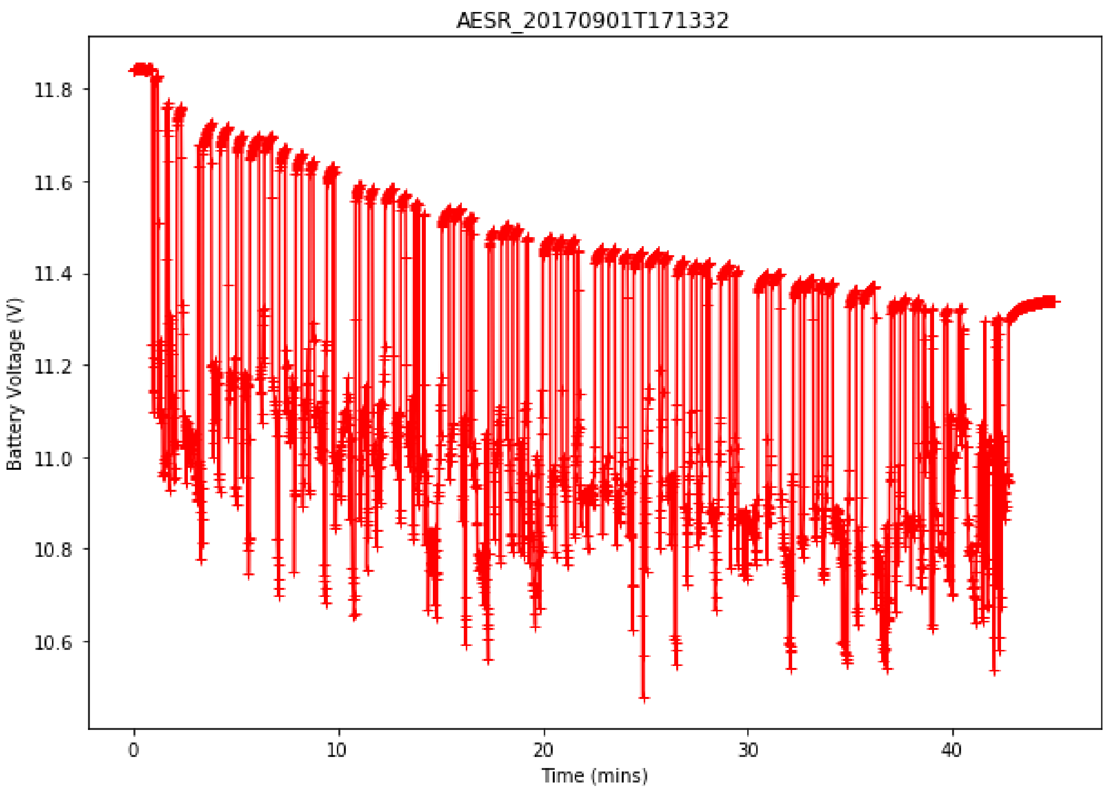
```
In [7]: # Want to add time calculation - seconds since start of sweep
        # Build a new numpy array with the time in seconds since start and the voltage

        data = []
        for row in c.find({'atype':'VOLT_MON'}):
```

```
        data.append( [
            ( row['_id'].generation_time - first['_id'].generation_time ).total_seconds()
            , row['param']['volt']
            ] )

    x = np.array(data,dtype = np.dtype('f4') ) # Create an np array using 4 byte precision u

    x
```

Out[7]:
```
array([[  0.00000000e+00,   1.18399410e+01],
       [  1.00000000e+00,   1.18410311e+01],
       [  2.00000000e+00,   1.18399410e+01],
       ...,
       [  2.69600000e+03,   1.13401613e+01],
       [  2.69700000e+03,   1.13407059e+01],
       [  2.69800000e+03,   1.13407059e+01]], dtype=float32)
```

In [8]:
```
plt.plot( x[:,0] / 60. ,x[:,1] , 'r+-')
plt.xlabel('Time (mins)')
plt.ylabel('Battery Voltage (V)')
plt.title('AESR_20170901T171332')
```

Out[8]:
```
<matplotlib.text.Text at 0x111009ef0>
```



3.3 Look at list of all Types of Data Available in Collection and printing some of the records

In [9]:
```
for ii in c.distinct('atype'): # A list of distinct atype's
    print('%-12s    %4d'%(ii , c.find({'atype':ii}).count() )) # How many records of eac
```

```
VOLT_MON        2684
TEMP            2684
PRES            2684
ADC             2684
ENVIRON         2684
GPS             2684
```

In [10]:
```
ii = 0
for row in c.find():
    pprint.pprint(row)
    ii += 1
    if(ii>10):
        break
```

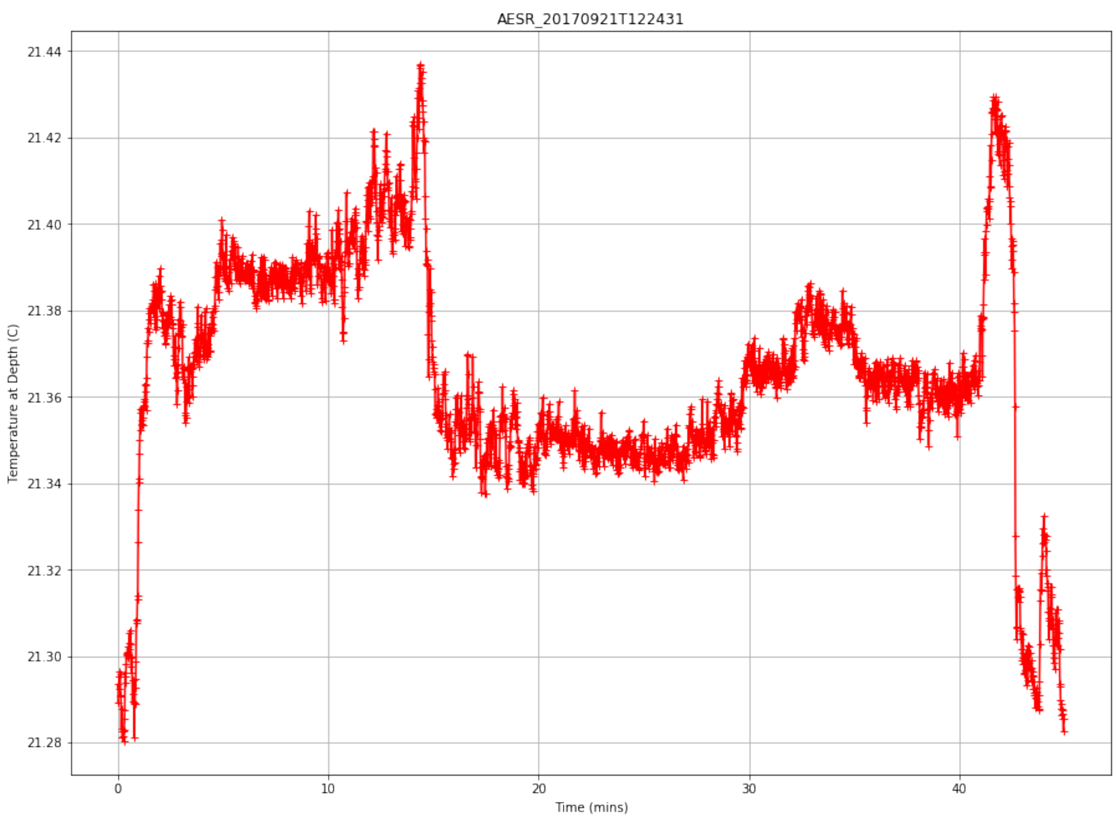

```
{'_id': ObjectId('59c3e7d6d6861b73e2cba15a'),
 'atype': 'VOLT_MON',
 'itype': None,
 'param': {'dir_volt': 21724, 'volt': 11.839941326334424},
 'ts': 1506011094.4221318}
{'_id': ObjectId('59c3e7d6d6861b73e2cba15b'),
 'atype': 'TEMP',
 'itype': None,
 'param': {'temp_c': 21.289114993670182},
 'ts': 1506011094.4383376}
{'_id': ObjectId('59c3e7d6d6861b73e2cba15c'),
 'atype': 'PRES',
 'itype': None,
 'param': {'pres_mbar': 1034.3361783252562, 'temp_c': 22.09001757112739},
 'ts': 1506011094.4961903}
{'_id': ObjectId('59c3e7d6d6861b73e2cba15d'),
 'atype': 'ADC',
 'itype': None,
 'param': {'adc_val': 10486,
           'mgL': 8.293326154240548,
           'volt': 1.9661850032044437},
 'ts': 1506011094.5108275}
{'_id': ObjectId('59c3e7d6d6861b73e2cba15e'),
 'atype': 'ENVIRON',
 'itype': None,
 'param': {'hum_per': 71.1607358020131,
           'pres_pas': 101588.59768460268,
           'temp_cel': 22.567456034902715},
```

```
 'ts': 1506011094.5195847}
{'_id': ObjectId('59c3e7d6d6861b73e2cba15f'),
 'atype': 'GPS',
 'itype': None,
 'param': {'lat': 41.735471667, 'lon': -71.325221667},
 'ts': 1506011094.5196419}
{'_id': ObjectId('59c3e7d7d6861b73e2cba161'),
 'atype': 'VOLT_MON',
 'itype': None,
 'param': {'dir_volt': 21726, 'volt': 11.841031359599599},
 'ts': 1506011095.4302835}
{'_id': ObjectId('59c3e7d7d6861b73e2cba162'),
 'atype': 'TEMP',
 'itype': None,
 'param': {'temp_c': 21.293594934844464},
 'ts': 1506011095.4454215}
{'_id': ObjectId('59c3e7d7d6861b73e2cba163'),
 'atype': 'PRES',
 'itype': None,
 'param': {'pres_mbar': 1033.5041671577055, 'temp_c': 22.090667765973922},
 'ts': 1506011095.502204}
{'_id': ObjectId('59c3e7d7d6861b73e2cba164'),
 'atype': 'ADC',
 'itype': None,
 'param': {'adc_val': 10891,
           'mgL': 8.630802703207495,
           'volt': 2.042124820703757},
 'ts': 1506011095.5133624}
{'_id': ObjectId('59c3e7d7d6861b73e2cba165'),
 'atype': 'ENVIRON',
 'itype': None,
 'param': {'hum_per': 70.61365612656536,
           'pres_pas': 101585.05394085418,
           'temp_cel': 22.562364603579045},
 'ts': 1506011095.5215447}
```

3.4 Temperature (degC) vs Time

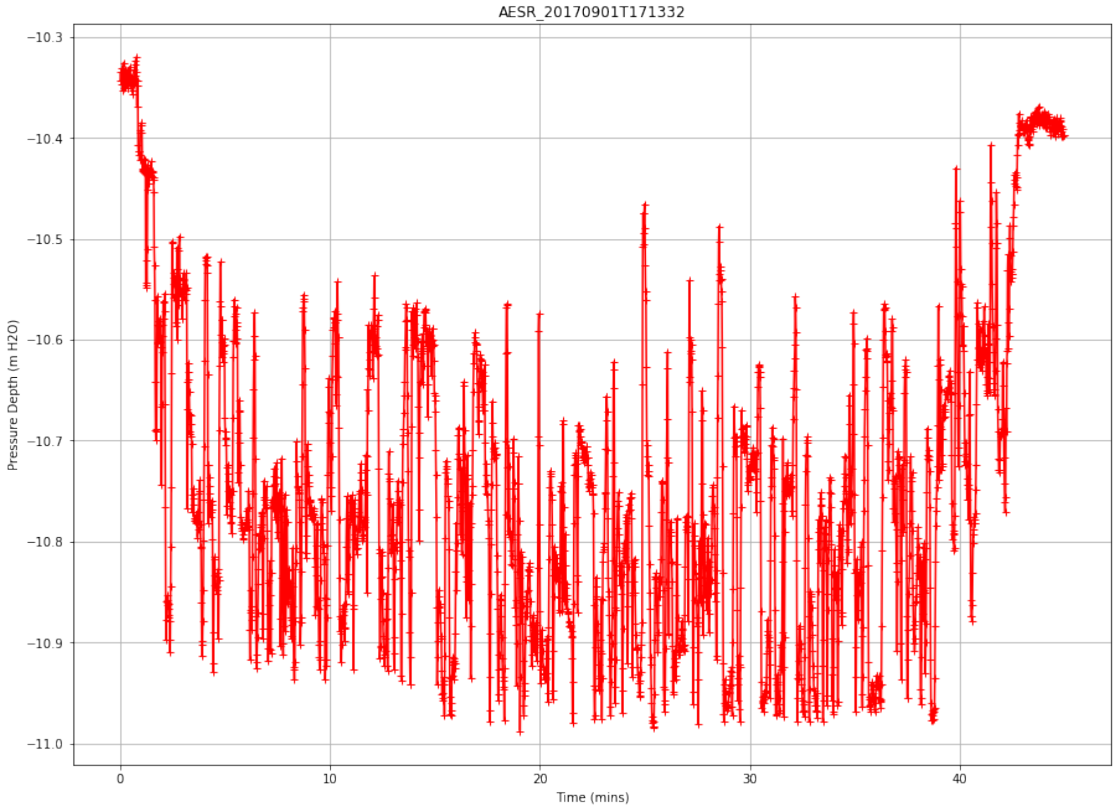
In [15]:
```
matplotlib.rcParams['figure.figsize'] = (15,11) # Reset plot size

sel_atype = 'TEMP'
sel_param = 'temp_c'
plot_ylabel = 'Temperature at Depth (C)'
plot_scale = 1
plot_offset = 0.

data = []
```

```
for row in c.find({'atype':sel_atype}):
    data.append( [
        ( row['_id'].generation_time - first['_id'].generation_time ).total_seconds()
        , row['param'][sel_param]
        ] )

x = np.array(data,dtype = np.dtype('f4') ) # Create an np array using 4 byte precision

# Remove any bad reads from thermometer set
x = x[ x[:,1] > 0. , :]

plt.plot( x[:,0] / 60. ,plot_scale * x[:,1] - plot_offset , 'r+-')
plt.xlabel('Time (mins)')
plt.ylabel(plot_ylabel)
plt.title(plot_title)
plt.grid(True)

matplotlib.rcParams['figure.figsize'] = (10,7) # Reset plot size
```



3.5 Pressure versus Time

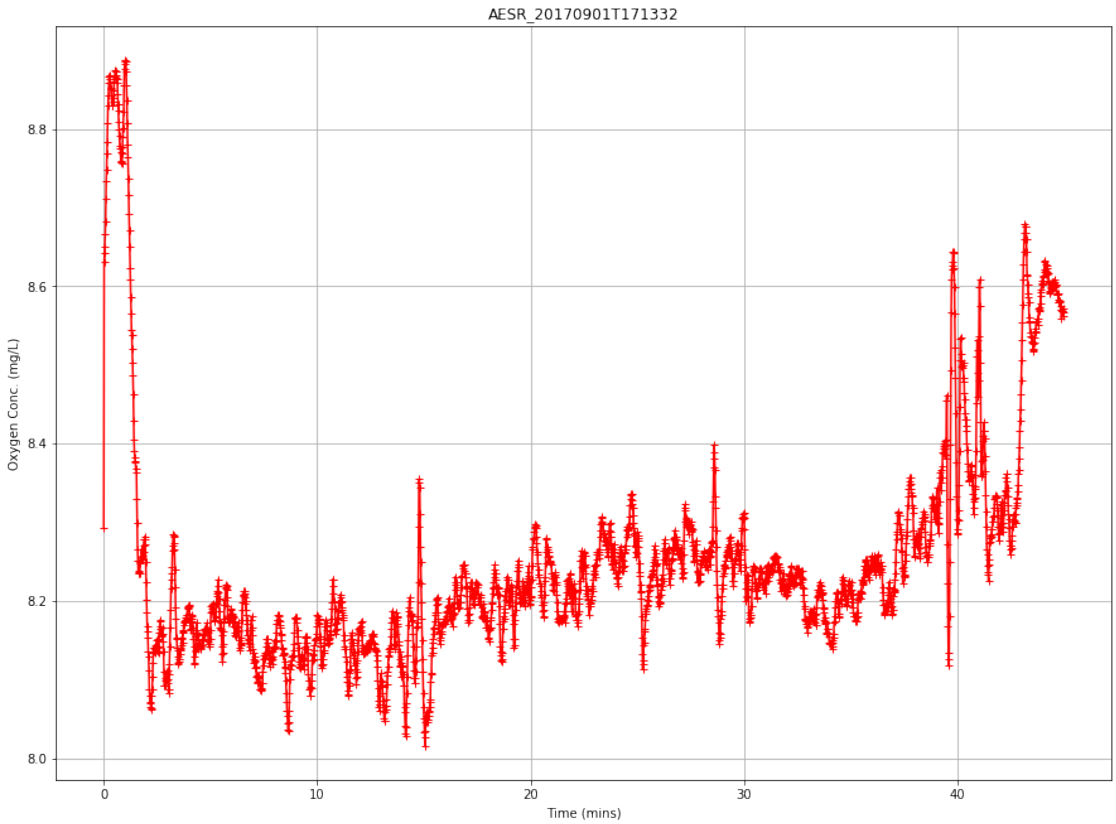
In [16]:
```
matplotlib.rcParams['figure.figsize'] = (15,11) # Reset plot size

sel_atype = 'PRES'
sel_param = 'pres_mbar'
plot_ylabel = 'Pressure Depth (m H2O)'
plot_title = 'AESR_20170901T171332'
plot_scale = -0.01
plot_offset = 0.

data = []
for row in c.find({'atype':sel_atype}):
    data.append( [
        ( row['_id'].generation_time - first['_id'].generation_time ).total_seconds()
        , row['param'][sel_param]
        ] )

x = np.array(data,dtype = np.dtype('f4') ) # Create an np array using 4 byte precision

plt.plot( x[:,0] / 60. ,plot_scale * x[:,1] - plot_offset , 'r+-')
plt.xlabel('Time (mins)')
plt.ylabel(plot_ylabel)
plt.title(plot_title)
plt.grid(True)

matplotlib.rcParams['figure.figsize'] = (10,7) # Reset plot size
```



3.6 Oxygen Levels versus Time

```
In [17]: matplotlib.rcParams['figure.figsize'] = (15,11) # Reset plot size

         sel_atype = 'ADC'
         sel_param = 'mgL'
         plot_ylabel = 'Oxygen Conc. (mg/L)'
         plot_title = 'AESR_20170901T171332'
         plot_scale = 1
         plot_offset = 0.

         data = []
         for row in c.find({'atype':sel_atype}):
             data.append( [
                 ( row['_id'].generation_time - first['_id'].generation_time ).total_seconds()
                 , row['param'][sel_param]
                 ] )

         x = np.array(data,dtype = np.dtype('f4') ) # Create an np array using 4 byte precision

         plt.plot( x[:,0] / 60. ,plot_scale * x[:,1] - plot_offset , 'r+-')
```

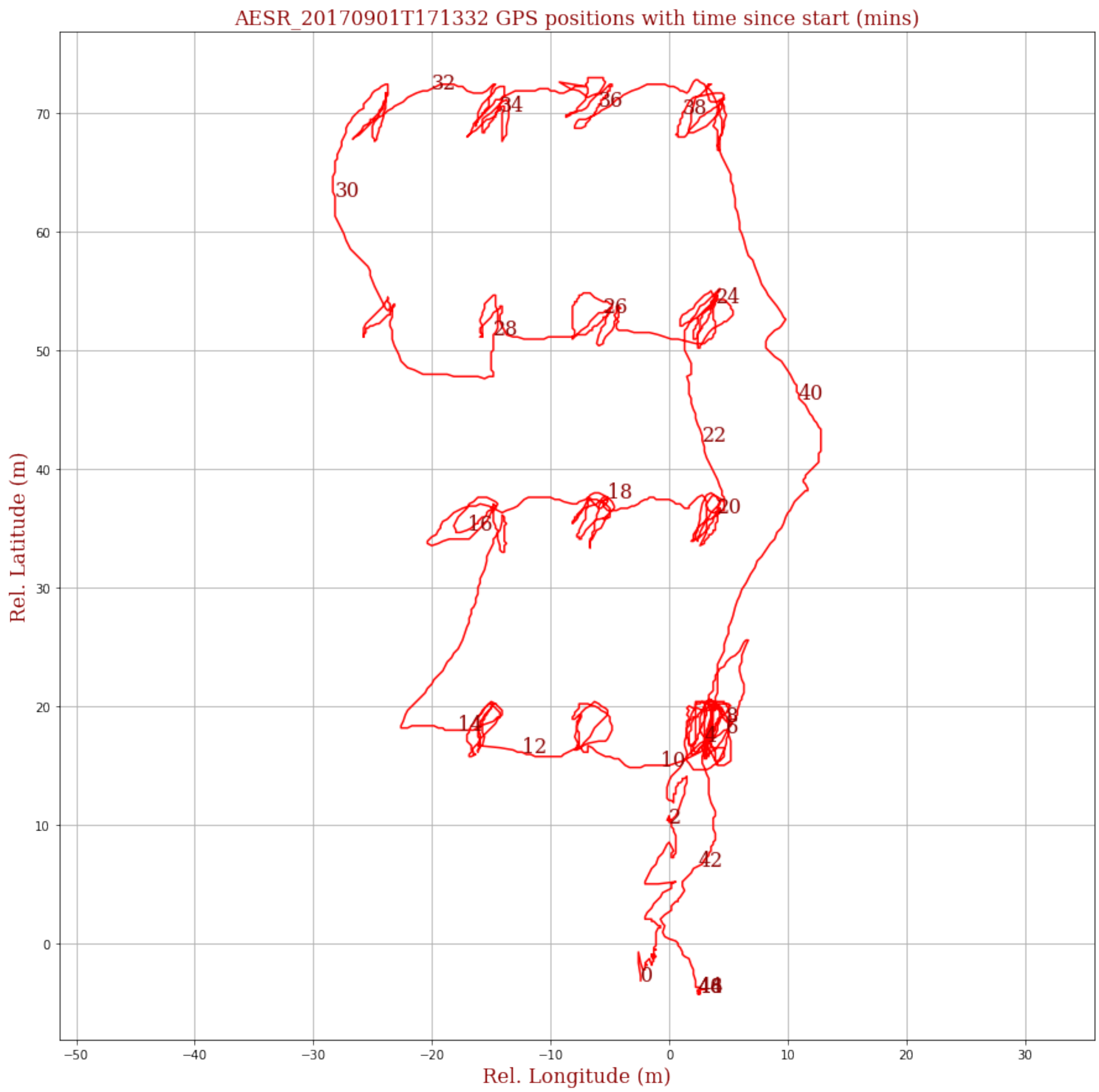
```
plt.xlabel('Time (mins)')
plt.ylabel(plot_ylabel)
plt.title(plot_title)
plt.grid(True)

matplotlib.rcParams['figure.figsize'] = (10,7) # Reset plot size
```



3.7 GPS Coordinates (x,y) with time (minutes) also shown

```
In [18]: # A Plot of the path of AESR with number every 2 minutes of real time

         matplotlib.rcParams['figure.figsize'] = (15,15) # Square plot for this map

         # Look for these tags in the mongo collection
         sel_atype = 'GPS'
         sel_param1 = 'lon'
         sel_param2 = 'lat'
         plot_ylabel = 'Position'
         plot_title2 = plot_title + ' GPS positions with time since start (mins)'

         launch_lon_lat = [-71.3252 , 41.7355] # (0,0) meters position for the plot - this shoul
```

```
scale_deg_to_m = 6371e3/57.296 # Scaling for latitude from degrees to m local on the gr

# Font for the plot and text labels
font = {'family': 'serif',
        'color':  'darkred',
        'weight': 'normal',
        'size': 16,
        }


# BUILD A NUMPY ARRAY WITH COLUMNS OF THE RELEVANT DATA
data = []
for row in c.find({'atype':sel_atype}):
    data.append( [
    ( row['_id'].generation_time - first['_id'].generation_time ).total_seconds()
        # Convert lon and lat to a local x,y coordinate
    , (row['param'][sel_param1]-launch_lon_lat[0]) * scale_deg_to_m # * math.cos( row['
    , (row['param'][sel_param2]-launch_lon_lat[1]) * scale_deg_to_m # lat
    ] )

x = np.array(data,dtype = np.dtype('f4') ) # Create an np array using 4 byte precision

plt.plot( x[:,1] , x[:,2] , 'r-')
plt.xlabel('Rel. Longitude (m)', fontdict = font)
plt.ylabel('Rel. Latitude (m)', fontdict = font)
plt.title(plot_title2 , fontdict = font )
plt.grid(True)
plt.axes().set_aspect('equal', 'datalim')

for ii in range(0,50,2):
    idx = (np.abs( x[:,0]/60. - ii)).argmin() # Find index of array entry closests to g
    plt.text( x[idx,1] , x[idx,2] , str(ii) , fontdict = font )


matplotlib.rcParams['figure.figsize'] = (10,7) # Reset plot size
```



4 Footnotes:

Taking inspiration from https://github.com/Altons/pymongo-tutorial/blob/master/pymongo-tutorial.ipynb

In [ ]: