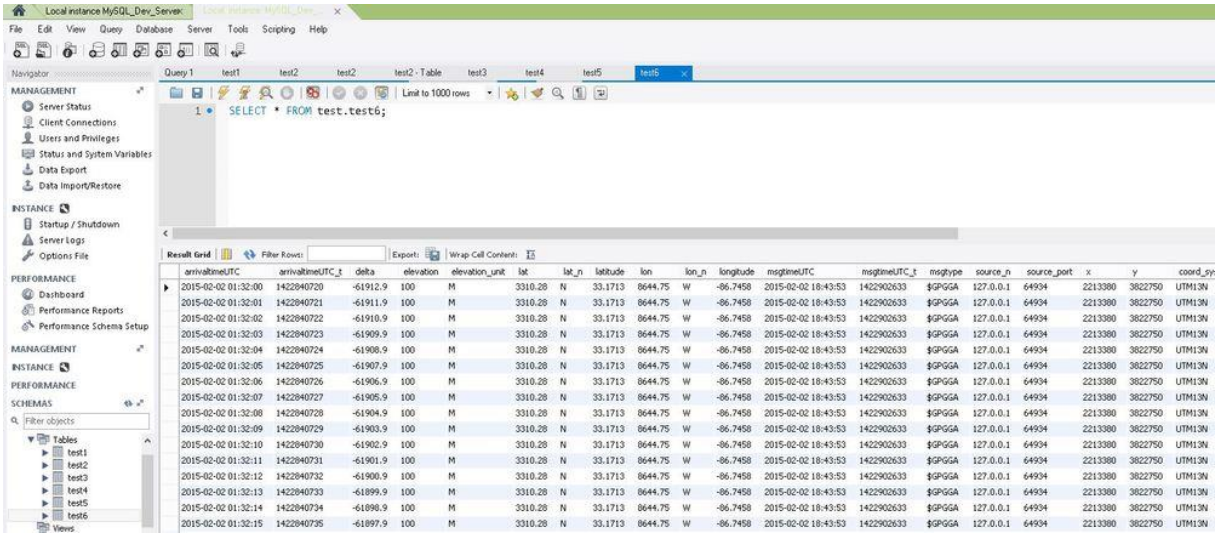
NMEA GPS Reciever, database, and coordinate projection in Python

A GPS NMEA web server application to capture GPGGA messages in Python, transform them into a local coordinate system using pyproj, and save them to a database in real time.

https://www.ryanbaumann.com/blog/2015/2/1/nmea-gps-reciever-database-and-coordinate-projection-in-python

https://github.com/ryanbaumann/NMEA_GPS_Server




```
#### Packages to import ###
import pandas as pd
import numpy as np
import pyproj
import datetime
import sys, os
from sqlalchemy import create_engine
import logging
import pymssql
import re

### Database Connection and Update Functions ###

def write_to_db(engine, tablename, dataframe):
    #Use pandas and sqlalchemy to insert a dataframe into a database

    try:
        dataframe.to_sql(tablename,
```

```
                        engine,
                        index=False,
                        if_exists=u'append',
                        chunksize=100)
        print "inserted into db"
    except: #IOError as e:
        print "Error in inserting data into db"

###logging function###
def log(msg):
    logging.basicConfig(format='%(asctime)s %(message)s',
                        datefmt='%m/%d/%Y %I:%M:%S %p',

filename='F:\\Python_Utilities\\GPS_Program\\gpslogfile.log')
    logging.warning(msg)

### Data processing function ###

def read_nmea(source, port, gpgga):
    #Read a pynmea2 object in, the 'gpgga' parameter, and create a pandas
dataframe
    format = '%Y-%m-%d %H:%M:%S'

    for msg in gpgga:

        arrivaltimeUTC = datetime.datetime.utcnow()
        arrivaltimeUTC_t = (arrivaltimeUTC -
datetime.datetime(1970,1,1)).total_seconds()
        today_utc = arrivaltimeUTC.date()
        msgtimeUTC = datetime.datetime.combine(today_utc,gpgga.timestamp)
        msgtimeUTC_t = (msgtimeUTC -
datetime.datetime(1970,1,1)).total_seconds()
        msgtype = '$GPGGA'
        delta = arrivaltimeUTC_t - msgtimeUTC_t

        values = {'source_n' : str(source),
                  'source_port' : str(port),
                  'msgtype' : str(msgtype),
                  'arrivaltimeUTC' : str(arrivaltimeUTC.strftime(format)),
                  'arrivaltimeUTC_t' : int(arrivaltimeUTC_t),
                  'msgtimeUTC' : str(msgtimeUTC.strftime(format)),
                  'msgtimeUTC_t' : int(msgtimeUTC_t),
                  'delta' : float(delta),
                  'lat' : float(gpgga.lat),
                  'lat_n' : str(gpgga.lat_dir),
                  'latitude' : float(gpgga.latitude),
                  'lon' : float(gpgga.lon),
                  'lon_n' : str(gpgga.lon_dir),
                  'longitude' : float(gpgga.longitude),
                  'elevation' : float(gpgga.altitude),
                  'elevation_unit' : str(gpgga.altitude_units)
                  }

        values_lst.append(values)

    #create the dataframe from the list of the messages
    dataframe(values_lst, index=[0])
    print dataframe

    #typecast the datetime columns as datetimes for database insertion
```

```
    dataframe['arrivaltimeUTC'] =
pd.to_datetime(dataframe['arrivaltimeUTC'])
    dataframe['msgtimeUTC'] = pd.to_datetime(dataframe['msgtimeUTC'])

    return dataframe

### Transform the coordinates, and insert results into the dataframe###

def transform_coords(dataframe):
    #Add projected coordinates to messages
    coord_sys_n = 'UTM13N'
    coord_sys_n2 = 'NAD83_ID_E_USft'
    wgs84=pyproj.Proj("+init=EPSG:4326")# Lat/Lon with WGS84 datum
    UTM13N=pyproj.Proj("+init=EPSG:32613") # NAD83 UTM zone 13N
    NAD83_ID_E=pyproj.Proj("+init=EPSG:2241") #NAD83 Idaho East (US Feet)
    latitude = dataframe['latitude'].values
    longitude = dataframe['longitude'].values
    try:  # !!! Update for each new coordinate system projection
        if latitude>=42.0000 and latitude<=44.7600 and longitude>=-113.2400
and longitude <=-111.0500:
            x, y = pyproj.transform(wgs84,NAD83_ID_E,longitude,latitude)
            dataframe['coord_sys_n']= coord_sys_n2
        else:
            x, y = pyproj.transform(wgs84,UTM13N,longitude,latitude)
            dataframe['coord_sys_n']= coord_sys_n
    except:
        print "projection error!  Assigning X and Y to zero"
        x, y = 0, 0

    #Insert the new values into the dataframe
    dataframe['x'] = x
    dataframe['y'] = y
    #dataframe['coord_sys_n']= coord_sys_n
    dataframe.fillna(0)  #Set X and Y values to zero if there was a
projection error

    return dataframe

### Start the GPS Server Listening service ###
from twisted.internet import reactor, protocol
from twisted.internet.protocol import DatagramProtocol
import pynmea2

#Database variables
tablename = 'gpsReports'
connString = 'mssql+pymssql://dbuser:dbpass@dbserver:dbport/dbname'
engine = create_engine(connString)
server_listen_port = 10110

class Read_Nmea(DatagramProtocol):
    #Read a UDP packet as an NMEA sentance
    streamReader = pynmea2.NMEAStreamReader()
    def datagramReceived(self, data, (host, port)):
        #A list of the incomming messages before writing to the db
        try:
            for line in data.split('\n'):
                nmea_msg = pynmea2.parse(line)
                if nmea_msg.sentence_type == 'GGA':
                    #If message is a GPGGA, continue
                    log(nmea_msg)
                    return nmea_msg
```

```
        except:
            print "error parsing message!"
            pass

reactor.listenUDP(server_listen_port, Read_Nmea())
reactor.run()
```