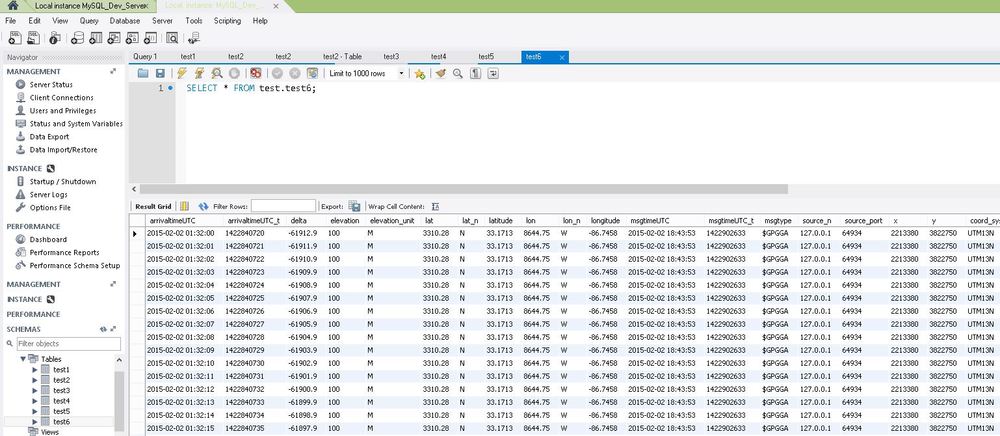
NMEA GPS Reciever, database, and coordinate projection in Python

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()