

NMEA GPS Receiver, 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

Local instance MySQL_Dev_Server

File Edit View Query Database Server Tools Scripting Help

Navigator

Query1 test1 test2 test2 test2 - Table test3 test4 test5 test6

MANAGEMENT

Server Status

Client Connections

Users and Privileges

Status and System Variables

Data Export

Data Import/Restore

INSTANCE

Startup / Shutdown

Server Logs

Options File

PERFORMANCE

Dashboard

Performance Reports

Performance Schema Setup

MANAGEMENT

INSTANCE

PERFORMANCE

SCHEMAS

Filter objects

Tables

test1

test2

test3

test4

test5

test6

Limit to 1000 rows

1 SELECT * FROM test.test6;

Result Grid

Filter Rows

Export

Wrap Cell Contents

	arrivalTimeUTC	arrivalTimeUTC_s	delta	elevation	elevation_unit	lat	lat_n	latitude	lon	lon_n	longitude	msgtimeUTC	msgtimeUTC_s	msgtype	source_n	source_port	x	y	coord_sys
▶	2015-02-02 01:32:00	1422840720	-61912.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:01	1422840721	-61911.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:02	1422840722	-61910.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:03	1422840723	-61909.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:04	1422840724	-61908.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:05	1422840725	-61907.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:06	1422840726	-61906.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:07	1422840727	-61905.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:08	1422840728	-61904.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:09	1422840729	-61903.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:10	1422840730	-61902.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:11	1422840731	-61901.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:12	1422840732	-61900.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:13	1422840733	-61899.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:14	1422840734	-61898.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N
	2015-02-02 01:32:15	1422840735	-61897.9	100	M	3310.28	N	33.1713	8644.75	W	-86.7458	2015-02-02 18:43:53	1422902633	\$GPGGA	127.0.0.1	64934	2213380	3822750	UTM13N

```
#### 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 pymysql
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_Uutilities\\GPS_Program\\gpslogfile.log')
    logging.warning(msg)

### Data processing function ###

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

    for msg in gpvga:

        arrivaltimeUTC = datetime.datetime.utcnow()
        arrivaltimeUTC_t = (arrivaltimeUTC -
datetime.datetime(1970,1,1)).total_seconds()
        today_utc = arrivaltimeUTC.date()
        msgtimeUTC = datetime.datetime.combine(today_utc, gpvga.timestamp)
        msgtimeUTC_t = (msgtimeUTC -
datetime.datetime(1970,1,1)).total_seconds()
        msgtype = '$GPVGA'
        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(gpvga.lat),
                  'lat_n' : str(gpvga.lat_dir),
                  'latitude' : float(gpvga.latitude),
                  'lon' : float(gpvga.lon),
                  'lon_n' : str(gpvga.lon_dir),
                  'longitude' : float(gpvga.longitude),
                  'elevation' : float(gpvga.altitude),
                  'elevation_unit' : str(gpvga.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 sentence
    streamReader = pynmea2.NMEAStreamReader()
    def datagramReceived(self, data, (host, port)):
        #A list of the incoming 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 GP GGA, continue
                    log(nmea_msg)
                    return nmea_msg

```

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

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