GPS Serial Library Documentation

Release 1.0

Brendan Doherty

Oct 18, 2019

Contents

1 Dependencies **3**

2 Installation **5**

3 What Is My Serial Device's Port Address? **7**

4 Table of Contents **9**
- 4.1 Simple test 9
- 4.2 gps_serial 10

5 Indices and tables **13**

Python Module Index **15**

Index **17**

Yet another GPS NMEA sentence parser, but this time with the threading module for expediting data parsing in background running threads. This was developed for & tested on the Raspberry PI.

CHAPTER 1

Dependencies

This library requires the py-serial library

CHAPTER 2

Installation

Currenty, there is no plan to deploy this single module library to pypi. but you can easily install this library using the following commands:

```
git clone https://github.com/DVC-Viking-Robotics/GPS_Serial.git
cd GPS_Serial
python3 setup.py install
```

The previous commands should automatically install the py-serial library. However, if you get import errors related to the `serial` module, make sure the py-serial library is install via:

```
pip3 install pyserial
```

Some cases may require the commands beginning with `python3` or `pip3` be prefixed with `sudo`.

CHAPTER 3

What Is My Serial Device's Port Address?

If you're going to use the GPIO pins, `RX` and `TX`, you must ensure that the `serial` interface is enabled by running:

```
sudo raspi-config
```

Important: make sure that the `serial console` feature is disabled. Otherwise, any data sent or received over these GPIO pins will be forwarded to a TTY console session if `serial console` feature is enabled (meaning this library will not be able to access the GPS module data).

It is worth noting that the port address for the GPIO serial pins is `/dev/ttyS0`. If you are using a USB connection, the address can be looked up using the py-serial's tools module:

```
python3 -m serial.tools.list_ports
```

You can then test which port in the outputted list is the GPS module by entering:

```
python3 -m serial.tools.miniterm /dev/ttyS0
```

where you replace the `/dev/ttyS0` part with the address you're testing. To exit the miniterm application use `ctrl + ]`

CHAPTER 4

Table of Contents

4.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/simple_test.py

```
"""
A simple test of the GPS-Serial library that automatically invokes the threading module.
"""
import time
from gps_serial import GPSserial

# you may want to adjust the port address that is passed to the constructor accordingly.
GPS = GPSserial('COM4')
while True:
    try:
        GPS.get_data() # pass `1` or `true` to print raw data from module
        if GPS.rx_status.startswith('Valid'):
            print('RxStatus:', GPS.rx_status, 'FixType:', GPS.fix)
            print('satelites\' quality:', GPS.sat_quality)
            print('satelites connected:', GPS.sat_connected)
            print('satelites in view:', GPS.sat_view)
            print('Course True North:', GPS.course_true, 'degrees')
            print('Course Magnetic North:', GPS.course_mag, 'degrees')
            print('speed in knots:', GPS.speed_knots, 'speed in kmph:', GPS.speed_kmph)
            print('Altitude:', GPS.altitude, 'meters')
            print('UTC: {}/{}/{} {}:{}:{}'.format(
                GPS.utc[1], GPS.utc[2], GPS.utc[0],
                GPS.utc[3], GPS.utc[4], GPS.utc[5]))
            print('lat:', GPS.lat, 'lng:', GPS.lng)
```

(continues on next page)

(continued from previous page)

```
            print('position dilution of precision:', GPS.pdop, 'meters')
            print('horizontal dilution of precision:', GPS.hdop, 'meters')
            print('vertical dilution of precision:', GPS.vdop, 'meters\n')
        else:
            print('Waiting for GPS fix')
        time.sleep(1)
    except KeyboardInterrupt:
        del GPS
        break
```

4.2 gps_serial

Yet another NMEA sentence parser for serial UART based GPS modules. This implements the threading module for [psuedo] asynchronous applications. CAUTION: The individual satelite info is being ignored until we decide to support capturing it from the GPS module's output.

gps_serial.**DEFAULT_LOC = {'lat': 37.96713657090229, 'lng': -122.0712176165581}**
The default/fallback location to use when waiting for a fix upon power-up of GPS device. This has been hard-coded to DVC Engineering buildings' courtyard.

class gps_serial.**GPSserial**(*address*, *timeout=1.0*, *baud=9600*)

Parameters

- **address** (*int*) – The serial port address that the GPS module is connected to. For example, on the raspberry pi's GPIO pins, this is `/dev/ttyS0`; on windows, this is something like `com#` where # is designated by windows.
- **timeout** (*int*) – Specific number of seconds till the threading `Serial`'s `~serial.Serial.read_until()` operation expires. Defaults to 1 second.
- **baud** (*int*) – The specific baudrate to be used for the serial connection. If left

lat
This attribute holds the latitude coordinate that was most recently parsed from the GPS module's data output.

lng
This attribute holds the longitude coordinate that was most recently parsed from the GPS module's data output.

utc
This attribute holds a tuple of time & date data that was most recently parsed from the GPS module's data output. This tuple conforms with python's time module functions.

speed_knots
This attribute holds the speed (in nautical knots) that was most recently parsed from the GPS module's data output.

speed_kmph
This attribute holds the speed (in kilometers per hour) that was most recently parsed from the GPS module's data output.

sat_connected
This attribute holds the number of connected GPS satelites that was most recently parsed from the GPS module's data output.

sat_view
This attribute holds the number of GPS satelites in the module's view that was most recently parsed from the GPS module's data output.

sat_quality
This attribute holds the description of the GPS satelites' quality that was most recently parsed from the GPS module's data output.

course_true
This attribute holds the course direction (in terms of "true north") that was most recently parsed from the GPS module's data output.

course_mag
This attribute holds the course direction (in terms of "magnetic north") that was most recently parsed from the GPS module's data output.

altitude
This attribute holds the GPS antenna's altitude that was most recently parsed from the GPS module's data output.

fix
This attribute holds the description of GPS module's fix quality that was most recently parsed from the GPS module's data output.

data_status
This attribute holds the GPS module's data authenticity that was most recently parsed from the GPS module's data output.

rx_status
This attribute holds the GPS module's receiving status that was most recently parsed from the GPS module's data output.

pdop
This attribute holds the GPS module's positional dilution of percision that was most recently parsed from the GPS module's data output.

vdop
This attribute holds the GPS module's vertical dilution of percision that was most recently parsed from the GPS module's data output.

hdop
This attribute holds the GPS module's horizontal dilution of percision that was most recently parsed from the GPS module's data output.

get_data(*raw=False*)
This function only starts the process of parsing the data from a GPS module (if any).

Parameters **raw** (*bool*) – `True` prints the raw data being parsed from the GPS module. `False` doesn't print the raw data. Defaults to `False`.

Returns the last latitude and longitude coordinates obtained from either object instantiation (*DEFAULT_LOC* values) or previously completed parsing of GPS data.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

gps_serial, 10

Index

A

altitude (*gps_serial.GPSserial attribute*), 11

C

course_mag (*gps_serial.GPSserial attribute*), 11
course_true (*gps_serial.GPSserial attribute*), 11

D

data_status (*gps_serial.GPSserial attribute*), 11
DEFAULT_LOC (*in module gps_serial*), 10

F

fix (*gps_serial.GPSserial attribute*), 11

G

get_data() (*gps_serial.GPSserial method*), 11
gps_serial (*module*), 10
GPSserial (*class in gps_serial*), 10

H

hdop (*gps_serial.GPSserial attribute*), 11

L

lat (*gps_serial.GPSserial attribute*), 10
lng (*gps_serial.GPSserial attribute*), 10

P

pdop (*gps_serial.GPSserial attribute*), 11

R

rx_status (*gps_serial.GPSserial attribute*), 11

S

sat_connected (*gps_serial.GPSserial attribute*), 10
sat_quality (*gps_serial.GPSserial attribute*), 11
sat_view (*gps_serial.GPSserial attribute*), 10
speed_kmph (*gps_serial.GPSserial attribute*), 10
speed_knots (*gps_serial.GPSserial attribute*), 10

U

utc (*gps_serial.GPSserial attribute*), 10

V

vdop (*gps_serial.GPSserial attribute*), 11