The Pythonic recipe to use a GPS module for your IoT projects

Sharad Shriram

sharad.sriram@gmail.com

April 5, 2016

Before we get started - a quik overview

We'll cover the following topics in this presentation

- What and how a GPS works?
- The 3 ways a GPS can start
- The data we get and how to interpret them

Interpreting the NMEA Data - For Reference

1. $GPBOD - Bearing, origin to destination
2. $GPBWC - Bearing and distance to waypoint, great circle
3. $GPGGA - Global Positioning System Fix Data
4. $GPGLL - Geographic position, latitude / longitude
5. $GPGSA - GPS DOP and active satellites
6. $GPGSV - GPS Satellites in view
7. $GPRMC - Recommended minimum specific GPS/Transit data
8. $GPR00 - List of waypoints in currently active route

1. $GPHDT - Heading, True
2. $GPRTE - Routes
3. $GPTRF - Transit Fix Data
4. $GPSTN - Multiple Data ID
5. $GPVBW - Dual Ground / Water Speed
6. $GPVTG - Track made good and ground speed
7. $GPWPL - Waypoint location
8. $GPXTE - Cross track

Ingredients needed

- A Laptop /SBC with Python installed
  DISCLAIMER : The demonstration done are in Python v.2.7
- A GPS module (Quectel L10 Development Kit)
- The necessary drivers depending on the operating system
- A few python libraries to make our jobs easy

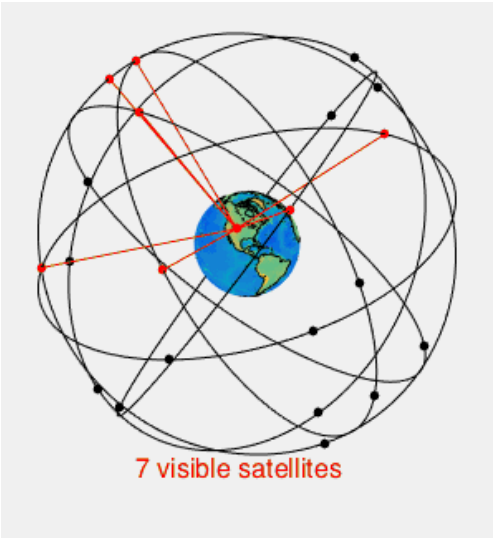
Figure: Quectel L10 GPS Development Kit - The hardware to be used today

GPS - Global Positioning System

- The ultimate watchdog - in the sense you cannot escape from the sights of GPS.
- In any weather or any place on the face of the Earth you are in .. **you are always being watched upon**.
- A GPS works based on **a constellation of 31 satellites** - that gives **information about time and position in any weather**.
- The GPS started out as a project for the use of US Armed Forces but was later released for civilian use.

How does a GPS work?

GPS determines the location of an object after getting pings from a minimum of 6 satellites - this is a cross verification method to give the accurate and precise position.



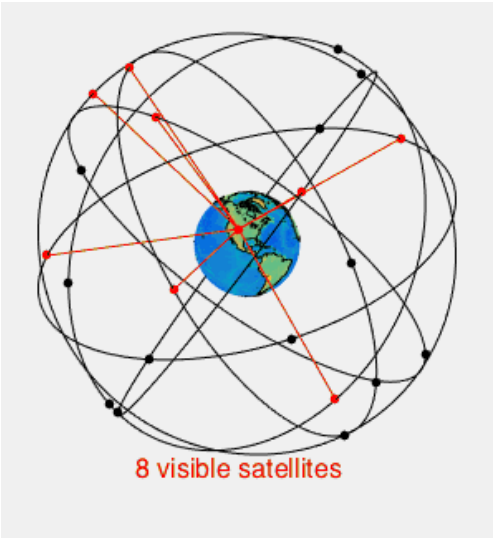


Figure: Here (in pic.) you see how the location is determined with 7 and 8 satellites.

Image courtesy : `https://upload.wikimedia.org/wikipedia/commons/2/27/GPS24goldenSML.gif`

The 3 Different Methods to Start a GPS Device

The **cold start** is when the GPS device dumps all the information, attempts to locate satellites and then calculates a GPS lock. This takes the longest because there is no known information.

The **warm start** is when the GPS device remembers its last calculated position, almanac used, and UTC Time, but not which satellites were in view. It then performs a reset and attempts to obtain the satellite signals and calculates a new position. This takes longer than a hot start but not as long as a cold start.

The **hot start** is when the GPS device remembers its last calculated position and the satellites in view, the almanac used and the UTC Time.This is the quickest GPS lock but it only works if you are generally in the same location as you were when the GPS was last turned off.

NMEA - The data representation standard of the GPS

- **NMEA 0183** is a combined electrical and data specification for communication between GPS receivers.
- It has been defined by, and is controlled by, the National Marine Electronics Association.
- The NMEA 0183 standard uses a simple ASCII, serial communications protocol that defines how data are transmitted in a "sentence" from one "talker" to multiple "listeners" at a time.
- It gives the data obtained from the sattelite, number of satellites,etc.

A sample stream of NMEA data obtained from the GPS hardware

$GPGGA,092750.000,5321.6802,N,00630.3372,W,1,8,1.03,61.7,M,55.2,M,,*76
$GPGSA,A,3,10,07,05,02,29,04,08,13,,,,,1.72,1.03,1.38*0A
$GPGSV,3,1,11,10,63,137,17,07,61,098,15,05,59,290,20,08,54,157,30*70
$GPGSV,3,2,11,02,39,223,19,13,28,070,17,26,23,252,,04,14,186,14*79
$GPGSV,3,3,11,29,09,301,24,16,09,020,,36,,,*76
$GPRMC,092750.000,A,5321.6802,N,00630.3372,W,0.02,31.66,280511,,,A*43

Interpreting the significance of each NMEA data field

$GPBOD - Bearing, origin to destination
$GPBWC - Bearing and distance to waypoint, great circle
$GPGGA - Global Positioning System Fix Data
$GPGLL - Geographic position, latitude / longitude
$GPGSA - GPS DOP and active satellites
$GPGSV - GPS Satellites in view
$GPHDT - Heading, True
$GPR00 - List of waypoints in currently active route
$GPRMA - Recommended minimum specific Loran-C data

Interpreting the significance of each NMEA data field (contd..)

$GPRMB - Recommended minimum navigation info
$GPRMC - Recommended minimum specific GPS/Transit data
$GPRTE - Routes
$GPTRF - Transit Fix Data
$GPSTN - Multiple Data ID
$GPVBW - Dual Ground / Water Speed
$GPVTG - Track made good and ground speed
$GPWPL - Waypoint location
$GPXTE - Cross-track error, Measured
$GPZDA - Date and Time

The Python recipe to use a GPS module

Before getting started with the code, youll need to install this helpful Python module called **PyNMEA**

$pip install pynmea

Now, we are all set to write our Python program.

Python code - Imports & Initializations

```
import time
import serial
import string
from pynmea import nmea #handles the GPS data

#settting up the serial connection with the necessary
ser = serial.Serial()
ser.port = "/dev/ttyACM0"
ser.baudrate = 9600
ser.timeout = 1

ser.open() #open serial port

#object to read the "\$GPGGA" string from the GPS
gpgga = nmea.GPGGA()
```

Python code - Setting up the code

```
#Run this code till its Judgement Day
while True:
        #Store the results into a txt file
        #Replace the txt by a csv file
        with open("GPS-dataset.txt","a") as f:
                #read a line of the incoming stream of
            data = ser.readline()
```

Python code - Read and Format latitude values

```
#check if it is starting with GPGGA
if data[0:6] == '$GPGGA':
#method to extract GPS info
gpgga.parse(data)
lats = gpgga.latitude
lat_dir = gpgga.lat_direction
#offseting the GPS data to correct the error
#1101.0884N is fixed and is corrected as 11.010884N
#latitude = float(lats) / 100

print "Latitude: "+ str(latitude)+ str(lat_dir)
```

Python code - Read longitude and add to output file

```
longs = gpgga.longitude
long_dir = gpgga.lon_direction
#offseting the GPS data to correct the error
#07658.5456E is fixed and is corrected as 76.585456E
# longitude = float(longs) / 100
print "Longitude: "+ str(longitude)+ str(long_dir)
#store the location
gps_value = str(latitude)+str(lat_dir)+\
 " "+str(longitude)+str(long_dir)+ "\n"
f.write(gps_value)
```

Note: Please provide suitable indentation to the code

-

Source code : `https://gist.github.com/ssharad/db28c4ba31c23c4a5fa971cc8f775cb7`

Thank You!