GPS Module Interfacing with Raspberry Pi

Hardware Guide:

For completing this lesson, you will require the following things along with your initial raspberry pi setup

1. GPS module
2. USB to TTL converter
3. Connecting wires

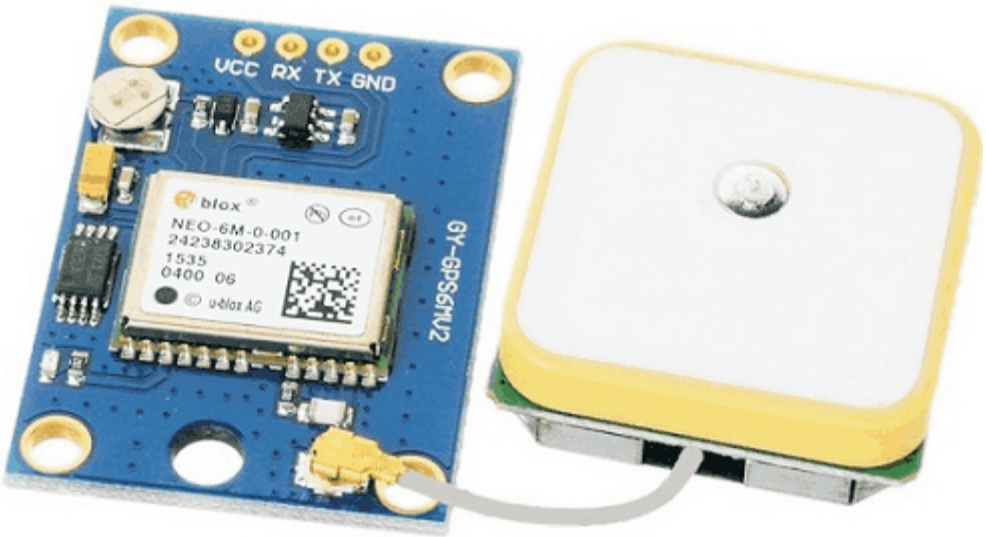
GPS Module:

Global Positioning System (GPS) makes use of signals sent by satellites in space and ground stations on Earth to accurately determine their position on Earth.

Radio Frequency signals sent from satellites and ground stations are received by the GPS. GPS makes use of these signals to determine its exact position.

The signals received from the satellites and ground stations contain time stamps of the time when the signals were transmitted.

Using information from 3 or more satellites, the exact position of the GPS can be triangulated.



GPS receiver module gives output in standard (National Marine Electronics Association) NMEA string format. It provides output serially on Tx pin with default 9600 Baud rate.

This NMEA string output from GPS receiver contains different parameters separated by commas like longitude, latitude, altitude, time etc. Each string starts with '$' and ends with carriage return/line feed sequence.

E.g.

$GPGGA,184237.000,1829.9639,N,07347.6174,E,1,05,2.1,607.1,M,-64.7,M,,0000*7D

$GPGSA,A,3,15,25,18,26,12,,,,,,,5.3,2.1,4.8*36

$GPGSV,3,1,11,15,47,133,46,25,44,226,45,18,37,238,45,26,34,087,40*72

$GPGSV,3,2,11,12,27,184,45,24,02,164,26,29,58,349,,05,26,034,*7F

$GPGSV,3,3,11,21,25,303,,02,11,071,,22,01,228,*40

$GPRMC,184237.000,A,1829.9639,N,07347.6174,E,0.05,180.19,230514,,,A*64

Wiring up your Circuit:

1. Connect the VCC Pin of GPS Module to 3.3V Pin of USB to TTL converter
2. Connect the GND Pin of GPS Module to GND Pin of USB to TTL converter
3. Connect the Tx Pin of GPS Module to Rx Pin of USB to TTL converter
4. Connect the Rx Pin of GPS Module to Tx Pin of USB to TTL converter.
5. Lastly connect the USB to TTL converter to USB port of Raspberry Pi.

Software Guide:

Open Terminal Window and type the following command to know to which USB port the GPS module is attached: ls /dev/ttyUSB*

We can find whether our GPS module is working properly and the connections are correct by typing the following command: sudo cat /dev/ttyUSB*

(Here replace * with the port number to which GPS module is attached. You should be seeing a lot of text pass by. That means it works. Type Ctrl + c to return.)

Use 'gpsd':

You can always just read that raw data, but its much nicer if you can hav e some Linux software prettify it. We'll try out gpsd which is a GPS-handling Daemon (background-helper)

Installing a GPS Daemon (gpsd)

The first step is installing some software on your Raspberry Pi that understands the serial data that your GPS module is providing via /dev/ttyUSB0.

Thankfully other people have already done all the hard work for you of properly parsing the raw GPS data, and we can use (amongst other options) a nice little package named 'gpsd', which essentially acts as a layer between your applications and the actual GPS hardware, gracefully handling parsing errors, and providing a common, well-defined interfaces to any GPS module.

To install gpsd, make sure your Pi has an Internet connection and run the following commands from the console:

1. sudo apt-get update
2. sudo apt-get install gpsd gpsd-clients python-gps

And install the software as it prompts you to do.

Raspbian Jessie systemd service fix:

Note if you're using the Raspbian Jessie or later release you'll need to disable a systemd service that gpsd installs. This service has systemd listen on a local socket and run gpsd when clients connect to it, however it will also interfere with other gpsd instances that are manually run (like in this guide). You will need to disable the gpsd system d service by running the following commands:

1. sudo systemctl stop gpsd.socket
2. sudo systemctl disable gpsd.socket

Should you ever want to enable the default gpsd systemd service you can run these commands to restore it (but remember the rest of the steps in this guide won't work!):

1. sudo systemctl enable gpsd.socket
2. sudo systemctl start gpsd.socket

Try out 'gpsd'

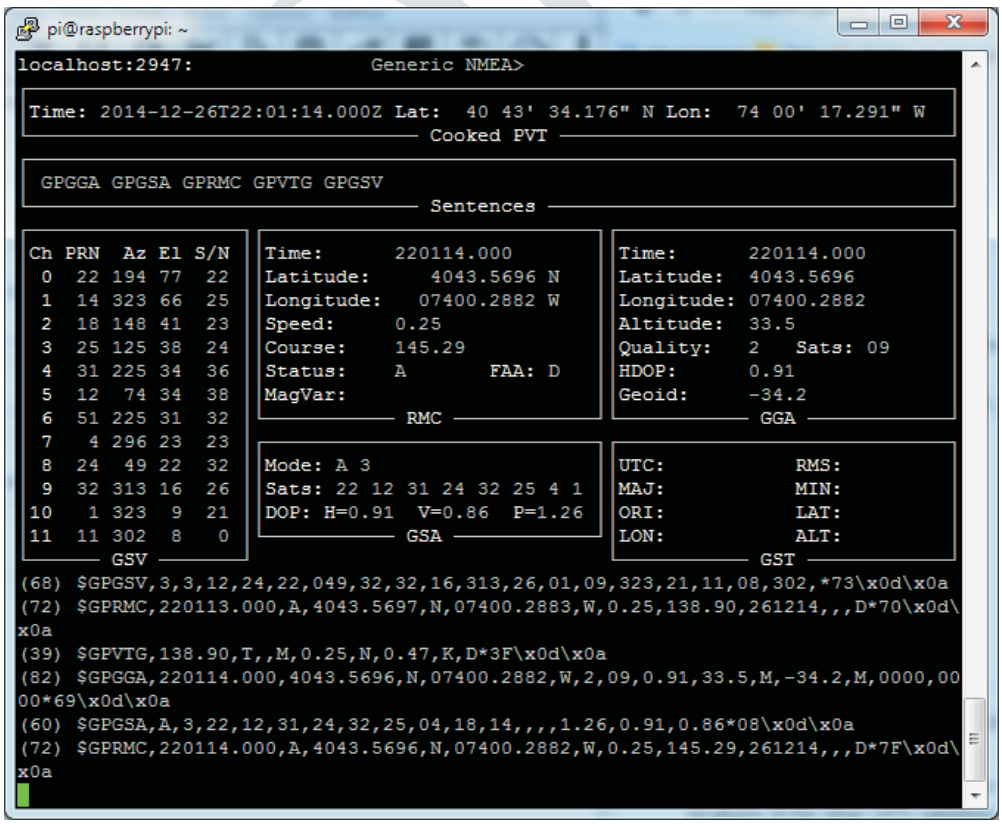
After installing gpsd and disabling the gpsd systemd service as mentioned above you're ready to start using gpsd yourself.

Start gpsd and direct it to use USB. Simply entering the following command (Here we are assuming that GPS module is connected to USB0):

1. sudo gpsd /dev/ttUSB0 -F /var/run/gpsd.sock

... which will point the gps daemon to our GPS device on the /dev/ttyAMA0 console

Try running **gpsmon** to get a live-streaming update of GPS data!



or cgps which gives a less detailed, but still quite nice output

1. cgps -s

```
Time:       2013-01-24T08:56:30.000Z   PRN:  Elev:  Azim:  SNR:  Used:
Latitude:                              11    80    287    37     Y
Longitude:                             1     59    288    26     Y
Altitude:   215.6 ft                   32    53    207    29     Y
Speed:      0.0 mph                    19    52    153    24     Y
Heading:    127.3 deg (true)           14    34    076    45     Y
Climb:      0.0 ft/min                 39    29    150    30     Y
Status:     3D FIX (7 secs)
```

You can abort gpsd by the following command

1. sudo killall gpsd

Using Python

Let's extract Latitude, Longitude and time information from NME! GPGG! string received from GPS module using Python. And print them on console (terminal). By using these latitude and longitude, locate the current position on Google Map.

Code:

```
'''
GPS Interfacing with Raspberry Pi using Pyhton
http://www.electronicwings.com
'''
import serial          #import serial pacakge
from time import sleep
import webbrowser      #import package for opening link in browser
import sys             #import system package

def GPS_Info():
    global NMEA_buff
    global lat_in_degrees
    global long_in_degrees
    nmea_time = []
    nmea_latitude = []
    nmea_longitude = []
    nmea_time = NMEA_buff[0]          #extract time from GPGGA string
    nmea_latitude = NMEA_buff[1]      #extract latitude from GPGGA string
    nmea_longitude = NMEA_buff[3]     #extract longitude from GPGGA string

    print("NMEA Time: ", nmea_time,'\n')
```

```
    print ("NMEA Latitude:", nmea_latitude,"NMEA Longitude:", nmea_longitude,'\n')

    lat = float(nmea_latitude)              #convert string into float for calculation
    longi = float(nmea_longitude)           #convertr string into float for calculation
    lat_in_degrees = convert_to_degrees(lat)    #get latitude in degree decimal format
    long_in_degrees = convert_to_degrees(longi) #get longitude in degree decimal format

#convert raw NMEA string into degree decimal form at
def convert_to_degrees(raw_value):
    decimal_value = raw_value/100.00
    degrees = int(decimal_value)
    mm_mmmm = (decimal_value - int(decimal_value))/0.6
    position = degrees + mm_mmmm
    position = "%.4f" %(position)
    return position



gpgga_info = "$GPGGA,"
ser = serial.Serial ("/dev/ttyUSB0")            #Open port with baud rate
GPGGA_buffer = 0
NMEA_buff = 0
lat_in_degrees = 0
long_in_degrees = 0

try:
    while True:
        received_data = (str)(ser.readline ())              #read NMEA string received
        GPGGA_data_available = received_data.find(gpgga_info)   #check for NMEA GPGGA string
        if (GPGGA_data_available>0):
            GPGGA_buffer = received_data.split("$GPGGA,",1)[1]  #store data coming after "$GPGGA,"
            NMEA_buff = (GPGGA_buffer.split(','))           #store comma separated data in buffer
            GPS_Info()                                  #get time, latitude, longitude

            print("lat in degrees:", lat_in_degrees," long in degree: ", long_in_degrees, '\n')
            map_link = 'http://maps.google.com/?q=' + lat_in_degrees + ',' + long_in_degrees
#create link to plot location on Google map
            print("<<<<<<<<press ctrl+c to plot location on google maps>>>>>>\n")
#press ctrl+c to plot on map and exit
            print("------------------------------------------------------------\n")

except KeyboardInterrupt:
    webbrowser.open(ma p_link)      #open current position information in google map
    sys.exit(0)

#end of file
```

You can download or copy the code from the following link:
http://www.electronicwings.com/raspberry-pi/gps-module-interfacing-with-raspberry-pi

The output of python code is as follows:

```
Python 3.4.2 Shell
File  Edit  Shell  Debug  Options  Windows  Help
NMEA Latitude: 1832.9724 NMEA Longitude: 07347.4944
lat in degrees: 18.5495  long in degree:  73.7916
<<<<<<<<press ctrl+c to plot location on google maps>>>>>>
------------------------------------------------------------
NMEA Time:  141750.000
NMEA Latitude: 1832.9724 NMEA Longitude: 07347.4945
lat in degrees: 18.5495  long in degree:  73.7916
<<<<<<<<press ctrl+c to plot location on google maps>>>>>>
------------------------------------------------------------
NMEA Time:  141751.000
NMEA Latitude: 1832.9724 NMEA Longitude: 07347.4943
lat in degrees: 18.5495  long in degree:  73.7916
<<<<<<<<press ctrl+c to plot location on google maps>>>>>>
------------------------------------------------------------
NMEA Time:  141752.000
NMEA Latitude: 1832.9724 NMEA Longitude: 07347.4942
lat in degrees: 18.5495  long in degree:  73.7916
<<<<<<<<press ctrl+c to plot location on google maps>>>>>>
------------------------------------------------------------
>>>
>>>
                                                    Ln: 107 Col: 4
```

Note: Please ensure that GPS module is visible to open sky or else it will not be able to produce desired output.