

Interfacing GPS module with Raspberry Pi

Installation Manual

- **GPS** stands for Global Positioning System and used to detect the Latitude and Longitude of any location on the Earth, with exact UTC time (Universal Time Coordinated).
- GPS module is the main component in our vehicle tracking system.
- This device receives the coordinates from the satellite for each and every second, with time and date.
- **GPS module** sends the data related to tracking position in real time, and it sends so many data in NMEA format.
- NMEA format consists of sentence starts from **\$GPGGA**, the coordinates, time and other useful information.
- **GPGGA** is referred to **Global Positioning System Fix Data**.
- GPGGA string contains following co-ordinates separated by commas.

Sr No	Identifier	Description	Sr No	Identifier	Description
1	\$GPGGA	Global Positioning system fix data	7	FQ	Fix Quality Data
2	HHMMSS.SSS	Time in hour minute seconds and milliseconds format.	8	NOS	No. of Satellites being Used
3	Latitude	Latitude (Coordinate)	9	HPD	Horizontal Dilution of Precision
4	N	Direction N=North, S=South	10	Altitude	Altitude from sea level
5	Longitude	Longitude(Coordinate)	11	M	Meter
6	E	Direction E= East, W=West	12	Height	Height
			13	Checksum	Checksum Data

<u>Hardware Requirements</u>	<u>Software Requirements</u>
<ul style="list-style-type: none">• Raspberry Pi Model B/B+ , SD Card• Ethernet Cable / Wi-Fi• Power Supply to Pi• Neo 6m v2 GPS Module• Jumper wires• Breadboard• 2x16 LCD Display• Potentiometer (to adjust Contrast)	<ul style="list-style-type: none">• Raspbian Stretch OS• Adafruit Python Char LCD Library

Neo 6m v2 GPS Module



- This board features the u-blox NEO-6M GPS module with antenna and built-in EEPROM. This is compatible with various flight controller boards designed to work with a GPS module.
- EEPROM is used for saving the configuration data when powered off.
- Power Supply Range: 3 V to 5 V
- Default Baud Rate: 9600 bps

Connect GPS Module to RPi

Connection is very simple. Requires 4 Female to Female Jumper Wires

Neo 6m V2 GPS Board Pin	Details	Raspberry Pi Physical Pin	Raspberry Pi Function
VCC	Power	Pin 1	3.3V Power
GND	Common Ground	Pin 39	GND
TXD	Data Output	Pin 10	(UART_RXD0) GPIO15
RXD	Data Input	Pin 8	(UART_TXD0) GPIO14

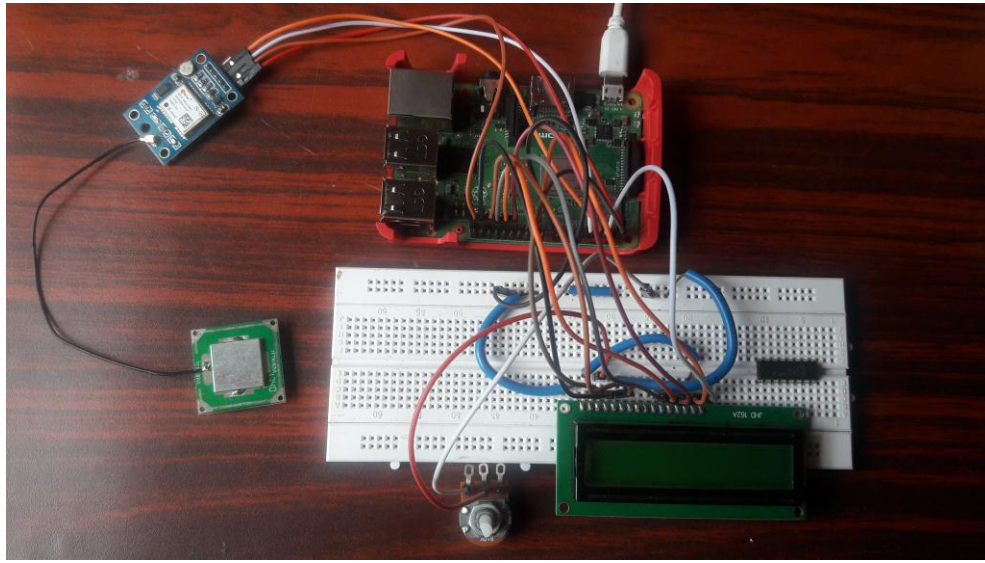
Connect 2x16 LCD Display to RPi

A 16x2 LCD is used for displaying all messages. A 10k Potentiometer is also used with LCD for controlling the contrast. I used following GPIO pins to connect LCD to Raspberry Pi's GPIO Pins:

Sr No	LCD Display Board Pin	RPI Physical Pin	Raspberry Function
4	RS	37	GPIO26
6	E	35	GPIO19
11	D4	33	GPIO13
12	D5	31	GPIO6
13	D6	29	GPIO5
14	D7	23	GPIO11

Connect GPS module to Raspberry Pi's GPIO Pins by using Female to Female Jumper wires

Connect 2x16 LCD Display to Raspberry Pi's GPIO Pins with the help of breadboard and jumper wires.



Step 1: Update Raspberry Pi

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo apt-get update  
  
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo apt-get upgrade
```

Step 2: edit the /boot/config.txt file

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo nano /boot/config.txt
```

At the bottom of this file, add the above lines

```
dtoverlay=pi3-disable-bt  
core_freq=250  
enable_uart=1  
force_turbo=1
```

The ***dtoverlay=pi3-disable-bt*** disconnects the bluetooth from the *ttyAMA0*, this is to allow us access to use the full UART power available via *ttyAMA0* instead of the mini UART *ttyS0*.

Save with Ctrl+X, yes and press Enter.

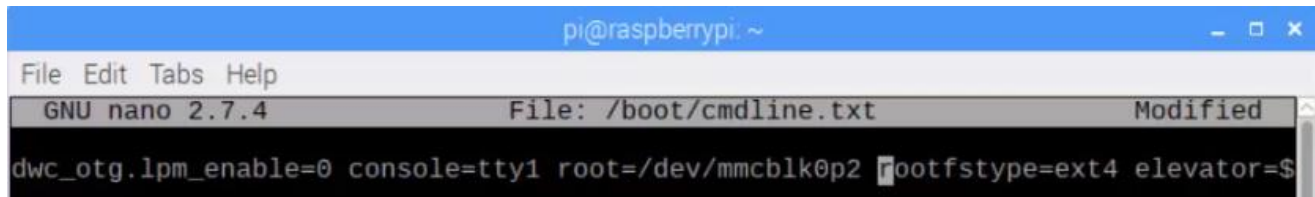
Step 3: Before proceeding, make a copy of cmdline.txt file

```
pi@raspberrypi:~ $ sudo cp /boot/cmdline.txt /boot/cmdline.txt.backup
```

Now, edit file cmdline.txt

```
pi@raspberrypi:~ $ sudo nano /boot/cmdline.txt
```

Remove console=serial0,115200 **and Modify** root=/dev/mmcblk0p2

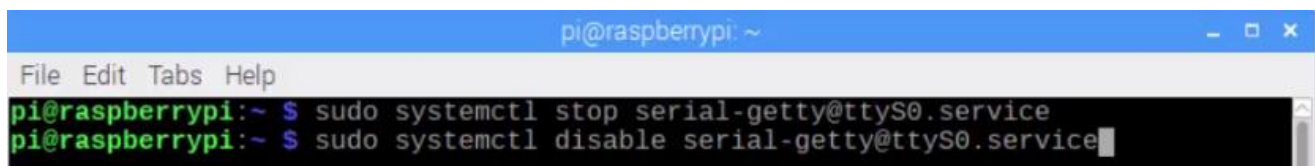


```
pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 2.7.4 File: /boot/cmdline.txt Modified
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline
```

Save with Ctrl+X, yes and press Enter.

Step 4: Reboot Raspberry Pi using the command *sudo reboot*

Step 5: Stop and disable the Pi's serial ttyS0 service



```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo systemctl stop serial-getty@ttyS0.service
pi@raspberrypi:~ $ sudo systemctl disable serial-getty@ttyS0.service
```

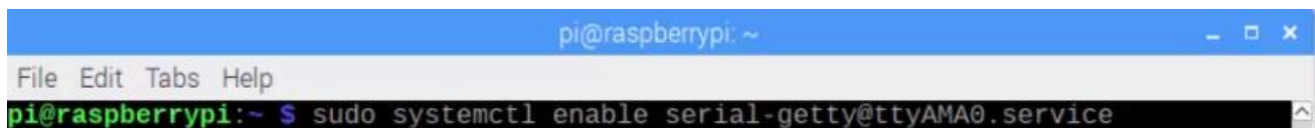
The following commands can be used to enable it again if needed

```
sudo systemctl enable serial-getty@ttyS0.service
```

```
sudo systemctl start serial-getty@ttyS0.service
```

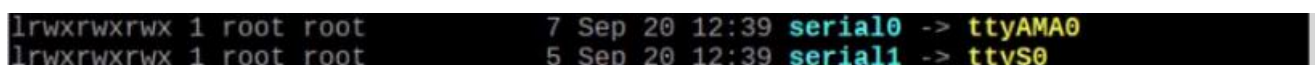
Step 6: Reboot Raspberry Pi using the command *sudo reboot*

Step 7: Now, Enable the ttyAMA0 service



```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo systemctl enable serial-getty@ttyAMA0.service
```

Verify it using *ls -l /dev* command



```
lrwxrwxrwx 1 root root 7 Sep 20 12:39 serial0 -> ttyAMA0
lrwxrwxrwx 1 root root 5 Sep 20 12:39 serial1 -> ttyS0
```

Step 8: Install minicom and pynmea2

Install minicom package which is used to connect to the GPS module and make sense of the data.

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo apt-get install minicom
```

Install pynmea2 library which is used to parse the received data.

```
pi@raspberrypi:~ $ sudo pip install pynmea2
```

Step 9: Use minicom command to test our GPS module is working fine.

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo minicom -D /dev/ttyAMA0 -b 9600
```

9600 represents the baud rate at which the GPS module communicates.

Here, we can see NMEA sentences. **NMEA format** consist several sentences, in which we only need one sentence. This sentence starts from **\$GPGGA** and contains the coordinates

```
$GPTXT,01,01,01,NMEA unknown msg*58  
$GPTXT,01,01,01,NMEA unknown msg*58  
$GPRMC,101752.00,A,1731.35655,N,07332.09973,E,0.503,,190918,,A*7B  
$GPVTG,,T,,M,0.503,N,0.932,K,A*2D  
$GPGGA,101752.00,1731.35655,N,07332.09973,E,1,07,1.04,2.0,M,-70.7,M,,*74  
$GPGSA,A,3,2120,10,2
```

Sometimes we received sentences which contains **unknown msg*58**, but this is not an error, actually it may takes some time to track your GPS module. (Even for the first time more than 20-30 minutes.)

I suggest keep your GPS module's antenna in open space (e.g. near the window)

To exit from above window, Press Ctrl+A, and Press x and Enter Key.

Step 10: The above same test can also be done using cat command

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo cat /dev/ttyAMA0  
  
$GPRMC,102259.00,A,1731.35273,N,07332.09827,E,0.245,,190918,,A*73  
$GPVTG,,T,,M,0.245,N,0.455,K,A*24  
$GPGGA,102259.00,1731.35273,N,07332.09827,E,1,08,0.99,1.8,M,-70.7,M,,*78  
$GPTXT,01,01,01,NMEA unknown msg*58
```

This sentence gives you Latitude found after two commas and Longitude found after four commas.

Step 11: Write Python script to display your current **Latitude and Longitude on LCD Display**.

Python Script (gpsdemo.py)

```
import time
import serial
import string
import pynmea2
import RPi.GPIO as gpio
import Adafruit_CharLCD as LCD

gpio.setmode(gpio.BCM)

lcd = LCD.Adafruit_CharLCD(rs=26, en=19,
                           d4=13, d5=6, d6=5, d7=11,
                           cols=16, lines=2)

lcd.message("MSD Gurukul\n Welcomes You")
time.sleep(2)
lcd.clear()
lcd.message("GPS Demo")
time.sleep(2)
lcd.clear()

port = "/dev/ttyAMA0" # the serial port to which the pi is connected.

#create a serial object
ser = serial.Serial(port, baudrate = 9600, timeout = 0.5)

try:
    while 1:
        try:
            data = ser.readline()
        except:
            print("loading")
        #wait for the serial port to churn out data

        if data[0:6] == '$GPGGA': # the long and lat data are always contained in the GPGGA string of
            the NMEA data

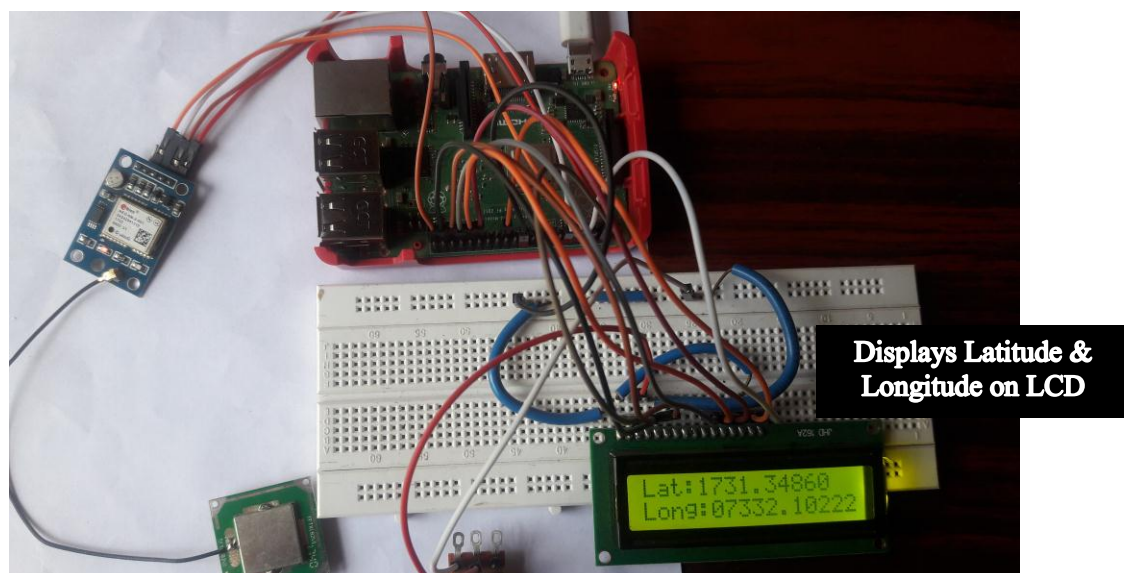
            msg = pynmea2.parse(data)
            latval = msg.lat #parse the latitude and print
            concatlat = "Lat:" + str(latval)
            print(concatlat)
            lcd.set_cursor(0,0)
            lcd.message(concatlat)
```



```
#parse the longitude and print
longval = msg.lon
concatlong = "Long:" + str(longval)
print(concatlong)
lcd.set_cursor(0,1)
lcd.message(concatlong)

time.sleep(0.5)#wait a little before picking the next data.
except KeyboardInterrupt:
    lcd.clear()
    lcd.message("Thank You")
    time.sleep(2)
```

Run Python script on terminal using sudo command (sudo gpsdemo.py)



That's all!!!

Thank you....