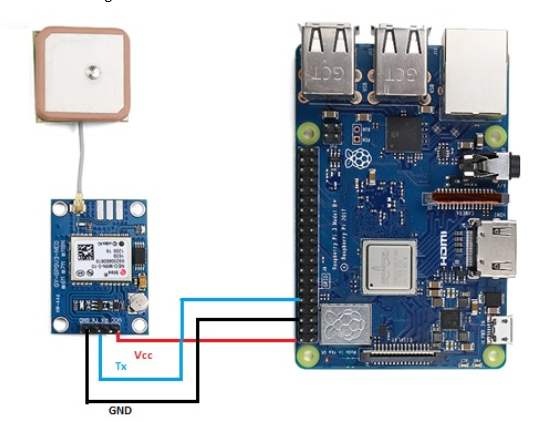
GPS



##### ****Step 1: Setting up the UART in Raspberry Pi****

sudo nano /boot/config.txt

At the bottom of the file, add the below lines,

dtparam=spi=on

dtoverlay=pi3-disable-bt

core\_freq=250

enable\_uart=1

force\_turbo=1

Now, Press ctrl+x to exit and press y and enter to save.

Raspbian uses the UART as a serial console and so we need to turn off that functionality. To do so we need to change the /boot/cmdline.txt file. For safety before editing the file make a backup of that using the following command,

sudo cp boot/cmdline.txt boot/cmdline\_backup.txt

sudo nano /boot.cmdline.txt

Replace the content with the below line,

dwc\_otg.lpm\_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-consoles

Press ctrl+x to exit and press y and enter to save.

Now reboot pi to see the changes,

sudo reboot

Now, we will check how our GPS module is working.

Before checking this out make sure that the yellow LED in the Neo 8M is blinking. Basically, the blinking of yellow LED means that the GPS module receiving the data perfectly. When the Yellow LED is blinking, run the following command. There are two serial ports in Raspberry pi 3: serial0 and serial1. Here, we have used serial0 because it will point to GPIO pins 14 and 15. Now to see the connection of port with serial0 use the below command,

ls -l /dev

##### ****Step 2: Disabling the Raspberry Pi Serial Getty Service****

After that, there are two possible outputs,

a. If in your output, Serial0 is linked with ttyAMA0, then to disable it use the below command,

sudo systemctl stop serial-getty@ttyAMA0.service

sudo systemctl disable serial-getty@ttyAMA0.service

b. If in your output Serial0 is linked with ttys0, then to disable it use the below command,

sudo systemctl stop serial-getty@ttys0.service

sudo systemctl disable serial-getty@ttys0.service

Now, again reboot the system using sudo reboot.

**Step 3: Activating ttys0**

In our system, we have disabled the ttyAMA0, the next thing is for us to enable the ttyso.

sudo systemctl enable serial-getty@ttys0.service

**Step 4: Install Minicom and pynmea2**

Use minicom python library to connect with the GPS module and make sense of the data.

sudo apt-get install minicom

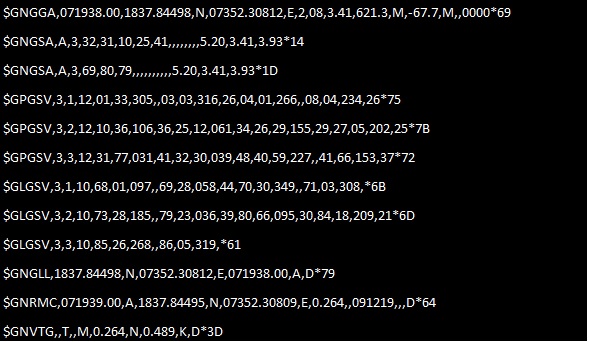
Use pynmea2 python library to parse the received NMEA data.

sudo pip install pynmea2

**Step 5: Testing output**

To test the GPS run the below command. The below data shows the data communication between the GPS Module and Raspberry pi microcontroller.

sudo cat /dev/ttyAMA0

[](https://robu.in/wp-content/uploads/2019/12/data-communication.jpg)

Now, finally, we will write the python code for interfacing of the GPS module with Raspberry pi.

import serial

import time

import string import pynmea2

while True:      port=“/dev/ttyAMAO”

ser=serial.Serial(port, baudrate=9600, timeout=0.5)

dataout =pynmea2.NMEAStreamReader()

newdata=ser.readline()

if newdata[0:6] == “$GPRMC”:

newmsg=pynmea2.parse(newdata)

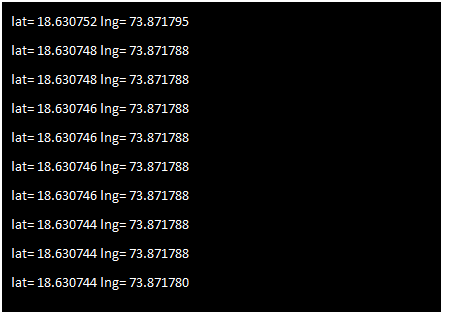
lat=newmsg.latitude

lng=newmsg.longitude

gps = “Latitude=” + str(lat) + “and Longitude=” +str(lng)

print(gps)

And here is the final output. it provides the data of your exact position in terms of Latitude and Longitude.

[](https://robu.in/wp-content/uploads/2019/12/final-output.png)