GPS HAT for Raspberry Pi

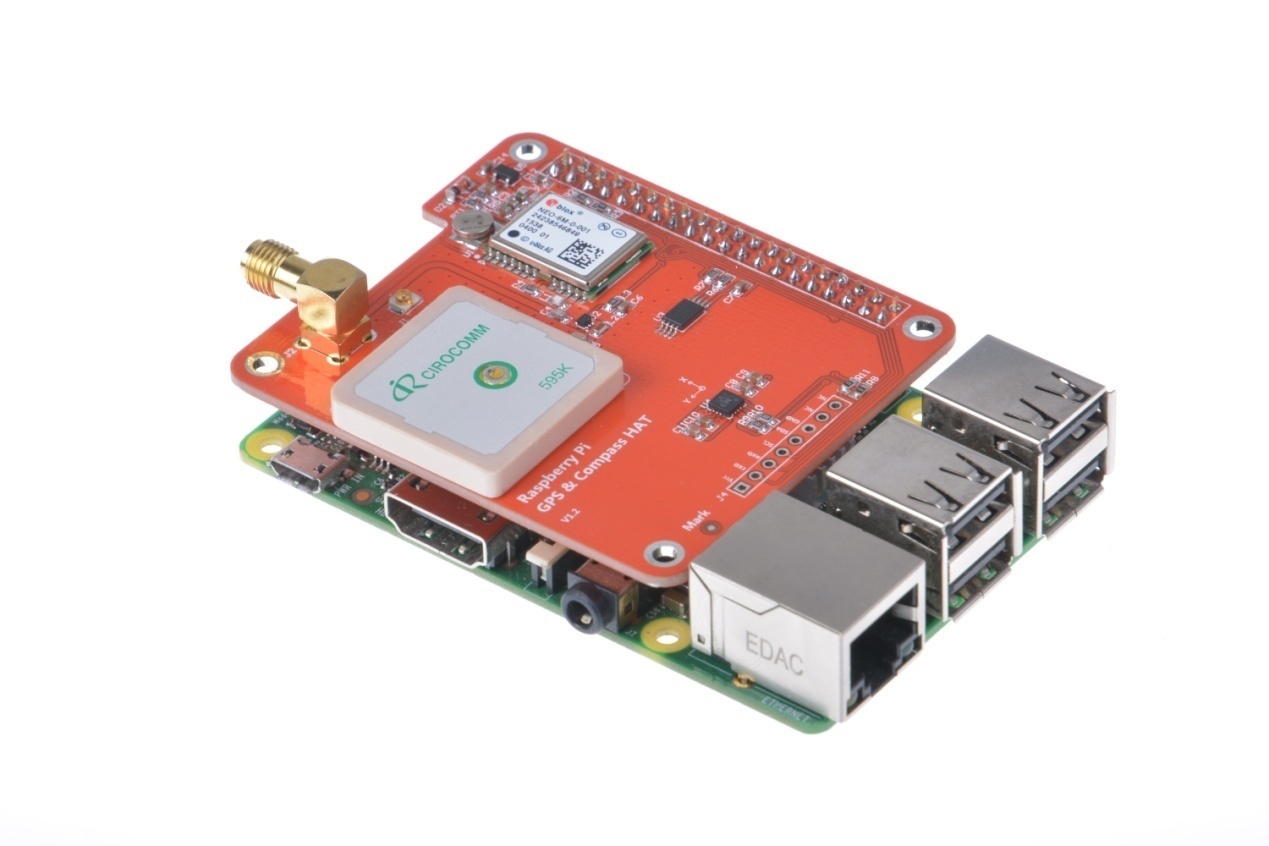


Table of Contents

[1.Overview 3](#_Toc440402000)

[2.Specifications 4](#_Toc440402001)

[3.Features 5](#_Toc440402002)

[4.Electrical & Sensitivity 5](#_Toc440402003)

[5.Pinouts 7](#_Toc440402004)

[5.1 Serial Console Pins 7](#_Toc440402005)

[5.2 PPS Pin 8](#_Toc440402006)

[5.3 Compass 8](#_Toc440402007)

[6.External Antenna 9](#_Toc440402008)

[7.Pinmap 11](#_Toc440402009)

[7.1 Raspberry PI interfaces 11](#_Toc440402010)

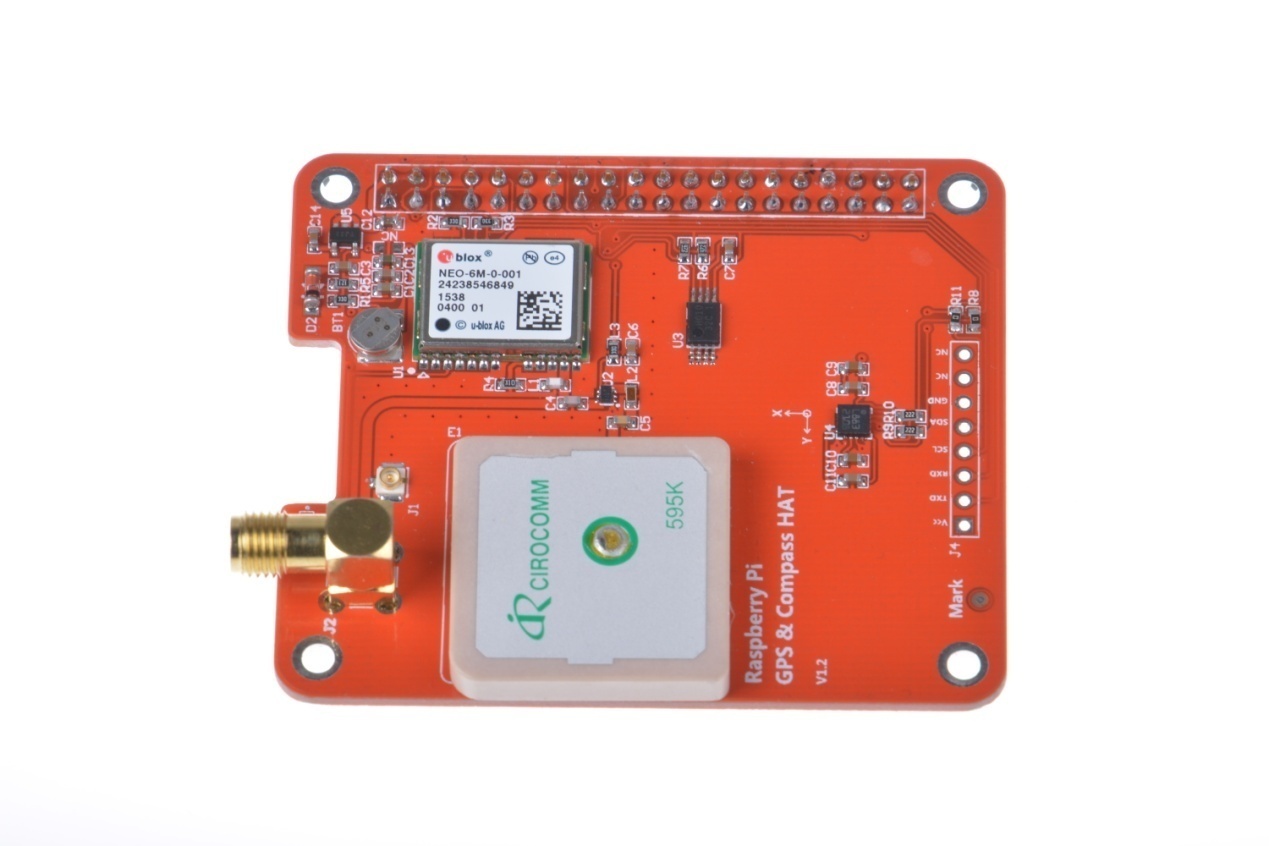
[7.2 GPS HAT interfce 12](#_Toc440402011)

[7.3 Pinmap 12](#_Toc440402012)

[8.Schematic 14](#_Toc440402013)

[9.User's manual 15](#_Toc440402014)

# 1.Overview



The GPS expansion board is designed for Raspberry Pi+ Models. Version 1 and version 2 are both supported. But it is NOT compatible with the original Raspberry Pi A and B boards. It provides GPS information via the serial ports to the Raspberry Pi. When it is connected to GPIO5,NTP PPS is available.

This GPS HAT is customized for Raspberry Pi interface based on NEO-6 GPS module. Through the serial port on Raspberry Pi, data returned from NEO-6M module can be received. In this way, current location and time can be obtained.

This GPS HAT is a multi-constellation smart GNSS module with an embedded 25 x 25 x 4 mm antenna. Universal Ublox NEO-6M module comes with a ceramic destined antenna, data backup battery, EEPROM storage , compass.

# 2.Specifications

|  |  |
| --- | --- |
| PCB size | 65mm X 55mm X 1.6mm |
| Input voltage | 5V |
| Interface | UART for GPS , IIC for compass |
| Baud rate | 9600(default),19200,38400,115200 |

•Multi-constellation GNSS module with integrated high performance 25 x 25\*4 mm antenna

• Compliant with GPS, SBAS.

• Plug & play use with embedded connector

•Low power processing core delivers current optimized multi-constellation tracking

• Ultra-sensitive -161 dBm (tracking) RF front-end

• Supports ephemeris file injection (A-GPS)

• Satellite Based Augmentation System (SBAS) compliant

# 3.Features

• 5-pin connector

• Embedded 25 x 25 x4 mm GPS antenna

• Frequency Bands: GPS L1.

• Standards: NMEA

• Jamming rejection

• Data logging

• A-GPS: ephemeris file injection

• EGNOS, WAAS, GAGAN and MSAS

It provides correction of positional errors due to ionospheric and orbital disturbances

# 4.Electrical & Sensitivity

• Current consumption

- OFF ‹ 7 uA

- Full power tracking: ‹ 37 mA (GPS)

- Full power acquisition: ‹ 47 mA (GPS)

• Sensitivity

- Acquisition: -147 dBm

- Navigation: -161 dBm

- Tracking: -161 dBm

• Power supply

- Range from 3 up to 5.5 V

• Positional accuracy (CEP50):

Autonomous Positional Error ‹ 3.5 m

• Accuracy

- Speed: ‹ 0.01 m/s

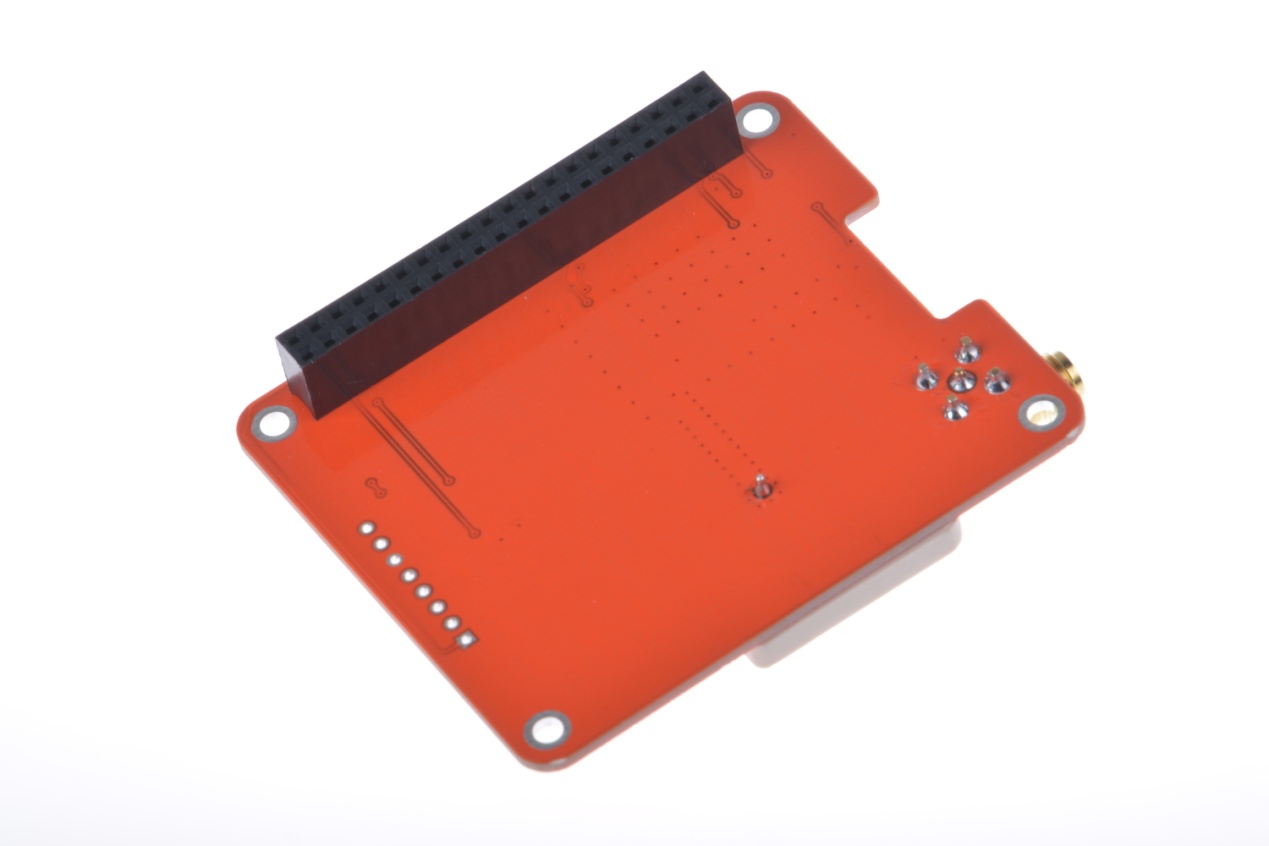
- Heading: ‹ 0.5 deg

• Time to first fix (90% @ -130 dBm)

- Hot start: 1 s

- Cold start: ‹ 27 s

# 5.Pinouts



2x20 way header is supplied and 2 suitable standoffs providing a very robust solution. It is compatible with Raspberry Pi's HAT physical layout with camera and display port notches. Additional HAT with a suitable header can be mounted on top of it.

It takes over the Pi's hardware UART to send/receive data to and from the GPS module. When the RX/TX pins are used with a console cable, it cannot be used.

## 5.1 Serial Console Pins

The Raspberry Pi has only one serial port, if serial is needed to chat with a GPS, it takes over the RXD and TXD pins**.**

## 5.2 PPS Pin

GPS's can output a 'pulse per second' for synchronizing the time. This challenge is resolved with a closed jumper that connects it to GPIO5.

## 5.3 Compass

The HMC5883L is a 3 axis digital compass which can communicate via I2C to the Raspberry Pi using only 2 data lines.

# 6. External Antenna



All Ultimate GPS modules have a built in patch antenna - this antenna provides -162 dBm(NEO-6M) sensitivity and is perfect for many projects. However, if you want to place your project in a box, it might not be possible to have the antenna pointing up, or it might be in a metal shield, or you may need more sensitivity. In these cases, an external active antenna should be employed.

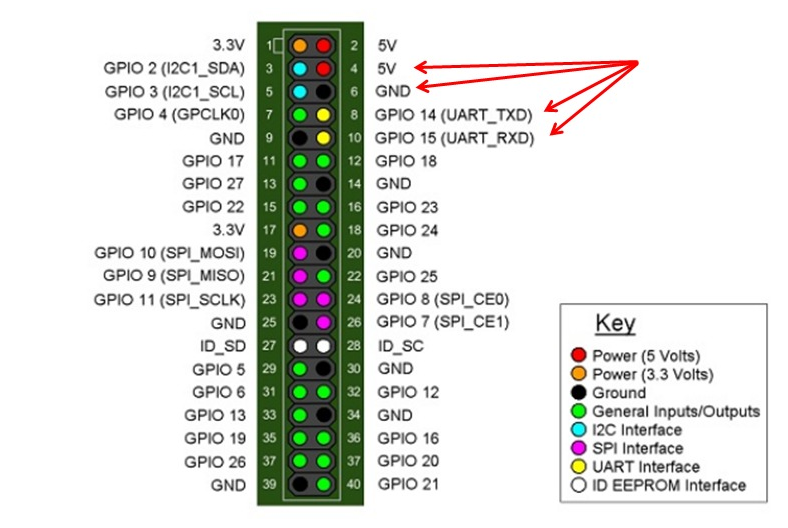
GPS antennas use SMA connectors or uFL->SMA adapter cable. Then the GPS antenna is connected to the other end of the cable.



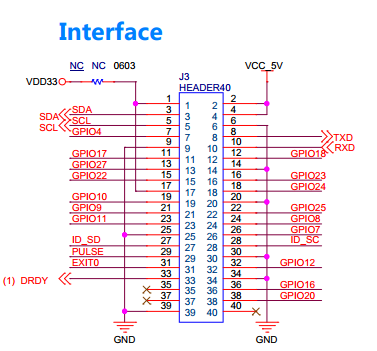
The uFL cable and The SMA connectors

# 7. Pin map

## 7.1 Raspberry PI interfaces



## 7.2 GPS HAT interfce

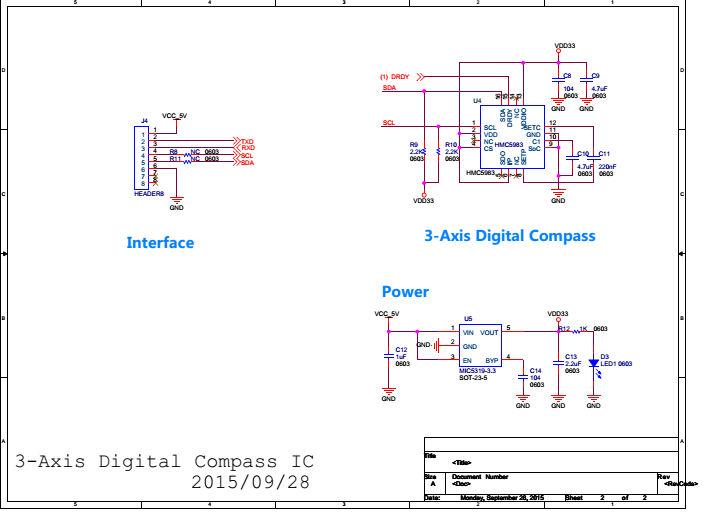


## 7.3 Pin map

|  |  |  |  |
| --- | --- | --- | --- |
| RPI PIN N.O. | Pin Name | Pin of GPS HAT | Description |
| 1 | +3.3V | - |  |
| 2 | +5V | +5V | GPS HAT Power |
| 3 | SDA | SDA | E-compass SDA |
| 4 | +5V | +5V | GPS HAT Power |
| 5 | SCL | SCL | E-compass SCL |
| 6 | GND | GND |  |
| 7 | GPIO4 | - |  |
| 8 | UART\_TXD | GPS\_RXD | Receiving end for gps  serial port |
| 9 | GND | GND |  |
| 10 | UART\_RXD | GPS\_TXD | Sending end for gps  serial port |
| 11 | GPIO17 | - |  |
| 12 | GPIO18 | - |  |
| 13 | GPIO27 | - |  |
| 14 | GND | GND |  |
| 15 | GPIO22 | - |  |
| 16 | GPIO23 | - |  |
| 17 | +3.3V | - |  |
| 18 | GPIO24 | - |  |
| 19 | GPIO10 | - |  |
| 20 | GND | GND |  |
| 21 | GPIO9 | - |  |
| 22 | GPIO25 | - |  |
| 23 | GPIO10 | - |  |
| 24 | GPIO8 | - |  |
| 25 | GND | GND |  |
| 26 | GPIO7 | - |  |
| 27 | ID\_SD | - |  |
| 28 | ID\_SC | - |  |
| 29 | GPIO5 | pulse | Pulse per second / Electronic brick interface |
| 30 | GND | GND |  |
| 31 | GPIO6 | - |  |
| 32 | GPIO12 | - |  |
| 33 | GPIO13 | - |  |
| 34 | GND | GND |  |
| 35 | GPIO19 | - |  |
| 36 | GPIO16 | - |  |
| 37 | GPIO26 | - |  |
| 38 | GPIO20 | - |  |
| 39 | GND | GND |  |
| 40 | GPIO21 | - |  |

# 8.Schematic

# 



# 9.User's manual

**This tutorial assumes you are using an last Raspbian install, have access to either LXTerminal or SSH and have an internet connection!**

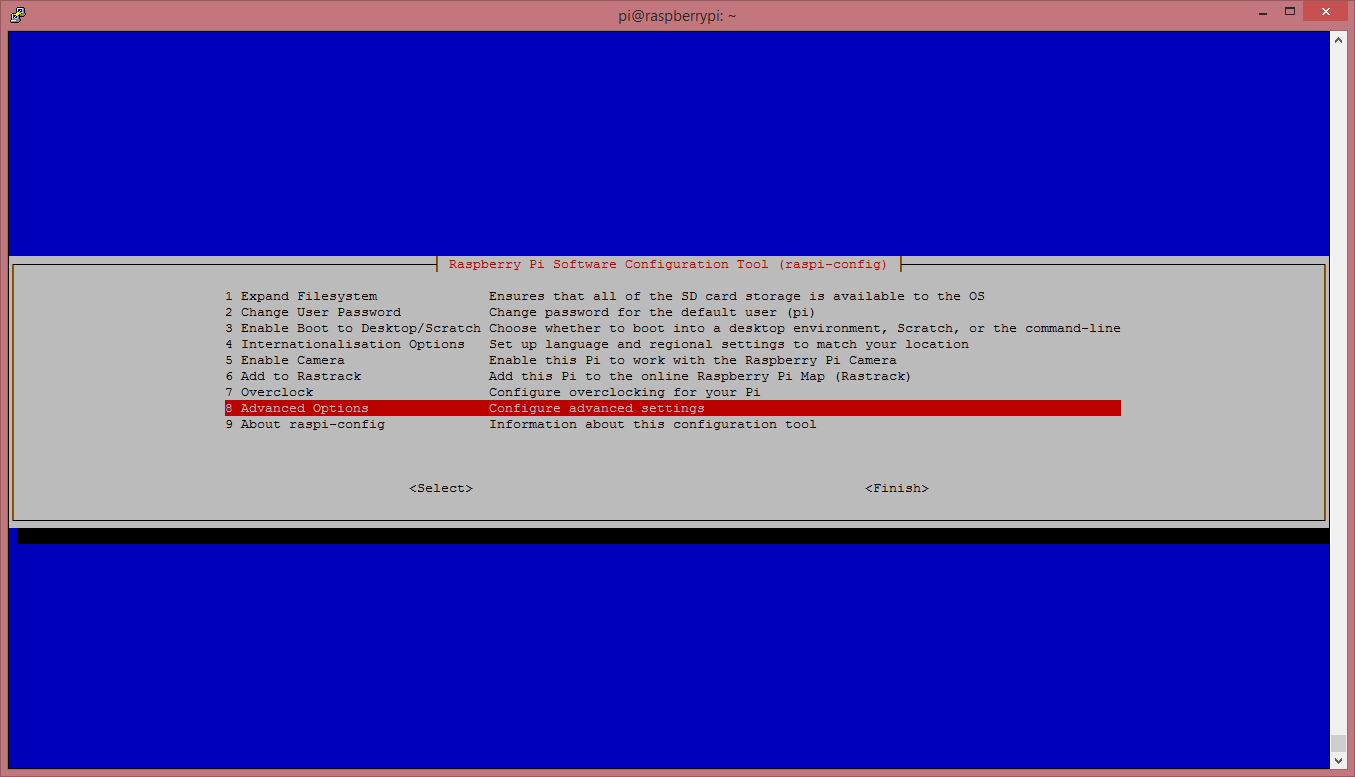
We're going to go through the steps on how to use a GPS module with your Raspberry Pi! In this tutorial we're going to use the Raspberry Pi GPS HAT!

By default, the Raspberry Pi serial port console login is enabled. We need to disable this before we can use the serial port for ourselves.

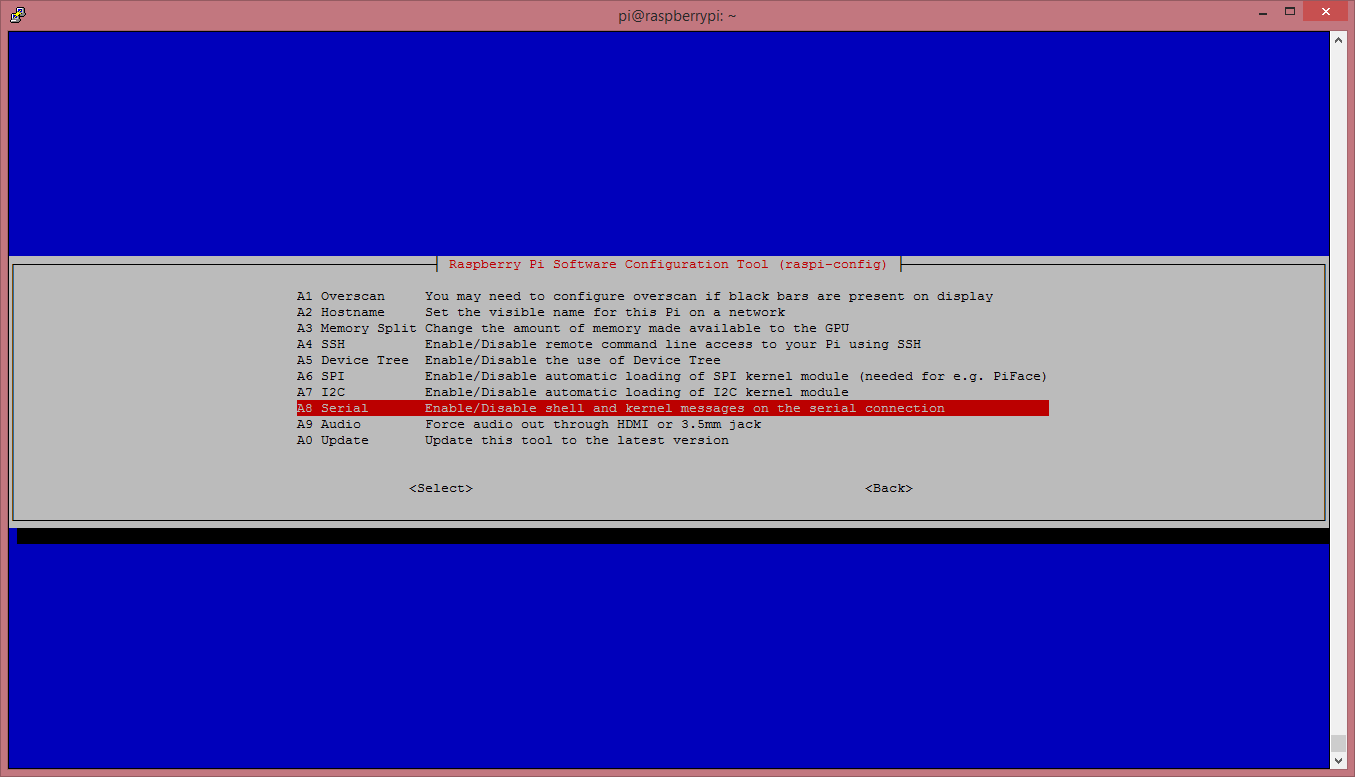
To do this, simply launch up the raspberry pi configuration tool:

sudo raspi-config

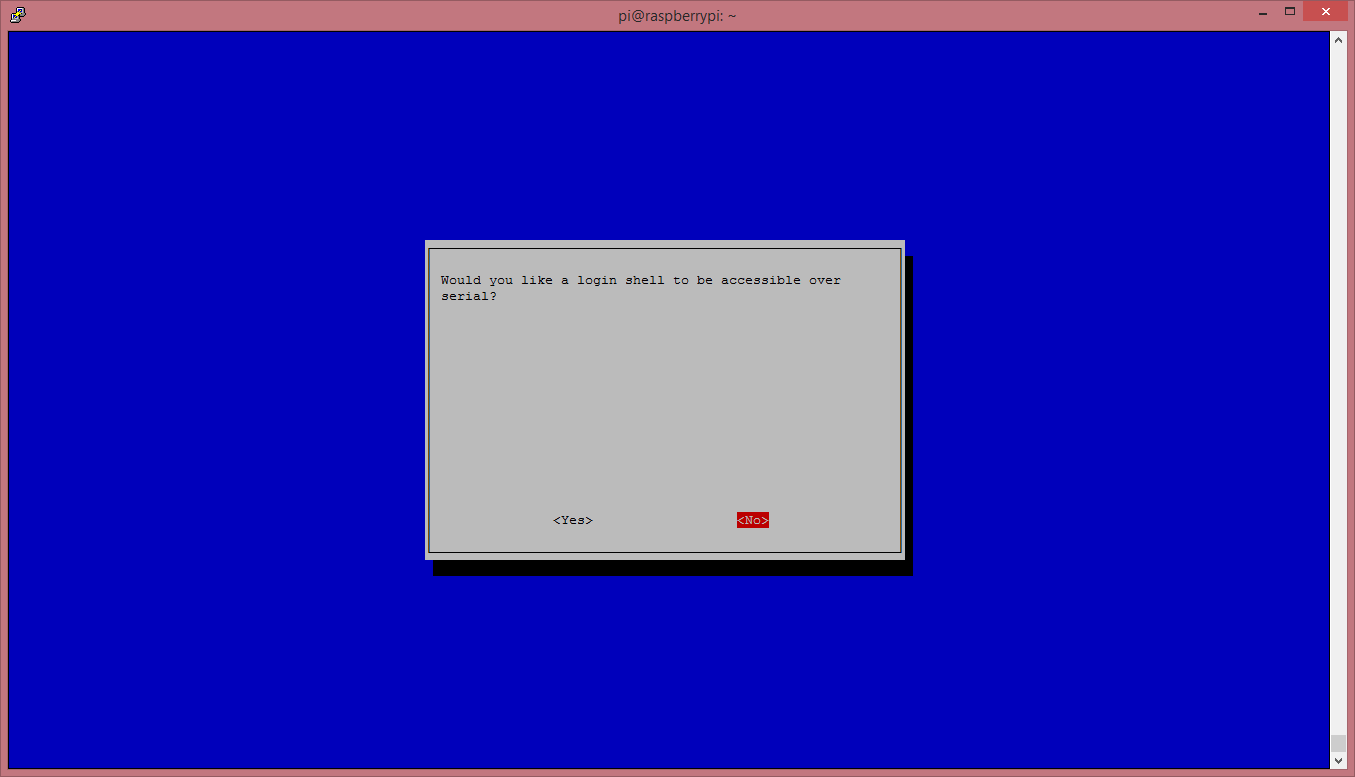
Then select option 8 – Advanced Options



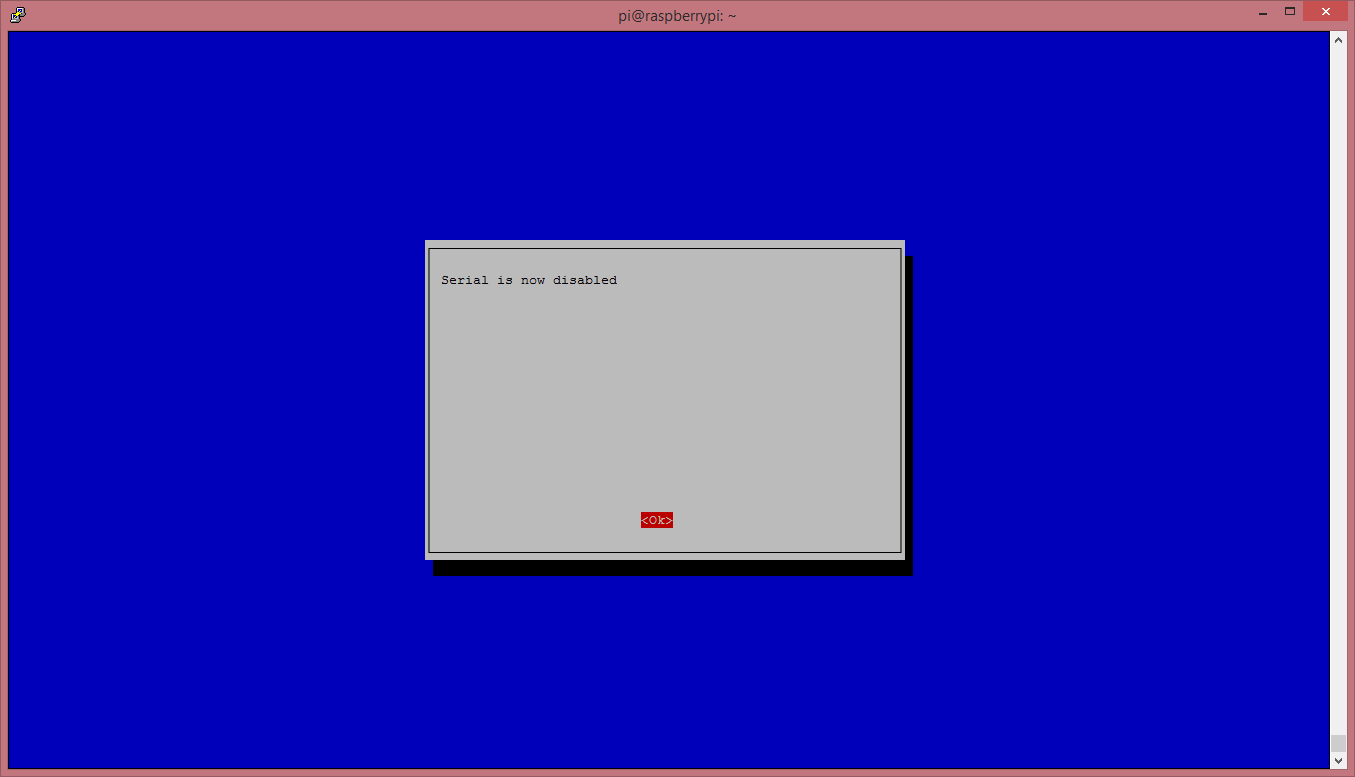
Then go to option A8 – Serial



Navigate to “No”



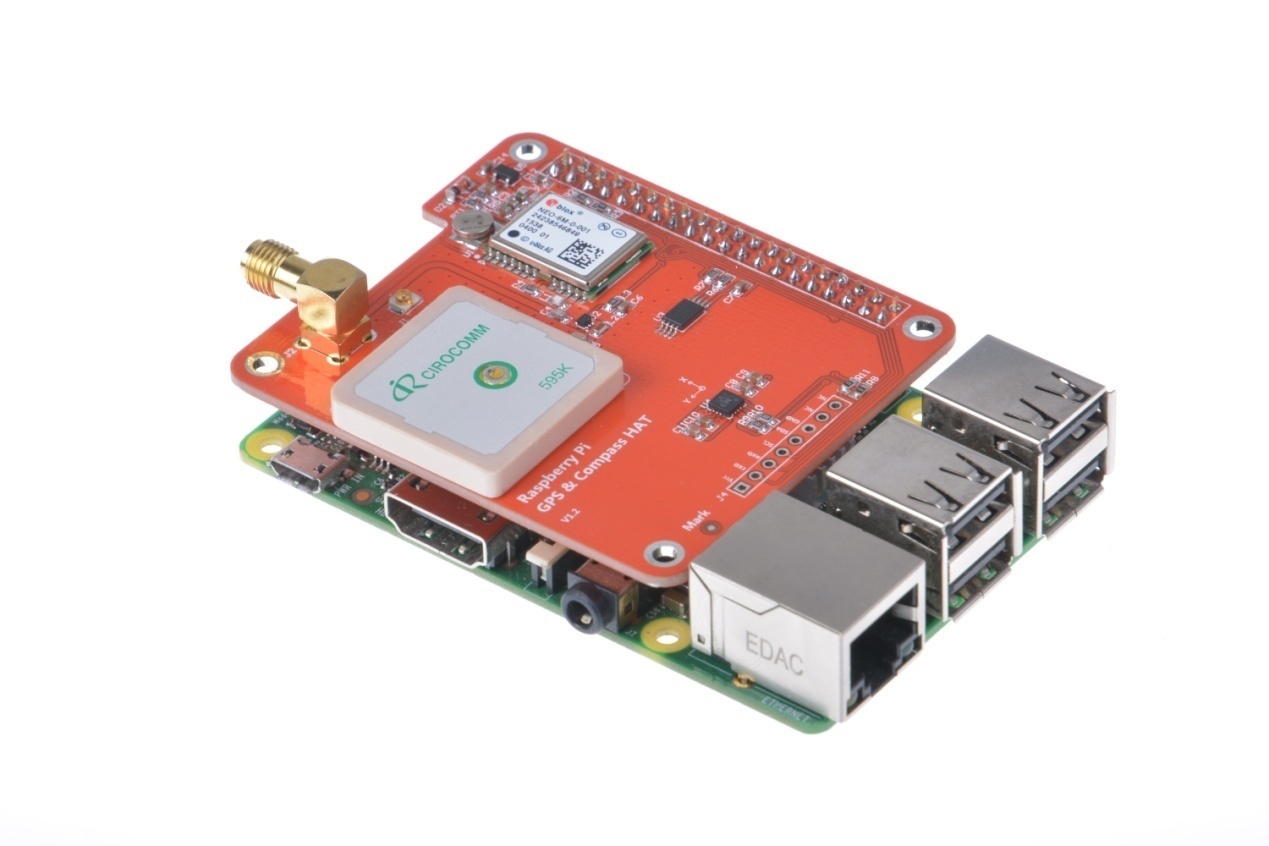
And finally select “Ok”



Now go to “Finish” and power off your Pi with the following command:

*sudo halt*

Whenthe Raspberry Pi is powered off, we can now plug our GPS HAT in and attach an aerial.



Once it is plugged in, we can turn on the Pi.

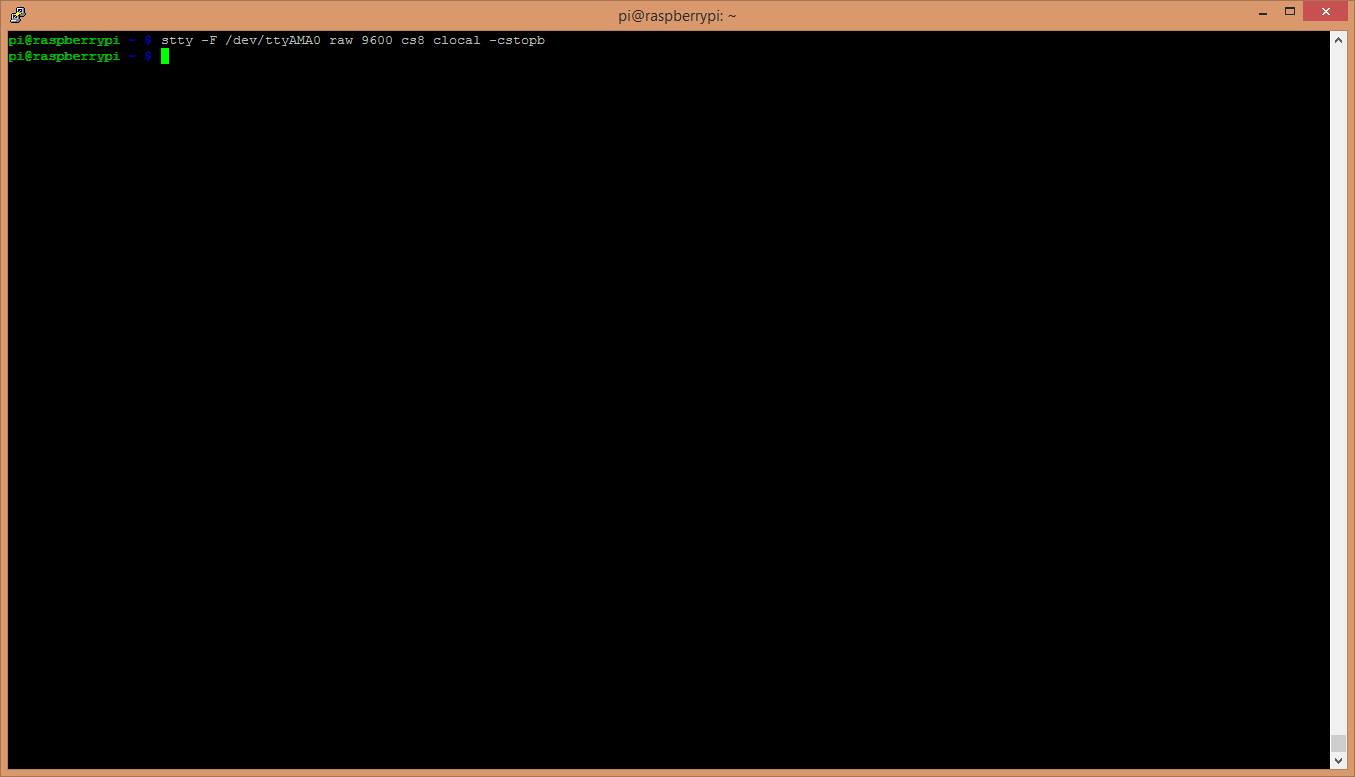
Before we go any further we need to make sure our GPS HAT has a “lock”. To find this out, you’ll need to refer to your GPS HAT manual, or if you are using the HAB Supplies GPS HAT, look for a blinking red led labelled “PPS”.the led is near SMA connectors, Keep in mind that it can take a long time for the HAT to get a lock, so be patient. If you are struggling to get a lock after 30mins try moving you’re aerial. For best results make sure the aerial is outside and has direct line of sight to the sky.

Once we have a GPS lock, we can do a quick test to make sure our Pi is able to read the data provided by the HAT.

So, log in to your Pi.  You can do this via SSH or via the normal method! **Please Note. We're running Raspbian from Terminal and have an internet connection!**

Start by setting up the serial port:

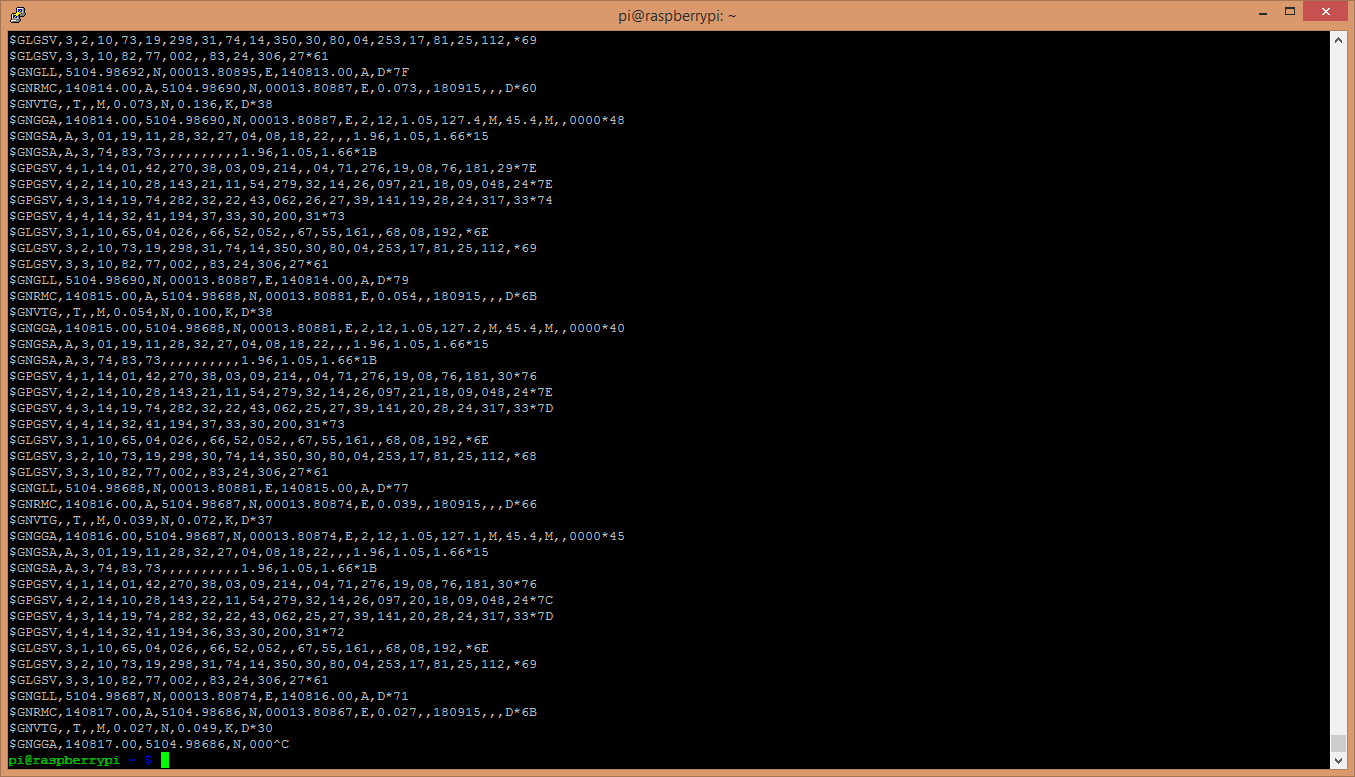
*stty -F /dev/ttyAMA0 raw 9600 cs8 clocal -cstopb*



Now simply run:

*cat /dev/ttyAMA0*

You should see something like this:



What you are seeing here is the raw GPS “NMEA sentence” output from the GPS module. The lines we are interested in are the ones beginning with $GPGGA (again, this might vary depending on the GPS HAT you have, but the line also contains “GGA” at the beginning.)For more NMEA information, refer to file< u-blox 6 Receiver Description .pdf>

If your $GPGGA lines are looking a little empty, and contains a lot of commas “,” with nothing in between them, then you don’t have a GPS lock.

Now it’s time to access this information in a python script!

We are going to use 2 libraries in our script:

1. serial
2. pynmea2

The first one, serial, we don’t need to install anything, this is a default library and will be pre-installed with Raspbian.

The second one, pynmea2, we need to install. So let’s do that! ([pynmea2](https://github.com/Knio/pynmea2) is an easy to use library for parsing NMEA sentences. We could write our own parser, but why re-invent the wheel!)

If you don’t already have “pip” installed, start by installing it:

*sudo apt-get install python-pip*

Once pip is installed we can then go ahead and install pynmea2 using pip:

*sudo pip install pynmea2*

Now we're going to start logging our GPS data using a Python script. This is a basic script that reads the serial port, passes each line to our pynmea2 parser and simply prints out a formatted string containing some information.

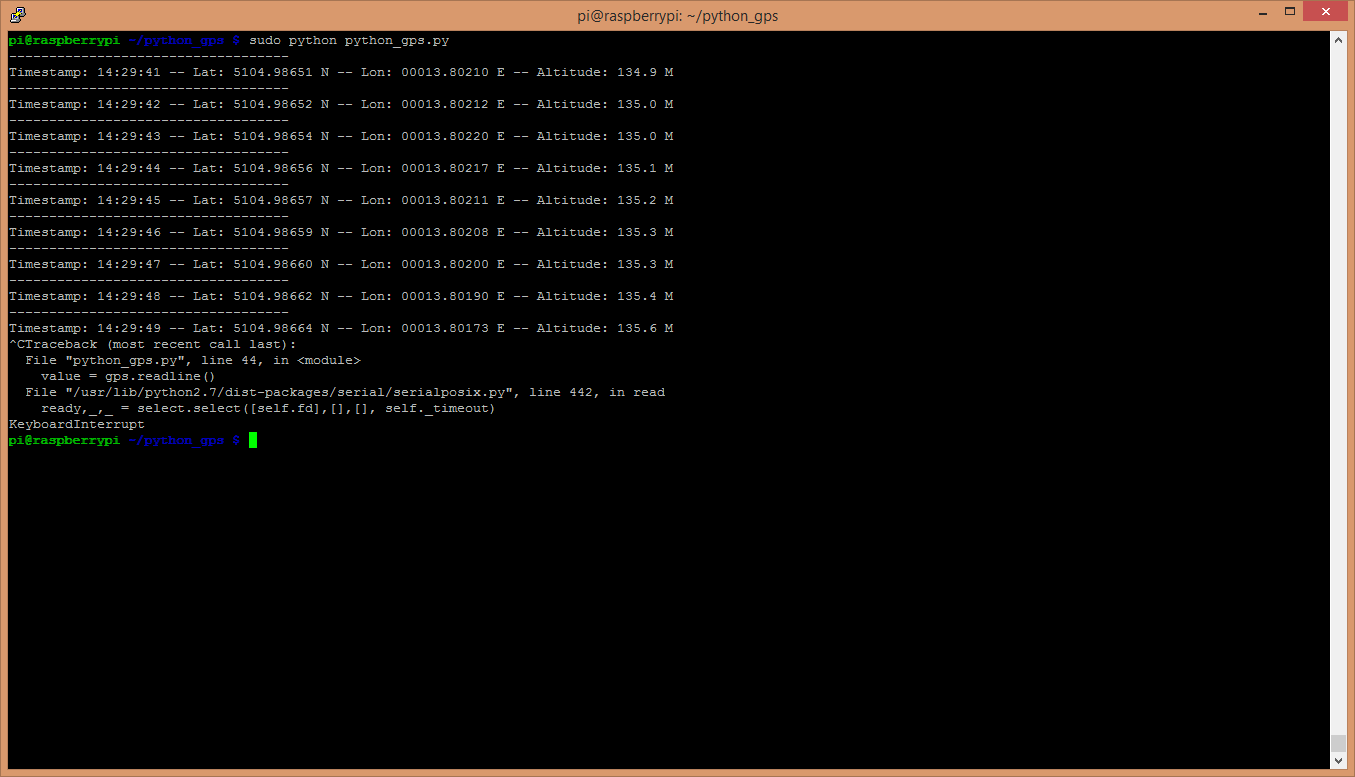
Then, download the python script to our Raspberry Pi, you can view it here -  <https://github.com/modmypi/GPS/blob/master/gps.py>. To do this, we need to use the following GitHub Clone command. This command downloads the Git repository to your current directoy, in this case the Raspberry Pi home directory. You can change this or create a new folder if you wish.

*git clone git://github.com/modmypi/GPS*

We now need to browse to the repo we just downloaded. So change the directory to the GPS folder:  
  
*cd GPS*  
  
We can now run our Python script! To start, simply type:

*sudo python gps.py*

You should see some results like these:



That's it! You're now tracking your GPS data!

**CODE**

Import serial  
import pynmea2  
  
def parseGPS(str):  
    if str.find('GGA') > 0:  
        msg = pynmea2.parse(str)  
        print "Timestamp: %s -- Lat: %s %s -- Lon: %s %s -- Altitude: %s %s" % (msg.timestamp,msg.lat,msg.lat\_dir,msg.lon,msg.lon\_dir,msg.altitude,msg.altitude\_units)  
  
serialPort = serial.Serial("/dev/ttyAMA0", 9600, timeout=0.5)  
  
while True:  
    str = serialPort.readline()  
    parseGPS(str)