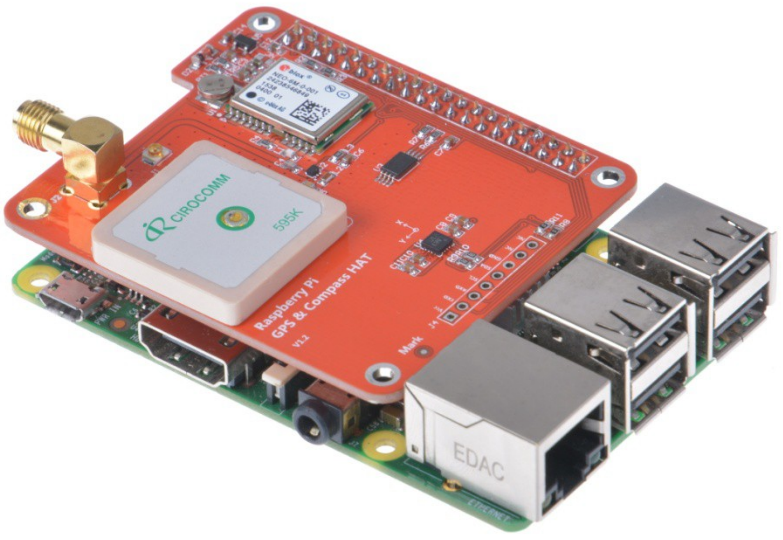
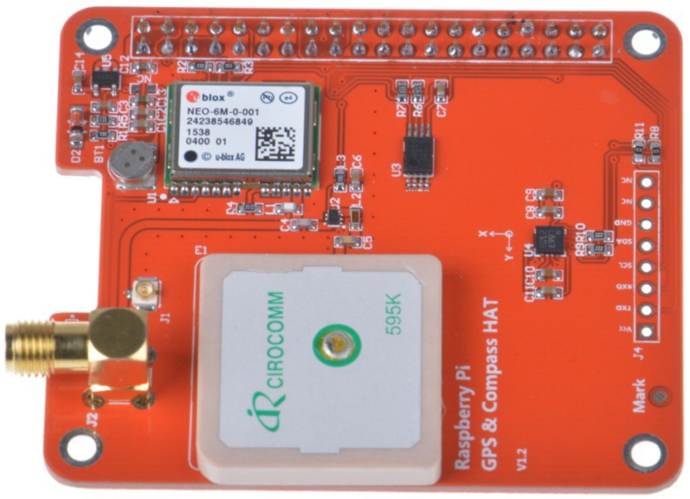
GPS HAT for Raspberry Pi



Overview

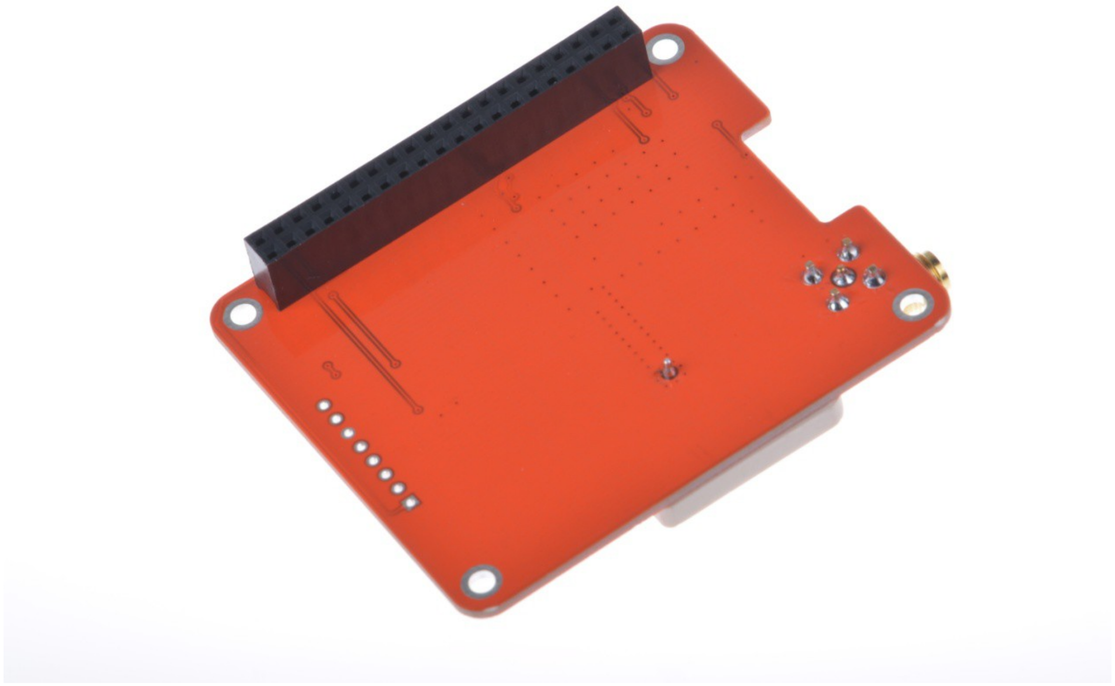


This is an GPS expansion board designed specifically for the both the version 1 and version 2 Raspberry Pi+ Models (This board is NOT compatible with the original Raspberry Pi A and B boards). This board is designed for applications that use a GPS connected via the

serial ports to the Raspberry Pi such as timing applications or general applications that require GPS information. To facilitate PPS the time pulse output is connected to GPIO5 so you can utilise this board to give NTP PPS discipline.
this board is equipped with the latest Ublox NEO-6M/NEO-M8N positioning module.

Here's the low-down on the GPS module:
-1 61dBm sensitivity, Update rate up to 5Hz, 50 channels (NEO-6M)
Only 30mA current draw (NEO-6M)
-1 67dBm sensitivity, Single GNSS update rate up to 18Hz,Concureent GNSS update rate up to 10hz, 72 channels support NASS,BEIDOU2,SBAS (NEO-M8N)
Built in Real Time Clock (RTC) - XH414H-IV01E backup battery for more of time keeping even if the Pi is off!
PPS output on fix, by default connected to pinGPIO5.
Internal patch antenna which works quite well when used outdoors + u.FL connector and SMA connector for external active antenna for when used indoors or in locations without a clear sky view
Fix status LED blinks to let you know when the GPS has determined the current coordinate.

Pinouts



2x20 way header is supplied and 2 suitable standoffs providing a very robust solution .The board follows Raspberry Pi's HAT physical layout with camera and display port notches. It is possible to stack additional HAT's on top of this board with a suitable header.

this HAT takes over the Pi's hardware UART to send/receive data to and from the GPS module. So, if you need to use the RX/TX pins with a console cable, you cannot also use this HAT.

Serial Console Pins

The Raspberry Pi has only one serial port, and you do need serial to chat to a GPS so we will take over the RXD and TXD pins.

PPS Pin

GPS's can output a 'pulse per second' for synchronizing the time. We have a breakout for this and a closed jumper that connects it to GPIO5.

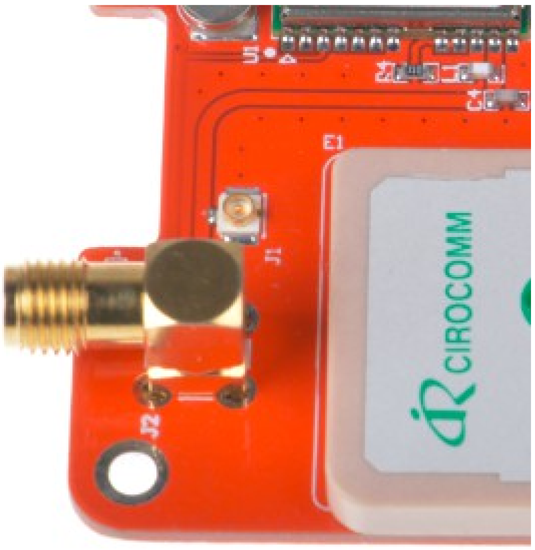
Compass

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

External Antenna

All Ultimate GPS modules have a built in patch antenna - this antenna provides -162 dBm(NEO-6M) or -167 dBm(NEO-M8n)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, you may want to use an external active antenna.

GPS antennas use SMA connectors or uFL->SMA adapter cable.Then connect the GPS antenna to the cable.



The uFL cable and The SMA connectors.

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

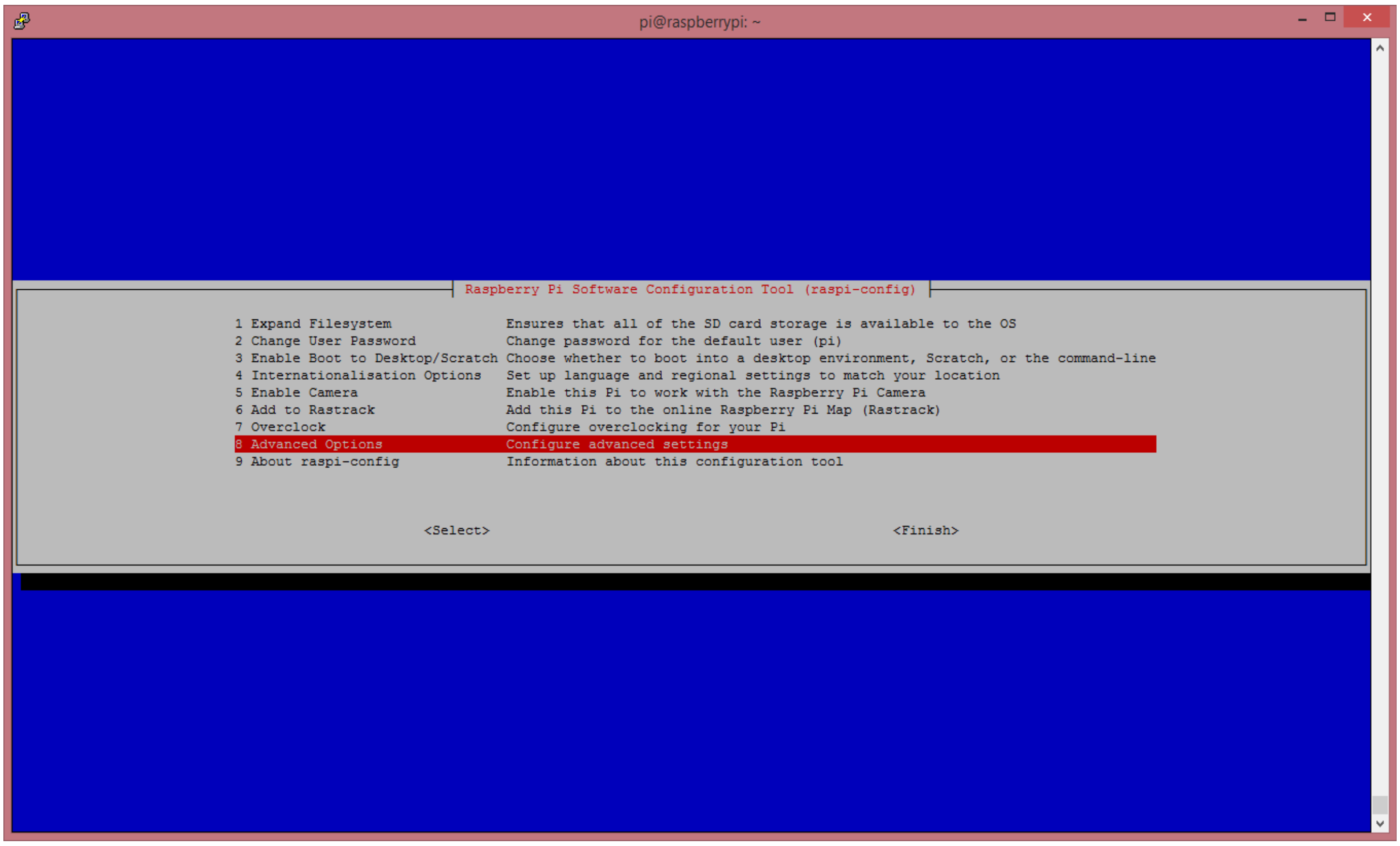
We're going to go through the steps on how to use a GPS module with your Raspberry Pi! In this tutorial we're going to use the Raspbery Pi GPS HAT!

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 load up the raspberry pi configuration tool:

sudo raspi-config

Then go to option 8 – Advanced Options



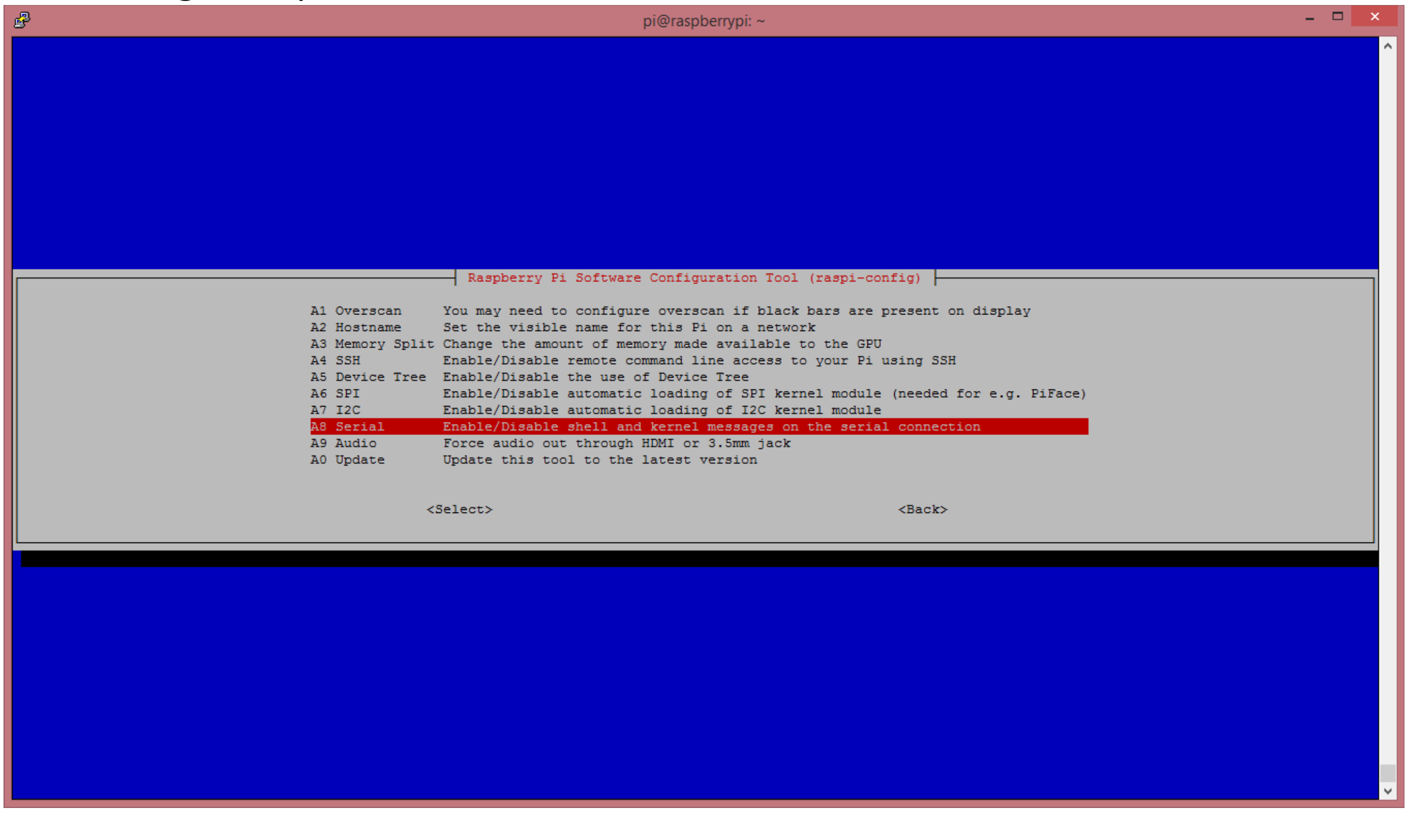
```
pi@raspberrypi: ~

Raspberry Pi Software Configuration Tool (raspi-config)

1 Expand Filesystem            Ensures that all of the SD card storage is available to the OS
2 Change User Password         Change password for the default user (pi)
3 Enable Boot to Desktop/Scratch Choose whether to boot into a desktop environment, Scratch, or the command-line
4 Internationalisation Options Set up language and regional settings to match your location
5 Enable Camera                Enable this Pi to work with the Raspberry Pi Camera
6 Add to Rastrack              Add this Pi to the online Raspberry Pi Map (Rastrack)
7 Overclock                    Configure overclocking for your Pi
8 Advanced Options             Configure advanced settings
9 About raspi-config           Information about this configuration tool

                <Select>                                   <Finish>
```

Then go to option A8 – Serial



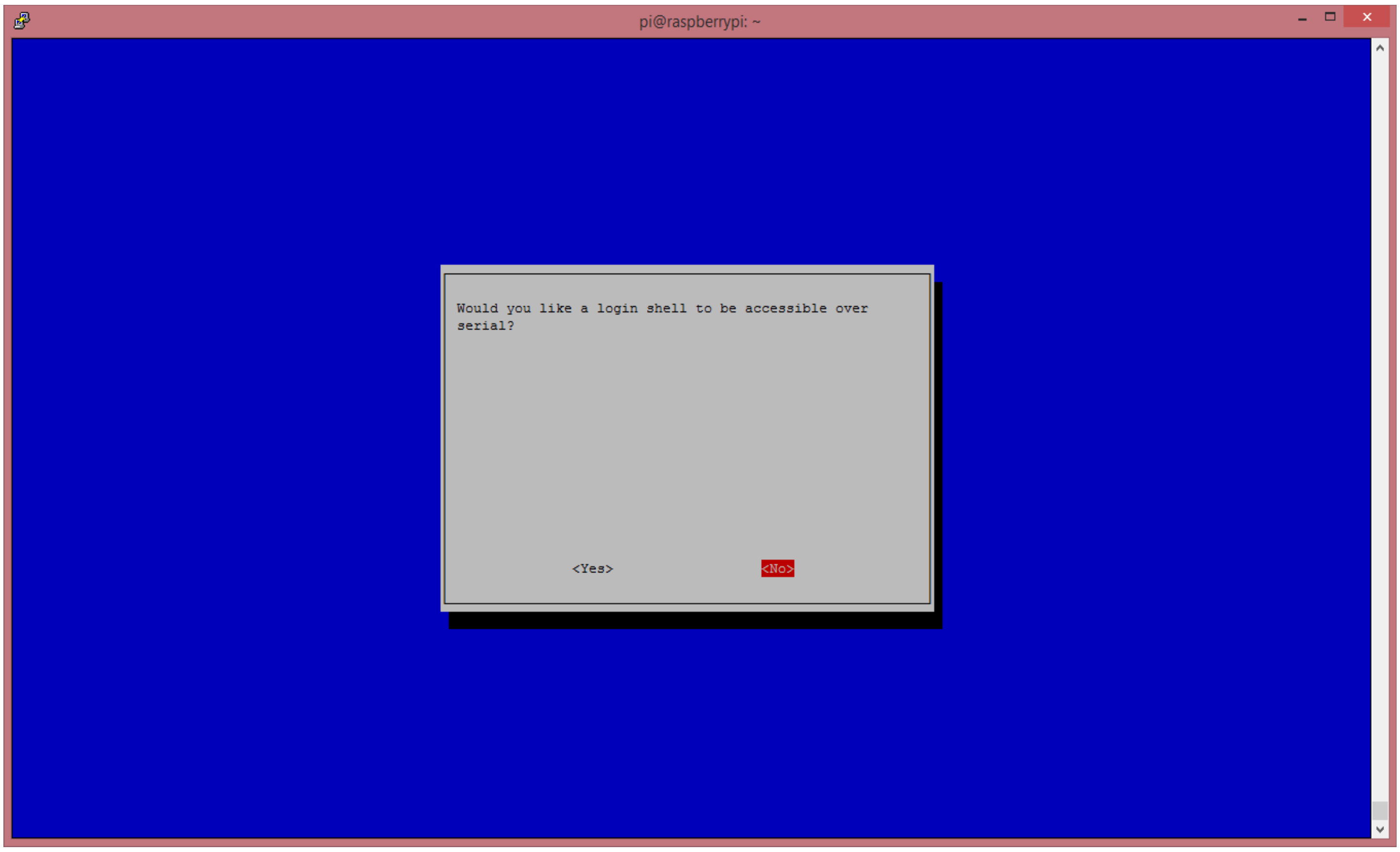
```
pi@raspberrypi: ~

                    Raspberry Pi Software Configuration Tool (raspi-config)

        A1 Overscan     You may need to configure overscan if black bars are present on display
        A2 Hostname     Set the visible name for this Pi on a network
        A3 Memory Split Change the amount of memory made available to the GPU
        A4 SSH          Enable/Disable remote command line access to your Pi using SSH
        A5 Device Tree  Enable/Disable the use of Device Tree
        A6 SPI          Enable/Disable automatic loading of SPI kernel module (needed for e.g. PiFace)
        A7 I2C          Enable/Disable automatic loading of I2C kernel module
        A8 Serial       Enable/Disable shell and kernel messages on the serial connection
        A9 Audio        Force audio out through HDMI or 3.5mm jack
        A0 Update       Update this tool to the latest version

                   <Select>                                                  <Back>
```

Over to "No"

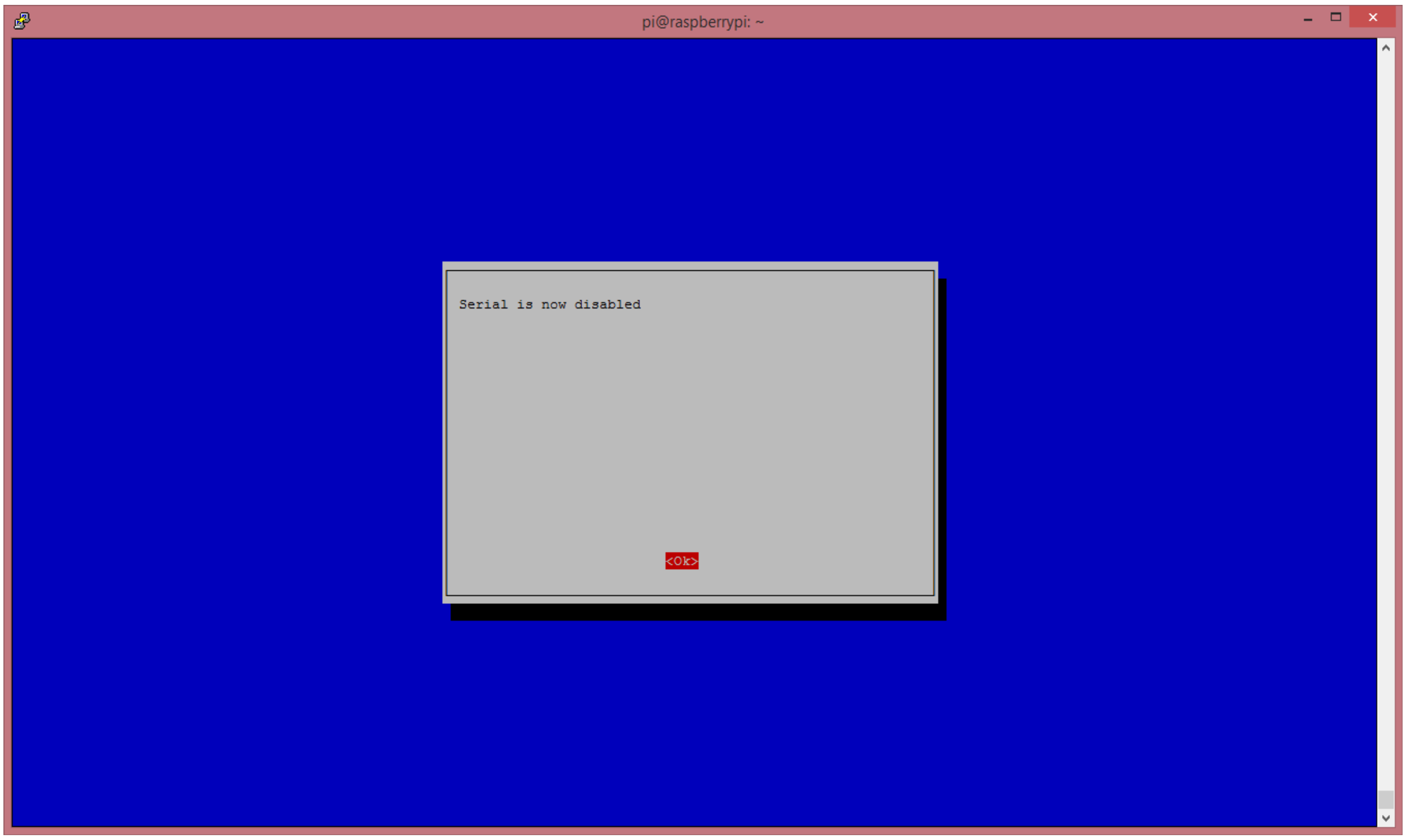


```
pi@raspberrypi: ~

 Would you like a login shell to be accessible over
 serial?

               <Yes>                  <No>
```

And finally “Ok”

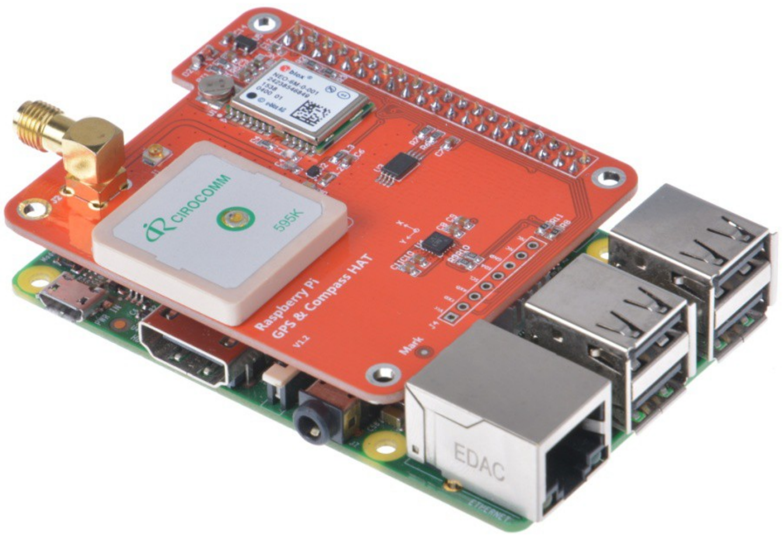


```
Serial is now disabled
```

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

sudo halt

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



Once everything is plugged in, we can power up the Pi.

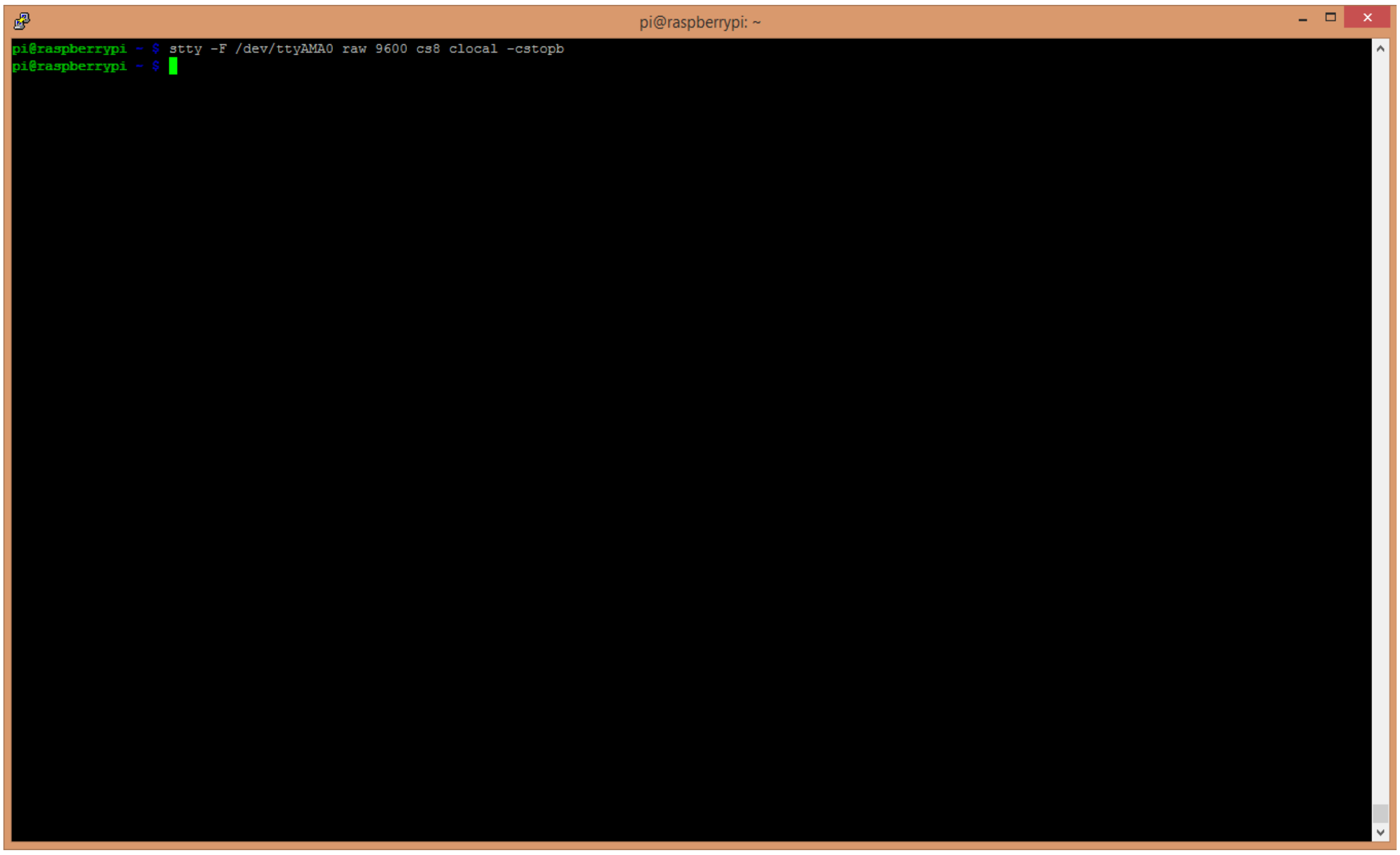
Before we go any further we need to make sure our GPS HAT has a “lock”. To find this out, you’ll need to refer to your GPS HAT manual, or if you are using the HAB Supplies GPS HAT, look for a blinking green led, labelled “timepulse”. Keep in mind that it can take a long time for the HAT to get a lock, 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 Raspian from Terminal and have an internet connection!**

Start by setting up the serial port:

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

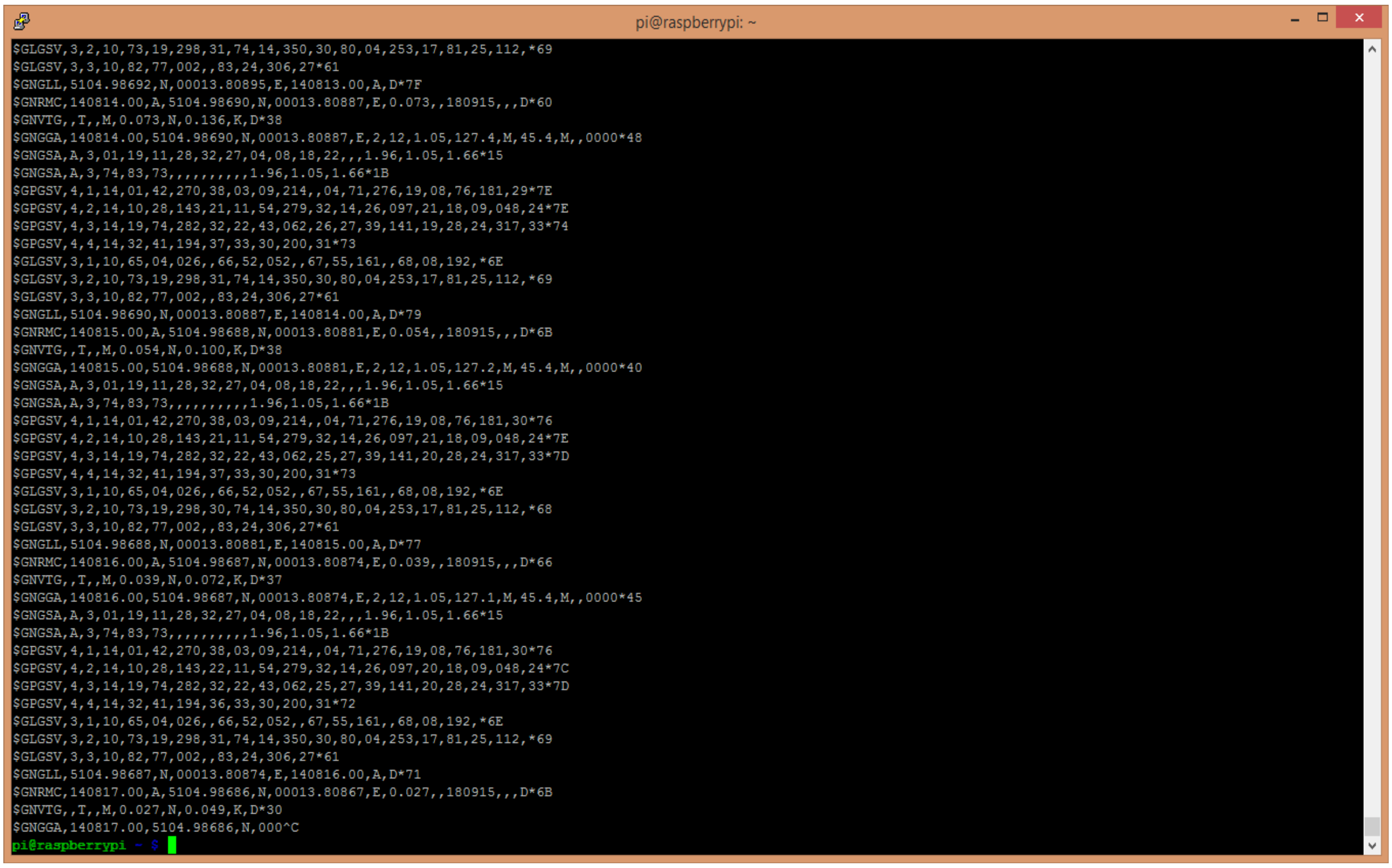


```
pi@raspberrypi ~ $ stty -F /dev/ttyAMA0 raw 9600 cs8 clocal -cstopb
pi@raspberrypi ~ $
```

Now simply run:

cat /dev/ttyAMA0

You should see something like this:



```
$GLGSV,3,2,10,73,19,298,31,74,14,350,30,80,04,253,17,81,25,112,*69
$GLGSV,3,3,10,82,77,002,,83,24,306,27*61
$GNGLL,5104.98692,N,00013.80895,E,140813.00,A,D*7F
$GNRMC,140814.00,A,5104.98690,N,00013.80887,E,0.073,,180915,,,D*60
$GNVTG,,T,,M,0.073,N,0.136,K,D*38
$GNGGA,140814.00,5104.98690,N,00013.80887,E,2,12,1.05,127.4,M,45.4,M,,0000*48
$GNGSA,A,3,01,19,11,28,32,27,04,08,18,22,,,1.96,1.05,1.66*15
$GNGSA,A,3,74,83,73,,,,,,,,,,1.96,1.05,1.66*1B
$GPGSV,4,1,14,01,42,270,38,03,09,214,,04,71,276,19,08,76,181,29*7E
$GPGSV,4,2,14,10,28,143,21,11,54,279,32,14,26,097,21,18,09,048,24*7E
$GPGSV,4,3,14,19,74,282,32,22,43,062,26,27,39,141,19,28,24,317,33*74
$GPGSV,4,4,14,32,41,194,37,33,30,200,31*73
$GLGSV,3,1,10,65,04,026,,66,52,052,,67,55,161,,68,08,192,*6E
$GLGSV,3,2,10,73,19,298,31,74,14,350,30,80,04,253,17,81,25,112,*69
$GLGSV,3,3,10,82,77,002,,83,24,306,27*61
$GNGLL,5104.98690,N,00013.80887,E,140814.00,A,D*79
$GNRMC,140815.00,A,5104.98688,N,00013.80881,E,0.054,,180915,,,D*6B
$GNVTG,,T,,M,0.054,N,0.100,K,D*38
$GNGGA,140815.00,5104.98688,N,00013.80881,E,2,12,1.05,127.2,M,45.4,M,,0000*40
$GNGSA,A,3,01,19,11,28,32,27,04,08,18,22,,,1.96,1.05,1.66*15
$GNGSA,A,3,74,83,73,,,,,,,,,,1.96,1.05,1.66*1B
$GPGSV,4,1,14,01,42,270,38,03,09,214,,04,71,276,19,08,76,181,30*76
$GPGSV,4,2,14,10,28,143,21,11,54,279,32,14,26,097,21,18,09,048,24*7E
$GPGSV,4,3,14,19,74,282,32,22,43,062,25,27,39,141,20,28,24,317,33*7D
$GPGSV,4,4,14,32,41,194,37,33,30,200,31*73
$GLGSV,3,1,10,65,04,026,,66,52,052,,67,55,161,,68,08,192,*6E
$GLGSV,3,2,10,73,19,298,30,74,14,350,30,80,04,253,17,81,25,112,*68
$GLGSV,3,3,10,82,77,002,,83,24,306,27*61
$GNGLL,5104.98688,N,00013.80881,E,140815.00,A,D*77
$GNRMC,140816.00,A,5104.98687,N,00013.80874,E,0.039,,180915,,,D*66
$GNVTG,,T,,M,0.039,N,0.072,K,D*37
$GNGGA,140816.00,5104.98687,N,00013.80874,E,2,12,1.05,127.1,M,45.4,M,,0000*45
$GNGSA,A,3,01,19,11,28,32,27,04,08,18,22,,,1.96,1.05,1.66*15
$GNGSA,A,3,74,83,73,,,,,,,,,,1.96,1.05,1.66*1B
$GPGSV,4,1,14,01,42,270,38,03,09,214,,04,71,276,19,08,76,181,30*76
$GPGSV,4,2,14,10,28,143,22,11,54,279,32,14,26,097,20,18,09,048,24*7C
$GPGSV,4,3,14,19,74,282,32,22,43,062,25,27,39,141,20,28,24,317,33*7D
$GPGSV,4,4,14,32,41,194,36,33,30,200,31*72
$GLGSV,3,1,10,65,04,026,,66,52,052,,67,55,161,,68,08,192,*6E
$GLGSV,3,2,10,73,19,298,31,74,14,350,30,80,04,253,17,81,25,112,*69
$GLGSV,3,3,10,82,77,002,,83,24,306,27*61
$GNGLL,5104.98687,N,00013.80874,E,140816.00,A,D*71
$GNRMC,140817.00,A,5104.98686,N,00013.80867,E,0.027,,180915,,,D*6B
$GNVTG,,T,,M,0.027,N,0.049,K,D*30
$GNGGA,140817.00,5104.98686,N,000^C
pi@raspberrypi ~ $
```

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 $GNGGA (again, this might differ depening on your GPS HAT you have, but look for the line that has "GGA" at the beginning.)

If your $GNGGA 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 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.

We now need to 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
```

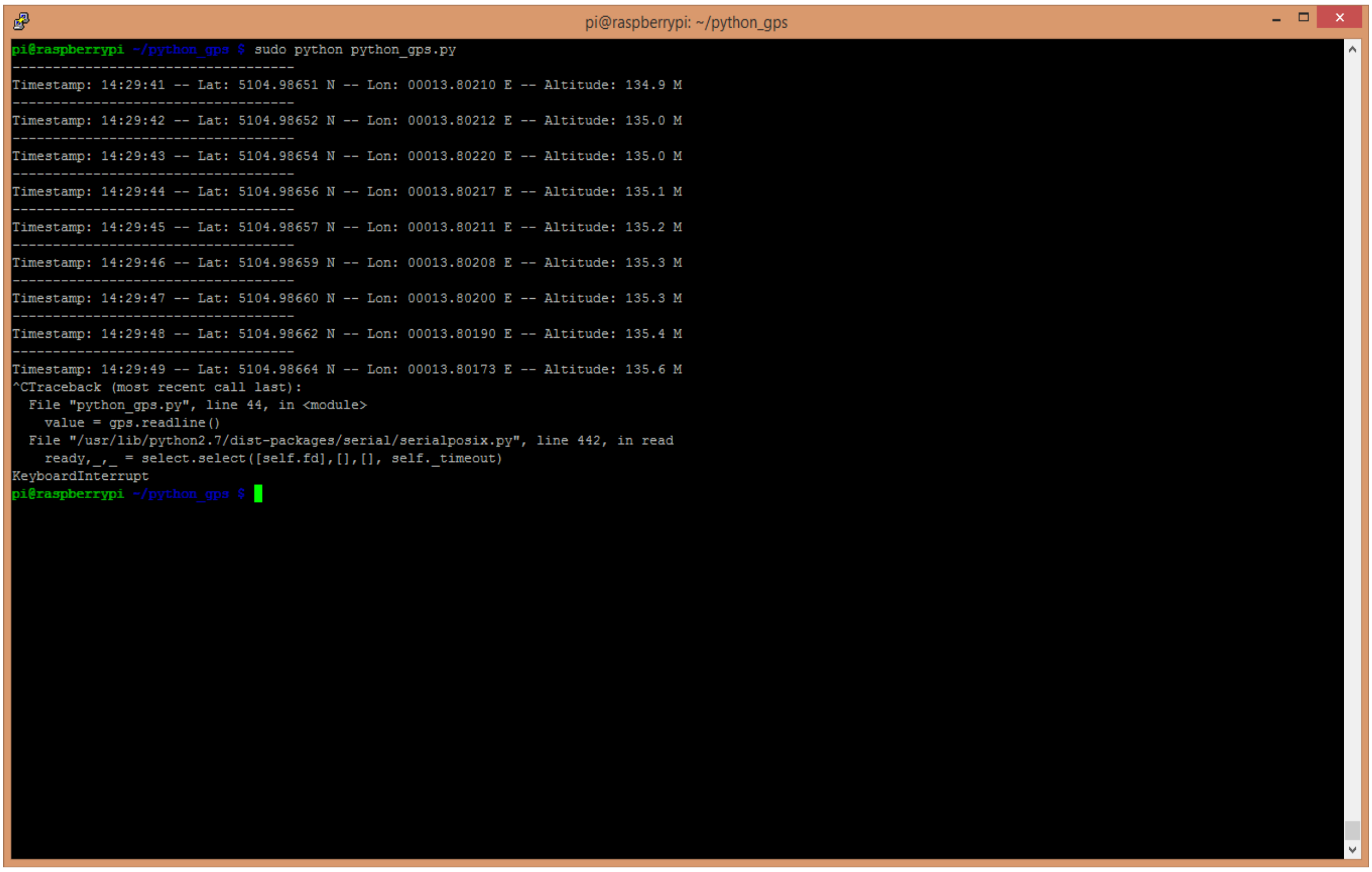
We now need to browse to the repo we just downloaded. So change the directory to the GPS folder:

```
cd GPS
```

We can now run our Python script! To start, simply type:

```
sudo python gps.py
```

You should see some results like these:



```
pi@raspberrypi ~/python_gps $ sudo python python_gps.py
-----------------------------------
Timestamp: 14:29:41 -- Lat: 5104.98651 N -- Lon: 00013.80210 E -- Altitude: 134.9 M
-----------------------------------
Timestamp: 14:29:42 -- Lat: 5104.98652 N -- Lon: 00013.80212 E -- Altitude: 135.0 M
-----------------------------------
Timestamp: 14:29:43 -- Lat: 5104.98654 N -- Lon: 00013.80220 E -- Altitude: 135.0 M
-----------------------------------
Timestamp: 14:29:44 -- Lat: 5104.98656 N -- Lon: 00013.80217 E -- Altitude: 135.1 M
-----------------------------------
Timestamp: 14:29:45 -- Lat: 5104.98657 N -- Lon: 00013.80211 E -- Altitude: 135.2 M
-----------------------------------
Timestamp: 14:29:46 -- Lat: 5104.98659 N -- Lon: 00013.80208 E -- Altitude: 135.3 M
-----------------------------------
Timestamp: 14:29:47 -- Lat: 5104.98660 N -- Lon: 00013.80200 E -- Altitude: 135.3 M
-----------------------------------
Timestamp: 14:29:48 -- Lat: 5104.98662 N -- Lon: 00013.80190 E -- Altitude: 135.4 M
-----------------------------------
Timestamp: 14:29:49 -- Lat: 5104.98664 N -- Lon: 00013.80173 E -- Altitude: 135.6 M
^CTraceback (most recent call last):
  File "python_gps.py", line 44, in <module>
    value = gps.readline()
  File "/usr/lib/python2.7/dist-packages/serial/serialposix.py", line 442, in read
    ready,_,_ = select.select([self.fd],[],[], self._timeout)
KeyboardInterrupt
pi@raspberrypi ~/python_gps $ 
```

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,m
sg.altitude_units)

serialPort = serial.Serial("/dev/ttyAMA0", 9600, timeout=0.5)

while True:
    str = serialPort.readline()
    parseGPS(str)
```