



MOBILE AS GPS

V2X Graduation Project Summary

By: Mahmoud Mohamed Kamal
<https://www.linkedin.com/in/mahmoudfierro98/>

UART Configuration

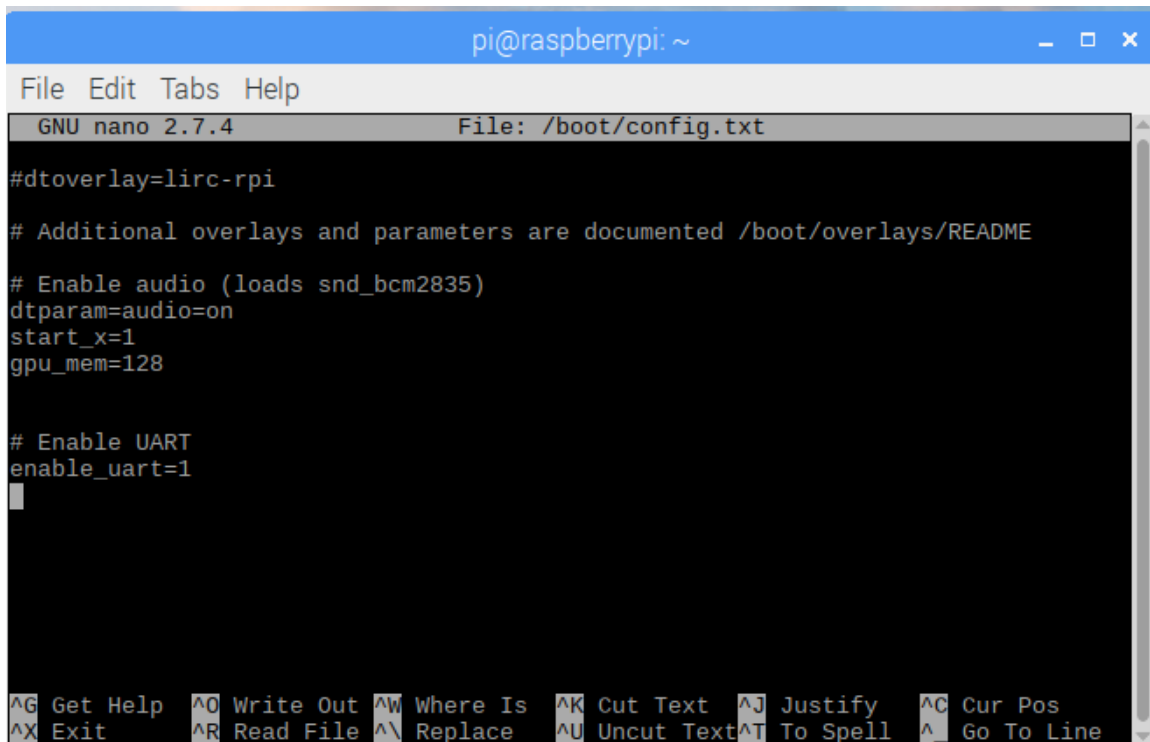
Enable UART in the Raspberry Pi, Open the Raspberry Pi terminal and insert the following commands.

Step 1: Open the config.txt file in the nano editor using the following command.

```
sudo nano /boot/config.txt
```

Add the following lines to the end of the file.

```
# Enable UART
enable_uart=1
```



```
pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 2.7.4 File: /boot/config.txt

#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
dtparam=audio=on
start_x=1
gpu_mem=128

# Enable UART
enable_uart=1

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

Figure 01: UART Configuration Step 1.

Then press Ctrl+x and press Y to save the file and close it.

Step 2: Disconnect the serial communication between the Raspberry Pi and the Bluetooth module.

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

Step 3: Open cmdline.txt in nano editor.

Delete the “console=serial0,115200” line and save the file.

```
sudo nano /boot/cmdline.txt
```

Step 4: Reboot your Raspberry Pi.

`sudo reboot`

Step 5: Install python-serial.

`sudo apt-get install python-serial`

`sudo apt-get install python3-serial`

Step 5: Use python to receive data from Microcontrollers.



```
UART.py *  
1 import serial  
2  
3 ser = serial.Serial("/dev/ttyS0",baudrate=9600,timeout=0.5)  
4 ser.write('Connected\r\n')  
5  
6 while True:  
7     data = ser.readline()  
8     print(data)
```

Bluetooth Configuration

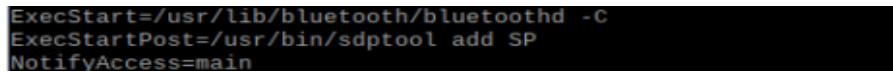
We used it for GPS.

Enable Bluetooth in the Raspberry Pi, Open the Raspberry Pi terminal and insert the following commands.

Step 1: Add the SP profile to the Pi. Edit this file:

`sudo nano /etc/systemd/system/dbus-org.bluez.service`

Step 2: Add the compatibility flag, '-C', at the end of the 'ExecStart=' line. Add a new line after that to add the SP profile. The two lines should look like this:



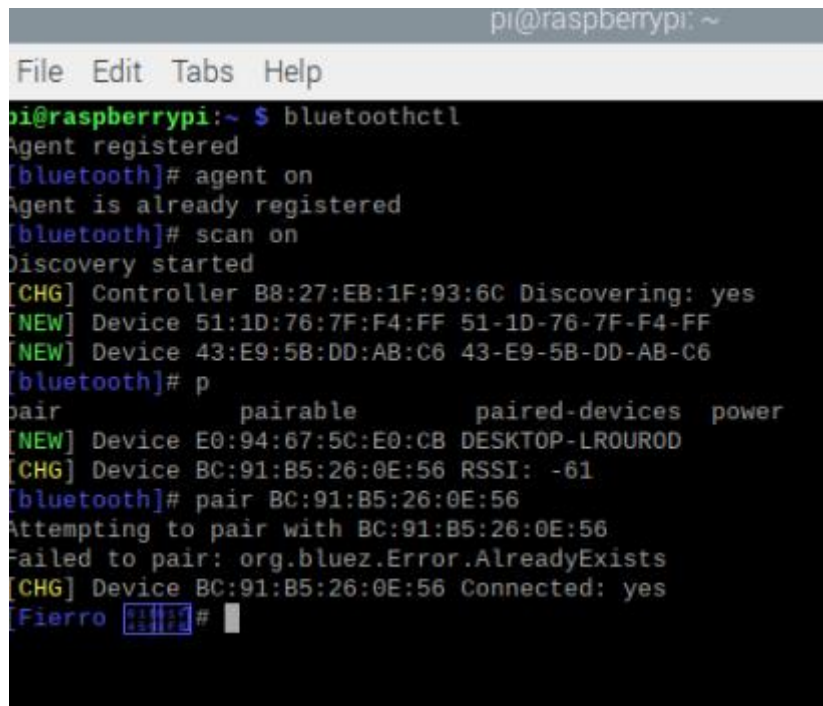
```
ExecStart=/usr/lib/bluetooth/bluetoothd -C  
ExecStartPost=/usr/bin/sdptool add SP  
NotifyAccess=main
```

Figure 02: Bluetooth Configuration Step 2.

Step 3: Save the file and reboot.

`Sudo reboot`

Step 4: Pair and trust your Pi and phone with bluetoothctl or with GUI.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ bluetoothctl  
Agent registered  
[bluetooth]# agent on  
Agent is already registered  
[bluetooth]# scan on  
Discovery started  
[CHG] Controller B8:27:EB:1F:93:6C Discovering: yes  
[NEW] Device 51:1D:76:7F:F4:FF 51-1D-76-7F-F4-FF  
[NEW] Device 43:E9:5B:DD:AB:C6 43-E9-5B-DD-AB-C6  
[bluetooth]# p  
pair pairable paired-devices power  
[NEW] Device E0:94:67:5C:E0:CB DESKTOP-LROUROD  
[CHG] Device BC:91:B5:26:0E:56 RSSI: -61  
[bluetooth]# pair BC:91:B5:26:0E:56  
Attempting to pair with BC:91:B5:26:0E:56  
Failed to pair: org.bluez.Error.AlreadyExists  
[CHG] Device BC:91:B5:26:0E:56 Connected: yes  
Fierro [bluetooth]#
```

Figure 03: Bluetooth Configuration Step 4 - Terminal.

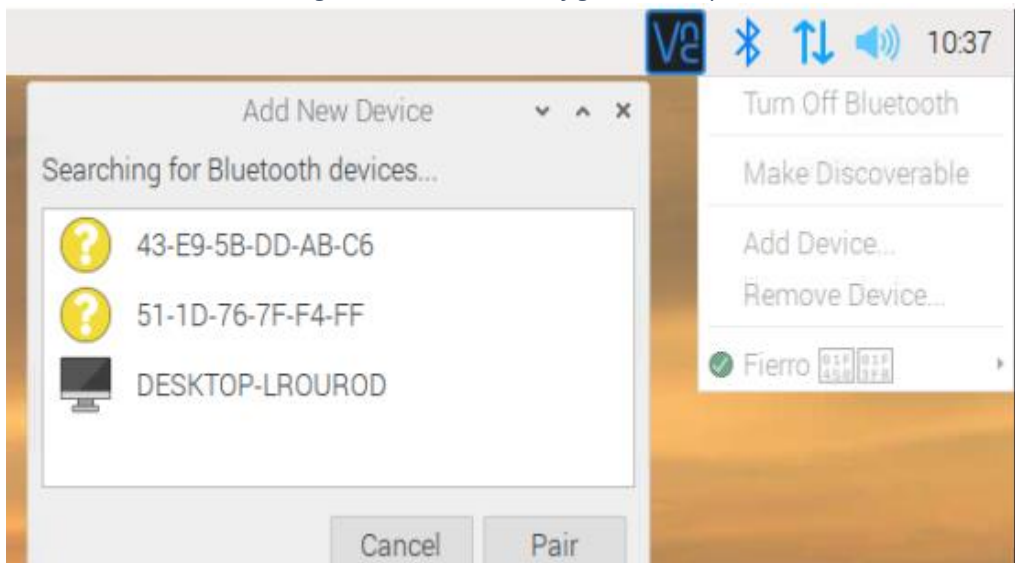


Figure 04: Bluetooth Configuration Step 4 - GUI.

Share GPS via Bluetooth:

Step 1: Pair your mobile phone with Raspberry pi.

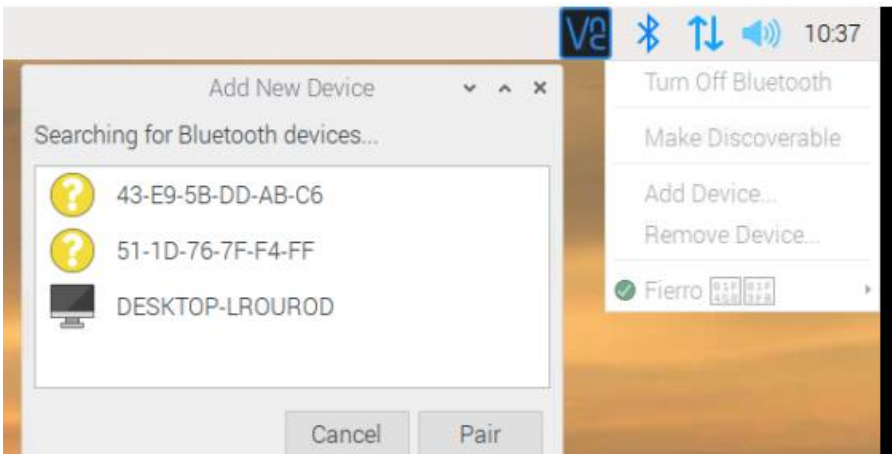


Figure 05: Share GPS via Bluetooth Step 1.

Step 2: Install Share GPS on your phone.



Figure 06: Share GPS via Bluetooth Step 2.

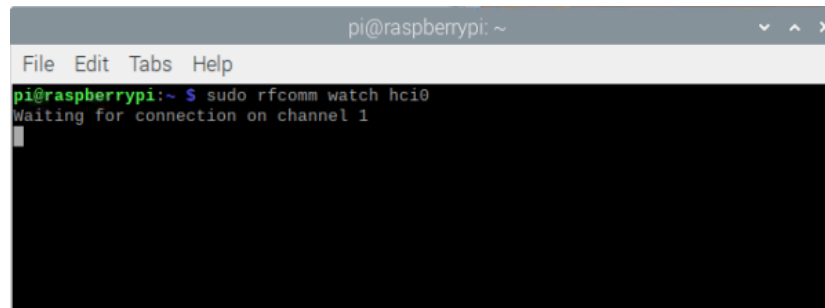
Step 3: Turn on Location on your phone and the app will start getting your latitude and longitude.

GPS STATUS		CONNECTIONS	
Status 3D Fix			
Latitude 31.27447		Altitude 15.7 m	
Longitude 30.00754		Direction --	
Distance Traveled 4.036 miles		Speed 0 mph	
Distance From Start 0.007 miles		Track Time 791:14:12	
STOP TRACK		CLEAR TRACK	

Figure 07: Share GPS via Bluetooth Step 3.

Step 4: wait serial connection by using this command:

sudo rfcomm watch hci0



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo rfcomm watch hci0  
Waiting for connection on channel 1
```

Figure 08: Share GPS via Bluetooth Step 4.

Step 5: Connect the app with raspberry pi.

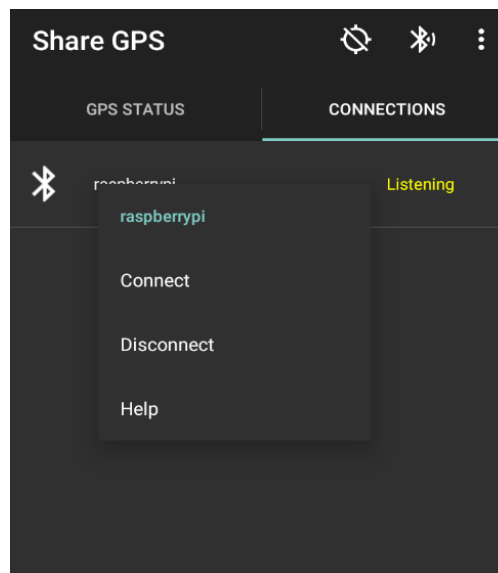
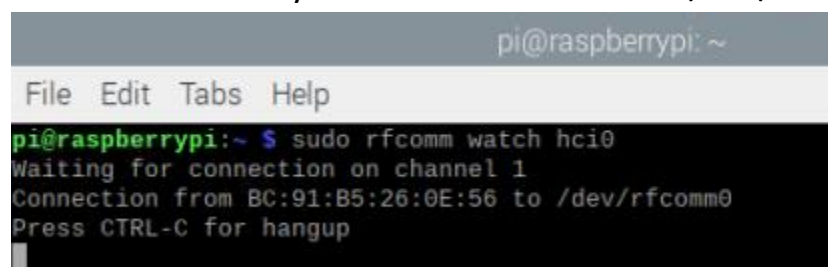


Figure 09: Share GPS via Bluetooth Step 5.

Step 6: Now app starts to connect and share location to raspberry pi.
And you can receive the reads by serial communication of /dev/rfcomm0.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo rfcomm watch hci0  
Waiting for connection on channel 1  
Connection from BC:91:B5:26:0E:56 to /dev/rfcomm0  
Press CTRL-C for hangup
```

Figure 10: Share GPS via Bluetooth Step 6 a.

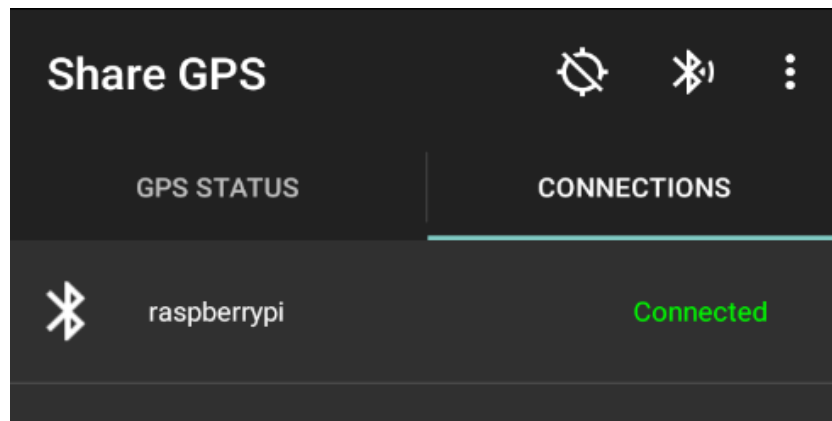


Figure 11: Share GPS via Bluetooth Step 6 b.

Step 7: By python and serial module we can get NMEA code that had your latitude and longitude.

```
GPS.py %
1 import serial
2
3 ser = serial.Serial("/dev/rfcomm0",baudrate=9600,timeout=0.5)
4
5
6 while True:
7     data = ser.readline()
8     print(data)
```

```
pi@raspberrypi: ~/Desktop
File Edit Tabs Help
pi@raspberrypi:~ $ cd Desktop/
pi@raspberrypi:~/Desktop $ python GPS.py

$GNGSA,A,2,05,20,29,,,,,,,,,3.7,3.5,1.0,1*3F
$GNGSA,A,2,68,85,,,,,,,,,3.7,3.5,1.0,2*33
$PQGSA,A,2,11,13,,,,,,,,,3.7,3.5,1.0,4*3C
$GNVTG,,T,,M,0.0,N,0.0,K,A*3D
$GNRMC,102446.00,A,3116.461976,N,03000.451996,E,0.0,,130621,3.0,E,A*0F
$GNGGA,102446.00,3116.461976,N,03000.451996,E,1,07,3.5,17.6,M,20.0,M,,*4F
$GPGSV,3,1,12,05,21,046,18,20,23,046,19,29,47,054,16,10,14,187,*78
$GPGSV,3,2,12,12,02,142,,15,04,105,,16,16,319,,18,74,310,*7A
$GPGSV,3,3,12,23,40,165,,25,34,149,,26,42,309,,31,26,247,*7E
$GLGSV,3,1,09,85,17,272,15,68,21,067,19,74,55,277,,73,39,196,*63
$GLGSV,3,2,09,80,00,000,,69,19,120,,84,48,337,,83,19,045,*64
$GLGSV,3,3,09,67,01,023,*5D
```

Figure 12: Share GPS via Bluetooth Step 7.

NMEA:

At the end we received a message that called NMEA code or NMEA message.

```
$GNRMC,055557.00,A,3116.459486,N,03000.462794,E,0.0,170721,3.0,E,A*0D
```

All we need is this line that has GMT, your latitude, and your longitude.

- GMT: **055557** This means the time of GMT 05:55:57.

- Position status: **A** (A = data valid, V = data invalid)

- Latitude:

```
,3116.461976,N,
```

This means $31^{\circ}16.461976'N = 31.27438^{\circ}N$

- Longitude:

```
03000.451996,E
```

This means $30^{\circ}00.451996'E = 30.00755^{\circ}E$

So, the app all his works is sharing GPS of mobile with Raspberry pi and sending NMEA code with your time and position.

Latitude	Altitude
31.27438	18.1 m
Longitude	Direction
30.00754	--
Distance Traveled	Speed
4.082 miles	0 mph
Distance From Start	Track Time
0.002 miles	791:50:49

Figure 13: NMEA Checking.

Advantages

- So accurate using many stiles to get your location.
- Indoor.
- Suitable for our project.
- Using Bluetooth of Raspberry pi.

Software

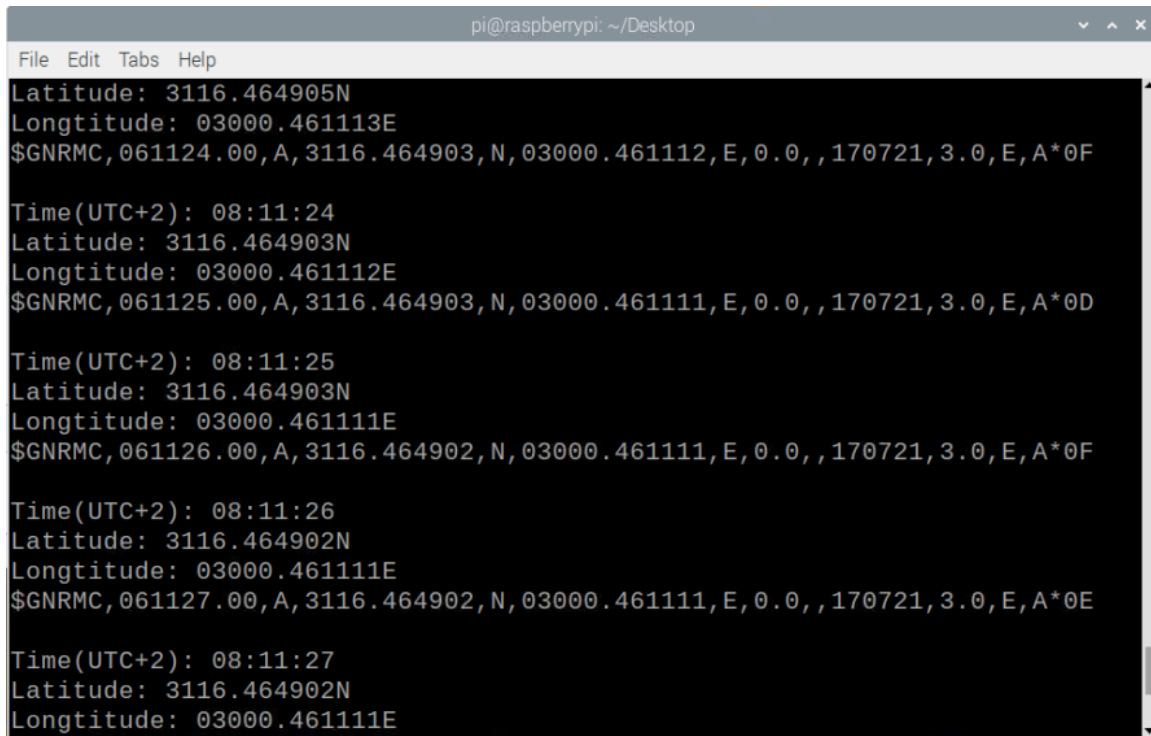
Using serial module to get data from Bluetooth of mobile and the parsing NMEA code to get specific line start with “\$GNRMC” sprite it to time, latitude and longitude.

We can send the data after parsing to another microcontroller through UART.

```

GPS.py
1 import serial
2
3 GPS = serial.Serial("/dev/rfcomm0",baudrate=9600,timeout=0.5)
4 #UART = serial.Serial("/dev/ttyS0",baudrate=9600,timeout=0.5)
5 #UART.write("ddd")
6 while True:
7     line = GPS.readline()
8     data = line.split(",")
9     ''' Parsing NMEA Code '''
10    if data[0] == "$GNRMC":
11        #UART.write(line)
12        #UART.write("\n")
13        ''' Get UTC of postion '''
14        time = data[1]
15        time = time.split(".")
16        utc = time[0]
17        utc2 = str(int(utc[1])+2)
18        hour = utc[0] + utc2
19        mins = utc[2:4]
20        secs = utc[4:6]
21        print(line)
22        print ("Time(UTC+2): "+hour+" "+mins+" "+secs)
23        #UART.write("Time(UTC+2): "+hour+" "+mins+" "+secs)
24        #UART.write("\n")
25        ''' Check Position status (A = data valid, V = data invalid) '''
26        if data[2] == "A":
27            ''' Get UTC of Latitude '''
28            Latitude_deg = data[3][0:2]
29            Latitude_min = data[3][2::]
30            Latitude_dir = data[4]
31            Latitude = Latitude_deg+Latitude_min+Latitude_dir
32            print("Latitude: "+Latitude)
33            #UART.write("Latitude: "+Latitude)
34            #UART.write("\n")
35            ''' Get UTC of Longitude '''
36            Longitude_deg = data[5][0:3]
37            Longitude_min = data[5][3::]
38            Longitude_dir = data[6]
39            Longitude = Longitude_deg+Longitude_min+Longitude_dir
40            print("Longitude: "+Longitude)
41            # UART.write("Longitude: "+Longitude)
42            #UART.write("\n")
43        else:
44            print "Not Fix - Data Invalid"

```

A screenshot of a terminal window titled 'pi@raspberrypi: ~/Desktop'. The window displays a series of GPS data outputs. Each output block includes the time in UTC+2, latitude, longitude, and a NMEA sentence (\$GNRMC). The data is updated every second, as indicated by the increasing time values from 08:11:24 to 08:11:27. The latitude and longitude values remain relatively constant, with minor fluctuations in the last decimal place. The NMEA sentence contains the same coordinates and additional data like speed and heading.

```
File Edit Tabs Help
Latitude: 3116.464905N
Longitude: 03000.461113E
$GNRMC,061124.00,A,3116.464903,N,03000.461112,E,0.0,,170721,3.0,E,A*0F

Time(UTC+2): 08:11:24
Latitude: 3116.464903N
Longitude: 03000.461112E
$GNRMC,061125.00,A,3116.464903,N,03000.461111,E,0.0,,170721,3.0,E,A*0D

Time(UTC+2): 08:11:25
Latitude: 3116.464903N
Longitude: 03000.461111E
$GNRMC,061126.00,A,3116.464902,N,03000.461111,E,0.0,,170721,3.0,E,A*0F

Time(UTC+2): 08:11:26
Latitude: 3116.464902N
Longitude: 03000.461111E
$GNRMC,061127.00,A,3116.464902,N,03000.461111,E,0.0,,170721,3.0,E,A*0E

Time(UTC+2): 08:11:27
Latitude: 3116.464902N
Longitude: 03000.461111E
```

Figure 14: GPS code output.

As shown refresh data every 1 second.