Utilizing GPS Information from the GlobalSat BU-353S4 on a UNIX system

Cody Wilson

4/1/15

Abstract

The BU-353S4 is a USB GPS receiver. The receiver outputs using the NMEA format. Using a UNIX system, the output of this receiver can be read and formatted, using a python script, for use in applications such as mapping, tracking, turn by turn directions, etc.

Keywords

GPS, UNIX, terminal, NMEA

Introduction

GPS (Global Positioning System) is a satellite navigation system that, with an unobstructed line of sight to four or more GPS satellites, can accurately provide location and time information anywhere on earth. GPS satellites were first launched in 1978 for military use, but was not fully available to the public until the year 2000. Civilian GPS receivers provide, on average, 7.8m of accuracy.

The BU-353S4 (herein referred to as the receiver) is a GPS receiver made by GlobalSat. The receiver is USB powered and outputs the GPS data in the standardized NMEA (National Marine Electronics Association) formats: GGA, GSA, GSV, and RMC.

This document walks you through connecting the receiver to a UNIX system and properly reading the information output by the receiver so that it can be formatted for further use.

Hardware/software

A UNIX based system with a USB port and an internet connection are required to complete the actions in this document. The Linux based Ubuntu is the operation system that is used throughout this document, however; the steps will be the same for most other UNIX based systems. You will need python and a text editing program to write the python programs. This document uses python 2.7 and VIM respectively.

Connecting the Receiver

In order to access the receiver once it is connected to the system, we need to know the device file that is created when connected. UNIX systems create device files for each device connected to the system. Before plugging the receiver into the system, open a file browser and enter the /dev/ folder as can be seen in Figure 1.

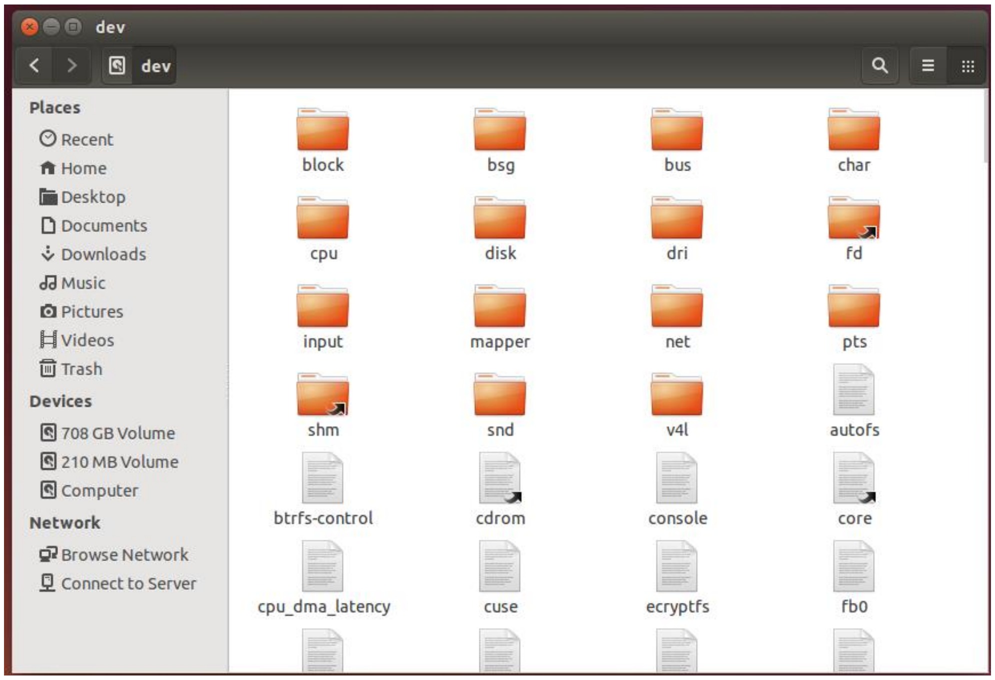


Figure 1

Once in the folder plug the device into one of the USB ports on your system. A new device file should appear, take note of its name. The device file should be named “ttyUSB0” or something similar as can be seen in Figure 2.

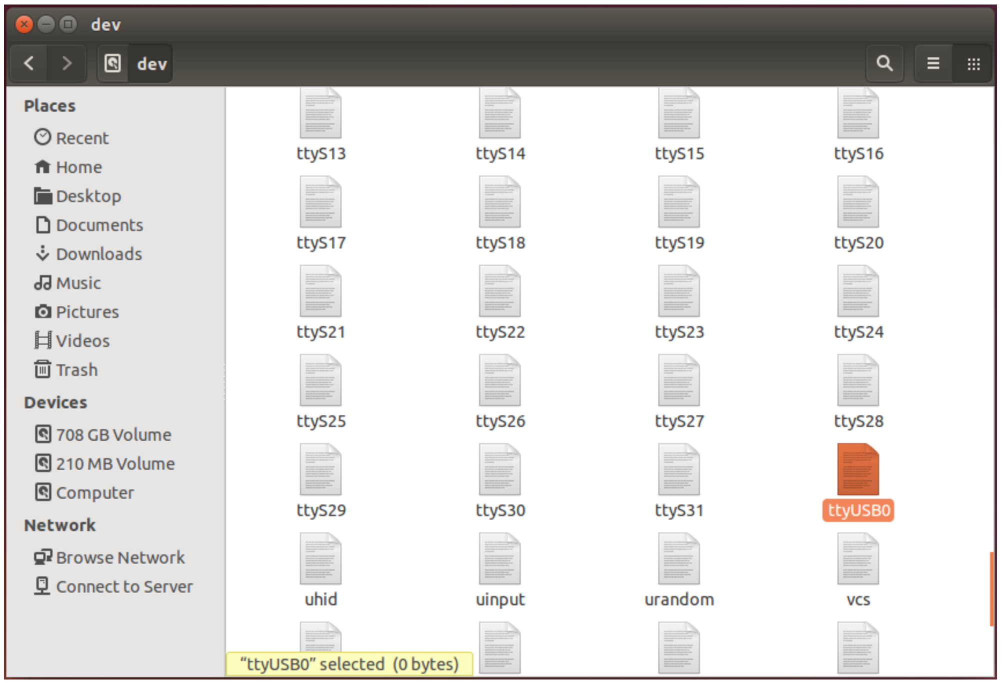
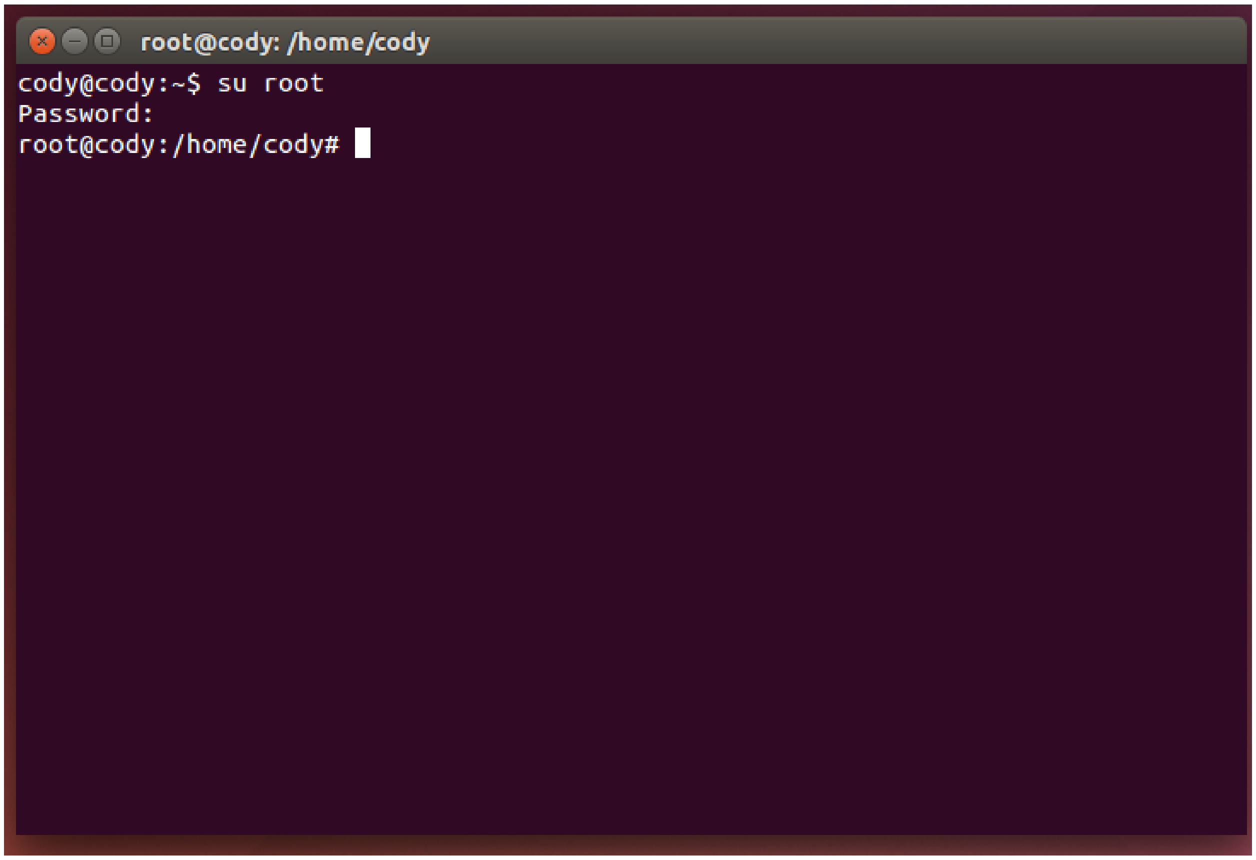


Figure 2

Reading the Output

The receiver is a serial character device, meaning information is sent out character by character with a start and stop character to signify the beginning and end of a line respectively. To begin reading data from the receiver you will need root access to the system. Root access gives you full control over the system. If the account you are currently logged into has root permissions you can skip ahead to the next step. Otherwise, open a terminal window and type “su root”, then hit enter. Now enter your root password, as seen in Figure 3. The “su” command stands for switch user.



```
cody@cody:~$ su root
Password:
root@cody:/home/cody#
```

Figure 3

Now that you have root permissions, enter the command:

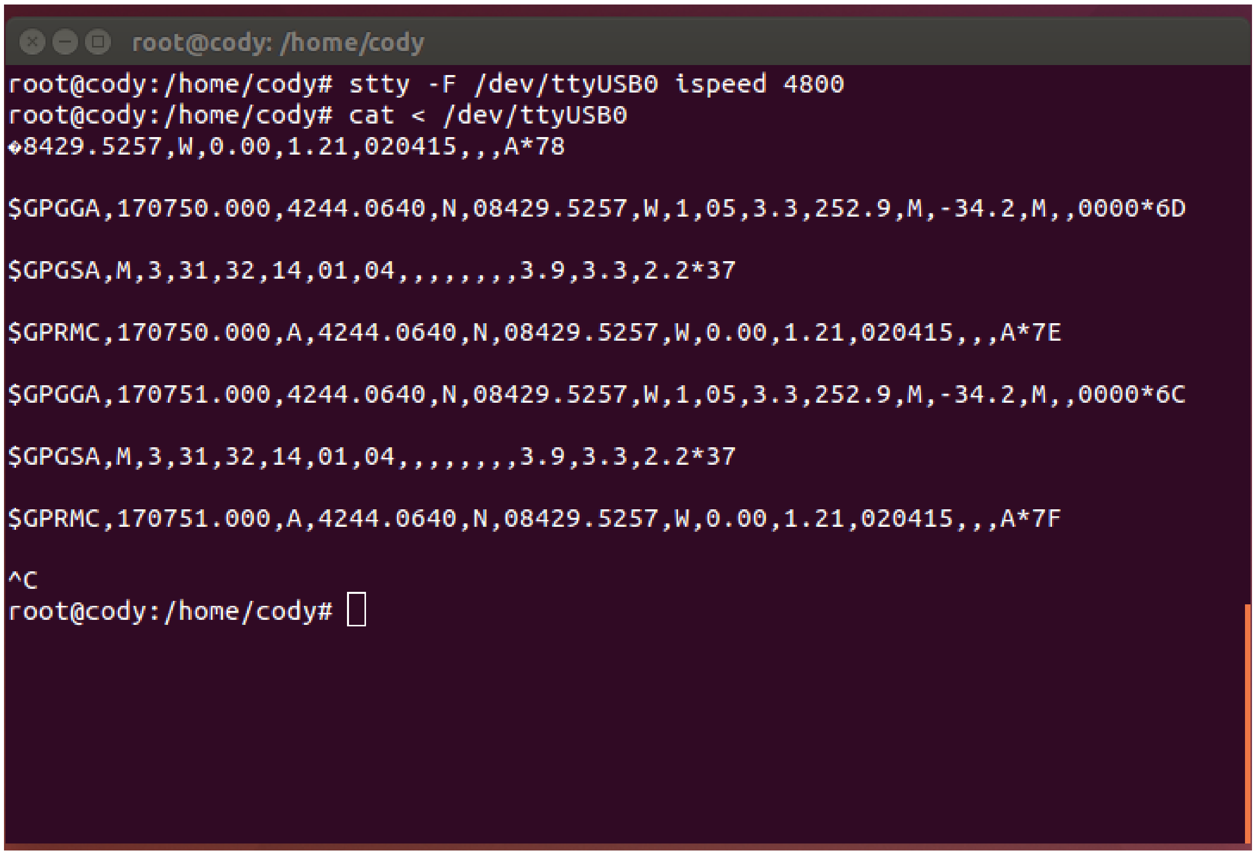
“stty -F /dev/ttyUSB0 ispeed 4800”

If your device was named something other than “ttyUSB0”, use that name in the above command. This command sets the access rate of the USB port on the system to match the transfer rate of the receiver at 4800.

Next, to see what the receiver is outputting, enter the command:

“cat < /dev/ttyUSB0”

Replacing ttyUSB0 with your device name should it be different. Now what you are seeing in the terminal is the output stream of the receiver. To stop the output, press “ctl+C” on your keyboard. This terminated the process. Your terminal should look similar to Figure 4.



```
root@cody:/home/cody# stty -F /dev/ttyUSB0 ispeed 4800
root@cody:/home/cody# cat < /dev/ttyUSB0
�8429.5257,W,0.00,1.21,020415,,,A*78

$GPGGA,170750.000,4244.0640,N,08429.5257,W,1,05,3.3,252.9,M,-34.2,M,,0000*6D

$GPGSA,M,3,31,32,14,01,04,,,,,,,,3.9,3.3,2.2*37

$GPRMC,170750.000,A,4244.0640,N,08429.5257,W,0.00,1.21,020415,,,A*7E

$GPGGA,170751.000,4244.0640,N,08429.5257,W,1,05,3.3,252.9,M,-34.2,M,,0000*6C

$GPGSA,M,3,31,32,14,01,04,,,,,,,,3.9,3.3,2.2*37

$GPRMC,170751.000,A,4244.0640,N,08429.5257,W,0.00,1.21,020415,,,A*7F

^C
root@cody:/home/cody# 
```

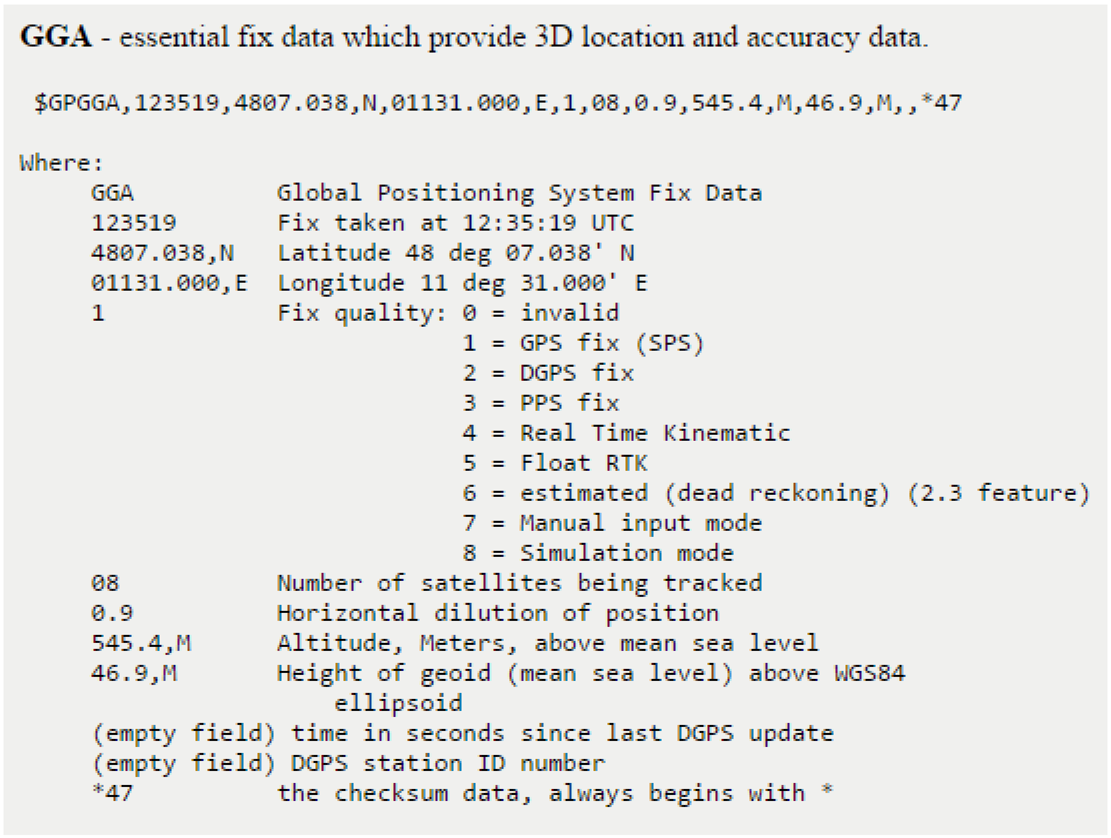
Figure 4

Notice the first line contains an odd character at the beginning. This is because the system started reading mid data stream and the incoming bits were not aligned to properly. When the end of line character is read the system properly aligns and begins outputting readable data.

Notice the variations of the lines. These are the different formats the receiver outputs. It cycles through them continuously. We are going to focus on the lines that begin with GPGGA, as this line contains the information we need.

Extracting the Location Data

As can be seen each line begins with the name of the standard (GPGGA) followed by the location data separated by commas. The GGA information can be seen below in Figure 5.



GGA - essential fix data which provide 3D location and accuracy data.

```
 $GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

Where:
     GGA          Global Positioning System Fix Data
     123519       Fix taken at 12:35:19 UTC
     4807.038,N   Latitude 48 deg 07.038' N
     01131.000,E  Longitude 11 deg 31.000' E
     1            Fix quality: 0 = invalid
                               1 = GPS fix (SPS)
                               2 = DGPS fix
                               3 = PPS fix
                               4 = Real Time Kinematic
                               5 = Float RTK
                               6 = estimated (dead reckoning) (2.3 feature)
                               7 = Manual input mode
                               8 = Simulation mode
     08           Number of satellites being tracked
     0.9          Horizontal dilution of position
     545.4,M      Altitude, Meters, above mean sea level
     46.9,M       Height of geoid (mean sea level) above WGS84
                      ellipsoid
     (empty field) time in seconds since last DGPS update
     (empty field) DGPS station ID number
     *47          the checksum data, always begins with *
```

Figure 5

The information we need are the longitude and latitude. This information can be read, parsed and formatted by a python script to be use in further applications.

To run a python script you will need to have python on your system. Python 2.7 is used in this document. If you do not have python open a terminal window and enter:

“apt-get install python2.7”

Follow the instructions and wait for python to finish downloading and installing.

You will also need a text editing program to write the python file. This document uses VIM, if you would like to use VIM and do not have it, open a terminal window and enter:

“apt-get install vim”

Follow the instructions and wait for VIM to download and install.

To begin writing the python file open a terminal and enter:

“VIM gps.py”

This will create and open a new file called “gps.py” and this will be where code will be written. The first line of the code should be “import serial”. “serial” is a python package for reading serial ports, adding this line allows that package to be used. The next line of code will be “ser = serial.Serial(‘/dev/ttyUSB0’ , 4800, timeout = 5)”

This line creates a serial variable called “ser” and sets the port it will read from and the read speed of that port. The timeout of 5 seconds prevents the program from getting stuck in an infinite loop trying to read the port.

Next we will create a while loop. Within the while loop we will add two lines:

“line = ser.readline()”

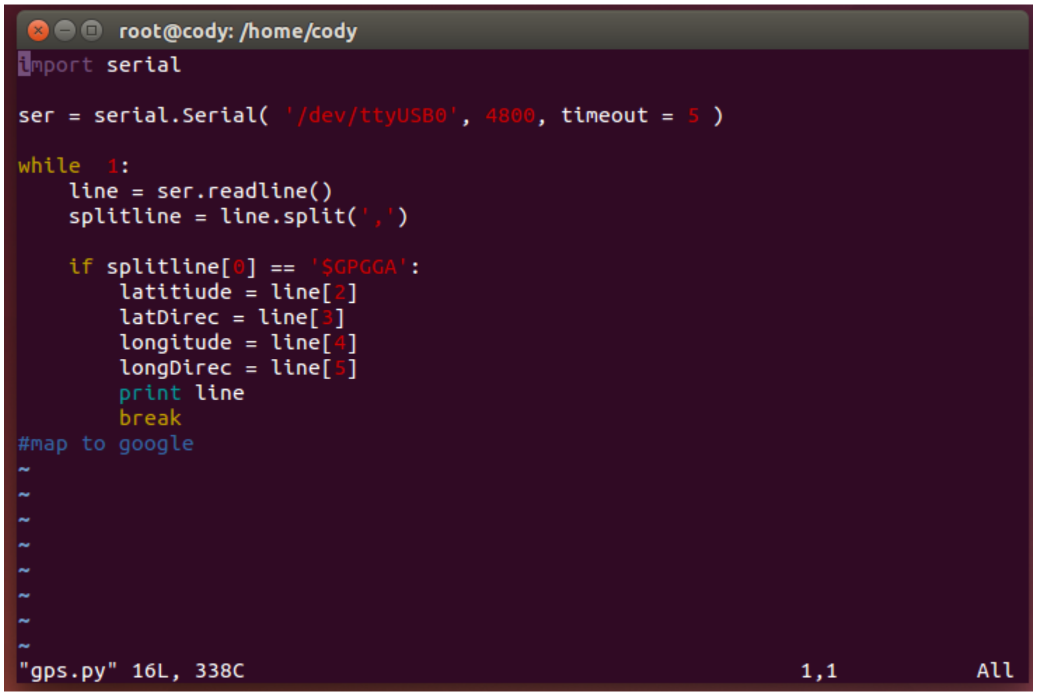
“splitline = line.split(‘,’)”

What this will do is read the receiver output in a loop and split the output by the commas separating the data.

Then after the last line and still in the while loop, add the line:

“if splitline[0] == ‘$GPGGA’:”

This if statement will check to see if the line begins with $GPGGA, which is the line that contains the useful information. If so, the information in that line can be formatted in a useful way. The whole of the code can be seen in Figure 6.



```
root@cody: /home/cody
import serial

ser = serial.Serial( '/dev/ttyUSB0', 4800, timeout = 5 )

while  1:
    line = ser.readline()
    splitline = line.split(',')

    if splitline[0] == '$GPGGA':
        latitiude = line[2]
        latDirec = line[3]
        longitude = line[4]
        longDirec = line[5]
        print line
        break
#map to google
~
~
~
~
~
~
~
~
"gps.py" 16L, 338C                                            1,1           All
```

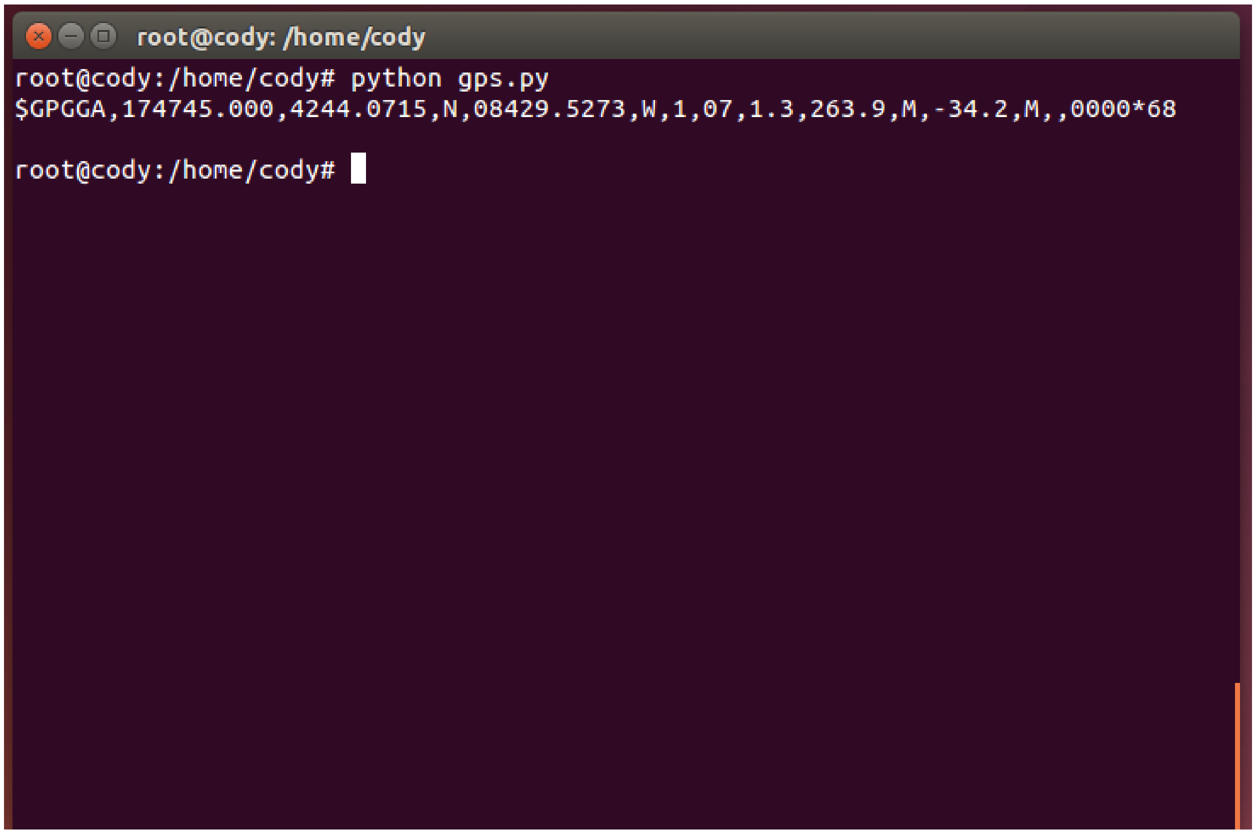
Figure 6

Within the If statement is where the useful information can be extracted and formatted. As an example a few lines were added to the code above which parse the line and save key information to variables such as longitude, latitude, and the directional information. Then the program prints the line and breaks the loop to stop running.

To run this code, make sure it is saved and your GPS is plugged in to your system. Then in a terminal window enter the directory that your file is contained in and enter the command:

“python gps.py”

This code will run and it will print out the line starting with $GPGGA. The output of the program can be seen in Figure 7.



```
root@cody:/home/cody# python gps.py
$GPGGA,174745.000,4244.0715,N,08429.5273,W,1,07,1.3,263.9,M,-34.2,M,,0000*68

root@cody:/home/cody#
```

Figure 7

The program can be manipulated to suit almost any need. Whether it be mapping, tracking or anything else you need GPS coordinates for.

Further Reading

"What Is a GPS? How Does It Work?" *What Is GPS? Everyday Mysteries*. Web. 2 Apr. 2015. <http://www.loc.gov/rr/scitech/mysteries/global.html>.

"GPS Accuracy." *GPS.gov:*. Web. 2 Apr. 2015. <http://www.gps.gov/systems/gps/performance/accuracy/>.

"NMEA Data." *NMEA Data*. Web. 2 Apr. 2015. <http://www.gpsinformation.org/dale/nmea.htm#GGA>.

"BU-353-S4." *USGlobalsat Corporate*. Web. 2 Apr. 2015. <http://usglobalsat.com/p-688-bu-353-s4.aspx#images/product/large/688.jpg>.