

1/15/17

Added the routine to the *load_usb_port.py*: `os.system('lsusb> usb.txt')`
gps-343.py imports the *usbnum* from the *load_usb_ports.py* . If *usbnum*>4 then keyboard is plugged in.
Otherwise, with no keyboard, the data is stored with the label Keyboard not connected.

The *load_usb_port.py* program lists the active USB ports redirected to the file *usb.txt*. It then loads this file into the python program and counts the lines, which represent each USB port in use. This number is the variable *usbnum*, which is imported into the main program.

1/13/17

Added under the capture data, sense if keyboard is present.
Usbnum.sh, This file makes a text file *usb.txt* showing all active usb devices
Load_usb_ports.py counts the number of USB ports from the *usb.txt* file and saves as *usbnum*
gps-343.py imports the *usbnum*. If *usbnum*>5 then keyboard is plugged in.

Must run the *usbnum.sh* script on startup. It is added to */etc/profile/*

1\

1/11/17

Version 342 added receiver lock and unlock to data to the *logfile.txt*

1/8/17






Version 341 Added flashing LED when latitude is > 0. This version uses the same Raspberry Pi pin, pin 11 that is used for the Enter Data routine.

1/5/17

Added flashing LED when latitude is > 0. T

1/3/17

Image files saved in *Documents/RaspberryPi/RaspPi_images*.

Name	Date modified	Type
 gps338.img	1/3/2017 6:35 PM	Disc Image File
 old082216	8/22/2016 3:25 PM	File
 Rasp2B082316	8/23/2016 8:07 AM	File
 raspSD	8/11/2016 10:57 AM	Disc Image File
 raspSD-8g-Old	8/11/2016 3:56 PM	File

Made an image file of the 8 gig, speed 4, Raspberry Pi 2 called gps338. Wrote this image to a 16 gig speed 10 SD chip for the Raspberry Pi 2.

Wrote the image Rasp2B082316 to a 16 gig speed 10 SD chip for the Raspberry Pi 2B. Need to add new versions of the gps program

Note: The first time I copied the image file to the SD card it took 3 hours. This crashed. It may have been caused by the WiPi. The second time I copied the image it took half an hour or less!

To make the program run on startup of the Raspberry pi:

In the **home/pi/.config** folder, create a new folder **autostart**.

In this folder, create the file **.desktop** with these three lines:

[Desktop Entry]

Type = Application

Exec = lxterminal -e "sudo python /home/pi/gps_2017/gps340.py"

(/home/pi/.config/autostart/.desktop)

This works! It auto starts the program in a terminal so it displays what is going on . It will not respond to a **ctrl-z** or a **ctrl-x** command. To stop the program, use **ctrl-alt-del** to launch the **Task Manager**, select lxterminal, right click on kill and click on yes.

Version 339, when launched on auto start writes to the datafile.txt in the /home/pi folder.

When launched manually it writes to the file in the /home/pi/gps_2017.

When I cat the folder in the gps_2017 folder it displays 9186, on the command line.

ie: **pi@raspberrypi ~/gps_2017 \$ 9186,**

The following works to autostart and open lxterminal. However the terminal close as the program ends. This may be OK.I must try this on actual program.

<http://raspberrypi.stackexchange.com/questions/8805/auto-login-into-lxde-and-auto-start->

[video-player-omxplayer](#)

The program that that used wrote 0-4 to a textfile.
I tried to make the file executable, but it will not run!
Here is the file test1.py

```
import time  
file = open('testfile.txt','a')  
x=0  
while x<5  
    print x  
    file.write(str(x))  
    time.sleep(1)  
    x=x+1  
print 'done!'  
file.close()
```

- 1- created an autostart folder in **/home/pi/.config/autostart**
- 2- In this folder I created a file: **.desktop** containing:

```
[Desktop]  
Type = Application  
Exec =lxterminal -e "python test1.py"
```

When I go to the **/home/pi/.config/autostart** folder and do an ls, it displays:
LXinput-setup.desktop

I would expect it to display **.desktop**

12/31/16 Version gps-339.py adds the. omxplayer command to force audio to analog output -o local

12/22/16 The issue with version 335 is that when the system boots up it does not run the program in a linux terminal. Therefore there is no keyboard assigned to the program!!

12/21/16 gps-335.py has the features needed to log the latitude and longitude data and add the location name to the **datafile.txt** file which can be opened with a spreadsheet.

Since the GPIO ports are being used, the program must be launched as root:

```
sudo python gps-335.py
```

12/19/16 Added data capture section in *gps-331.py* .

Since there is no ON KEY() function in Python, the Raspberry GPIO ports are being used.

When the capture button is pressed, the latitude and longitude data is saved to the *datafile.txt* file. To be added: time, location, text, capture multiple times etc. Also carriage return.

12/15/16 Updated the program to include gain control for each audio file.

The latest version is *gps-321.py* . All files are in the */home/pi/gps_2017* folder.

Added new variables in the spreadsheet *nb_gain[]* and *sb_gain[]*.

Added new variables to *load_csv_data.py*. Also added the variable *gain[]*.

This is a dummy variable used to change from northbound to southbound in main program.

The *logfile.txt* is in the */home/pi* directory.

There is a leapfrog file in the */home/pi/gps_2017* folder with the name *Changes* that does not copy to the flash drive??

I edited a version of *gps-321.py* in the laptop, using Idle. It would not run in the Raspberry Pi??

Note that the time is EDST.

12/2/16

iUsing */ect/rc.local* instead of */etc/profile* solves the echo problem in the Pi 2B unit

11/28/16

oplayer's auto-detection of the correct audio output device fails, you can force output over hdmi with:

```
omxplayer -o hdmi example.mp3
```

or you can force output over the headphone jack with:

```
omxplayer -o local example.mp3
```

11/23/16 The problem with the Raspberry Pi 2B playing echos is an omxplayer problem. When using pygame, the problems no longer exists. NO NO There is still a problem with the audio when played in autostart mode! The effect is much less with pygame but there is still a problem!

One of the problems with pygame was that the volume was much lower than omxplayer.

From the command line, enter the folowing: *mixer set PCM -- 100%*

100% is 4db
95% is -1.3db
93% is -3.3db
91% is -5.6db
85% is -12db

11/15/16

Adding volume control to each clip.

Modified play_mp3s_nb.py

The system does not read the line:

os.system("omxplayer --vol nb_gain[x] %s" % track)

It only works if the volume number is a real number and not a variable:

os.system("omxplayer --vol -3000 %s" % track)

10/31

Tried

<https://neverbenever.wordpress.com/2015/02/11/how-to-autostart-a-program-in-raspberry-pi-or-linux/>

First Way

This launches the terminal but not the GPS program

Second Way does not work at all. Although the ./autostart command does launch the program manually!

Tried to autostart in terminal from

<https://www.raspberrypi.org/forums/viewtopic.php?t=17051&p=170858>

It added an autostart directory to the /home/pi/.config directory.

I commented the line in the /etc/profile file .

It does auto-start, but not in a terminal window??

10/3/16 Adding volume control to system.
track is the name of the mp3 file

```
str(x) is the volume level -3000 to 1500  
os.system("omxplayer --vol "+str(x) + " %s " % track)
```

Note spaces after *--vol* and before *%s*

For version *gps-321.py* and above.

Add column 5 or E in the *gps_data.csv*.

Make the default value 600

Change in *gps-321.py* line 137

Change *play_mp3s_nb.py* lines 21, 23

Change *play_mp3s_sb.py*

Change *load_csv_data.py* lines 8, 17, 27,

Add *adjust_volume.py*

9/20/16

Latest version is *gps-312.py*

Files needed in the *home\pi* directory:

gps-312.py

load_csv_data.py

gps_data.csv

The mp3 audio files must be stored in the *home\pi\audio_tracks* directory.

The *gps_data.csv* file is made from an Excel file exported or saved as if you are using

Libre Office from a spreadsheet that looks like Figure 1.

Test for 9-23-18 [Compatibility Mode] - Microsoft Excel Starter

	A	B	C	D
1	35.661544500	motor_car_house	motor_car_house.mp3	motor_car_house.mp3
2	35.663549000	bonsal_crossing	bonsal_crossing.mp3	bonsal_crossing.mp3
3	35.665534305	location_3	location_3_north.mp3	location_3_south.mp3
4	35.667519610	path_1	path_1.mp3	path_1.mp3
5	35.668666950	mile_post_1	mile_post_1.mp3	mile_post_1.mp3
6	35.670630054	location_7	location_7_north.mp3	location_7_south.mp3
7	35.672593159	location_8	location_8_north.mp3	location_8_south.mp3
8	35.674556263	location_9	location_9_north.mp3	location_9_south.mp3
9	35.676519367	path_2	path_2.mp3	path_2.mp3
10	35.680130208	mile_post_2	mile_post_2.mp3	mile_post_2.mp3
11	35.681215228	horton_road	horton_road.mp3	horton_road.mp3
12	35.683864309	stairway	stairway.mp3	stairway.mp3
13	35.685526515	path_3_calc	path_3_calc.mp3	path_3_calc.mp3
14	35.687756114	bridge	bridge.mp3	bridge.mp3
15	35.690047684	mile_post_3	mile_post_3.mp3	mile_post_3.mp3
16	35.691551587	location_16	location_16_north.mp3	location_16_south.mp3
17	35.693055491	location_17	location_17_north.mp3	location_17_south.mp3
18	35.694559394	location_18	location_18_north.mp3	location_18_south.mp3
19	35.696063297	path_4	path_4.mp3	path_4.mp3
20	35.697294414	new_hill_yard_marker	new_hill_yard_marker.mp3	new_hill_yard_marker.mp3
21	35.697669272	new_hill_south_switch	new_hill_south_switch.mp3	new_hill_south_switch.mp3
22	35.699658911	new_hill_whistle_marker	new_hill_whistle_marker.mp3	new_hill_whistle_marker.mp3
23				
24				

Figure 1 The *gps-data.xls* file

This EXEL file has four columns, A, B, C and D. These columns contain the latitude data, the location name, the audio file name played going northbound and the audio file name played going southbound.

The *load_csv_data.py* file loads the *gps_data.csv* data

```
1 35.661544500,motor_car_house,motor_car_house.mp3,motor_car_house.mp3
2 35.663549000,bonsal_crossing,bonsal_crossing.mp3,bonsal_crossing.mp3
3 35.665534305,location_3,location_3_north.mp3,location_3_south.mp3
4 35.667519610,path_1,path_1.mp3,path_1.mp3
5 35.668666950,mile_post_1,mile_post_1.mp3,mile_post_1.mp3
6 35.670630054,location_7,location_7_north.mp3,location_7_south.mp3
7 35.672593159,location_8,location_8_north.mp3,location_8_south.mp3
8 35.674556263,location_9,location_9_north.mp3,location_9_south.mp3
9 35.676519367,path_2,path_2.mp3,path_2.mp3
10 35.680130208,mile_post_2,mile_post_2.mp3,mile_post_2.mp3
11 35.681215228,horton_road,horton_road.mp3,horton_road.mp3
12 35.683864309,stairway,stairway.mp3,stairway.mp3
13 35.685526515,path_3_calc,path_3_calc.mp3,path_3_calc.mp3
14 35.687756114,bridge,bridge.mp3,bridge.mp3
15 35.690047684,mile_post_3,mile_post_3.mp3,mile_post_3.mp3
16 35.691551587,location_16,location_16_north.mp3,location_16_south.mp3
17 35.693055491,location_17,location_17_north.mp3,location_17_south.mp3
18 35.694559394,location_18,location_18_north.mp3,location_18_south.mp3
19 35.696063297,path_4,path_4.mp3,path_4.mp3
20 35.697294414,new_hill_yard_marker,new_hill_yard_marker.mp3,new_hill_yard_marker.mp3
21 35.697669272,new_hill_south_switch,new_hill_south_switch.mp3,new_hill_south_switch.mp3
22 35.699658911,new_hill_whistle_marker,new_hill_whistle_marker.mp3,new_hill_whistle_marker.mp3
23
```

Figure 2 The *gps_data.csv* file

There are two utilities also in the `\home\pi` directory that will play all of the audio clips:

play_mp3s_nb.py

play_mp3s_sb.py

The spreadsheet file should also be in this directory.

When changing location data, the old data can be saved in sub-directories. For example there is a `\gps` sub-directory and a `\fearr` sub-directory where the data is stored.

The 300 series changed the gps data from four text files to one .csv files.

The system creates a logfile.txt which records the date and time and file name of the audio files it plays

The correct date is still a problem with the model 2.

Although it did show the correct date once. May be it was caused by running cgps program before launching the gps program.

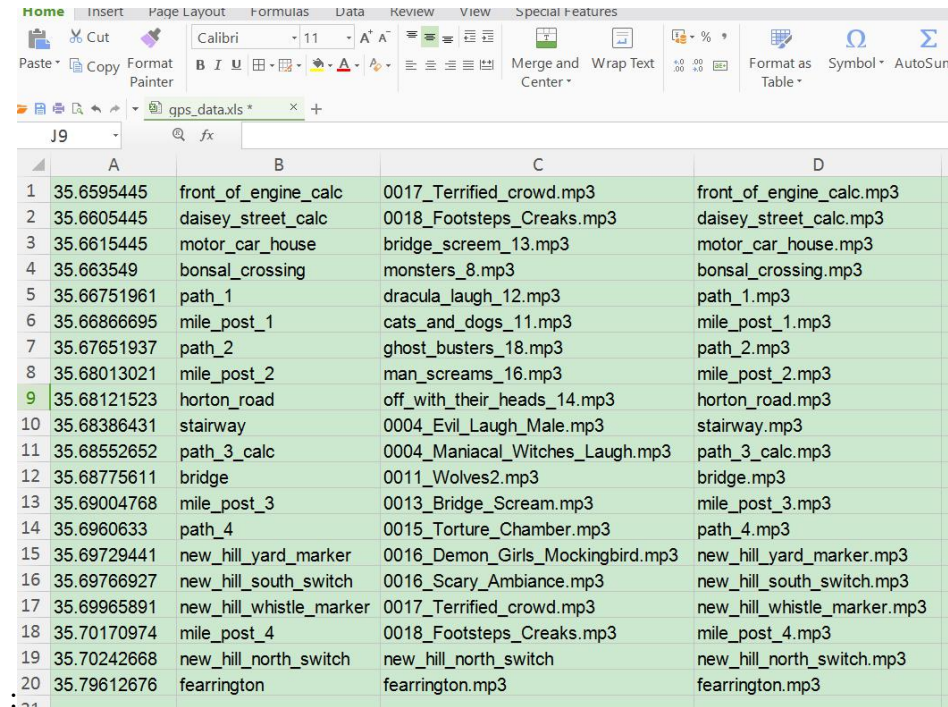
9/8/16 New file for loading gps data from spreadsheet. It is located in the `c:\python27\mystuff\csv directory` in the windows machine.

The file is `load_gps_data.py`. Replace existing line in gps file with: **from load_gps_data import***

The spreadsheet should have the data in columns A, B,C and D

Where column A is the latitude, column B is the location, column C is the northbound mp3 file and column D is the southbound mp3 file.

`gps_data.xls` file:



	A	B	C	D
1	35.6595445	front_of_engine_calc	0017_Terrified_crowd.mp3	front_of_engine_calc.mp3
2	35.6605445	daisy_street_calc	0018_Footsteps_Creaks.mp3	daisy_street_calc.mp3
3	35.6615445	motor_car_house	bridge_scream_13.mp3	motor_car_house.mp3
4	35.663549	bonsal_crossing	monsters_8.mp3	bonsal_crossing.mp3
5	35.66751961	path_1	dracula_laugh_12.mp3	path_1.mp3
6	35.66866695	mile_post_1	cats_and_dogs_11.mp3	mile_post_1.mp3
7	35.67651937	path_2	ghost_busters_18.mp3	path_2.mp3
8	35.68013021	mile_post_2	man_screams_16.mp3	mile_post_2.mp3
9	35.68121523	horton_road	off_with_their_heads_14.mp3	horton_road.mp3
10	35.68386431	stairway	0004_Evil_Laugh_Male.mp3	stairway.mp3
11	35.68552652	path_3_calc	0004_Maniacal_Witches_Laugh.mp3	path_3_calc.mp3
12	35.68775611	bridge	0011_Wolves2.mp3	bridge.mp3
13	35.69004768	mile_post_3	0013_Bridge_Scream.mp3	mile_post_3.mp3
14	35.6960633	path_4	0015_Torture_Chamber.mp3	path_4.mp3
15	35.69729441	new_hill_yard_marker	0016_Demon_Girls_Mockingbird.mp3	new_hill_yard_marker.mp3
16	35.69766927	new_hill_south_switch	0016_Scary_Ambiance.mp3	new_hill_south_switch.mp3
17	35.69965891	new_hill_whistle_marker	0017_Terrified_crowd.mp3	new_hill_whistle_marker.mp3
18	35.70170974	mile_post_4	0018_Footsteps_Creaks.mp3	mile_post_4.mp3
19	35.70242668	new_hill_north_switch	new_hill_north_switch	new_hill_north_switch.mp3
20	35.79612676	fearrington	fearrington.mp3	fearrington.mp3

`gps_data.csv` file:

```
35.6595445,front_of_engine_calc,0017_Terrified_crowd.mp3,front_of_engine_calc.mp3
35.6605445,daisy_street_calc,0018_Footsteps_Creaks.mp3,daisy_street_calc.mp3
35.6615445,motor_car_house,bridge_scream_13.mp3,motor_car_house.mp3
35.663549,bonsal_crossing,monsters_8.mp3,bonsal_crossing.mp3
35.66751961,path_1,dracula_laugh_12.mp3,path_1.mp3
35.66866695,mile_post_1,cats_and_dogs_11.mp3,mile_post_1.mp3
35.67651937,path_2,ghost_busters_18.mp3,path_2.mp3
35.68013021,mile_post_2,man_screams_16.mp3,mile_post_2.mp3
35.68121523,horton_road,off_with_their_heads_14.mp3,horton_road.mp3
35.68386431,stairway,0004_Evil_Laugh_Male.mp3,stairway.mp3
35.68552652,path_3_calc,0004_Maniacal_Witches_Laugh.mp3,path_3_calc.mp3
35.68775611,bridge,0011_wolves2.mp3,bridge.mp3
35.69004768,mile_post_3,0013_Bridge_scream.mp3,mile_post_3.mp3
35.6960633,path_4,0015_Torture_Chamber.mp3,path_4.mp3
35.69729441,new_hill_yard_marker,0016_Demon_Girls_Mockingbird.mp3,new_hill_yard_marker.mp3
35.69766927,new_hill_south_switch,0016_Scary_Ambiance.mp3,new_hill_south_switch.mp3
35.69965891,new_hill_whistle_marker,0017_Terrified_crowd.mp3,new_hill_whistle_marker.mp3
35.70170974,mile_post_4,0018_Footsteps_Creaks.mp3,mile_post_4.mp3
35.70242668,new_hill_north_switch,new_hill_north_switch,new_hill_north_switch.mp3
35.79612676,fearrington,fearrington.mp3,fearrington.mp3
```

9/6/16 Setting up a spreadsheet to load gps files.
CSV.py loads the lats[], locs[], mp3s_nb[] and the np3s_sb[] lists from a .csv file.
Needs cleanup and renaming of files but it works. This eliminates the copy and paste from spreadsheet to text files.

Enter data in spreadsheet
Export to .csv file.
Program does the rest

<http://www.pythonforbeginners.com/systems-programming/using-the-csv-module-in-python/>

The CSV module contains the following functions:

```
csv.reader  
csv.writer  
csv.register_dialect  
csv.unregister_dialect  
csv.get_dialect  
csv.list_dialects  
csv.field_size_limit
```

<https://newcircle.com/s/post/1572/python-for-beginners-reading-and-manipulating-csv-files>

8/30/16 Change path_3_calc latitude
Original was : 35.685810211500
It was change last week to: 35.6855xxx
Now it is: 35.68524282

8/28/16 Changed the delay from 1 sec to 5 sec. It seems to cure issue.
Also added the version number to the file logfile.txt
Latest version is *gps-105.py*.

8/27/16 Checked version 104 on ride day.
The 2B will not play audio when powered by the inverter.
It played the audio clips twice with echo

8/24/16

Created a simulation program *gps-simulation.py* based on gps-01.py to test resetting

the latitude from 45 to 35 if difference is $>\text{delta} * 3$. This seems to work so it was implemented in `gps-02.py` which will be tested on 8/26

8/23/16

Created `gps-102.py` to add a separate field for the sound track. It uses `load_gps_files5.py`. This has not been implemented as of this writing.

8/22/16

Added `load_gps_files5.py` to load `mp3s[]` which stores the `.mp3` files. Changed the `logfile.txt` to show the `mp3` file name, but the display does not look nice. Need to add a CR to the `file.write()` command.

8/22/16

For the Raspberry Pi 2B the `/etc/profile` was edited and the following two lines were added:

```
/home/pi/gpssock.sh &  
Python /home/pi/gps-01.py
```

To eliminate the echo.

8/19/16

Destroyed the NOOBs on both units. Had to use the image files stored in `/Documents/RaspPi_images`. Made an update image for the model 2 `old082216.img`

8/12/16 In the model 2B, the track plays twice, creating an echo and the log shows it was recorded twice when I edited the `/etc/profile` by adding

```
./gps.sh
```

8/11/16

MUST ADD `&` to the `/etc/profile` line ie: `sudo python /home/pi/gps-95.py &`

In the autorun setup

However the `gps` program is running in the background. We cannot see it!!!

8/9/16

<http://www.raspberrypi-spy.co.uk/2015/02/how-to-autorun-a-python-script-on-raspberry-pi-boot/>

How To Autorun A Python Script On Raspberry Pi Boot

9

By [Matt](#) on February 5, 2015 [Python](#), [Raspbian](#)



There are lots of techniques for running a script when the Pi boots and which one you choose will depend on exactly what the script does and what you expect. In this post I'll explain a technique where the Pi automatically logs in as the Pi user and immediately executes a Python script.

This has one major advantage over another popular method (see [Running A Python Script At Boot Using Cron](#)) in that because the terminal is up and running text output from the script is visible before you are returned to a usable command line prompt.

Auto Login Setup (optional)

The first step is to enable the Pi to login automatically without requiring any user intervention. This step is optional.

At the command prompt or in a terminal window type :

```
sudo nano /etc/inittab
```

followed by Enter. Find the line :

```
1:2345:respawn:/sbin/getty 115200 tty1
```

and add a # character to the beginning of the line to disable it so it looks like :

```
#1:2345:respawn:/sbin/getty 115200 tty1
```

Under the line add the following :

```
1:2345:respawn:/bin/login -f pi tty1 </dev/tty1 >/dev/tty1 2>&1
```

where “pi” is the username.

Type “Ctrl+X” to exit, then “Y” to save followed by “Enter” twice.

Prepare Script

My test script is called “[myscript.py](#)” and is located in /home/pi/. This is what it contains :

```
#!/usr/bin/python
1print "*****"
2print "* This is a test script. There are many like it,          *"
3print "* but this one is mine. My script is my best friend.   *"
4print "* It is my life. I must master it as I must master     *"
5print "* my
6life.
7          *"
print "*****"
```

You can download this directly to your Pi by using the following command :

```
wget https://bitbucket.org/MattHawkinsUK/rpispymisc/raw/master/python/
myscript.py
```

Auto-run Script Setup

Now we need to tell the operating system to run the script for the Pi user. In the command prompt or in a terminal window type :

```
sudo nano /etc/profile
```

Scroll to the bottom and add the following line :

```
sudo python /home/pi/myscript.py
```

where “/home/pi/myscript.py” is the path to your script.

Type “Ctrl+X” to exit, then “Y” to save followed by “Enter” twice.

Reboot and Test

To test if this has worked reboot your Pi using :

```
sudo reboot
```

When it starts up your script will run and you will see something like this :

```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Feb  5 19:28:57 2015
*****
* This is a test script. There are many like it,      *
* but this one is mine. My script is my best friend. *
* It is my life. I must master it as I must master  *
* my life.                                           *
*****
pi@raspberrypi ~ $
```

Due to the technique we've used the script is run whenever the Pi user logs in. This means if you create other terminal sessions (via SSH for example) the script will run each time.

Troubleshooting

If it doesn't work here are some things to try :

- Run your script manually and check it works correctly
- Use my example script and check that works
- Double check the initial steps

8/8/16

THIS DESTROYED THE gpsfiles

NameError: golbal name 'gps' is not defined in:

Gpsd = gps(

Running A Python Script At Boot Using Cron

[42](#)

By [Matt](#) on July 27, 2013 [Python](#), [Raspbian](#)

```

# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
@reboot python /home/pi/MyScript.py &

```



There may be times when you want to run a Python script when your Raspberry Pi boots up. There are a number of different techniques to do this but I prefer the method that uses “cron”.

Cron is a job scheduler that allows the system to perform tasks at defined times or intervals. It is a very powerful tool and useful in lots of situations. You can use it to run commands or in this case, a Python script.

Step 1 – Create A Python Script

The first step is creating your Python script. This will be the script that will run at boot time. It is important to remember its name and location. In this example I will assume the script is called “MyScript.py” and it is located in “/home/pi”.

Double check you’ve got the correct path by typing :

```
cat /home/pi/MyScript.py
```

This should show the contents of your script.

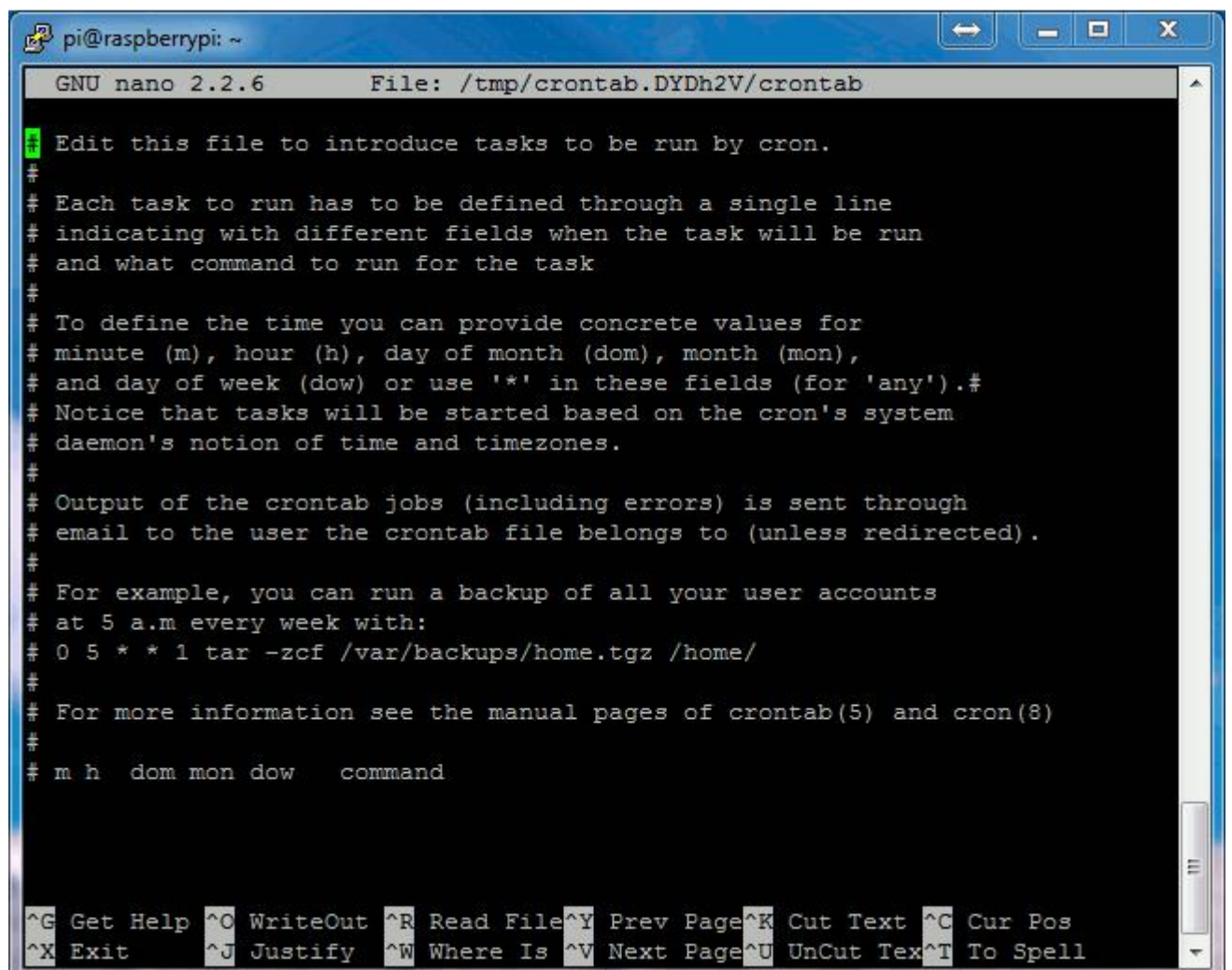
Make sure your script works and does what you expect it to. Once you are running at boot it isn't so easy to debug so don't rush!

Step 2 – Add A New Cron Job

To create a new job to Cron we will modify the “crontab”. This is a table that contains the list of jobs that Cron will monitor and run according to it's details. To edit it we use the command :

```
sudo crontab -e
```

Each user of the system (ie “pi”) can have its own Crontab but in this case we want to add it as an admin so we prefix our “crontab -e” command with “sudo”. You should see something that looks like this :



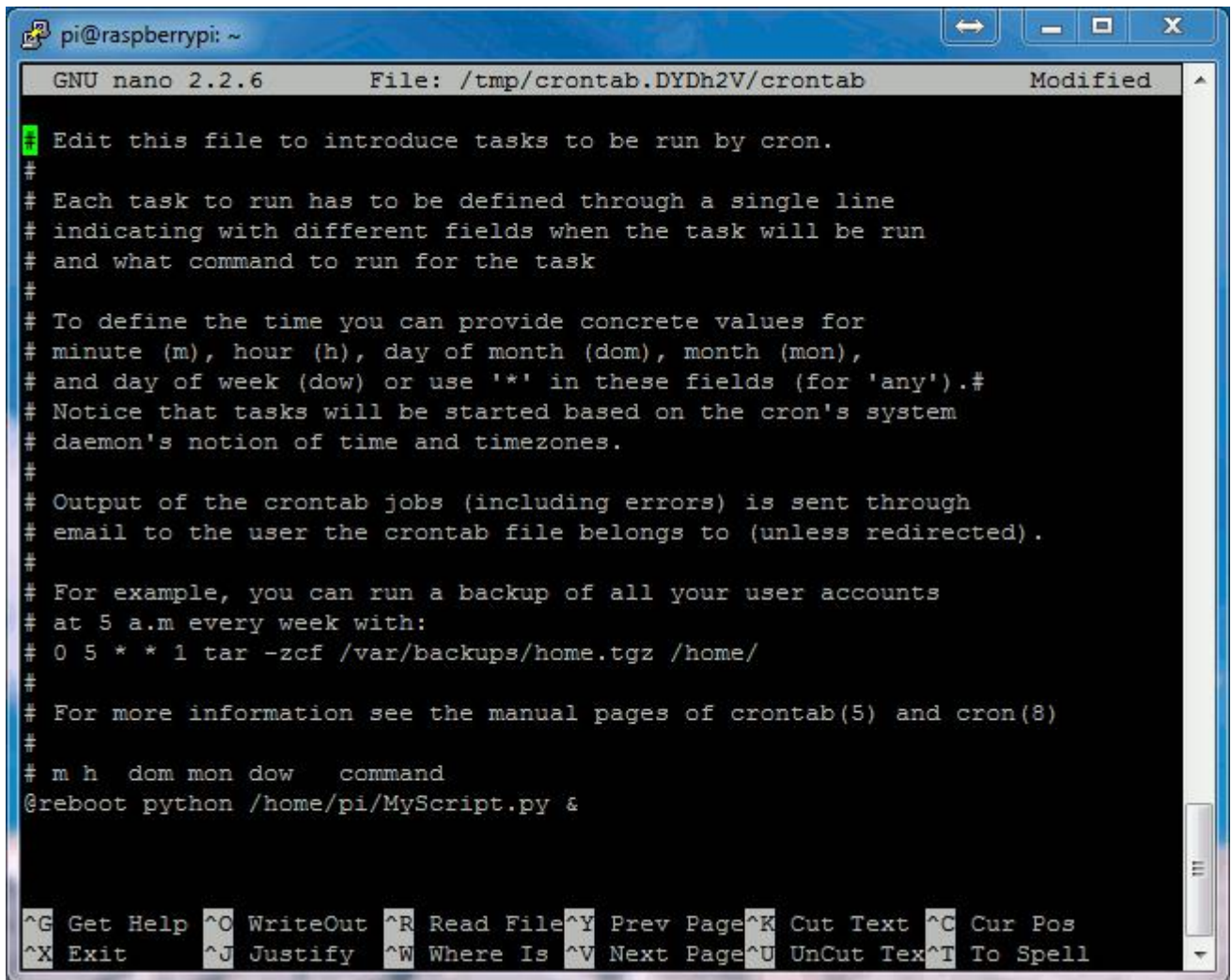
```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /tmp/crontab.DYDh2V/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Tex ^T To Spell
```

Using your cursor keys scroll to the bottom and add the following line :


```
@reboot python /home/pi/MyScript.py &
```

This tells Cron that every boot (or reboot or start-up) we want to run Python with the script MyScript.py. The “&” at the end of the line means the command is run in the background and it won’t stop the system booting up as before.

Your screen should look something like this :



```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /tmp/crontab.DYDh2V/crontab Modified
Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
@reboot python /home/pi/MyScript.py &

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Tex ^T To Spell
```

To save these changes click “CTRL-X”, then “Y” and finally “Return”. You should now be back at the command prompt.

To start testing you can now reboot using :

```
sudo reboot
```

Once setup your Python script will run whenever your reboot or start-up your Pi. There may be times when you reboot and don't want the script running. To stop it you can find out its process number and "kill" it. To do this type :

```
ps aux | grep /home/pi/MyScript.py
```

This should give you a line starting with "root" and ending in the path to your script. Immediately after the "root" should be a process number. For example :

```
root 1863 0.0 1.0 24908 4012 ? S1 19:45 0:00 python
/home/pi/MyScript.py
```

In this case we can stop the process using :

```
sudo kill 1863
```

Final Thoughts

If you are feeling adventurous you can write your Python script to automatically exit if a certain condition is met so you don't need to ever "kill" it. Ideas include :

- Test the GPIO pins and quit if a switch is being pressed. Maybe two switches being held down.
- Test if a network connection is available and quit if it is. This may indicate you are testing (a camera for example) and you only want the script to auto-run when there is no network present.
- Check for the existence of a particular file. This would allow you to create a named file to prevent the script from running at next boot.

There are other techniques to run scripts at boot up and you might want to Google "rc.local" or "init.d". I prefer the Cron method because it is so simple.

For additional information on the powerful features of Cron take a look at the [Wikipedia Cron page](#).

8/6/16

Found volume control for omxplayer:

```
omxplayer --vol N audio.mp3 where: -1500<N<5000
```

So to play the file ***bridge.mp3***, from the command line:

```
omxplayer --vol 600 bridge.mp3
```

In *gps-95.py* the line is:

```
os.system("omxplayer --vol 1200 %s" % sound)
```

8/4/16

I changed the files in the *gps* file to absolute names with paths.
However when I add the */home/pi/* to the *load_gps_files* I get a syntax error??

I added a couple of lines to the *gps* file to write to a *gpsboot.txt* file. It did not do this!!
Although it does write to the file when I run */etc/rc.local!!!*

I added tow files to the autostart file

It ran the *boot_file.py* This file appends a text line to the file *bootlog.txt*. When I reboot, it writes it three times to the file.

It did not run the *gps* file.

The line I added is *python /home/gps-94.py &*

Could the it be that it did not open an lx terminal??

When I run the file */etc/rc.local* it runs the *gps* file BUT I cannot ctrl-z out of it!!

8/2/16

Trying to make an autostart file

/etc/rc.local

I added the line */usr/bin/python3 /home/pi/boot_file.py*

The system restarts but it does not open a terminal widow to display the results of the *boot_file.py*

REMEMBER to add the *&* after the file name or will NEVER boot if the file is a loop.

7/26/16 To Do

Fix sound level in *pygame*

Fix mile post 4

Get accurate data for locations

Add direct and return trip locations
Add offset to locations
Reset the *locs[x]* list when we reach New Hill.

7/25/16 To fix the problem below we stripped the the last character off of the *locs[x]* variable ie *locs[x][0:-2]* Don't know why??

If you print *locs[x] + 'xxx'* , the xxx appears on the next line.

In the *play_locs1.py* file this worked fine for x=0 to 13. When x=14 I had to strip 4 characters ie *locs[x][0:-4]* ????. Why does fearrington work it is 15??

The *pa.py* file (play audio), plays the audio clip in the current directory using pygame

On the command line enter:

```
python filename ie if the clip is horton_road.mp2, enter python  
horton_road
```

If I play a file with omxplayer it is 12db hotter than playing with pygame????

Files needed load:

```
gpssock.sh  
gps_files3.py  
logfilenum.txt  
gps-74.py  
latitude.txt  
location.txt  
play_locs.py  
pa.py
```

7/18/16

It appears that the variable *locs[x]* has a carriage return in it!
It was probably inserted in the *load_gps_files3.py*.

When I assemble the file name sound

```
sound = 'home/pi/Music/' + str(x) + locs[x] + '.mp3'
```

When I print sound

```
print sound
```

It prints the .mp3 on the next line ie:

```
/home/pi/Music/15-Fearrington
```

.mp3

The pygame can't open the file

7/17/16

To play audio from the analog port, **from the command** line:

pi@raspberrypi: \$ amixer cset numid=3 1 Sets output to **analog** port

pi@raspberrypi: \$ amixer cset numid=3 2 Sets output to **HDMI** port

To play a .wav file ie a file called Steel.wav:

aplay Steel.wav

To play an mp3 file called steet.wav:

omxplayer steel.mp3

To play a file from a Python program I used pygame. After several failed attempts I found a listing that described the problem.

```
# Play audio file in Python
import pygame
import time

#pygame.init()

pygame.mixer.init()
pygame.mixer.music.load('crowd.mp3')

pygame.mixer.music.play(0)
```

The program stopped before the track would play.

I could here audio hiss when I ran the program below, but the file would not play. Here is the program called audio4.py,that allowed the file to play.

```
# Play audio file in Python
```

```

import pygame
import time

#pygame.init()

pygame.mixer.init()
pygame.mixer.music.load('crowd.mp3')

pygame.mixer.music.play(0)
while pygame.mixer.music.get_busy():
    time.sleep(1)

print "Done"

```

Files needed load:

gpssock.sh
gps_files3.py
logfilenum.txt
gps-6.py
latitude.txt
location.txt

7/14/16

add_1_store.py

This module opens the file ***logfilenum.txt*** It extracts the number stored in it. This becomes the next logfile(num).txt that will be opened in the gps program.

It then adds 1 to this number and saves it in the ***logfilenum.txt***.

```

1 w='logfilenum.txt'
2 f = open(w, 'r')
3 print w
4 f.readline()
5 filenum = f.read()
6 print filenum
7 print int(filenum) +1
8 f.close
9 f = open(w, 'w')
10 f.write ("This stores the file number\n")
11
12 f.write(str(int(filenum)+1))
13 f.close

```

When it reads the file in line 5, it starts at line 2 of the ***logfilenum.txt*** so there is a dummy

line in the file, **This stores the file number**. The next line is where the number is stored.

test9.py

The added lines are in bold:

```
import time
from add_1_store import filenum
logfile = 'logfile'+filenum+'.txt'
print logfile
from load_gps_files3 import* #loads the locations and latitudes
print"#####"
print
print"Searching for location"
print
print
file = open(logfile,'w')
ttime = .001 # Dummy variable for simulating GPS receiver
delta =.0006 # The spread between the latitude and play point
```

Each time the *test9.py* is run the logfile.txt digit is increased ie After *logfile15.txt* is used, the next time *test9.py* is run *logfile16.txt* stores the data.

7/9/16

So far there are three parts of the system

There are two text files that show the locations of the points along the railroad tracks that the sound effects will be played.

One file lists the latitude and the other lists the description.

These files are *latitudes.txt* and *locations.txt*.

These files are loaded into the GPS system with the *load_gps_files3.py* program

The *test8-1.py program* uses the above file to load the locations and latitudes.

It then simulates the GPS receiver data and searches for the sound effects locations.

When it find one, it plays the audio file. This has not been tested so far.

It then writes the data which will include the time and speed to the file *logfile.txt* .

It adds 10 to the latitude in the *lats[x]* list, so it wont restart the audio file.

The parameter ttime should set to .001. This file still has to be combined into the GPS reader file. The current version of this program is *ted14.py*.

There is a *clearfile.py* which will erase the data in the *logfile.txt*. This is for testing purposes only.

Things to do:

Combine test8-1 with ted14

Save *logfile.txt* under a new name to include the date and time.

Reset the *lats/x/* list when we reach New Hill.

Check direction to play different files on return trip.