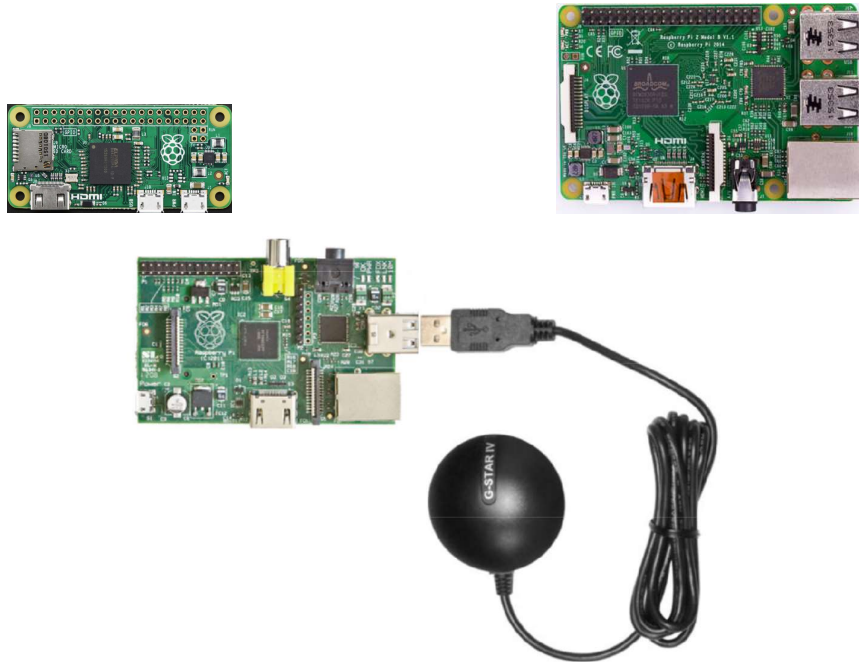


Python GPS1 system 12/12/23 *gps-409-2-5.py*

This file is printed *gps_system_description.pdf*.



Overview

The GPS system was designed to play audio clips along the route of the New Hope Valley Railway. The initial motivation was to play sound effects during the Halloween train rides. At a specified latitude and longitude, the device would play a specific sound effect.

After the system is powered up, it may take the GPS receiver a couple of minutes to lock on to enough satellites to retrieve valid data. The system then **runs autonomously**. On startup, the LED indicator will come on after the Linux system loads and the python application has been launched. This takes less than a minute. The indicator will begin to flash when the GPS receiver is locked and is sending valid data to the system. This could take several minutes, depending on the atmospheric conditions.

In normal operation the LED will flash approximately once per second. The LED will remain on when the system is capturing and storing positional data. (See ctrl-c functions)

If a background audio track exists, it will begin to play

In 2023, an additional feature was added. The system will now play audio clips before the train leaves the station. The clips will play 15, 10, 5 and 1 minute before the train departs the station. See Appendix 13 for details.

Requirements

The system was designed around a Raspberry Pi computer. The software was written in Python 2.7. To configure and maintain the system, the users needs to have a minimum amount of knowledge of Linux, the text-based operating system on the Raspberry Pi and a working knowledge spreadsheets.

The GPS receiver chosen is the Globecast BU353. .

It is a sealed weather proof unit with a USB interface and costs about \$35.

See Appendix 8 for details on building the system on the Raspberry Pi.

Executive summary

You need to have a list of the latitudes at which to you want to play a sound track. Use the **Ctrl-c** feature to capture the data. See below for details.

You need to create a .csv file that contains the latitude, the filename, the volume level (usually 0). This file can be made with EXEL or Libre calc, which is installed on the Raspberry Pi.

The column in the spreadsheet are:

- A latitude
- B longitude (set to zero)
- C The location name

- D Audio file to be played going north bound
- E Audio gain level for northbound clip (usually 0)
- F Audio file to be played going south bound
- G Audio gain level for southbound clip (usually 0)

The latitudes can be captured by using the utility in the ctrl-c function.

A	B	C	D	E	F	G	H
35.79642534	-79.07393458	863hatfield	863_Hatfield.mp3	0	863_Hatfield.mp3	0	
35.79477313	-79.07571446	Langdon-Millcroft	langdon_millcroft_n.mp3	0	langdon_millcroft_s.mp3	0	
35.79452104	-79.08334893	Mail Klost	Silent_0003.mp3	0	Silent_0003.mp3	0	
35.79708756	-79.08751786	East Camden-Weatherfield	east_camdon_weatherfield_n.mp3	0	east_camdon_weatherfield_s.mp3	0	
35.79940474	-79.08897619	E. Camden ΓCδ McDowell	east_camdon_mcdowell_n.mp3	0	east_camdon_mcdowell_s.mp3	0	
35.80020662	-79.0898087	East Camden-East Madison	east_camdon_east_madison_n.mp3	0	east_camdon_east_madison_s.mp3	0	
35.79945827	-79.09108501	W. Camden - W. Madison	W_Camden_W_Madison.mp3	0	W_Camden_W_Madison.mp3	0	
35.79714203	-79.08962034	W.Camden ΓCδ Caldwell	W_Camden_Caldwell.mp3	0	W_Camden_Caldwell.mp3	0	
35.79590404	-79.08885486	W. Camden - Weathersfield	W_Camden_Weathersfield.mp3	0	W_Camden_Weathersfield.mp3	0	
35.79689167	-79.09593706	51-501 Weathersfield	Silent_0003.mp3	0	Silent_0003.mp3	0	
35.79864801	-79.09055416	W. Camden - Caswell	W_Camden_Caswell.mp3	0	W_Camden_Caswell.mp3	0	
35.79528517	-79.08514481	E. Camden - Rutherford	E_Camden_Rutherford.mp3	0	E_Camden_Rutherford.mp3	0	
35.7946373	-79.08284167	Millcroft ΓCδ E. Camden	millcroft_east_camdon_n.mp3	0	millcroft_east_camdon_s.mp3	0	
35.79514922	-79.08187463	Millcroft ΓCδ Duplin	millcroft_duplin_n.mp3	0	millcroft_duplin_s.mp3	0	
35.79514411	-79.0819777	Millcroft ΓCδ Harnett	millcroft_harnett.mp3	0	millcroft_harnett.mp3	0	
35.79429548	-79.07911952	Millcroft ΓCδ Burke place	millcroft_burke_n.mp3	0	millcroft_burke_s.mp3	0	
35.79619775	-79.07492875	N. Langdon - Brookfield	N_Langdon_Brookfield.mp3	0	N_Langdon_Brookfield.mp3	0	
35.79706685	-79.0739176	Brookfield ΓCδ Hatfield	hatfield_brookfield_n.mp3	0	hatfield_brookfield_s.mp3	0	
35.79723252	-79.08965542	West Camden- Caldwell	W_Camden_Caldwell.mp3	0	W_Camden_Caldwell.mp3	0	
35.79788599	-79.08799305	E. Camden - Yancey	east_camdon_yancey_n.mp3	0	east_camdon_yancey_s.mp3	0	

Example of a gps_data.csv file

DO NOT leave line 1 blank!

Save this file with the name: **gps_data**. Save as type, **.csv**. This is a **Comma delimited file**. The file name will be **gps_data.csv**

This file must be in the **/home/pi/gps_2017** directory.

The **gps_data.csv** file can be edited offline, on a desktop computer, with EXEL. Copy this file to a flash drive with the name **USB_DISK**.

If this flash drive is in the Raspberry Pi it will upload the **gps_data.csv** file to the **/home/pi/gps_2017** folder on boot up. **Disabled**.

You need the sound tracks that will be played at each latitude. They can be **.mp3** files or **.wav** files.

Make sure the audio files are in the **/home/pi/gps_2017/audio_tracks** directory.

There are utilities to check to see if files are missing and to play each file. See Utilities below.

The system creates the following files: **logfile.txt**, **play.txt** and **datafile.txt** .

The last three lines in the `gps_data.csv` file are unique. The background audio track should be entered in the next to last line. The latitude of the last three lines are set to 39, 40 and 50.

To play a background audio track, enter the data on the next to the last row. Enter the name of the track in column 4, the gain in column 5 and the number of times the track needs to be repeated in column 7.

To display the name of the csv file, enter the name in column 3 of the last row.

37	35.69866409	-79.97	half_way_to_whistle	silence_0001.wav	0	silence_0001.wav	0
38	35.69965891	-79.97	new_hill_whistle_marker	silence_0001.wav	0	silence_0001.wav	0
39	39	-79.97	per_departure-gain		599		
40	40	-79.97	Background Music	Xmas_Music_Tracks_2023.mp3	20	silence_0001.wav	8
41	50	-79.97	Xmas_2020	silence_0001.wav	0	silence_0001.wav	0
42							

A	B	C	D	E	F	G
1	2	3	4	5	6	7

In the above example, line 39, with the latitude set to 39, the departure audio tracks will play with the volume set to 599 (5.99db).

Line 40, with the latitude set to 40, the background audio track, in column 4, is **xmas_Music_Tracks_2023.mp3**, with the gain, in column 5 is set to **20**, in column 6 and it will be repeated **8** times, as set in column 7.

Line 41, with the latitude set to 50, shows the name for the **gps_data.csv** file, in column 4

Note: During normal operations the system does NOT need to have a monitor connected to the Raspberry Pi. When the train is at Bonsal, the operator can connect to the system using the remote software VNC from the computer in the yard office.

The monitor display is:

```
pi@raspberrypi: ~
File Edit Tabs Help
--- New Hope Valley Railway ---

GPS system for the New Hope Valley Railway
Created by Ted Dunn with software written by Dan Mandle
With the help of Robert Malkin of Duke University
-----
latitude      35.796246094
longitude     -79.074193958
time utc      2023-12-14T13:56:59.000Z + altitude (m) 156.739
speed (m/s)   0.612
climb         -0.024
track         241.6123
mode          3
Altitude :   514 Feet

Delta         0.00015      55 ft.
TIMETAG =: EST  OFFSET =: 5
Current latiude  35.796246094
Previous latitude 35.796246094
Speed : 1.368 mph
Traveling to Bonsal
2023-12-14T13:56:59.000Z
Local Time = 08:56:59 AM EST 12/14/23
GPS_DATA.CSV in use: Xmas_2020
Pre-departure audio gain: 599      gps_data.csv pre_dep_vol 599

Background track Xmas_Music_Tracks_2023.mp3 Gain = 20 Repeating 8 times
['gps1.py']
409-2-5 Pre gain
-----
('Trains today: ', '10:30 AM      ')
2023-12-14 08:55:56.712504
```

Data file in use
Pre-departure level

Background Muisic track

gps1.py Program Flow

- 1- Import from libraries
- 2- Set GPIO on Rasp Pi
- 3- Set variables
- 4- Upload and Download to Flash Drive

- 5- Load location data from .CSV file
- 6- Format date and time to EST and 12 Hr
- 7- Create Thread to play audio clip
- 8- Create Thread for background music
- 9- Load schedule.csv to play audio tracks before the train departs the station
- 10- Set Delta to find location
- 11- Run gpssock
- 12- Run the GPS Thread from Dan Mendel
- 13- Save data to logfile.txt
- 14- Check for gps lock
- 15- Find DST data
- 16- Keyboard interrupt ctrl-c from Robert Malkin
- 17- Store Latitude and Longitude to datafile.txt or change time or play audio track

Details

All of the files are located in the **/home/pi/gps_2017** folder.

To make it faster and easier to copy the program, the following folders were moved from the **/home/pi/gps_2017** folder, to the **/home/pi** folder

```
gps_audio_archive
gps_audio_must
gps_data_normal_ride
gps_old_audio_tracks
```

To begin the process, we have to know the latitude of the locations that will play the audio clip. This version does not use the longitude data.

The GPS1 system uses only latitude. This allows the system to play different audio tracks when going northbound and southbound.

Next, you need to know the file names of the audio clips that are to be played.

The file in which this information is assembled is named: **gps_data.csv**.

This is a spreadsheet file, saved in the **.csv** format (Comma Separated Values). The Python library has a module to read **.csv** files. This data, in this **.csv** file, is read into the system by the program **load_csv_data.py**.

The `gps_data.csv` contains seven fields:

- Latitude
- Longitude
- Location
- Audio track northbound
- Volume level northbound
- Audio track southbound
- Volume level southbound

Example of a `gps_data_csv` file

	A	B	C	D	E	F	G	H
1	35.79642534	-79.07393458	863hatfield	863_Hatfield.mp3	0	863_Hatfield.mp3	0	
2	35.79477313	-79.07571446	Langdon-Millcroft	langdon_millcroft_n.mp3	0	langdon_millcroft_s.mp3	0	
3	35.79452104	-79.08334893	Mail Klost	Silent_0003.mp3	0	Silent_0003.mp3	0	
4	35.79708756	-79.08751786	East Camden-Weathersfield	east_camdon_weathersfield_n.mp3	0	east_camdon_weathersfield_s.mp3	0	
5	35.79940474	-79.08897619	E. Camden ΓÇô McDowell	east_camdon_mcdowell_n.mp3	0	east_camdon_mcdowell_s.mp3	0	
6	35.80020662	-79.0898087	East Camden-East Madison	east_camdon_east_madison_n.mp3	0	east_camdon_east_madison_s.mp3	0	
7	35.79945827	-79.09108501	W. Camden - W. Madison	W_Camden_W_Madison.mp3	0	W_Camden_W_Madison.mp3	0	
8	35.79714203	-79.08962034	W.Camden ΓÇô Caldwell	W_Camden_Caldwell.mp3	0	W_Camden_Caldwell.mp3	0	
9	35.79590404	-79.08885486	W. Camden - Weathersfield	W_Camden_Weathersfield.mp3	0	W_Camden_Weathersfield.mp3	0	
10	35.79689167	-79.09593706	51-501 Weathersfield	Silent_0003.mp3	0	Silent_0003.mp3	0	
11	35.79864801	-79.09055416	W. Camden - Caswell	W_Camden_Caswell.mp3	0	W_Camden_Caswell.mp3	0	
12	35.79528517	-79.08514481	E. Camden - Rutherford	E_Camden_Rutherford.mp3	0	E_Camden_Rutherford.mp3	0	
13	35.7946373	-79.08284167	Millcroft ΓÇô E. Camden	millcroft_east_camdon_n.mp3	0	millcroft_east_camdon_s.mp3	0	
14	35.79514922	-79.08187463	Millcroft ΓÇô Duplin	millcroft_duplin_n.mp3	0	millcroft_duplin_s.mp3	0	
15	35.79514411	-79.08119777	Millcroft ΓÇô Harnett	Millcroft_Harnett.mp3	0	Millcroft_Harnett.mp3	0	
16	35.79429548	-79.07911952	Millcroft ΓÇô Burke place	millcroft_burke_n.mp3	0	millcroft_burke_s.mp3	0	
17	35.79619775	-79.07492875	N. Langdon = Brookfield	N_Langdon_Brookfield.mp3	0	N_Langdon_Brookfield.mp3	0	
18	35.79706685	-79.0739176	Brookfield ΓÇô Hatfield	hatfield_brookfield_n.mp3	0	hatfield_brookfield_s.mp3	0	
19	35.79723252	-79.08965542	West Camden - Caldwell	W_Camden_Caldwell.mp3	0	W_Camden_Caldwell.mp3	0	
20	35.79788599	-79.08799305	E. Camden - Yancey	east_camdon_yancey_n.mp3	0	east_camdon_yancey_s.mp3	0	
21								
22								

When the csv file is loaded into the system it adds another field the location number. In this case it would be 0 to 13. This can be used to play the audio clips using the `play.py` utility program.

Note: There cannot be an empty row at the top of this spreadsheet. If there is, the `load_csv_data.py` program will crash.

Hint: You can store different data files by adding a descriptive suffix to the name. For example, save the `gps_data.csv` as `gps_data_halloween_10-20-18.csv`. Remember to load all of the audio files listed in the `gps_data.csv` file into the `audio_tracks` folder. Use the `check_for_audio_files.py` utility to insure you have loaded all of the required files.

```

pi@raspberrypi:~/gps_2017 $ ls gps*.csv
gps_data102218.csv          gps_data_xmas_12-3.csv
gps_data.csv               gps_data_xmas_12-5.csv
gps_data_fearr3.csv       gps_data_xmas_12-8.csv
gps_data_halloween_10-13-18.csv  gps_data_xmas.csv
gps_data_halloween_10-20-18.csv  gps_data_xxx.csv
gps_data_halloween_2017.csv

```

Above is a listing of the data files in the system.

Use the **cp** command to change the **gps_data.csv** file. For example to use the **gps_data_fearr3.csv** file, at the command line in the **/home/pi/gps_2017** directory, enter: **cp gps_data_fearr3.csv gps_data.csv**

WARNING: The cp command will overwrite the existing file without warning!

The main program that operates when the train is running is named **gps1.py**. The program that loads the data from the spreadsheet is, as stated above, **load_csv_data.py**. This program is run by the import command in the main program.

When the GPS receiver has acquired enough satellites to lock, the date is stored in the text file **logfile.txt**. This file is then read by the **find_dst_import.py** routine to determine whether the time standard is DST or EST.

As stated above the audio files are stored in the folder **/home/pi/gps_2017/audio_tracks**. The names are formatted in the name_length.mp3. The length is four digits and is the duration of the clip in seconds.

For example: There is an audio clip of a “demented man walking” that is 16 seconds in duration. The file name is: **demented_man_walking_0016.mp3**.

This format was selected so that the audio clips can be easily sorted by length.

When the system is running, and an audio clip is played, the program records the location number, the direction of travel (n or s), date and

time, audio file name, location and the file name of the gps program file. This information is stored in the **logfile.txt** file.

Note: The logfile.txt inserts a blank character at the beginning of the cell. ie (**motor_car_house**) is stored as (**motor_car_house**)

It can easily be imported into a spreadsheet for future use.

Tip: When creating or editing the *gps_data.csv* file, it is best to copy the audio file names from the **audio_tracks** directory and paste them into the spreadsheet file. This will reduce typographic errors.

To do this, create a text file of the audio tracks, go to the **/home.pi.gps_2017/audio_tracks** folder. Use the command: **ls > audio_files.txt.**(The “>” redirects the “ls” command from the screen to the text file.) Then open the file **audio_files.txt**, in a spreadsheet to copy and paste the names into the **gps_data.csv** file.

Similarly, the latitude data from the **datafile.txt** should be pasted into this spreadsheet file.

Ctrl-c operations

Label:

ctrl-c functions

```
est dst tone1k tone
pre prevol nb sb music voice
```

The Ctrl-c performs the following functions:

Note: The ctrl-c stops the background audio track. To restart the track enter background.

The original purpose of the Ctrl-c function was to capture the latitude information rather than manually typing it in.

This function has been expanded.

This logging will work without a monitor connected to the Raspberry Pi.

Using the Ctrl-c function.

Press the **Ctrl** and the **c** keys. The LED will stop flashing and remain on.

If there is a background audio track running, the LED will continue to flash. Press the **Ctrl** and the **c** keys again and the LED will stop flashing and remain on.

Note: The first Ctrl-c will be detected by the mplayer program and will stop playing the background audio track. The next Ctrl-c will be detected by the operating program and will pause the system for the operator to enter the commands.

Operations after ctrl-c is pressed

est	Sets system time to eastern standard time
dst	Set system time to eastern daylight saving time
tone	Plays a 440 Hz tone for 20 minutes
tone1k	Plays a 1000 Hz tone
voice	Plays a voice track for 50 minutes
pre	Plays the four audio tracks that will play before departure
prevol	Allows user to change the volume of the four pre-departure audio tracks
nb	Plays the north bound tracks
sb	Plays the south bound clips

To stop playing these tracks, press the “**q**” key.

- 1- In order to log the position of a location that you wish to have an audio file play, press the **ctrl-c** keys. The LED will stay on. Type in the name of this location and press enter. The LED will begin to flash. The data is saved in the file **datafile.txt**. Note that the datafile records the latitude and the longitude. The longitude is not used in this version.

This file can be opened by a spreadsheet program and the data used to make the **gps_data.csv** file.

- 2- To play a music track, press the **ctrl-c** keys, type the word **music** and press the **enter** key. To stop playing press the **q** key.
- 3- To restart the background audio track, press **ctrl-c**, type the word **background** and press enter.

Utility files:

check_for_audio_files.py
check_for_audio_filesPC.py
play_nb.py
play_sb.py
playcsv.py
play.py
text_file_2_array.py (This file is created used by the *play.py* utility.)

These files help to manage the audio clips stored in the **audio_tracks** folder.

The *check_for_audio_files.py* loads the *gps_data.csv* file and compares the audio file names found, with the files in the **audio_tracks** directory. If it cannot find a file, it lists the file in the *missing_audio.txt* file. If the flash drive is mounted on the system, the *missing_audio.txt* file is copied to it.

To use these utility programs:

Open a new window

At the command line, change to the `gps_2017` directory by typing:

`cd gps_2017` (change directory)

At the command line enter: **`python {name of utility}`**

For example:

At the command line enter: **`python check_for_audio_files.py`**

The **`check_for_audio_filesPC.py`** runs on a PC system.

```
35 - silence_0001.wav is a file in audio_tracks/
35 - its_beginning_to_look_0320.wav is NOT a file in audio_tracks/
36 - silence_0001.wav is a file in audio_tracks/
36 - silence_0001.wav is a file in audio_tracks/
37 - various_1500.wav is a file in audio_tracks/
37 - silence_0001.wav is a file in audio_tracks/
38 - silence_0001.wav is a file in audio_tracks/
38 - silence_0001.wav is a file in audio_tracks/
Found Flash Drive
Completed process

Missing audio files
2018-12-10 17:21:57

its_beginning_to_look_0320.wav
pi@raspberrypi:~/gps_2017 $
```

This is what the screen looks like if a file is missing.

```
35 - silence_0001.wav is a file in audio_tracks/
35 - its_beginning_to_look_0320.wav is a file in audio_tracks/
36 - silence_0001.wav is a file in audio_tracks/
36 - silence_0001.wav is a file in audio_tracks/
37 - various_1500.wav is a file in audio_tracks/
37 - silence_0001.wav is a file in audio_tracks/
38 - silence_0001.wav is a file in audio_tracks/
38 - silence_0001.wav is a file in audio_tracks/
Found Flash Drive
Completed process

Missing audio files
2018-12-10 17:15:51

pi@raspberrypi:~/gps_2017 $
```

This is what the screen looks like if there are no missing audio files.

The utility files **`play_nb.py`** and the **`play_sb.py`** will play all of the audio tracks that will be played on the route based on the contents of the **`load_csv_data.csv`** file. To play them in the proper sequence, the **`gps_data.csv`** file should be sorted by the latitude column.

To use these utility programs:

Open a new window

At the command line, change to the `gps_2017` directory by typing:

`cd gps_2017` (change directory)

At the command line enter: **`python play_nb.py`** or

`python play_sb.py`

Tip: When using these utility programs, pressing the **q** key will stop the audio clip that is playing and start the next clip. See all of the `omxplayer` commands in appendix 1.

play.py

This utility program will play any clip in the `audio_tracks` folder at any volume level

At the command line, in the `gps_2017` folder type:

`python play.py`

The screen will display all of the audio files in the **`audio_tracks`** folder and assign an index number to each file. It will then prompt you to enter the file number.

It will then prompt you to enter a gain level. This can be any number between -2500 and +1000. To double the gain of the audio clip enter 600. This will increase the volume level by 6 db (decibels). Similarly to reduce the level by half, enter -600. This will decrease the volume level by -6 db .

The program will then play the audio file with the gain level that was entered.

If the gain was OK, you can save the information for future reference to the file **`saved.txt`** by entering the **y** key when prompted.

If you wish to replay the audio clip, launch the program again and when prompted to enter a file name, hit the Enter key. If you wish to change the gain, at the prompt, enter a new gain number and hit Enter key. If you wish to play the file at the same level just hit the ENTER key and it will play it at the previous gain level.

The previous file name and previous gain level are stored in the **`play_last.txt`** file.

Note that it is more convenient to have subfolders which contain the audio files for specific events. For example, a Halloween effects folder and Bonsal test files folder. When using these audio files, copy them from the subfolder to the **`audio_tracks`** folder.

playcsv.py

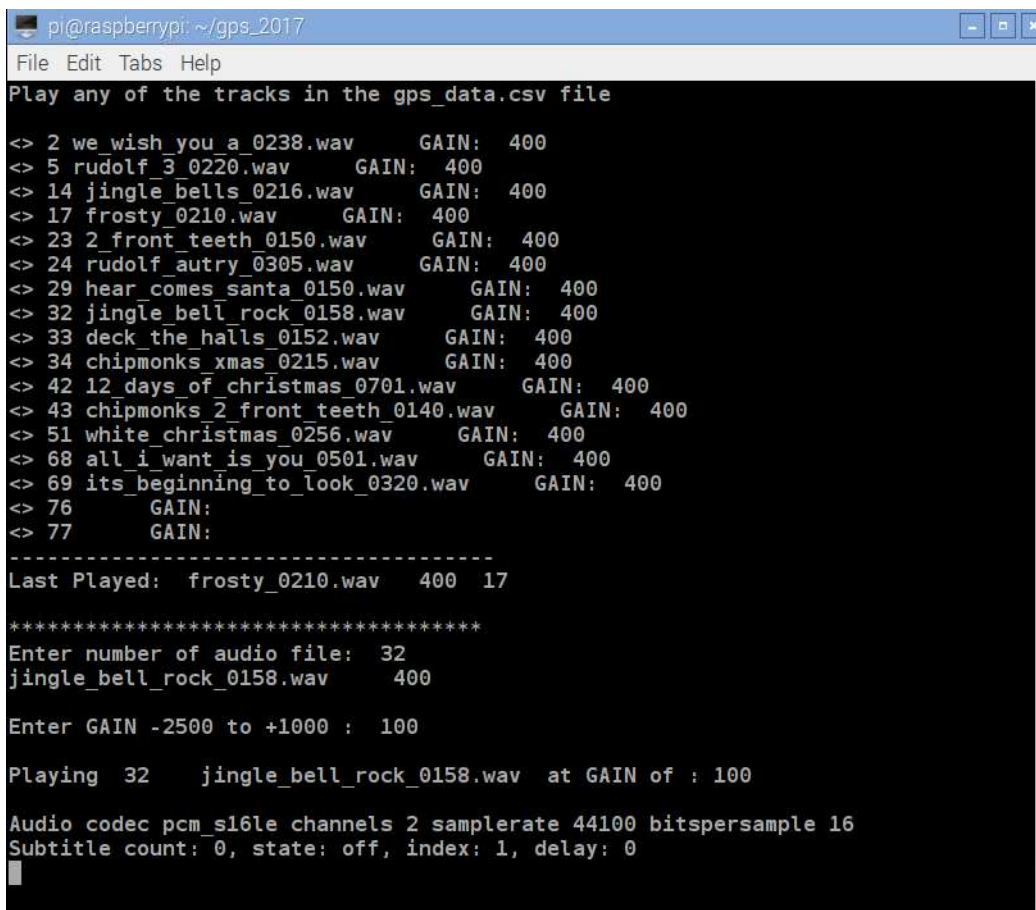
This utility program will play any clip in the **`gps_data.csv`** file at any volume level.

At the command line, in the `gps_2017` folder type: **`python playcsv.py`**

Below shows the screen when the program is run. Number 32 was entered, the **`jingle_bell_rock_0158.wav`** file. The gain entered was **100** and the audio clip was played. The clip playback can be cancelled by pressing the **q** key (for quit).

Note that the screen shows the **'Last Played: "** file was **`frosty_0210.wav`**. If you wish to replay that file, just hit the Enter key. Then enter the desired gain.

You can save the data to the **`saved.txt`** file by entering **Y** or **y**.



```
pi@raspberrypi: ~/gps_2017
File Edit Tabs Help
Play any of the tracks in the gps_data.csv file

<> 2 we_wish_you_a_0238.wav      GAIN: 400
<> 5 rudolf_3_0220.wav          GAIN: 400
<> 14 jingle_bells_0216.wav     GAIN: 400
<> 17 frosty_0210.wav           GAIN: 400
<> 23 2_front_teeth_0150.wav    GAIN: 400
<> 24 rudolf_autry_0305.wav     GAIN: 400
<> 29 hear_comes_santa_0150.wav GAIN: 400
<> 32 jingle_bell_rock_0158.wav  GAIN: 400
<> 33 deck_the_halls_0152.wav   GAIN: 400
<> 34 chipmonks_xmas_0215.wav   GAIN: 400
<> 42 12_days_of_christmas_0701.wav GAIN: 400
<> 43 chipmonks_2_front_teeth_0140.wav GAIN: 400
<> 51 white_christmas_0256.wav  GAIN: 400
<> 68 all_i_want_is_you_0501.wav GAIN: 400
<> 69 its_beginning_to_look_0320.wav GAIN: 400
<> 76      GAIN:
<> 77      GAIN:

-----
Last Played:  frosty_0210.wav  400  17

*****
Enter number of audio file: 32
jingle_bell_rock_0158.wav  400

Enter GAIN -2500 to +1000 : 100

Playing 32  jingle_bell_rock_0158.wav  at GAIN of : 100

Audio codec pcm_s16le channels 2 samplerate 44100 bitspersample 16
Subtitle count: 0, state: off, index: 1, delay: 0
█
```

You can save the data to the **`saved.txt`** file by selecting **Y**.

Below are the last few lines in the **`saved.txt`** file.

```
demented_man_howling_0006.mp3,-200 ,
adams_famaly_0085.mp3,-700 ,
bridge_scream_0013.mp3,-700 ,
bubble_bubble_0011.mp3,-700 ,
bubble_bubble_0011.mp3,500 ,
casper_0065.mp3,-700 ,
cats_and_dogs_0011.mp3,-700 ,
2 front teeth_0150.wav
-200chipmonks_xmas_0215.wav200
200deck_the_halls_0152.wav,300 ,
we_wish_you_a_0238.wav,123
12_days_of_christmas_0701.wav,-987
jingle_bell_rock_0158.wav,100
pi@raspberrypi:~/gps_2017 $
```

In summary:

The following files need to be in the operational directory: (/home/pi/gps_2017)

gps1.py (The main operating program)
gps_data.csv (The spreadsheet file containing the data)
load_csv_data.py (Loads the spreadsheet data)
gpssock.sh (see Appendix 8)
play_nb.py (Utility to play audio tracks)
play_sb.py (Utility to play audio tracks)
play.py (Utility to play and change the audio tracks)
text_file_2_array.py (This file is used by the **play.py** file.)
playcsv.py (Utility to play only files in gps_data.csv)
check_for_audio_files.py
.desktop (This file is in the /home/pi/.config/autostart folder see Appendix 8)

The following files are needed for version gps-409-xxx.py

schedule.csv This file contains the schedule of the rides for the current year

1.mp3 In **the audio_files** folder.

2.mp3

3.mp3

4.mp3

tone.mp3

The following files will be created:

logfile.txt (logs whenever an audio clip is played)
datafile.txt (store latitude data captured by ctrl-c)
bootlog.txt (Stores data from the last startup)
missing_audio.txt (Created by the **check_for_audio_files.py** file.)
saved.txt (Stores file names and gain of all the audio files played)
play_last.txt (Stores data of the last audio files played by **play.py**)
play_last_csv.txt (Stores data of the last audio files played by (**playcsv.py**))

Other files and folders 11/02/23

The **audio_archive** folder contains all of the audio files that have been uses. The **audio_archive** folder has been moved out of **the ...gps_2017** folder to the **/home/pi** folder to make it faster to copy the **gps_2017** folder over the network.

A new folder was created: **/home/pi/gps_data_normal_ride.**

This folder was added to make it easier to change the GPS system, when changing from normal rides to special event rides.

This folder contains the *gps_data.csv* file and the corresponding audio files required.

The *gps_data.csv* file should be copied to the *gps_2017* folder.

The audio files should be copied to the *gps_2017/audio_tracks* folder.

```
pi@raspberrypi:~/gps_data_normal_ride $ ls
1.mp3                bridge_north_bound.mp3  leaving_bonsal.mp3
1-readme            bridge_southbound_2.mp3 on_behalf_of.mp3
1sec_tone.mp3       cuts_and_fills.mp3     remain_seated.mp3
2.mp3               goldston_depot_1-17.mp3 run_around_2.mp3
3.mp3               gps_data.csv           tone1k.mp3
4.mp3               grade_crossing.mp3     tone.mp3
all_volunteer_organization.mp3 History-3-20L.mp3
approaching_bonsal.mp3 horton_rd.mp3
```

The sound effects tracks are played using *omxplayer*. This is part of the NOOB's installation.

The background music track is played using *mplayer*.

The media player *mplayer* must be installed. It is used to play the background music track.

At the command prompt, enter: *sudo apt-get install mplayer*.

It may take 10 minutes to load.

The *mplayer* does not paly audio on the Pi-4.

A Flash drive is optional. It needs to have the name: **USB_DISK**.

It can be used to configure the system “offline”, by adding files to this drive from another PC and to download and analyze the data saved by the system offline without removing the system from operation.

When the system boots up it will:

- 1- Download the files **logfile.txt** , **datafile.txt** and **gps_data.csv** as **gps_data_downloaded.csv**.
- 2- If **gps_data.csv** file is on the FD, it is uploaded to the system’s SD card.
- 3- If the file **est_dst.txt** is on the FD it will change the time standard to Eastern Standard or Daylight Savings time.

The Flash Drive should have two text files **est.txt** and **dst.txt**.

For Eastern Standard time the text file contains: **5EST**

For Daylight Savings time the text file contains: **4DST**

To change the time standard, copy either file to a new file **est_dst.txt**.

With the flash drive plugged into the system, the files will automatically be uploaded when the system is rebooted.

As stated above, the audio clips used are stored in a separate folder.

In the **/home/pi/gps_2017** directory, create a folder **audio_tracks**

Place the audio files, in .mp3 or .wav format, into the folder:

/home/pi/gps_2017/audio_tracks.

Tip: WAV files can be edited using Audacity, a free audio editing program. MP3 files were used in a previous version of the system that used an mp3 player and cannot be edited.

Tip: Files can be captured using Sound Trap, a free streaming audio recorder.

AUTOSAVE to Flash Drive Introduced in version *gps-404.py*.

A feature was added that automatically copies the logfile.txt file to a flash drive, if it is plugged into a USB port. The flash drive must have the name **USB_DISK**. This is accomplished when the speed is less than 3 MPH and the latitude has changed from the last time the file was saved.

If the system is stationary, the logfile will only be saved once.

```
##### Autosave logfile to flash drive named USB_DISK
    if (gpsd.fix.speed*2.236<3 and abs(lattsaved -latt) > 2*delta):
        print '                Copying logfile to flashdrive'
        copyfile('/home/pi/gps_2017/logfile.txt','/media/pi/USB_DISK/logfile.txt')
        lattsaved =latt
        print lattsaved

#### Autosave logfile to flash drive named USB_DISK
    if (gpsd.fix.speed*2.236<3 and abs(lattsaved -latt) > 2*delta)and os.path.isdir('/media/pi/USB_DISK'):
        print '                Copying logfile to flashdrive'
        copyfile('/home/pi/gps_2017/logfile.txt','/media/pi/USB_DISK/logfile.txt')
        lattsaved =latt
        print lattsaved
```

The routine **if clause** checks the **speed**, the change in **latitude** and whether the **Flash Drive** is mounted. If it finds the drive it copies the logfile to it. If there is no drive in the USB port, the programs continue to run without incident.

This allows the GPS System to continue to operate while the logfile data is analyze in another system.

UPLOAD GPS DATA version 404-2

The problem: To change the GPS data, the system must be setup with a monitor, mouse and keyboard and the *gps_data.csv* file must be modified.

The solution: Install a flash drive (named *USB_DISK*) with the new GPS data in the file *gps_data.csv*.

Reboot the system.

The system does the following:

- 1- Checks if there is a flash drive named *USB_DISK* in the system.
- 2- Saves a copy of the *logfile.txt* on the flash drive.
- 3- Saves a copy of the *gps1.py* on the flash drive.
- 4- Checks if there is a *gps_data.csv* file is on the flash drive
- 5- Checks if there is a file *est_dst.txt* on the flash drive.
- 6- Checks if there is a *datafile.txt* on the system drive.
- 7- Saves a copy of the results to the file *bootlog.txt* to both the */home/pi/gps_2017* directory and the flash drive.

rm_logfile.txt

You can delete the *logfile.txt* file from the system drive (the SD Card) automatically when the system boots up by placing a file called *rm_logfile.txt* on the flash drive. Remember that the *logfile.txt* is always saved to the flash drive and if the *rm_logfile.txt* is on the flash drive it will be erased from the SD Card. Note that the *rm_logfile.txt* is also erased from the flash drive. It must be placed on the flash drive each time you want to erase the *logfile.txt*. It does not matter what is in the *rm_logfile.txt*. As long as this file is present on the flash drive, it will erase the *logfile.txt* on the system drive (the SD card) and then erase the *rm_logfile.txt* from the flash drive. Just copy any file to *rm_logfile.txt*.

For example: *cp est.txt rm_logfile.txt*

Remember that the audio files listed in the *gps_data.csv* file must be present in the */home/pi/gps_2017/audio_tracks* directory.

Use the utility *check_for_audio_files.py* to verify that the audio clips are correct.

Note: There is a 4 second delay during boot up. Otherwise the system will not see the flash drive!

There is another 10 second delay to allow the operator to read the display output of the script if a monitor is connected to the system.

Tip:

Aside from the est.txt and the dst.txt files on the flash drive, copy the versions of the gps_data.csv files with descriptions on the flash drive. This will make it easier to change the gps_data.csv file on another computer.

Other Stuff

Text to speech software has improved dramatically in the last few years. These online programs will accept a text file and convert the text to an audio file. As of 6/9/23, the website:

https://speechify.com/text-to-speech-online/?landing_url=https%3A%2F%2Fspeechify.com%2Ftext-to-speech-online%2F is what I have been using.

You will notice that in the program, when ever files are accessed or loaded, the absolute path is used ie: ***file = open('/home/pi/gps_2017/logfile.txt','a')***.

This was done because of the way the program was launch on powering up the Raspberry Pi using the ***autoexec.sh*** script called by the ***.desktop*** file in the ***autostart*** folder. See Appendix 8, section 9 for more details. If the program was run from the ***gps_2017*** directory, relative paths could be used.

The software needed to read the GPS receiver can be installed from this link:

<http://www.danmandle.com/blog/getting-gpsd-to-work-with-python/>

There is another procedure that I used to install the GPS receiver from this link:

<http://www.instantsupportsite.com/self-help/raspberry-pi/raspberry-globalsat-353s4-install/>

This worked when the initial installation was performed on both the 2 and the 2B systems.

However, I tried it recently and found that the `/lib/udev/gpsd.hotplug` file does not exist and I was unable to read the data from the receiver.

The SD chip used on the Raspberry Pi 2B works on the 3B and the Pi Zero.

If a Raspberry Pi 2B or 3B are used the time and date are correct

If a Raspberry Pi 2 is used the time will be correct, but the date is off by 7168 days!

Once this is installed, the system is functional and can be used.

Notes:

The system can play a different audio track at a location depending in the direction of the train. There are times that you may not want to play a track in one direction. In the folder `audio_clips` there is a file named ***silent.mp3*** for this purpose.

The system to date only monitors the latitude data. When the train proceeds past Bonsal Crossing Rd., the train' heading is approximately north-west.

In warm weather, the announcer usually sits in north end of car 101. In cold weather he sits in the caboose. He offset between these two positions is .7 times the distance between these two points.

Or 150 ft. Times .7 or 105 ft. This translates into .00027degrees.

The GPS receiver chosen is the Globecast BU353.

BU-353-S4



It is a sealed weather proof unit with a USB interface and costs about \$35.

<http://www.instantsupportsite.com/self-help/raspberry-pi/raspberry-globalsat-353s4-install/>

Note: This installation worked fine when it was originally used in August 2016.

It is not working at this time.

There are also problems with it on the RaspPi-2B. The RaspPi-2 works fine. See anomalies section for details and solution

The program uses python code developed by Dan Mandle. Here is the link:

<http://www.danmandle.com/blog/getting-gpsd-to-work-with-python/>

The first few lines of for **gps1.py** code are shown below:

import argv was used to display the file name of the program automatically

The remainder of the imports were from Dan Mendle.

A special thank you to Robert Malkin of Duke University. He is a volunteer at the Visual World Investigate Lab at the Museum of Natural Sciences in Raleigh, NC. He was very helpful in making the **ctrl-c** routine work. I could not have done this without him.

System Setup

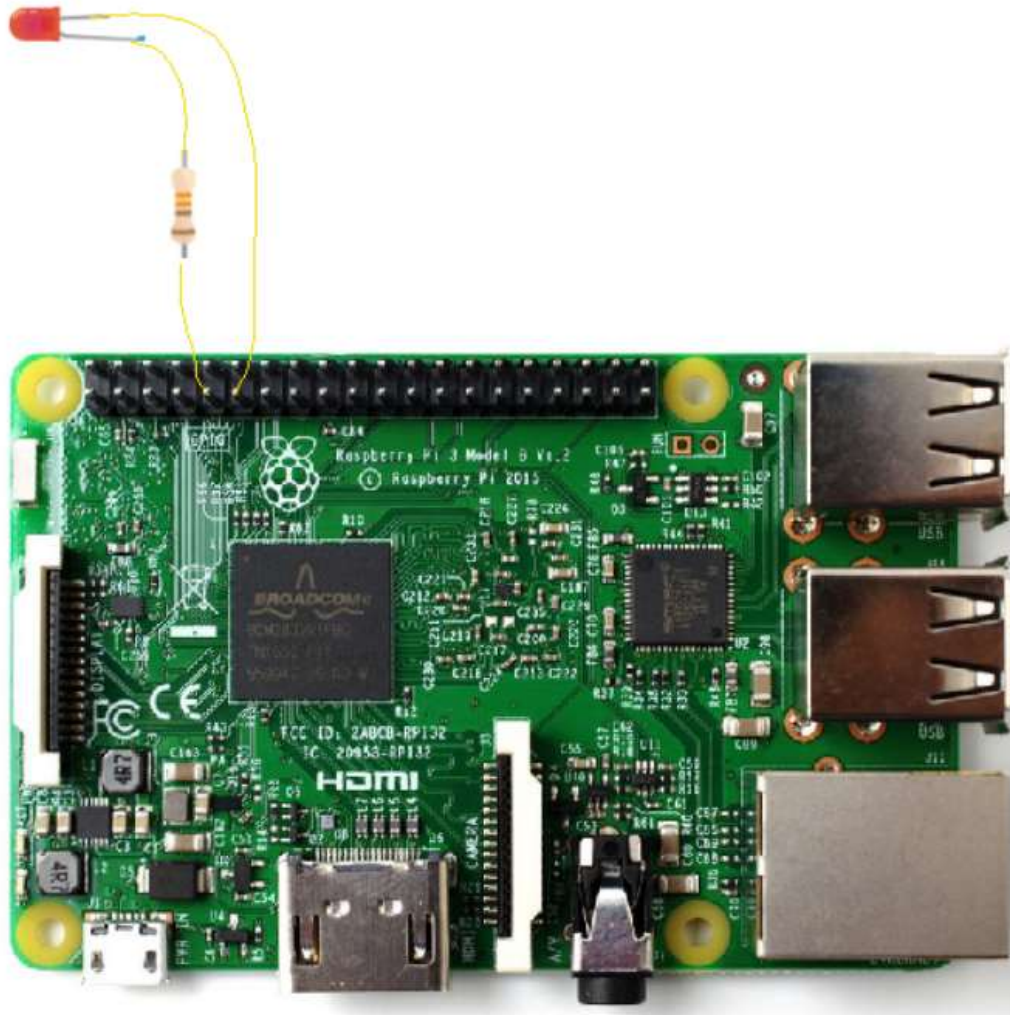
Hardware:

- Raspberry Pi 3 with LED connected to _____
- Globcast BU353 GPS receiver
- Power supply
- USB keyboard

Software:

- Pre-configure 16 gig SD chip.

Adding the LED



GPS description - Other details

Auto Upload

Delays the running of the shell script for 3 seconds.

Otherwise the script does not run! A delay of 1 second does not work

A delay of 2 seconds works. Set the delay to 3 seconds.

When the [cp_stuff.sh](#) is copie, it sometimes must be made executable

readtime module

The **est_dst.txt** file contains one line.

For Eastern Standard time it is **5EST**

For Daylight Savings time it is **4DST**

The file is read with the "with open(....."

The line is converted to a string with `y= str(content)`

The offset is converted to an integer from `y`

The timetag is stripped from the string `y`

The `gps_data.csv` file is imported by

from `load_csv_data import` command

The file is [load_csv_data.py](#)

The `EDST(utc-gpsd)` function converts the time to every day format.

The format of `utc-gpsd` is : 2018-04-08T22:43:10.000Z

Which outputs DST = 6:42:10 PM 04/8/18

There is an issue with this function during the last day of the month

For the last day of month, before midnight, the day is set to zero

ie 04/0/18. It should be 03/31/18

.....

Revision 409-2-5 12/13/23

Added variable *pre_dep_vol* which controls the volume level of the pre-departure audio tracks. This the default value of this variable extracted from the *gps_data.csv* file. This value can be changed using the *ctrl-c* function *prevol*.

The *build_file_names3.py* was incorporated in to the program and is no longer needed. See Appendix 13 for details.

This version changed the audio output to both for the Raspberry Pi Zero.

Revision 7/24/2023

Version 409-2-4 Fix error in Ctrl-c when GPS is not locked

If GPS is not locked and Ctrl-c is accessed, the system crashes.

This happens when the date and time of the Ctrl-c action is logged in the datafile.txt.

Changed From

To

```
else:
    print 'No Keyboard connected. Saving Data'
    datapoint = 'No keyboard'
    threading.Thread(target=play_background_clip).start()
    #time.sleep(2)
    GPIO.output(11,0)
    file.write(datapoint+',')
    if gps_lock >0:
        file.write(EDST(gpsd.utc)+'\n')
    else:
        file.write('No time available ' + '\n')
    file.close()

else:
    print 'No Keyboard connected. Saving Data'
    datapoint = 'No keyboard'
    threading.Thread(target=play_background_clip).start()
    #time.sleep(2)
    GPIO.output(11,0)
    file.write(datapoint+',')
    if gps_lock >0:
        file.write(EDST(gpsd.utc)+'\n')
    else:
        file.write('No time available ' + '\n')
    file.close()
```

Revision 4/28/20 *gps-407-3-2.py*

The update uses the function *dst(gpsd.utc)* to determine the starting dates of DST and EST. This eliminates the need for the *find_dst_import.py* and *read_data.txt* files.

Revision 3/20/20 *gps-407-3.py* added auto EST-DST correction. This uses the *find_dst_import.py* routine to find the dates of DST and EST for that year. The file *find_dst_import.py* file was added the *gps_2017* directory and the highlighted lines were added to make the *gps-404-3.py* file. When the GPS receiver has locked on to the satellites, the date information is written to the *read_data.txt* file. This data is read by the *find_dst_import.py* to determine the off_set. This is only performed once, since the *gps_lock* and *check_timetag* variables are set to 1.

if latt >35 and gps_lock == 0:

```
    gps_lock = 1
    print 'GPS Locked'
    file = open('/home/pi/gps_2017/logfile.txt','a')
    file.write(EDST(gpsd.utc) + ' '+ 'GPS Receiver Locked \n')
    file.write(' '+str(script))
```

```
file.close()
file = open('read_data.txt', 'w') # Saves time for find_dst_import
file.write(gpsd.utc)
file.close()
```

```
check_timetag = 1
from find_dst_import import tag
timetag = tag
if tag == "DST":
    off_set = 4
else:
    off_set = 5
```

The `est_dst.txt` file is not needed and was deleted from the `gps_2017` directory

The highlighted lines were deleted since they were no longer needed:

`else:`

```
    print 'NO DISK FOUND.'  
    file.write('NO DISK FOUND.'+'\n')  
    time.sleep(3)
```

```
#####
```

```
##### Read offset and timestamp from est_dst.txt
```

```
##### readtime.py
```

```
with open('/home/pi/gps_2017/est_dst.txt') as g:
```

```
    content = g.readlines() # reads the data
```

```
y=str(content) # converts the data to a string
```

```
offset = int(y[2:3]) # strips the offset data
```

```
timetag= y[3:6] # strips the timetag data
```

```
g.close()
```

```
#####
```

```
from load_csv_data import*
```

```
def EDST(utc): # Strips time from gpsd.utc
```

```
    timee=utc[11:19]
```

```
    hr=utc[11:13]
```

See Appendix 11 for the details on finding the DST dates in the `find_dst_import.py`.

Revisions : 2/2/18 added autosave logfile to USB_DISK

The latest version is **gps-404.py**. This has the auto save of the **logfile.txt** to a flash drive with the name **USB_DISK** .

9/18/18 Version gps-405 removed the cp_stuff.sh file and included the features within the main program.

10/30/18 The latest version is 405-5

12/11/18 revised play.py Enter index number instead of file names . Requires the text_file_2_array.py file

Created playcsv.py to play the audio files in current gps_data.csv file, except the silence_0001.wav files.

1/16/19 Added the thread to play the audio clips

Added

```
#####  
#Creates a thread to play an audio Clip  
def play_clip():  
    playing_clip = 1  
    os.system(xxx %sound)
```

Added:

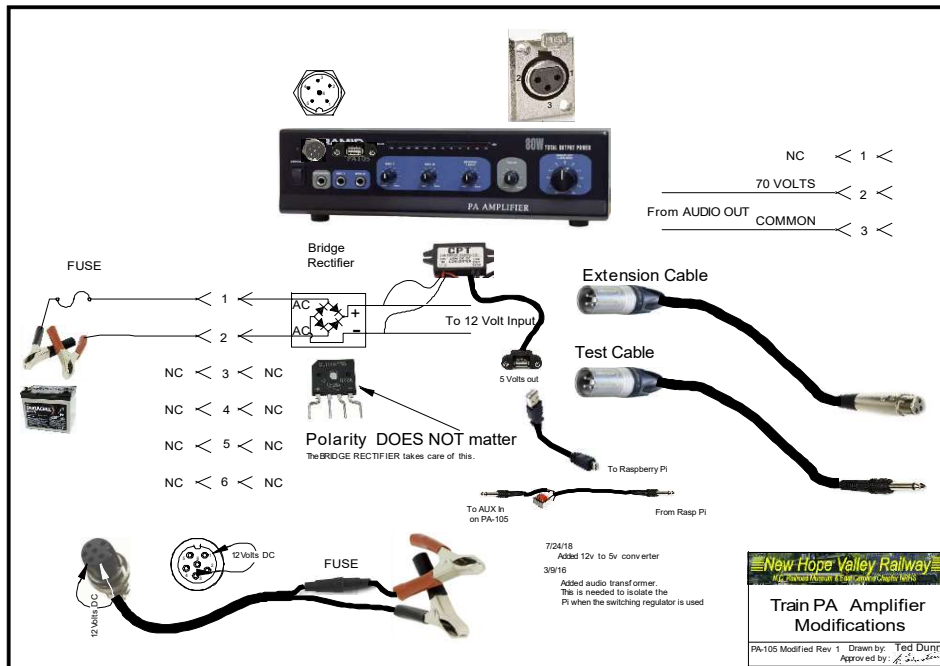
```
file = open('/home/pi/gps_2017/logfile.txt','a')  
for x in range(0,len(locs)):  
    if abs(lats[x] -latt) < delta:  
        #os.system('killall omxplayer.bin') #stops playing audio clip  
        playing_clip = 0 # Stops extra flashing  
        time.sleep(.5)  
        folder = '/home/pi/gps_2017/audio_tracks/'  
        files = mp3s_nb[x]  
        gain[x] = nb_gain[x]  
        if direction == 's':
```

```

files = mp3s_sb[x]
gain[x] = sb_gain[x]
sound = folder +files #
print x, files, gain[x]
###xxx = 'omxplayer --vol '+ gain[x] +' %s'
xxx = 'omxplayer -o local --vol '+ gain[x] +' %s'
if files <>"silence.0001.wav":
    os.system('killall omxplayer.bin') #stops playing audio clip
if files <>"silence.0001.wav":
    threading.Thread(target=play_clip).start() #Plays clip throughj threas
#os.system(xxx %sound) # used to lay the clip
lats[x] =lats[x] + 10 # takes the last played file out of list
file.write('\n' + str(x)+',' ) # The number of the location list

```

3/9/19 When the Raspberry Pi is powered by the switching regulator installed in the amplifier, it creates high frequency oscillations that show up as noise in the audio. To eliminate this interference, a 1:1 audio transformer was added between the output of the Raspberry Pi and the input of the amplifier.



12/18/19

Added a background audio track to the system. Version **gps-407-1.py**.

A thread was added to play an audio track continuously

def play_background_clip():

```
    os.system("mplayer /home/pi/gps_2017/audio_tracks/xmas_90_minute_track.mp3 -loop 10 -af  
    volume=-20")
```

```
    threading.Thread(target=play_background_clip).start()
```

Since we are using omxplayer for the individual audio tracks and we use the **killall** command to stop omxplayer, we are using mplayer to play the background music track. Note that it was necessary to use the absolute path for the location of the audio track. If we use the relative path, the system would not play the track on boot up.

Table of Contents

Appendix 1	Omxplayer commands
Appendix 2	USB_DISK boot routine
Appendix 3	Pi-Zero analog audio
Appendix 4	Controlling the Pi with PUTTY and FILEZILLA
Appendix 5	Controlling the Pi with VNC
Appendix 6	Setting static IP address
Appendix 7	Stopping an audio clip
Appendix 8	Building the GPS system on the SD card.
Appendix 9	Run a python program on boot up
Appendix 10	Mplayer commands
Appendix 11	Auto-correct for Daylight Savings Time
Appendix 12	Emulate a key press in a Python script.
Appendix 13	Play audio tracks before the train leaves the station.
Appendix 14	Using VNC with 3.5 inch display

Appendix 1 OMXPLAYER keyboard commands (<https://elinux.org/Omxplayer>)

Key Action

1	decrease speed
2	increase speed
<	rewind
>	fast forward
z	show info
j	previous audio
stream	
k	next audio stream
i	previous chapter
o	next chapter
n	previous subtitle
stream	
m	next subtitle
stream	
s	toggle subtitles
w	show subtitles
x	hide subtitles
d	decrease subtitle
delay (- 250 ms)	
f	increase subtitle
delay (+ 250 ms)	
q	exit omxplayer
p / space	pause/resume
-	decrease volume
+ / =	increase volume
left arrow	seek -30 seconds
right arrow	seek +30 seconds
down arrow	seek -600 seconds
up arrow	seek +600 seconds

OMXPLAYER documentation:

<https://www.raspberrypi.org/documentation/raspbian/applications/omxplayer.md>

To adjust the volume use the command: `omxplayer --vol 'xxxx' 'filename'`

Where xxx is in millibels. For a 6 db change xxxx= 600.

A +6 decibel (db) change doubles the audio level. A -6 db change halves the audio level.

http://coretechgroup.com/dbm_calculator/

zero dbm is 1 milliwatt in 600 ohms is .7745 vrms			
	VRMS	P-P	
+6 dbm	1.549193	4.38178	
+3 dbm	1.095445	3.098387	
0 dbm	0.774597	2.19089	
-3 dbm	0.547723	1.549193	
-6 dbm	0.387298	1.095445	

Appendix 2 USB Boot routine

When a USB flash drive is installed in a Raspberry Pi, it mounts in the */media/pi* folder.

Opens *bootlog.txt* file to write.

Checks for the presence of the USB Flash Drive with the name *USB_DISK*.

If True:

Copies *logfile.txt*, *gps1.py* and *gps_data.csv* as *gps_data_downloaded* to Flash Drive.

Note: This will overwrite any existing files on the USB Flash Drive.

Checks if ***rm_logfile.txt*** is on the Flash Drive.

If True:

Deletes the ***logfile.txt*** from the system SD card.

Deletes the ***rm_logfile.txt*** from the Flash Drive.

Checks if ***gps_data.csv*** is on Flash Drive.

If True:

Saves the ***gps_data.csv*** file as ***old_gps_data.csv*** on SD card.

Saves the ***gps_data.csv*** file as ***old_gps_data.csv*** on FD.

Copies ***gps_data.csv*** file from Flash Drive to the system SD card.

Deletes the ***gps_data.csv*** file from the Flash Drive.

Checks if the ***est_dst.txt*** file in on the Flash Drive.

If True:

Copy ***est_dst.txt*** to system SD card

Delete ***est_dst.txt*** file from Flash Drive.

Checks if there is a ***datafile.txt*** file on the system SD card.

If True:

Copies ***datafile.txt*** to the Flash Drive.

Checks if there is a ***rm_datafile.txt*** on the Flash Drive.

If True:

Deletes the ***datafile.txt*** from the system SD card

Deletes the ***rm_datafile.txt*** from the Flash Drive.

Closes ***bootlog.txt*** file

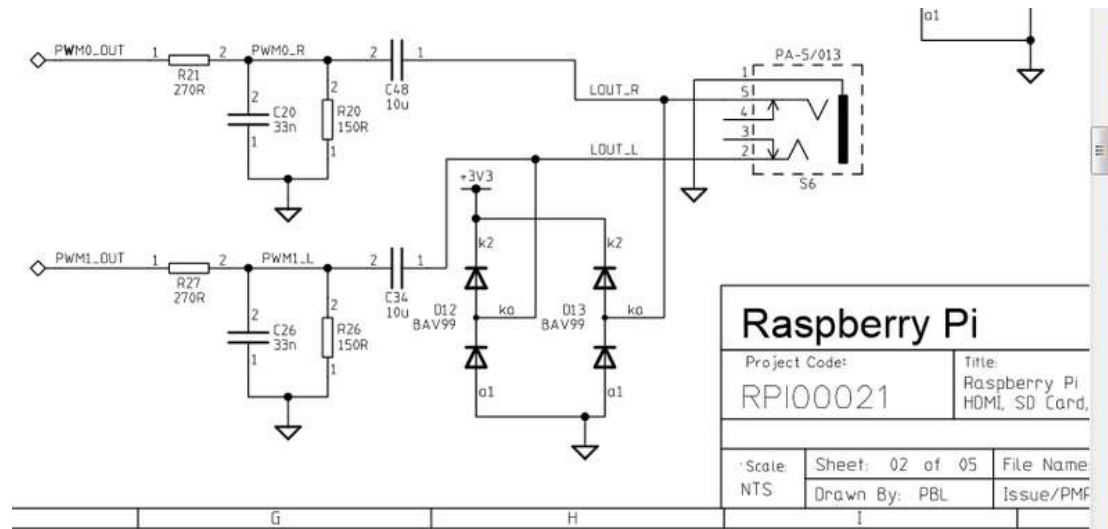
Copies ***bootlog.txt*** file to Flash Drive.

Appendix 3 Pi-Zero analog audio output

How Other Pi's Create Audio

<https://learn.adafruit.com/introducing-the-raspberry-pi-zero/audio-outputs>

GPIO #18 is also known as PWM0 and in the original Pi was coupled with a very basic RC filter to create the audio output:

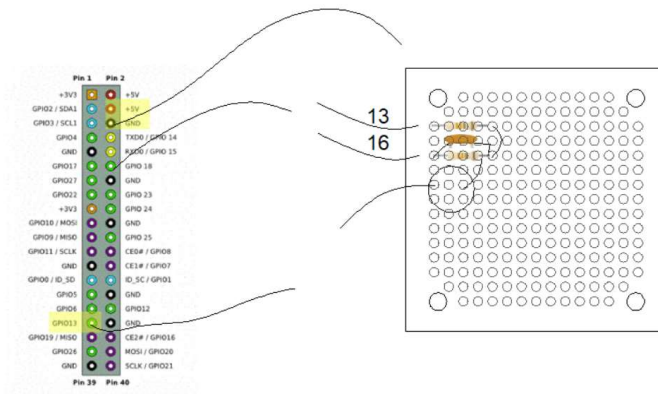
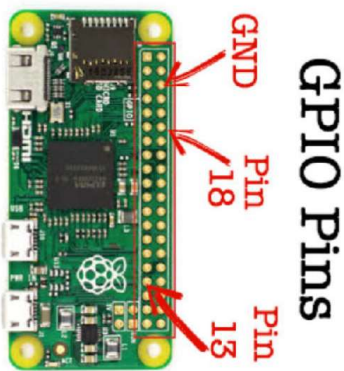
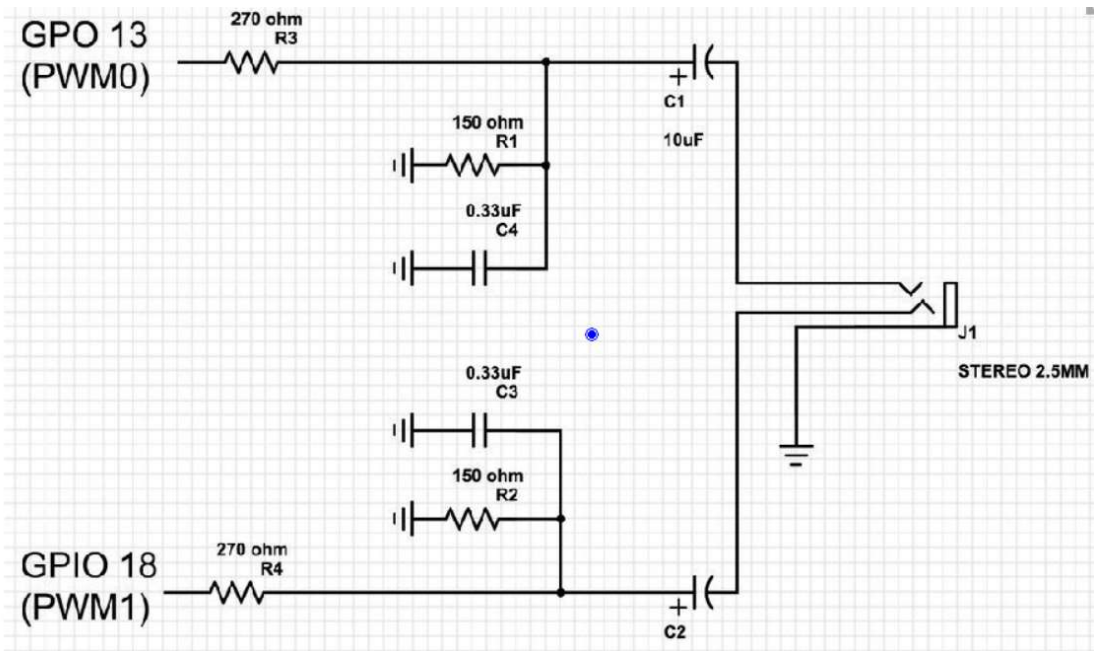


If you don't mind getting a few 150 and 270 ohm resistors, and two each of about 33nF (also known as 0.033uF) and 10uF capacitors, you can basically recreate those two filters.

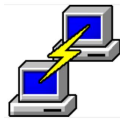
Now all you need is access to PWM0_OUT and PWM1_OUT, which are...on GPIO #40 and #45 and are not brought out on the Pi Zero. Tragedy? Give up? No! You can get to **PWM0** on GPIO #18 (ALT5) and **PWM1** on GPIO #13 (ALT0) or GPIO #19 (ALT5)

*Simply adding the following line to your **/boot/config.txt** will reconfigure the pins at boot without any external software or services:*

dtoverlay=pwm-2chan,pin=18,func=2,pin2=13,func2=4



Appendix 4 Controlling the Rasp from a remote computer.



Putty



FileZilla

Putty is an open source software that will allow you to open a text window on a remote computer. You must know the IP address of the Raspberry Pi (see below) and the user name and password. They are respectively **pi** and **raspberrypi**. Simply download putty and install on your computer

FileZilla is a free FTP client that needs to be installed on the remote computer.

After the FTP server software has been installed on the Raspberry Pi, the FTP client can be used to transfer files between the two systems. See the link below.

For these programs to load, you need to know the IP address of the Rasp Pi.

In a text window enter **ifconfig**

```
pi@raspberrypi: ~  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Tue Nov 13 08:22:33 2018 from 192.168.1.106  
pi@raspberrypi:~$ ifconfig  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:65536  Metric:1  
          RX packets:294137 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:294137 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1  
          RX bytes:70934662 (67.6 MiB)  TX bytes:70934662 (67.6 MiB)  
  
wlan0     Link encap:Ethernet  HWaddr 00:0f:13:37:0f:81  
          inet addr:192.168.1.108  Bcast:192.168.1.255  Mask:255.255.255.0  
          inet6 addr: fe80::f2b8:901d:4cb7:784a/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MPU:1500  Metric:1  
          RX packets:32319 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:4926 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:5948309 (5.6 MiB)  TX bytes:781003 (762.6 KiB)  
  
[1]+  Done                  /home/pi/singing_pumpkins/./gpssock  
pi@raspberrypi:~$
```

Install putty on the pc and you will be able to control the Rasp Pi from a text window on the pc.

The SSH must be enabled in the Rasp Pi. In a text widow, enter raspi-config.....

ftp or File Transfer Protocol. This will allow for moving files from the Rasp Pi to and from a windows system. An ftp server is installed on the Rasp Pi and an ftp client is installed on the PC. **FileZilla works** well.

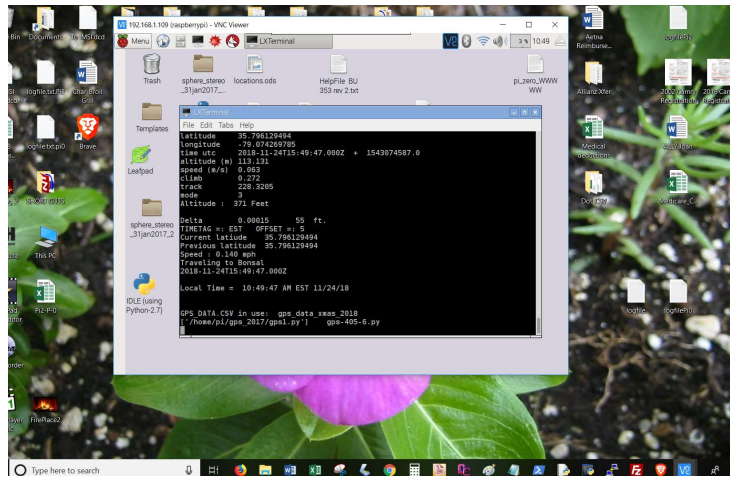
See this link: <https://howtoraspberrypi.com/setup-ftp-server-raspberrypi/> for details on the installation of the ftp server on the Rasp Pi.

In a text window enter ***sudo apt install proftpd***.

Appendix 5 GUI Control of the Raspberry Pi from a remote computer.



Real VNC Viewer



Detailed installation:

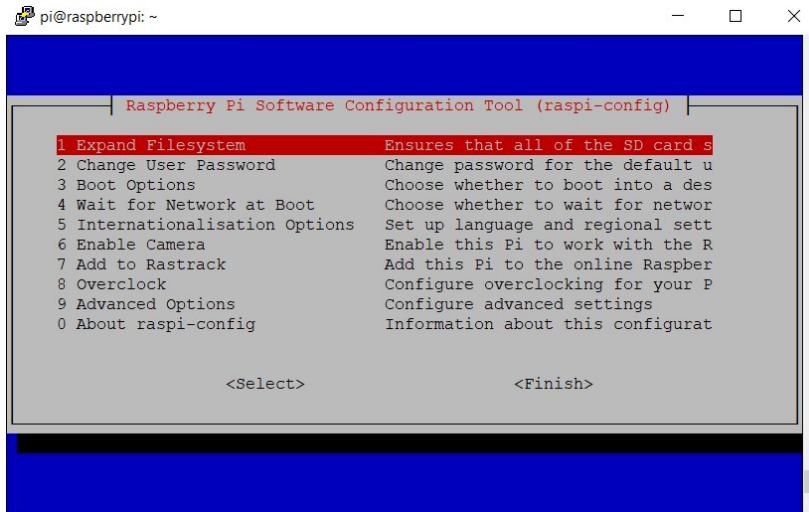
<https://www.raspberrypi.org/documentation/remote-access/vnc/README.md>

Run ***sudo apt-get update***

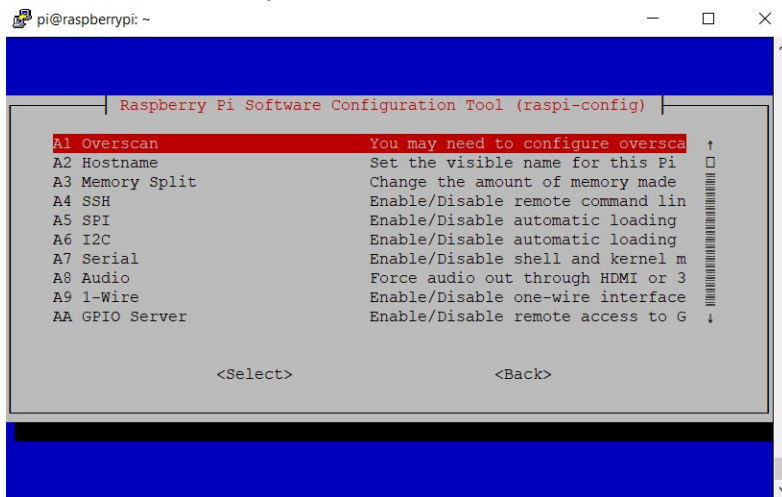
Run ***sudo apt-get install realvnc-vnc-server realvnc-vnc-viewer***

Run ***sudo raspi-config***

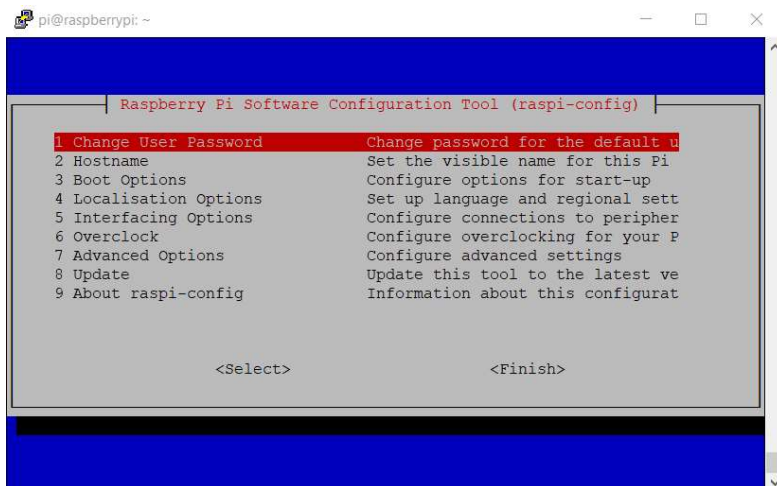
After the install was completed, I ran *raspi-config* but could not find VNC!



Select Advanced options



Use down arrow to scroll down and show additional options, to A0 Update. It is below the last item shown. Select A0 Update. This will take several minutes to complete. It will then display:

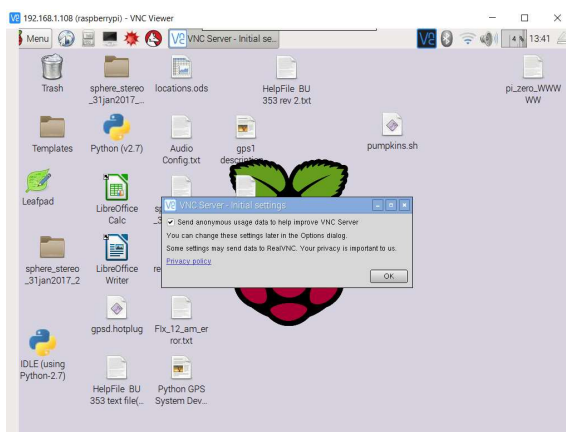


Now you can select Interfacing Options.

Select VNC and enable it.

Select finish.

Reboot the Rasp Pi and you will see:



If you have RealVNC viewer on the controlling unit, it will connect!

Appendix 6 Setting Static IP Address

Works on Pi-Zero. **NOT on Pi-3:**

<https://www.raspberrypi.org/learning/networking-lessons/rpi-static-ip-address/>

If the desired static IP address of the raspberry pi is 192.168.1.201,

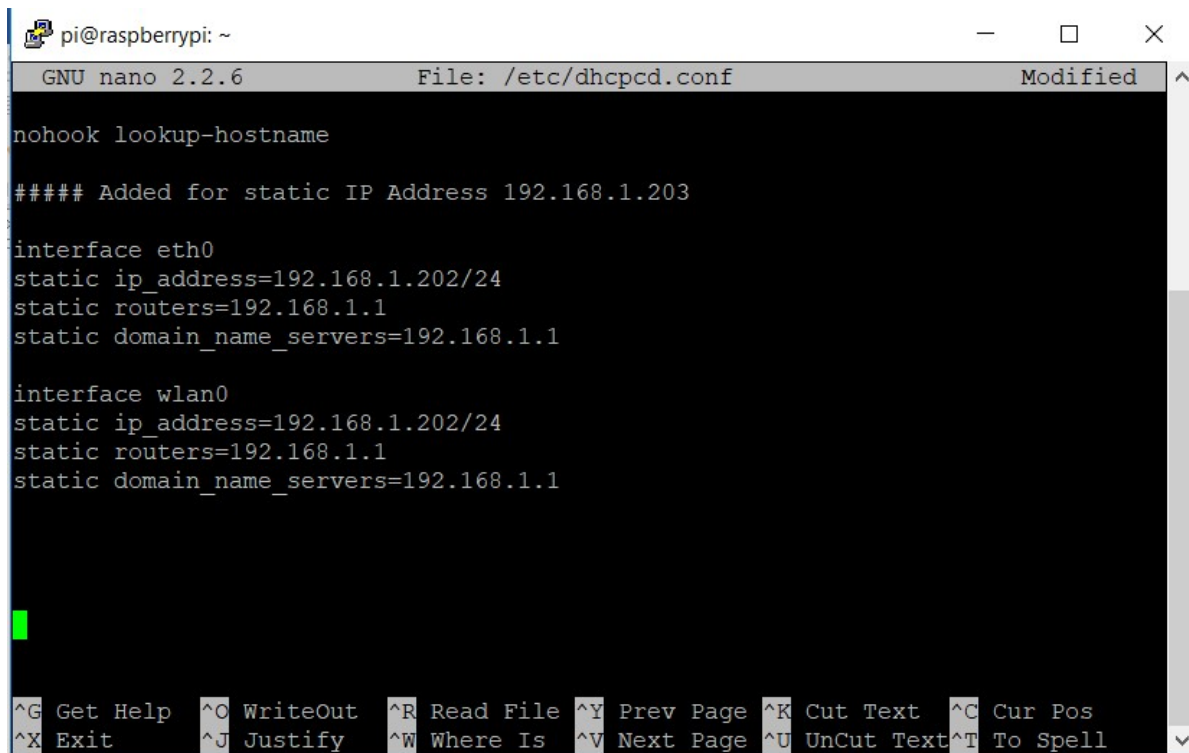
Add the following at the bottom of the `/etc/dhcpd.conf` file:

```
interface eth0

static ip_address=192.168.0.201/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1

interface wlan0

static ip_address=192.168.0.201/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1
```



```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/dhcpd.conf Modified
nohook lookup-hostname
##### Added for static IP Address 192.168.1.203
interface eth0
static ip_address=192.168.1.202/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1

interface wlan0
static ip_address=192.168.1.202/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1

^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 Text ^T To Spell
```


Appendix 7. Stopping the audio clip *gps-406-6-1.py*

There is a problem if the audio clip is longer than the time between locations. If this is the case, the clip will continue to play and the next clip will be skipped.

In the current system, the program runs until it finds a location that matches one in the `gps_data.csv` file. It then halts the program and plays the clip. When the clip is finished, the program continues to run until it finds another location. If the audio clip is long and is playing while another location is past, the system will skip that location. By using a thread to play the audio clip, the system continues to run while the thread plays the audio clip. It can detect the next location and stop the clip using the `killall` function. The current location's clip can then be played.

To solve this problem, a thread was added (`play_clip()`) to play the audio clip.

```
def play_clip():  
    print'Beginning Thread play_clip'  
    os.system('xxx %sound')
```

When a new location is detected, the command :

```
os.system('killall omxplayer.bin')
```

is issued to stop `omxplayer`, if it is still running.

Then the command:

```
threading.Thread(target=play_clip).start()
```

Is issued to play the next clip

There is of course another problem. There are place holder silent audio clips in the `gps_data.csv` file. The name of these clips is `silent_0001.wav`

The program will look to see if the audio clip starts with ***sile*** . If it find this it skips stopping the playback. Below are the 4 lines of code that inhibit the thread when a silent file is detected

```
if files[0:4] <>"sile":  
    os.system('killall omxplayer.bin') #stops playing audio clip
```

```

if files[0:4] <>"sile":
    threading.Thread(target=play_clip).start() #Plays clip througj threas

```

Below is the segment of the code that plays the audio clips at a specific latitude

```

file = open('/home/pi/gps_2017/logfile.txt','a')
for x in range(0,len(locs)):
    if abs(lats[x] -latt) < delta:
        #os.system('killall omxplayer.bin') #stops playing audio clip
        playing_clip = 0 # Stops extra flashing
        time.sleep(.5)
        folder = '/home/pi/gps_2017/audio_tracks/'
        files = mp3s_nb[x]
        gain[x] = nb_gain[x]
        if direction == 's':
            files = mp3s_sb[x]
            gain[x] = sb_gain[x]
        sound = folder +files #
        print x, files, gain[x]
        ###xxx = 'omxplayer --vol '+ gain[x] +' %s'
        xxx = 'omxplayer -o local --vol '+ gain[x] +' %s'
        if files[0:4] <>"sile":
            os.system('killall omxplayer.bin') #stops playing audio clip
        if files[0:4] <>"sile":
            threading.Thread(target=play_clip).start() #Plays clip througj threas
#os.system(xxx %sound) # used to lay the clip
    lats[x] =lats[x] + 10 # takes the last played file out of list
    file.write('\n' + str(x)+',') # Tne number of the location list
    file.write (str(gpsd.fix.speed *2.236)[0:5]+' mph' + ' ' + direction +',')
    file.write (str(gpsd.fix.track)+',')
    file.write(EDST(gpsd.utc)+',')
    file.write(' '+files +',') #the audio file name
    file.write (gain[x]+',')
    file.write(' '+locs[x]+',') #the location name
    file.write(' '+str(gpsd.fix.latitude)+',')
    file.write(' '+str(version)+'\n')
    #lats[x] =lats[x] + 10 # takes the last played file out of list
    last = x ##### The last location's lats to reset it
    x = x+1
    time.sleep(1)
file.close()
lattold = latt

```

Appendix 8: Building the GPS system on the Raspberry Pi SD card

1- Copy NOOBS to SD card

Use a 16 gig type 10 SD chip.

Format the SD chip with SDFormatter

Download the latest NOOBS zip file.

Extract the zip file and copy the files to the SD memory card.

2- Boot Rasp Pi Select only Raspian

3- Install gps software

Do not connect the gps receiver

At the command line enter:

```
sudo apt-get install python gpsd gpsd-clients
```

The software will install.

Enter "Y" when prompted

4- Install mplayer software

At the command line:

```
sudo apt-get install mplayer
```

The software will install.

Copy the gps_2017 folder containing the operating program from a flash drive to the /home/pi directory

5- Copy the gps_2017 folder containing the operating program from a flash drive to the /home/pi directory

Instructions to create the gpssock file.

This file should already be in the gps_2017 folder

Create the **gpssock** file in the **gps_2017** folder

This file contains three lines:

either

```
sudo killall gpsd
```

```
sudo gpsd /dev/ttyUSB0 -F /var/run/gpsd.sock
```

```
sudo service ntp restart
```

or

```
sudo systemctl stop gpsd.socket
```

```
sudo systemctl disable gpsd.socket
```

```
sudo gpsd /dev/ttyUSB0 -F /var/run/gpsd.sock
```

Make it executable:

```
chmod +x gpssock
```

5-Install the GPS receiver

6- Run the gpssock file

At the command line, enter:

```
./gpssock
```

The **.** / is the command to run an executable file.

7- Run the trest program. At the command line enter::

cgps -s

You should see the gps data

```
pi@raspberrypi: ~
File Edit Tabs Help
Time: 2019-03-23T18:25:47.000Z
Latitude: 35.796132 N
Longitude: 79.074209 W
Altitude: 363.9 ft
Speed: 0.0 mph
Heading: 0.0 deg (true)
Climb: 0.0 ft/min
Status: 3D FIX (41 secs)
Longitude Err: +/- 32 ft
Latitude Err: +/- 58 ft
Altitude Err: +/- 216 ft
Course Err: n/a
Speed Err: +/- 79 mph
Time offset: -0.763
Grid Square: FM051t
PRN: Elev: Azim: SNR: Used:
16 46 039 31 Y
27 66 096 33 Y
7 31 309 28 Y
4 39 303 30 Y
8 56 174 33 Y
9 51 283 24 Y
23 57 220 32 Y
26 21 057 19 Y
31 01 112 00 N
11 06 180 00 N
21 03 043 00 N
131 32 232 00 N
```

7 Add the line ***/home/pi/gps_2017/.gpssock &*** at the bottom of the ***/etc/profile*** file.

NOTE: The apersand (&) character is important.

This file is needed otherwise the gps receiver will not communicate with the Raspberry Pi. The exact reason for this is unclear.

7-a **Alterative to the above, add the line:**

os.system('./gpssock') (Added on gps_407-2-2-1-1)

This restarts the gps software without changing the ***profile*** file.

The downside to this is that the ***cgps -s*** program will not run until ***gps1.py*** is run.

8- Install mplayer. This program is needed to play the background music track. (Added on gps_407-1)

At the command prompt, enter: ***sudo apt-get install mplayer***. It may take 10 minutes to load.

9- Launch the GPS program

At the command line enter:

sudo python /home/pi/gps_2017/gps1.py


```
pi@raspberrypi: ~
File Edit Tabs Help
--- New Hope Valley Railway ---
1553365969.0
GPS system for the New Hope Valley Railway
Created by Ted Dunn with software written by Dan Mandle
With the help of Robert Malkin of Duke University
-----
latitude    35.79610266
longitude   -79.074232628
time utc    2019-03-23T18:32:49.000Z + 1553365969.0
altitude (m) 115.55
speed (m/s)  0.0
climb        0.0
track        0.0
mode         3
Altitude : 379 Feet

Delta        0.00015      55 ft.
TIMETAG =: DST  OFFSET =: 4
Current latiude  35.79610266
Previous latitude 35.796105701
Speed : 0.0 mph
Traveling to Bonsal
2019-03-23T18:32:49.000Z

Local Time = 02:32:49 PM DST 03/23/19

GPS_DATA.CSV in use: halloween_2017
['/home/pi/gps_2017/gps1.py']
gps-405-5-1.py --- Using isdir() instead of cp_stuff.sh
```

9- **Setup to Auto start on power up**

In order to have the system run automatically when power is applied the file: **.desktop** must be in the folder **/home/pi/.config/autostart**.

Create the **autostart** folder in the **.config** directory

Create the **.desktop** file in the **autostart** directory

Here are the three lines in the **.desktop file**:

Revised 3/23/19 for NOOBS 3.0

[Desktop Entry]

Type = Application

Exec = lxterminal -e sudo python /home/pi/gps_2017/gps1.py

The **.config** folder exists in the **/home/pi** folder. Create the **autostart** folder in the **.config** folder. Then create the **.desktop** file in the **autostart** folder.

Appendix 9 Run a Python program automatically on startup

This method will start a Python Program in a terminal widow.
The program is **xyz.py** and it is in the **/home/pi/abc** directory.

To do this, an **autoexec.sh** file is created in the startup directory, **/home/pi**. This is an executable linux shell file. It will run the Python file in the correct directory.

The name autoexec was chosen in remembrance of the unique DOS batch file, autoexec.bat. This file was automatically run on bootup in the DOS operating system. In Linux, the name of the file is not important.

To run this shell script on startup, it must be in the **/home/pi/.config/autostart/desktop** file.

- 1- Create the **autostart** directory in the **/home/pi/.config** directory.
 - a. Start from the **/home/pi** directory.
 - b. Change to the **.config** directory: **cd .config**
 - c. Create the **autostart** directory: **mkdir autostart**

- 2- Create the **.desktop file** in the autostart directory:
- 3- Move to the autostart directory
- 4- **cd autostart**
- 5- Create the **.desktop** file
- 6- **sudo nano .desktop**
 - a. Enter the follow 3 lines in the text editor:
 - i. **[Desktop Entry]**
Type = Application
Exec = lxterminal -e ./autoexec.sh (The **./** is used to run executable files.)

 - ii. Press **ctrl-o**
 - iii. Press **Enter** to save the file.
 - iv. Press **ctrl-x** to exit the text editor.

- 7- Create the **autoexec.sh** file to run the **xyz.py** Python program in the **/home/pi/abc** directory.
 - a. Start in the **/home/pi** directory.
 - i. Create the **autoexec.sh** file: **sudo nano autoexec.sh**
 - ii. Enter the following 2 lines:
cd abc
python xyz.py
 - iii. Press **ctrl-o**
 - iv. Press **Enter** to save the file.
 - v. Press **ctrl-x** to exit the text editor.

 - b. Make the **autoexec.sh** file executable **chmod +x autoexec.sh**.

Note. The autoexec.sh file is not necessary. To run the file it could be placed in the last line of the .desktop file: **Exec = lterminal -e ./python home/pi/abc/xyz.py.**

The advantage of the autoexec.sh file is that it makes it easier to change the autorun program.

- 1-The file to be edited is in the **/home/pi directory** instead of the **/home/pi/.config/autostart** directory.
- 2- Several **autoexe.xx** files can be save and easily be rename to the **autoexec.sh** file.

For example if one wishes to run a program name **gps1.py** which is located in the **/home.pi/gps_2017** directory.

For example if one wishes to run a program name **gps1.py** which is located in the **/home.pi/gps_2017** directory.

The autoexec.sh file in the /home/pi directory will have the following lines:

```
cd gps_2017  
sudo python gps1.py
```

Appendix 10 MPLAYER keyboard controls

<http://www.keyxl.com/aaa2fa5/302/MPlayer-keyboard-shortcuts.htm>

MPlayer Keyboard Shortcuts

Keyboard control

<- and ->	Seek backward/forward 10 seconds.
up and down	Seek forward/backward 1 minute.
pgup and pgdown	Seek forward/backward 10 minutes.
[and]	Decrease/increase current playback speed by 10%.
{ and }	Halve/double current playback speed.
backspace	Reset playback speed to normal.
< and >	Go backward/forward in the playlist.
ENTER	Go forward in the playlist, even over the end.
HOME and END	next/previous playtree entry in the parent list
INS and DEL (ASX playlist only)	next/previous alternative source.
p / SPACE	Pause (pressing again unpauses).
.	Step forward. Pressing once will pause movie, every consecutive press will play one frame and then go into pause mode again (any other key unpauses).

q / ESC	Stop playing and quit.
+ and -	Adjust audio delay by +/- 0.1 seconds.
/ and *	Decrease/increase volume.
9 and 0	Decrease/increase volume.
(and)	Adjust audio balance in favor of left/right channel.
m	Mute sound.
_ (MPEG-TS and libavformat only)	Cycle through the available video tracks.
# (DVD, MPEG, Matroska, AVI and libavformat only)	Cycle through the available audio tracks.
TAB (MPEG-TS only)	Cycle through the available programs.
f	Toggle fullscreen (also see -fs).
T	Toggle stay-on-top (also see -ontop).
w and e	Decrease/increase pan-and-scan range.
o	Toggle OSD states: none / seek / seek + timer / seek + timer + total time.
d	Toggle frame dropping states: none / skip display / skip decoding (see -framedrop and -hardframedrop).
v	Toggle subtitle visibility.
j	Cycle through the available subtitles.
y and g	Step forward/backward in the subtitle list.
F	Toggle displaying forced subtitles .
a	Toggle subtitle alignment: top / middle / bottom.
x and z	Adjust subtitle delay by +/- 0.1 seconds.
r and t	Move subtitles up/down.
i (-edlout mode only)	Set start or end of an EDL skip and write it out to the given file.
s (-vf screenshot only)	Take a screenshot.
S (-vf screenshot only)	Start/stop taking screenshots.
l	Show filename on the OSD.
! and @	Seek to the beginning of the previous/next chapter.
D (-vo xvmc, -vf yadif, -vf kerndeint only)	Activate/deactivate deinterlacer.

Hardware accelerated video output

1 and 2	Adjust contrast.
3 and 4	Adjust brightness.
5 and 6	Adjust hue.

7 and 8 Adjust saturation.

SDL Video Output Driver

c Cycle through available fullscreen modes.

n Restore original mode.

Multimedia Keyboard

PAUSE Pause.

STOP Stop playing and quit.

PREVIOUS and NEXT Seek backward/forward 1 minute.

GUI Support (if compiled in)

ENTER Start playing.

ESC Stop playing.

l Load file.

t Load subtitle.

c Open skin browser.

p Open playlist.

r Open preferences.

If TV or DVB support compiled

h and k Select previous/next channel.

n Change norm.

u Change channel list.

Navigate menus (if DVNAV support)

keypad 8 Select button up.

keypad 2 Select button down.

keypad 4 Select button left.

keypad 6 Select button right.

keypad 5 Return to main menu.

keypad 7 Return to nearest menu (the order of preference is: chapter->title->root).

keypad ENTER Confirm choice.

Teletext support (if compiled)

X Switch teletext on/off.

Q and W Go to next/prev teletext page.

mouse control

button 3 and button 4 Seek backward/forward 1 minute.

button 5 and button 6 Decrease/increase volume.

joystick control

left and right	Seek backward/forward 10 seconds.
up and down	Seek forward/backward 1 minute.
button 1	Pause.
button 2	Toggle OSD states: none / seek / seek + timer / seek + timer + total time.
button 3 and button 4	Decrease/increase volume.

3.11. Software Volume adjustment

Some audio tracks are too quiet to be heard comfortably without amplification. This becomes a problem when your audio equipment cannot amplify the signal for you. The `-softvol` option directs MPlayer to use an internal mixer. You can then use the volume adjustment keys (by default **9** and **0**) to reach much higher volume levels. Note that this does not bypass your sound card's mixer; MPlayer only amplifies the signal before sending it to your sound card. The following example is a good start:

```
mplayer quiet-file -softvol -softvol-max 300
```

The `-softvol-max` option specifies the maximum allowable output volume as a percentage of the original volume. For example, `-softvol-max 200` would allow the volume to be adjusted up to twice its original level. It is safe to specify a large value with `-softvol-max`; the higher volume will not be used until you use the volume adjustment keys. The only disadvantage of a large value is that, since MPlayer adjusts volume by a percentage of the maximum, you will not have as precise control when using the volume adjustment keys. Use a lower value with `-softvol-max` and/or specify `-volstep 1` if you need higher precision.

The `-softvol` option works by controlling the `volume` audio filter. If you want to play a file at a certain volume from the beginning you can specify `volume` manually:

```
mpplayer quiet-file -af volume=10
```

This will play the file with a ten decibel gain. Be careful when using the `volume` filter - you could easily hurt your ears if you use too high a value. Start low and work your way up gradually until you get a feel for how much adjustment is required. Also, if you specify excessively high values, `volume` may need to clip the signal to avoid sending your sound card data that is outside the allowable range; this will result in distorted audio.

Added the ability to play a background audio track.

The track must be in the /home/pi/gps_2017/audio_tracks folder

To adjust the sound level, a keyboard must be connected.

Increase volume 0 key

Decrease volume 9 key



To play a background audio track, enter the data on the next to the last row. Enter the name of the track in column 4, the gain in column 5 and the number of times the track needs to be repeated in column 7.

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7
35.69729441	-79.97	path_4	silence_0001.wav	0	silence_0001.wav	0
35.69766927	-79.97	new_hill_south_switch	silence_0001.wav	0	silence_0001.wav	0
35.69866409	-79.97	half_way_to_whistle	silence_0001.wav	0	silence_0001.wav	0
35.69965891	-79.97	new_hill_whistle_marker	silence_0001.wav	0	silence_0001.wav	0
35.79608	-79.97	Fearrington	silence_0001.wav	0	silence_0001.wav	0
40	79	Background	xmas_90_minute_track.mp3	-3	repeat	2
50	-79.97	Xmas_12/21/19 With Background	silence_0001.wav	0	silence_0001.wav	0

The last two lines in the gps_data.csv file are unique. The background audio track should be entered in the next to last line. The latitude of the bottom line should be set to 50 and the line above should be set to 40

Appendix 11 Find Daylight Savings Time *dst(gpsd.utc)* 4/28/2020

This python function, *dst(utc)*, calculates the start date of Daylight savings time and the start date of Eastern standard time, for the date entered.

DST begins on the second Sunday in March and Ends on the first Sunday in November.

Today's date is entered as a string in the format of yyyy-mm-dd.

The year, month and date are extracted and assigned to the variable *tday*.

The algorithm to find the start and end date of DST is as follows:

Find the number of the day of January first *jan1=date(y,1,1)* of the current year.

Find the day of the week for March 1.

Find the date of the second Sunday.

Find the number of the day for the second Sunday in March, DST start.

Find the day of the week for November 1.

Find the date of the first Sunday.

Find the number of the day for the first Sunday in November, EST start.

If the number of the day of today falls within DST start and EST start, today is on DST.

Otherwise, today is on EST

See Appendix 11-1 for all of the options of when DST starts and when EST starts.

The function imports the date functions from the standard library.

In Linux each day is assigned a number beginning on January 1, 1970

The number of the January 1 is assigned the variable *jan1*.

```
jan1 = date(yr,1,1)
```

It generates three dictionaries. The *dst{}* dictionary and the *est{}* dictionary contain the offset of the first day of the month to the start of DST in March and the start of EST in November. See Appendix 1.

The *dow{}* dictionary shows the days of the week based on the numeric value from datetime. 0 is Sunday and 6 is Saturday.

The current date *gpsd.utc* is in the format of yyyy-mm-dd.

The yr, mon, and da variables are extracted from gpsd.utc.

```
yr = int(gpsd.utc)[0:4]
```

As an example, if gpsd = 2020-03-17, yr =2020

```
mon = int(gpsd.utc)[5:7]
```

```
da = int((gpsd[8:10]))
```

Today is the variable **tday**.

```
tday = date(yr,min,da)
```

Today's day of the week is dayofweek

The number for the day of January 1 of the year above is assigned to the variable **jan1**.

```
Jan1 = date(yr,1,1)
```

The day of the week of March 1, of the year above is assigned to the variable mar1.

```
mar1 = datetime.date(yr, 3, 1).weekday() # days 0-6
```

Similarly, for November 1.

From the **dst{}** and **est{}** dictionaries the starting dates for daylight savings time and eastern standard time are the variables **dst_start** and **est_start**.

```
dst_start = dst.get(mar1) # This is the offset from Mar 1
```

```
est_start = est.get(nov1)
```

The number of the starting dates for daylight savings time and eastern standard times are:

```
start_dst= int((date(yr,3,dst_start)-jan1).days)
```

```
start_est = int((date(yr,11,est_start)-jan1).days)
```

Finally the timetag, either DST or EST is determined by:

```
if ttday-start_dst <0 or start_est - ttday <=0:
```

```
    tag= 'EST'
```

```
else:
```

```
    tag ='DST'
```

```
print 'Our time zone is on ',tag
```

```
timetag = tag
```

```
return (timetag)
```

An interesting observation regarding calendars is that March 1 and November 1, always occur on the same day of the week, regardless of the year.

Listing of *dst(x)*

```
def dst(utc):
    #####
    ###Finds the start of EST and DST from today's date
    import datetime
    from datetime import date

    dst = {0:14,1:13,2:12,3:11,4:10,5:9,6:8} # off-set from Mar 1 to start
of DST
    est = {0:7,1:6,2:5,3:4,4:3,5:2,6:1} # off_set from Nov 1 to start of
EST
    dow =
    {0:'Monday',1:'Tuesday',2:'Wednesday',3:'Thursday',4:'Friday',5:'Saturday'
,6:'Sunday'}

    xxx = gpsd.ut
    yr = int(xxx[0:4])
    mon = int(xxx[5:7])
    da = int(xxx[8:10])
    tday =date(yr,mon,da)

    dayofweek = datetime.date(yr, mon, da).strftime("%A")

    # find Jan 1 of year
    jan1 = date(yr,1,1)

    #find mar1
    mar1 = datetime.date(yr, 3, 1).weekday()# days 0-6
    nov1 = datetime.date(yr, 11,1).weekday()
    print 'March 1 is on a :', dow.get(mar1)
    print 'November 1 is on a :', dow.get(nov1)
    dst_start = dst.get(mar1) # This is the offset from Mar 1
    est_start = est.get(nov1)

    print 'Today is :',xxx
    print 'Today is ',dayofweek,' ',tday
    print 'Daylight savings time starts on Sunday March' ,
dst.get(mar1),'th in ',yr
    print 'Estern Standard time starts on Sunday
November',est.get(nov1),'th in ',yr

    print xxx
    yr= int(xxx[0:4])
    mon = int(xxx[5:7])
    da = int(xxx[8:10])
    tday =date(yr,mon,da)
    dayofweek = datetime.date(yr, mon, da).strftime("%A")

    # find Jan 1 of year
    jan1 = date(yr,1,1)
```

```

#find mar1
mar1 = datetime.date(yr, 3, 1).weekday()# days 0-6
nov1 = datetime.date(yr, 11,1).weekday()
print 'March 1 is on a :', dow.get(mar1)
print 'November 1 is on a :', dow.get(nov1)
dst_start = dst.get(mar1) # This is the offset from Mar 1
est_start = est.get(nov1)

print 'Today is :',xxx
print 'Today is ',dayofweek,' ',tday
print 'Daylight savings time starts on Sunday March' ,
dst.get(mar1),'th in ',yr
print 'Eastern Standard time starts on Sunday
November',est.get(nov1),'th in ',yr
ttday =int((tday-jan1).days) # Number of days between today and 1/1
print dst_start
print est_start
print 'Number of days from Jan 1 to today ;',ttday
print 'tday: ',tday
print date(yr,3,dst_start)
print date(yr,11,est_start)
start_dst= int((date(yr,3,dst_start)-jan1).days)
start_est = int((date(yr,11,est_start)-jan1).days)

print start_dst
print start_est

if ttday-start_dst <0 or start_est - ttday <=0:
    tag= 'EST'
else:
    tag ='DST'
print 'Our time zone is on ',tag
timetag = tag
return (timetag)

```

The start of Daylight Savings Time is the second Sunday in March

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Python Day Number	6	0	1	2	3	4	5
If March 1 is Mon		1	2	3	4	5	6
	7	8	9	10	11	12	13
DST starts	14						
If March 1 is Tue			1	2	3	4	5
	6	7	8	9	10	11	12
DST starts	13	14					
If March 1 is Wed				1	2	3	4
	5	6	7	8	9	10	11
DST starts	12	13	14				
If March 1 is Thu					1	2	3
	4	5	6	7	8	9	10
DST starts	11	12	13	14			
If March 1 is Fri						1	2
	3	4	5	6	7	8	9
DST starts	10						
If March 1 is Sat							1
	2	3	4	5	6	7	8
DST starts	9						
If March 1 is Sun	1	2	3	4	5	6	7
DST starts	8	9	10	11	12	13	14

The start of Eastern Standard Time is the first Sunday in November

Python Day Number	Sun	Mon	Tue	Wed	Thu	Fir	Sat
	6	0	1	2	3	4	5
If Nov 1 is Mon		1	2	3	4	5	6
EST starts	7						
If Nov 1 is Tue			1	2	3	4	5
EST starts	6	7					
If Nov1 is Wed				1	2	3	4
EST starts	5	6	7				
If Nov 1 is Thu					1	2	3
EST starts	4	5	6	7			
If Nov 1 is Fri						1	2
EST starts	3	4	5	6	7		
If Nov 1 is Sat							1
EST starts	2	3	4	5	6	7	
If Nov 1 is Sun	1	2	3	4	5	6	7
EST starts							

Appendix 12 Emulate a key press in a Python script.

This technique is listed to vary the volume of the audio tracks based on the speed of the train.

Xdotool

This works for numbers and letters. Not for special characters!

The command line is:

```
xdotool key [enter key]
```

I first tested that it would work using p for pause in the program listed below.

I had one additional problem. It did not work for special characters. And I needed + and -, for omxplayer!

I tried using ascii codes for special characters.

If you type ctrl-shift u and then the ascii code from the command line it does work.

```
xdotool key ctrl+shift u 2b
```

2b is the hex for 43, the ascii for +.

But there is no way to do this in a python script (that I could find).

I posted a message on stack overflow and they provided the solution:

```
xdotool key plus
```

```
and
```

```
xdotool key minus
```

And it works in a python script.

```
os.system('xdotool key minus')
```

Below is the python script to test the procedure


```

#####
## ascci for 'p' is 112 base 10, 1110000 base 2, 70 in base 16
## ascci for '+' is 43 base 10, 101011 base 2, 2B in base 16
## ascci for 'p' is 45 base 10, 1110000 base 2, 2D in base 16
## xdotool ctrl+shift u 70 returns p
import os
from gps import *
from time import *
import time
import threading
a=1
#####
#Creates a thread to play an audio Clip
def play_clip():
    playing_clip = 1
    os.system('xxx %sound')
#####
xxx = 'omxplayer -o both --vol 0 %s'
sound ='tone.mp3'
threading.Thread(target=play_clip).start()
print 'starting tone'
time.sleep(5)
while a==1:
    #os.system('echo -n "-" > /proc/$(pidof omxplayer.bin)/fd/0')
    #os.system('echo - > omxplayer.bin')
    print"ttt"
    time.sleep(1)
    os.system('xdotool key minus')
    time.sleep(1)
    os.system('xdotool key plus')

```

Appendix 13. Play audio tracks before the train leaves the station.

gps-409-2.py 5/8/23

In the effort to automate the announcer on the train, we needed to make announcement before the train leaves the station. This information is in the EXEL workbook used to create the Dispatcher's Reports, **2023_Trains.xlsm**. In this workbook, the worksheet **2023_Trains**, shown below contains the departure times.



2023 Schedule and Ridership

Revised 4/17

Total Ridership YTD 3423

6/18/2023

Index	Date		Trains	1	2	3	4	5	Riders	Average	Theme
1	02/01/23	Wed	1	10:30 am					1	1	Photo shoot
2	03/31/23	Fri	1	11:00 am					3	3	VIP Train
3	04/01/23	Sat	5	9:30 am	11:00 am	12:30 pm	2:15 pm	3:45 pm	410	82	Hop into Spring-1
4	04/02/23	Sun	5	9:30 am	11:00 am	12:30 pm	2:15 pm	3:45 pm	707	141	Hop into Spring-2
5	04/12/23	Wed	1	10:30 am					140	140	Group ride
6	04/15/23	Sat	4	11:00 am	12:00 pm	1:00 pm	2:00 pm				Operate-A-Loco
7	04/21/23	Fri	1	10:30 am					193	193	Group ride
8	04/23/23	Sun	1	2:30 pm					121	121	Charter
9	05/06/23	Sat	4	1:00 pm	2:30 pm	4:00 pm	5:30 pm		375	94	Brew & Choo
10	05/12/23	Fri	1	10:30 am					63	63	Group ride
11	05/13/23	Sat	4	11:00 am	12:00 pm	1:00 pm	2:00 pm				Operate-A-Loco
12	05/17/23	Wed	1	10:30 am					99	99	Group ride
13	05/20/23	Sat	2	10:30 am					39	20	Fuel train
14	05/21/23	Sun	3	1:00 am	2:30 pm	4:00 pm			254	85	Slow Down Sunday
15	05/26/23	Fri	1	10:30 am					156	156	Group ride
16	06/03/23	Sat	4	1:00 pm	2:30 pm	4:00 pm	5:30 pm		339	85	Brew & Choo

The worksheet must be saved as a .csv (comma separated value) file, with the name **schedule.csv**. This file needs to be in the working directory, **/home/pi/gps_2017**.

A python **thread** was created in the **gps1.py** program to perform this function: **play_clips_before_departing**.

The system was designed to play audio clips 15, 10, 5 and 1 minute before the train departs.

The files are:

- 1.mp3** is played 15 minutes before departure.
- 2.mp3** is played 10 minutes before departure.
- 3.mp3** is played 5 minutes before departure.
- 4.mp3** is played 1 minute before departure.

The play back volume is set by the variable **pre_dep_vol**.

The added row in the **gps_data.csv** file, adds a line with the latitude of 39 and sets the gain at 600. It is the third from the last row in the spreadsheet.

When the data from this file is imported, the *pre_dep_vol* is set to *nb_gain[len(locs)-3]*.

Last few lines of the *gps_data.csv* file:

37	35.69866409	-79.97	half way to whistle	silence_0001.wav	0	silence_0001.wav	0
38	35.69965891	-79.97	new hill whistle marker	silence_0001.wav	0	silence_0001.wav	0
39	39	-79.97	per_departure-gain		599		
40	40	-79.97	Background Music	Xmas_Music_Tracks_2023.mp3	20	silence_0001.wav	8
41	50	-79.97	Xmas_2020	silence_0001.wav	0	silence_0001.wav	0
42							

The *build_file_names3.py* was incorporated into the program and this is no longer needed.

A new **Ctrl-c** function was added to change the value of this variable. It is displayed on the screen along with the value that was found in the *gps_data.csv* file.

To change the *pre_dep_vol* level:

Press the **Ctrl-c** twice.

The LED will stop flashing and remain on.

Enter *prevol*.

Enter the desired *pre_dep_vol* level.

Note that the value entered is in milli-DB. -600 will reduce the value to half.

Screen capture setting level to 300:

```
Background track Xmas_Music_Tracks_2023.mp3 Gain = 20 Repeating 8 times
['gps1.py']
409-2-5 Pre gain
-----
('Trains today: ', '10:30 AM ')
2023-12-15 07:48:12.400922
^CEnter location OR est or dst OR music OR background: prevol
Enter a new volume: 300
```

Last few lines of the display monitor:

```
Data file in use Local Time = 08:56:59 AM EST 12/14/23
GPS_DATA.CSV in use: Xmas_2020
Pre-departure level Pre-departure audio gain: 599 gps_data.csv pre_dep_vol 599
Background Music track Background track Xmas_Music_Tracks_2023.mp3 Gain = 20 Repeating 8 times
['gps1.py']
409-2-5 Pre gain
-----
('Trains today: ', '10:30 AM ')
2023-12-14 08:55:56.712504
```

A script was created for each file and the texts were uploaded to a text to speech website:

https://speechify.com/text-to-speech-online/?landing_url=https%3A%2F%2Fspeechify.com%2Ftext-to-speech-online%2F

The audio files created were down loaded.

Audio files can also be edited with the open source and free software **Audacity**.

The following functions were used in this thread:

def convert24() This converts 12 hour time to 24 hour time

def convert_to_min() Converts hh:mm:ss to minutes from midnight

def con_min() Converts minutes to hh:mm in datetime. Note that this function was copied from an internet site.

Get schedule from Dispatcher's Report 2023_Trains EXEL file.

This file is saved as a comma separated value file: **schedule.csv**.

The program then finds the schedule for today.

It then creates a list of the train times: **t[]** for the day. These train times are stings extracted from the .csv file.

If there are less than 5 trains in a day, the blank trains times in the **t[]** list are replaced with **2:00 AM** times. This is a quick fix. Otherwise, the thread will crash if there are any empty times in the csv file.

The system stores the times for 5 trains and each train has 4 pre-departure announcements.

The advanced times for the announcement are 15, 10, 5 and 1 minute before the train departure times.

These advanced times are stored in a list: **ad[]**.

To change these times, the python program will have to be edited.

This is the line of code that determines the time before departure

```
ad = [0,15,10,5,1] ## Advance times before departure
```

The audio files that will be played at the advanced times are named: **1.mp3, 2.mp3, 3.mp3** and **4.mp3**. These file names are stored 5 times in the **file_list[]** list. These audio files must be in the **/home/pi/gps_2017/audio_tracks** folder.

They are stored 5 times so they can be accessed by the same dummy variable that detects the time of day.

The program then creates a list of all of the advanced announce times for the day and stores them in the Advanced Times list **at[]**. This is done in the nested for loop using the **at.append** command.

The **at[]** list is then sorted and it will now match up with the **file_list[]** discussed above.

The next part of the Thread uses a for loop to check if the time of day equals one of the advanced times in the **at[]** list:

```
ttt == at[x]
```

When this occurs, the system plays the corresponding audio file from the **file_list[x+1]**.

It then records the time to the text file.

The program then sleeps for 61 seconds. Otherwise, the audio track would play again.

Appendix 14. Using VNC on Pi with a 3.5 inch display

When using a 3 ½ inch display on a Rasp pi, there are several issues that need to be addressed:

There is as problem using VNC to transfer files from the Rasp Pi to the pc.

There are additional problems when the FULLSCREEN mode is used.

There is an issue that you also have to deal with when transferring files to the Rasp Pi from a PC. The files can ONLY be sent from the PC to the Desktop folder on the Rasp Pi.

These issues are exhibited in the speedometer program. It consists of a Pi Zero driving a 3 ½ inch display. The graphics for the display were generated using pygame which is set to FULLSCREEN mode. The program is set to run when power is applied to the Rasp Pi.

Here is the procedure:

Reboot the unit by powering down and then re-starting.

While the *autoexec.sh* file is launching the python script, stop the execution using **Ctrl-c**.

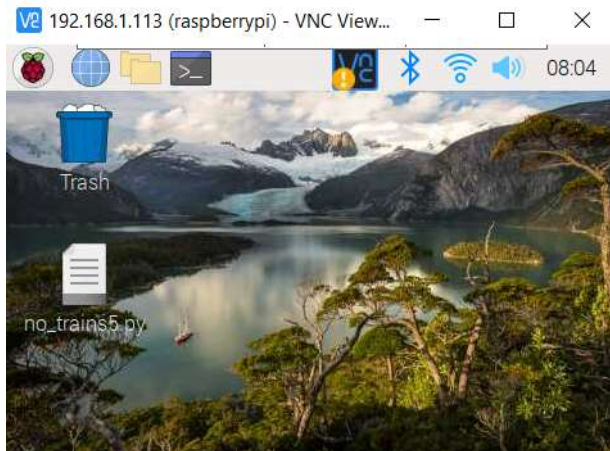


Fig 1

From this display on the tft screen and the VNC screen, if you click on the



The display is will show as Fig 2.

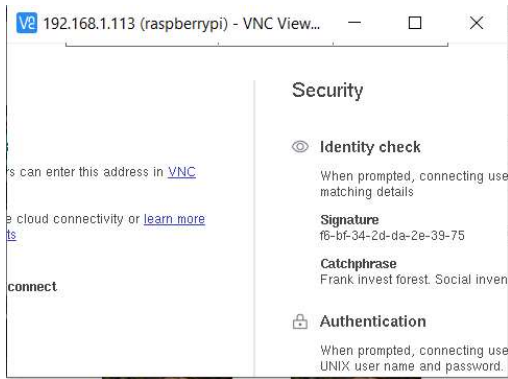


Fig 2

There is no way to exit this without rebooting!

so, **do not** do this.

The display needed will look like Fig 3.

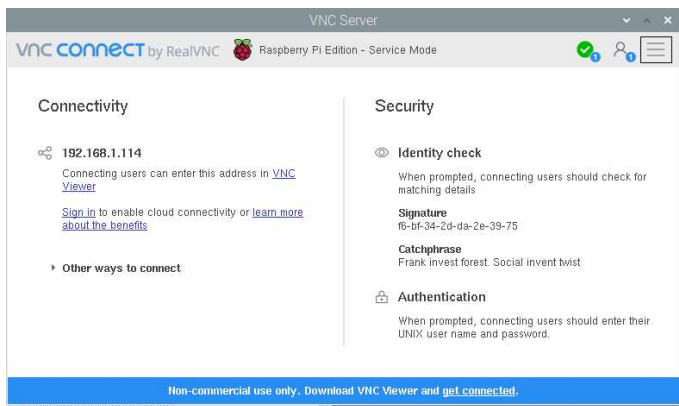


Fig 3

So then you can click on this icon



to transfer files as shown in Fig 4.

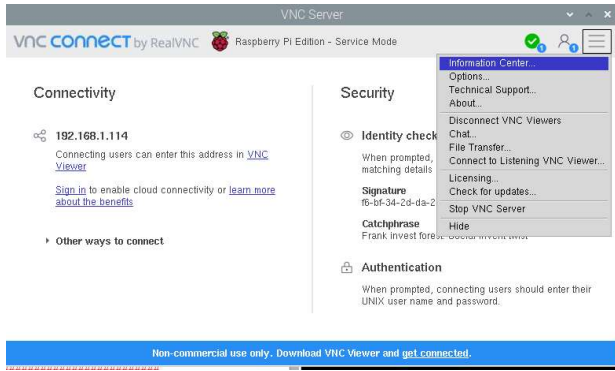


Fig 4

To get to this screen, you have to reboot again sending the display to the hdmi port, even though there is nothing connected to the hdmi port.

Reboot From a terminal window by entering `./hdmi`. Then use **Ctrl-c**, to again, to kill python script that the autoexec.sh has launched. This will stop the FULLSCREEN mode. See Fig 5.

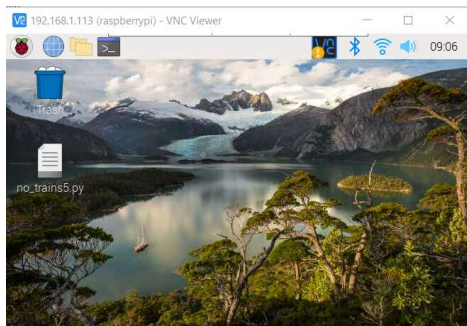


Fig 5

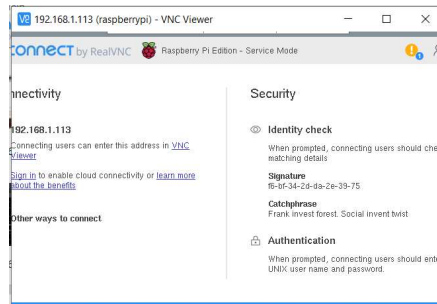


Fig 6

Now when click on , you will see the display Fig 6.

There is an area in Fig 7. that you can click on to move the frame and expose the necessary icon shown in Fig 8.

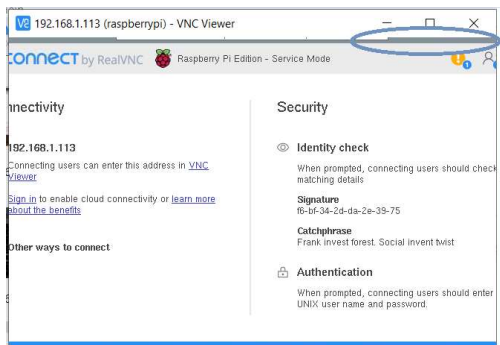


Fig 7

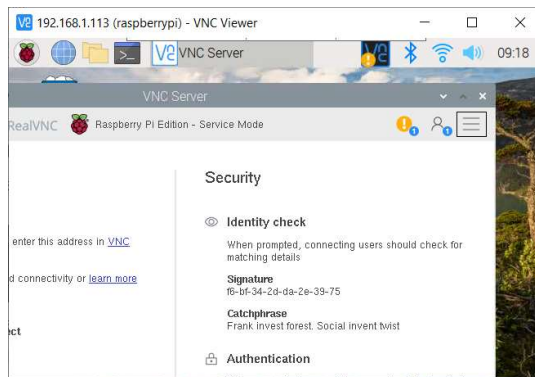


Fig 8



This is what we need to click on to send files from the Rasp Pi to the VNC Viewer on the PC.

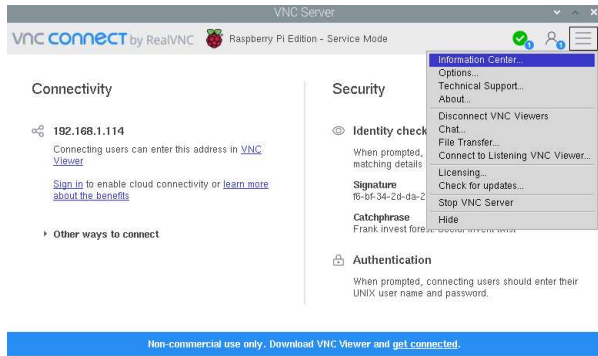


Fig 9.

To return to normal, reboot using `./tft` from a terminal window.