Speedometer Description 10/3/22 *(tft11-59-6-1.py)*

http://dunn-itvideoservicesllc.yolasite.com/speedometer.php

Our “old” engines, #399, #1686 and #17 are not equipped with speedometers.
The two “new” engines, #671 and #1640 are equipped with speedometers and several of our engineers noted that is was a useful feature.

A speedometer, based on a GPS receiver, was fabricated to be used in our old engines. The unit is self-contained and does not require any mechanical connection to the engine other than 12 volts DC.

The speedometer uses a Globecast BU-353 GPS receiver, a Raspberry Pi-Zero Linux computer, with a 3.5 in display. The system requires 5 volts and uses a step-down DC-DC convertor to obtain this from the engine’s electrical system.

The software is a python script. It displays the speed on a 3.5” graphic display.
The unit also displays the time and date and it automatically adjusts for DST and EST.

The latest version displays speed limit warnings for speeds above 15 mph on the main line and 10 mph at the grade crossing, the bridge and Bonsal and New Hill yards.

Technical Details

The software is installed in the ***/home/pi/speedometer*** directory and the main program is ***speedometer.py***. In addition to the main program, the system requires that the shell script ***gpssock*** be installed in the /home/pi/speedometer directory.
Additionally, to auto-start the system on power up, a shell script, ***autoexec.sh*** must be in the ***/home/pi directory***, and a ***.desktop*** file must be installed in the ***/home/pi/.config/autostart*** directory. (The ***autoexec.sh*** is named after the after the ***autoexec.bat*** file in MSDOS)

A complete list and a detailed set of instructions are in build for the SD card for the Raspberry Pi.
There is also a detailed description of all of the custom software used in the system.

The speedometer is housed in a Harbor Freight tilt bin plastic storage cabinet. The groove in the drawer allows the Pi-0 to slide into it. The SD card will not fit within the groove. A slot was made in the drawer to allow the SD card to be inserted after the assembly has been installed. A plastic pin holds the drawer from opening.
The DC-DC buck converter will work with an input voltage range of 40 volts to 7 volts and provide the 5 volts needs. There are other convertors available that will work up to 80 volts if need be.

The version tft11-59-6 adds compass arrow indication North.

Version tft11-2.py automatically sets the local time to EST or DST.
Since the system determines the offset, Version ***tft11-2dow,py*** replaces the EST or DST with the day of the week in the time display.

There are several documented errors:

1- When the system is booted after 8:00 PM local time the day of the week displays the next day. This is cause by the conversion from Universal time to local time.
2- Additionally, when the system was booted on the evening of 8/31/20, the date displayed was ***09/0/20***.
3- The other issue is that the date for 9/2/20 should be displayed as ***'09/02/20***'. It displays the date as '***09/2/20***',
4- When VNC is used to log into the system the display is small and it is difficult to work with.
5- 1/15/21. On two occasions it was noticed the system displayed the wrong date:

Note that 8/26/2004 is NOT a Friday. 8/26/1904 is a Friday.

When this happened, the speed was displayed correctly.

Re starting the system restored normal operation.

A **failed attempt** to fix problems 1, 2 and 3 was made in version ***tft11-2-1dow.py***.

The following files are needed in the ***/home/pi/speedometer*** folder:

gpssock
nhvrr350.jpg
speedometer.py
autoexec.sh
clearscreen.jpg
speed_limit_40x50.jpg
speed_limit_10.jpg
arrow_up.jpg
arrow_down.jpg
arrow_clear.jpg
Track3_50.jpg

Software Description. tft11-2.py

The system uses a GPS receiver to determine the speed. It also extracts the date and time information from the GPS serial data stream.

The ***gps_lock*** determines if the data is valid. If it is not, the error signal will be displayed. “Waiting for valid data”.

When the receiver locks, the time and date are extracted and the function ***dst(utc)*** determines if the time is Daylight Savings or Eastern Standard time and sets the offset from Universal time. This procedure is performed only once when the receiver locks on to the satellite. The ***update_timetag*** is set from 0 to 1.

The date and time are displayed in a format by the function ***EDST(utc)***.
Pygame is used to display the data graphically.
A good portion of the program deals with the date and time display and getting the DST and EST offsets automatically.
For a detailed description of the program, see Appendix 3

Revisions 12/7/20
In the original version, the speed display showed the tenth of a mile when the speed was below 10 mph. It then stopped displaying the tenth of a miles. It did this by only displaying the first two characters of the speedd variable (***speed =speed[0:2]***).
The update changed the truncated speed display using the ***round*** function. The ***round()*** function display .0.
If the ***speedd*** is ***11.634***, the ***round(11.624)*** displays ***12.0***. The ***[0:2]*** truncates the rounded string to ***12***.

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      # This section is for the Speedometer
    #####  from /home/pi/learn_pygame/pigame12.py
        ####
        mph = " MPH"

        if len(speedd)>3:
           #speedd =speedd[:2]
           speedd = str (round(gpsd.fix.speed *2.236))[0:2] # rounds the speed

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

12/15/2020 Version ***tft11-41_speed_alert.py*** displays a 15 MPH speed limit sign when the speed is greater than 15 mph. It requires the file: ***speed_limit_40x50.jpg*** file.

Three lines were added to the program

The variable ***speedlimit*** is the converted Speed Limit sign above,
In the last two lines below the if statement displays the ***speedlimit*** icon if the speed is greater than 15 mph.

03/06/2021 Version ***tft11-54.py***

This version added a 10 mph limit at Bonsal Crossing, Horton Road, the Bridge and New Hill yard.. Note the ffearr location is for testing purposes only.

1 degree of latitude is 69.2 miles. 250 Ft is .0007 degrees. This is what we will make delta. We will subtract delta from the new hill yard latitude and add delta to the motor car house latitudes.

```
## latitudes for speed alert
fearr = 35.796198504
motor_car_house = 35.6615445
horton_rd = 35.68121523
bridge = 35.68775611
new_hill = 35.69729441
delta =.00015  # The spread between the latituude and play point

    speed_limit_10 = pygame.image.load('/home/pi/speedometer/speed_limit_10.jpg').convert()
```

This section checks the speed and location and displays the warning.

```
    if abs(fearr -latt) < delta and gpsd.fix.speed *2.236 > 10:
        print fearr -latt
        print delta
        screen.blit(speed_limit_10,(100,115))

    elif abs(horton_rd -latt) < delta and gpsd.fix.speed *2.236 > 10:
        screen.blit(speed_limit_10,(100,115))

    elif abs(bridge -latt) < delta and gpsd.fix.speed *2.236 > 10:
        screen.blit(speed_limit_10,(100,115))

    elif latt< motor_car_house and gpsd.fix.speed *2.236 > 10:
        screen.blit(speed_limit_10,(100,115))

    elif latt> new_hill and gpsd.fix.speed *2.236 > 10:
            screen.blit(speed_limit_10,(100,115))

    if gpsd.fix.speed *2.236 > 15:
```

screen.blit(speedlimit,(100,115))

03/06/2021 Version ***tft11-57.py***

Changed the from to

New Hope Valley Railway

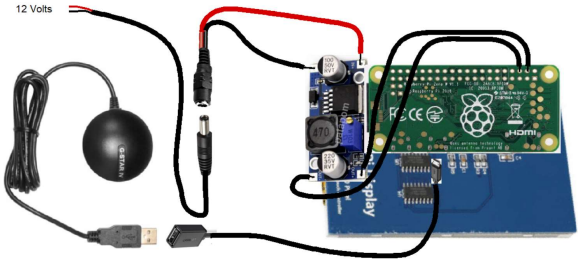
Add ***nhvrr350.jpg*** to /home/pi/***speedometer*** folder.
Centered date and time display.

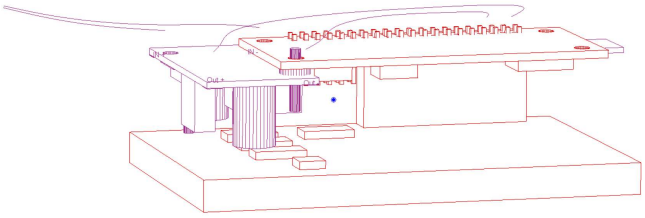
10/3/22 Version ***tft11-59-6-1.py***

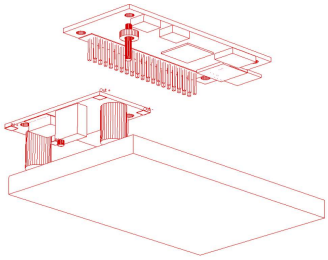
This version added the North Tracking arrow:

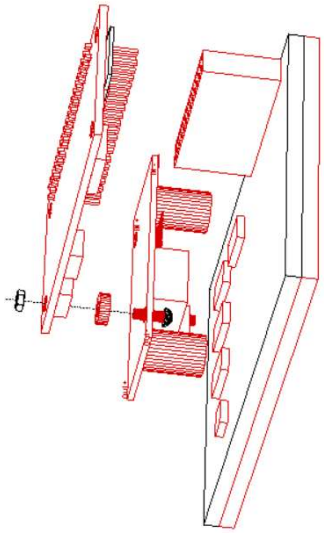
```
def show_track(message):
  font = pygame.font.Font(None,50)
  text = font.render(message, 1,(200,200,200))
  screen.blit(text,(10,90))

  show_track(str(gpsd.fix.track)[0:2])

  #Rotating Arrow
    track_arrow =pygame.image.load('track3_50.jpg').convert()

          rotated_image = pygame.transform.rotate(track_arrow, gpsd.fix.track)

       screen.blit(rotated_image,(15,125))
```

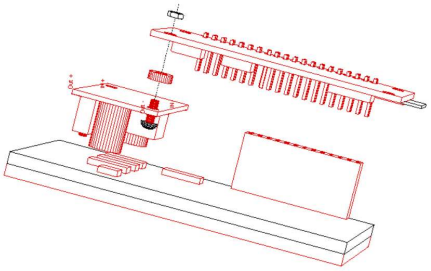
Assembly Drawing











Appendix 1

3/3/20 Build Speedometer SD card.

The easiest way to do it is to use an image file of an existing system and write it to a new 16 gig SD card. It is VERY important to use ONLY SanDisk brand SD card.
Some other brands (for example PNY) have slightly less usable memory ie 16K less, and the image file is too large to write to this card. The 15,558,144 KB image file dated 1/28/21 is ***tft11-41.img***.

a) Load Noobs
b) Select WiFi
c) Install VNC
d) Install CGPS
e) Build gpssock file
f) Install TFT display driver
g) Load speedometer folder
h) Make the system auto start on bootup.
i) Inhibit sleep mode

e) The ***gpssock*** is a Linux shell script. It is needed to initialize the gpsd software.

It contains the following three lines:
sudo systemctl stop gpsd.socket
sudo systemctl disable gpsd.socket
sudo gpsd /dev/ttyUSB0 -F /var/run/gpsd.sock

The gpssock script file just created must be made executable. To do this, in the terminal enter the command: ***sudo chmod +x gpssock***
When the ***ls*** command is executed the ***gpssock*** file will be yellow or green, indicating that is executable. See below.

```
pi@raspberrypi: ~/speedometer
File  Edit  Tabs  Help
 clrscreen.jpg
 day_before.py
 desktop.ini
 downot.py
 dst-2.py
 dst_correct_dow2.py
 dst_correct_dow.py
 dst.py
 find_dst_import.py
 find_dst_import.pyc
 gpssock
 gpssock.sh
 logoTFT.jpg
'Material Lilst.xlsx'
 nhvrr350.jpg
 Pictures
'Screen Grab.dcd'
 Screen_grab_tft11-52.jpg
 speed_alert.py
 speed_limit_10.jpg
 Speed_limit_150x230.jpg
 Speed_limit_300x500.jpg
 speed_limit_40x50.jpg
 Speed_limit_.jpg
'Speedometer Description 03-19-21.docx'
 speedometer.py
 test_datetime.py
 tft11-41_speed_alert.py
 tft11-51.py
 tft11-52.py
 tft11-53.py
 tft11-54.py
 tft11-55.py
 tft11-56.py
 tft11-57.py
 Title.jpg
 x-gpssock
 x.py
pi@raspberrypi:~/speedometer $
```

f) Load TFT software

Open Terminal of Raspberry Pi (You may need to connect a keyboard and HDMI LCD to Pi for driver installing)

At the command line run

```
git clone https://github.com/waveshare/LCD-show.git
```

After the program `loads:`

```
cd LCD-show
Run the script LCD35B-show
```

Note: Network connection is required while installing driver to your Pi, or else the touch won't work properly. #If the LCD you have is old version, use this command:

```
./LCD35B-show
```

#If the LCD you have is V2 version, use this command:

```
./LCD35B-show-V2
```

3. After system rebooting, the RPi LCD is ready to use.

Method 2. Using Ready-to-use image

The image file with pre-installed driver is located in the IMAGE directory of the CD, or you can download it from #Image. Extract the .7z file and you will get an .img file. Write the image to your micro SD card (How to write an image to a micro SD card for your Pi? See RPi Image Installation Guides for more details). Then insert the card to your Pi, power up and enjoy it.


```
File  Edit  Tabs  Help
pi@raspberrypi:~ $ cd LCD-show
pi@raspberrypi:~/LCD-show $ ls
\boot                       LCD4C-show
cmdline.txt                 LCD4-show
cmdline.txt-noobs           LCD5-show
dtc.sh                      LCD7-1024x600-show
etc                         LCD7-800x480-show
inittab                     LCD-hdmi
LCD101-1024x600-show        mk_arcade_joystick_rpi-master
LCD154-show                 nes
LCD28-show                  README.md
LCD32C-show                 rpi-fbcp
LCD32-show                  usr
LCD35B-show                 waveshare32b-overlay.dtb
LCD35B-show-V2              waveshare32c-overlay.dtb
LCD35C-show                 waveshare35a-overlay.dtb
LCD35-HDMI-480x320-show     waveshare35b-overlay.dtb
LCD35-HDMI-800x480-show     waveshare35b-v2-overlay.dtb
LCD35-show                  waveshare35c-overlay.dtb
LCD43-show                  waveshare4c-overlay.dtb
LCD43-show-V2               xinput-calibrator_0.7.5-1_armhf.deb
LCD4-800x480-show
pi@raspberrypi:~/LCD-show $ \
```

Toggle between LCD and HDMI display

There are a pair to ***shell scripts*** that will toggle the displays.

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
pi@raspberrypi:~ $
pi@raspberrypi:~ $
pi@raspberrypi:~ $ ls
2020-10-04-215440_1280x1024_scrot.png  Downloads   speedometer
2021-12-01-192151_1920x1080_scrot.png  g           ted.py
autoexec.gps                           gps_2017    ted.py.save
autoexec.GPS                           gps_2020    Templates
autoexec.s                             gps31.py    tft
autoexec.sh                            gpsosck.sh  up
autoexec.sh.save                       hdmi        usb.txt
autoexec.tft                           LCD-show    Videos
autoexec.tft.save                      MagPi       x.py
Desktop                                Music       zzRebuild_no_Autostart
dn                                     Pictures    zzRebuild_no_Autostart.save
Documents                              Public
pi@raspberrypi:~ $
```

From the ***/home/pi directory,***
enter: ***./hdmi*** or ***./tft*** to toggle the display.

Once this LCD is enabled, meanwhile the default settings for HDMI are changed. If you want to use another HDMI monitor, please run the following command:

```
cd LCD-show/
./LCD-hdmi
```

This toggles the mode to LCD display: #If the LCD you have is old version, use this command:

```
./LCD35B-show
```

#If the LCD you have is V2 version, use this command:

```
./LCD35B-show-V2
```

Load files into the /home/pi/speedometer

The following files are needed in the ***/home/pi/speedometer*** folder:

nhvrr350.jpg
speedometer.py
autoexec.sh
clearscreen.jpg
speed_limit_40x50.jpg
speed_limit_10.jpg
arrow_up.jpg
arrow_down.jpg
arrow_clear.jpg
Track3_50.jpg

Turn off sleep mode

Stop Sleep mode:

```
sudo nano /etc/xdg/lxsession/LXDE-pi/autostart
```

Add these lines:

```
@xset s noblank
@xset s off
@xset -dpms
```

h) Build the auto start system to run the program on turn-on.

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.

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***: ***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. Pres ***ctrl-x*** to exit the text editor.

3- Create the ***autoexec.sh*** Linux script 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. Pres ***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 = lxterminal -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 ***autoexec.****xx* files can be save and easily be rename to the ***autoexec.sh*** file.

3-In this case, the 2 lines in the autoexec.sh script are:
cd speedometer
sudo python speedometer.py

The autoexec.sh file is launched by the ***.desktop*** file. This file is in the ***/home/pi/.config/autostart*** folder and contains the following three lines:

[Desktop Entry]
Type = Application
Exec = lxterminal -e ./autoexec.sh

- Harborfreight

Storehouse

4 in. Stacking Tilt Bin

Appendix1- 2 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 **raspberry**. 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::f2bd:501d:4cb7:784a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU: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.

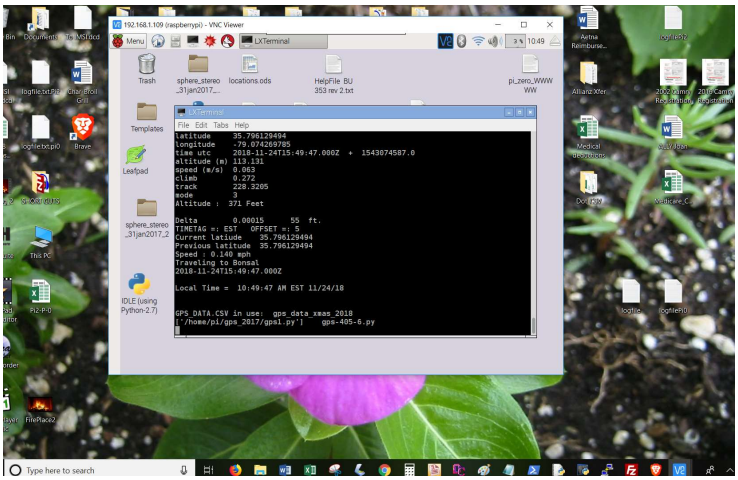
An ftp server is installed on the Rasp Pi and an ftp client is installed on the PC. **FileZilla work**s well.

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

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

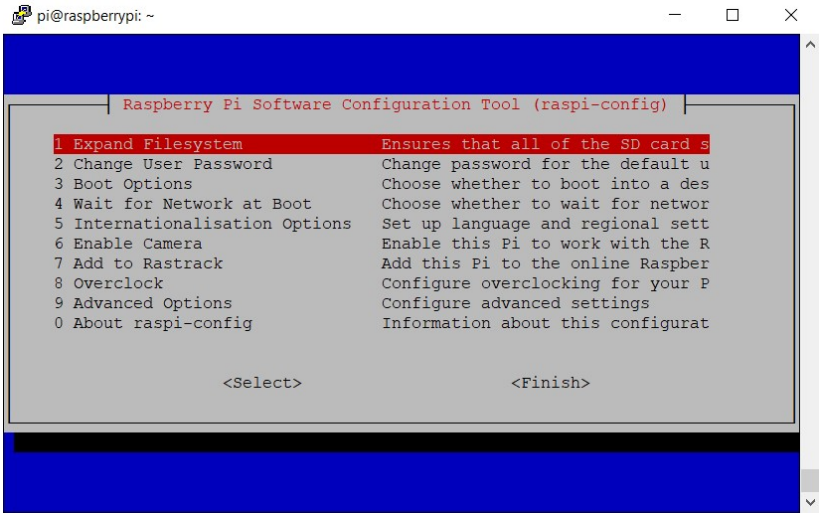
Appendix 1-3 GUI Control of the Raspberry Pi from a remote computer.

Real VNC Viewer

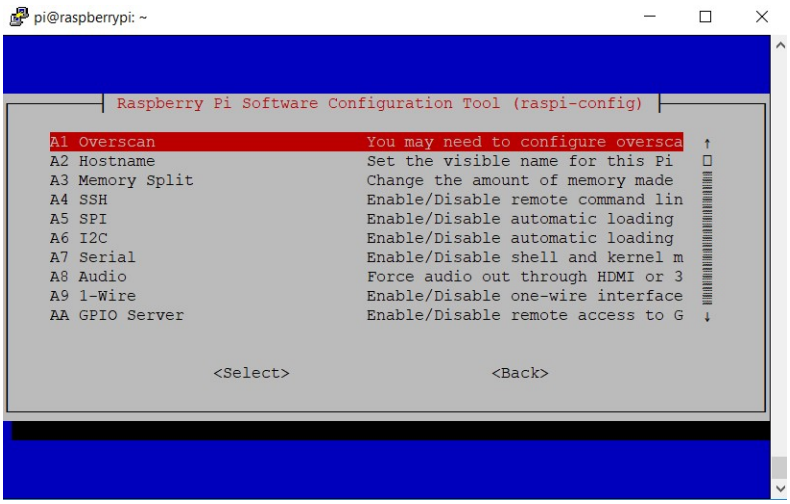


Detailed installation:
https://www.raspberrypi.org/documentation/remote-access/vnc/README.md
Run ***sudo apt-get update***
Run ***sudo apt-get install realvnc-vnc-server realvnc-vnc-viewer***
Run ***sudo raspi-config***

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

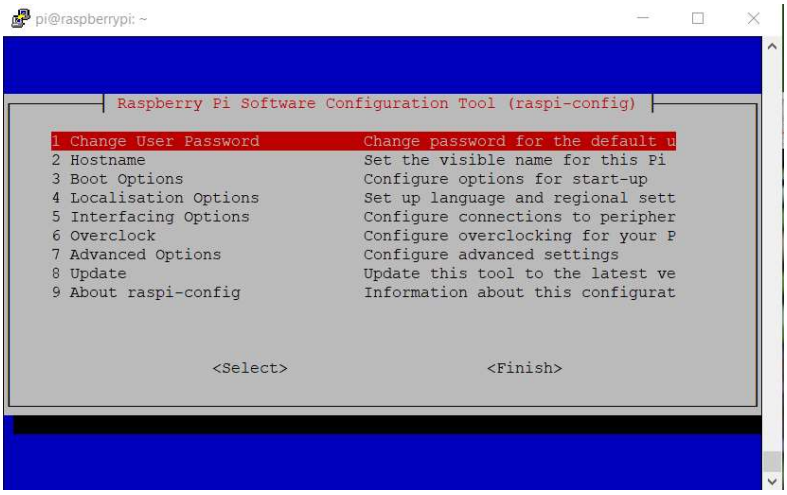


Select Advanced options



Use down arrow to scroll down and show additional options, to A0 Update. It is below the last item shown.

Select AO Update. This will take several minutes to complete. It will then display:

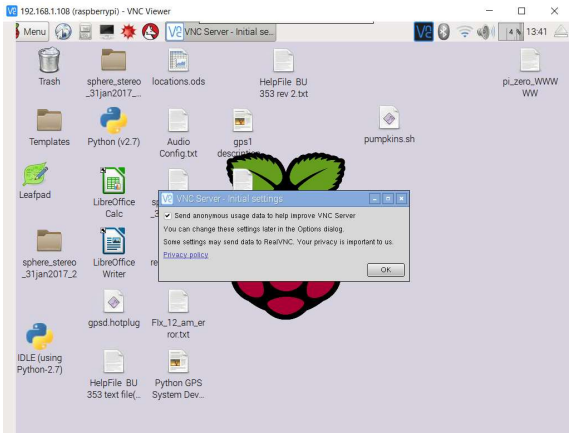


Now you can select Interfacing Options.

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 1-4

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 promted

Run the gpssock file

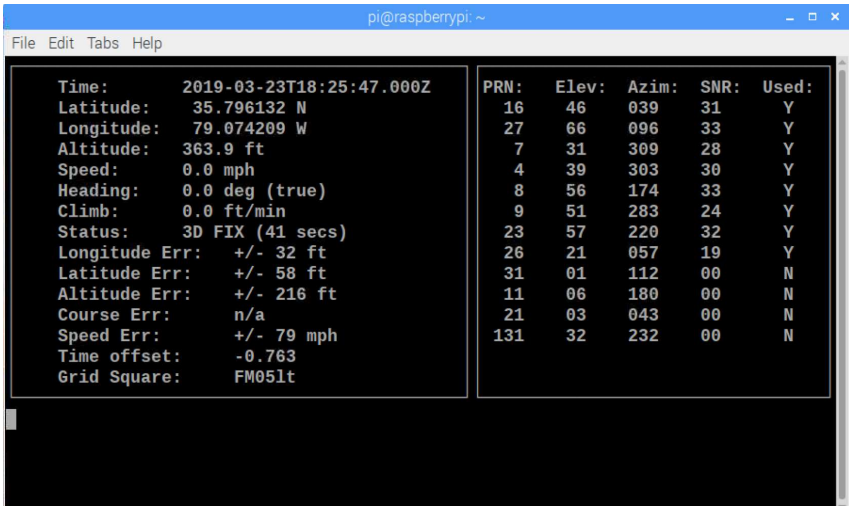
At the command line in the /home/pi/speedometer directory, enter:
./gpssock

The **. /** is the command to run an executable file. **This file is needed otherwise the gps receiver will not communicate with the Raspberry Pi.** The exact reason for this is unclear.

Run the rest program. At the command lineenter::

cgps -s

You should see the gps data



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

Appendix 2

Determine the dates of DST and EST. 05/14/2020

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.
In North America, 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 2-1 for all of the options of when DST starts and when EST starts.
For example: If March 1st occurs on Wednesday, the second Sunday is on March 12th .
The ***est{}*** and ***dst{}*** dictionaries were generated from the data in Appendix 2-1,

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{} dictionaly 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 2-1.
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 ***dstr{}*** 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,dayofweek)
```

The dayofweek variable is retrieve as: ***datetag = dst(gpsd.utc)[1]***.
It replaces the ***timetag*** variable in the ***EDST(x)*** function which generates the time display.

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 'Estern 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,dayofweek)
```

Appendix 2-1

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

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

Appendix 3 Detailed decription of tft11-2.py

From the libraries: gps, time, threading and shutil are imported. (shutil may not be needed)

Variables are selected, (lattold is not needed
pygame is initialized.

The function ***dst(utc)*** is defined. This function returns the ***timetag***, either EST or DST.
See Appendix 2 for a detailed explanation of this function

The function ***EST(utc)*** is defined. This function returns the date and time string variable that is displayed.

The shell script ***gpssock*** is run. This is needed to start the gps receiver.
The ***GpsPoller*** is the routine from Dan Mandle that extract the data from the serial stream.

The next section sets up the display. It first prints the window label showing the file name and a brief description. Then loads the file of the graphic, sets the display dimension and displays the graphic

The next sections defines the display sizes and locations of the elements. The on labeled display is the speed graphic.

The ***if speed =='nan:*** section checks that the GPS receiver is locked.

The ***if speed !='nan:*** checks that if the data is valid and the update_***timetag*** variable is 0, the system gets the ***timetag*** variable from the ***dst(utc)*** function. It then sets the ***update_timetag*** variable to 1 so that it ignores this step.

The speed is assigned to the variable ***speedd = str(round(gpsd.fix.speed * 3.235,1))***. The ***speed*** variable extracted from the GPS data is in KPH
If the speed is less than10 MPH the display shows tenths of miles per hour. Otherwise it deletes the decimal part.

The next two lines displays the date and time and updates the display.

The last section does not apply to this program, it could not be deleted at this time.


New Hope Valley Railway

Appendix 4: The listing of the speedometer.py (tft11-41_speed_alert.py)

```
###


### Gps System for the New Hope Valley Railway
###  Created by Ted Dunn  7/28/2016

#! /usr/bin/python
# Written by Dan Mandle http://dan.mandle.me September 2012
# License: GPL 2.0

###  Version 405-3  Added copy gps_data.csv to USB_DRIve as gps_data_download.csv
###  Version 405-4  Adds version of gps_data.csv (print locs[len(locs)-1])
### Speed.py   9/10/19
###  speed20.py  1/17/20  added os.system('./gpssock')
###  speed21 reduce delay removed 4 sec delay



### ver = 'speed30-8-2'   # Changed size of date
### ver = 'speed30-8-3'   # Changed ESTime to be Nov-Feb
### ver = 'speed30-8-4'   # Display Waiting for valid data

### tft6-1 sets display to 480X320
### tft10 Adds find_dst_import
### tft10-1 removes flash drive
####  tft11-1 Uses function
###   tft11-2dow 5/15/20  replaces EST or DST with Day of week
###   tft11-41_speed_alert 12/15/20
ver = 'tft11-41_speed_alert'

version = 'gps-405-4.py --- Using isdir() instead of cp_stuff.sh'
nhv = '--- New Hope Valley Railway ---'
from sys import argv    #used to get file name

#--from load_usb_ports import usbnum  ## Find active number of USB ports for capture data
```


```
import os
from gps import *
from time import *
import time
import threading
from shutil import copyfile  #### Used for isdir() funciot


#####################################
lattsaved = 0  ## used for auto save to flash drives
gps_lock = 0
lattold =0    ## Used to deterime direction of travel
script = argv   # This is the file name
timetag = 'XYZ'
off_set = 1
update_timetag=0


#################################################################
#This section is for the Speedometer
import pygame
pygame.init()
screen = pygame.display.set_mode((0,800))  # Sets the screen size
screen.fill((0,0,0))  #Fills the bbackgroud with black
#################################################################
#################################################################
#Define Colours
WHITE = (255,255,255)
BLUE = (0,0,255)
BLACK = (0,0,0)
GRAY = (128, 128, 128)
MAROON = (128, 0, 0)
NAVYBLUE = (0, 0, 128)
OLIVE = (128, 128, 0)
PURPLE = (128, 0, 128)
TEAL = (0,128,128)
PINK = (226,132,164)
MUTEDBLUE = (155,182,203)
PLUM = (221,160,221)
```


```
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.utc
    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 'Estern 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,dayofweek)
```


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


   timee=utc[11:19]
   hr=utc[11:13]
   dayy=int(gpsd.utc[8:10])
   hrnum=int(hr) - int(off_set)  #hour - zolu offset
   hrstr=str(hrnum)
   if hrnum<0:       # Previous day

     hrnum=hrnum+24
     dayy=dayy-1
     hrstr =str(hrnum)

   if hrnum > 12:
      hrnum =hrnum-12
      hrstr= str(hrnum)
      daynite ='PM'
   else:
      daynite = 'AM'
      if hrnum==12:  ### fixes 12 AM error
         daynite = 'PM'
   if hrnum<10:       #adds 0
      hrstr ='0'+hrstr
   #ttime =hrstr + utc[13:19] + ' ' +daynite +' '+timetag+' '+gpsd.utc[5:7]+'/'
+str(dayy)+'/'+gpsd.utc[2:4]
   ttime =hrstr + utc[13:19] + ' ' +daynite +' '+daytag[0:3]+' '+gpsd.utc[5:7]+'/'
+str(dayy)+'/'+gpsd.utc[2:4]

   ####str(dst(gpsd.utc))[1]
   return (ttime)

gpsd = None #seting the global variable
```


```
os.system('./gpssock')  # 1/17/20
os.system('clear') #clear the terminal (optional)

class GpsPoller(threading.Thread):
  def __init__(self):
    threading.Thread.__init__(self)
    global gpsd #bring it in scope
    gpsd = gps(mode=WATCH_ENABLE) #starting the stream of info
    self.current_value = None
    self.running = True #setting the thread running to true

  def run(self):
    global gpsd
    while gpsp.running:
      gpsd.next() #this will continue to loop and grab EACH set of gpsd info to clear the buffer

if __name__ == '__main__':
  gpsp = GpsPoller() # create the thread
  gpsp.start() # start it up

  while True:
      try:

        while True:
      #It may take a second or two to get good data
      #print gpsd.fix.latitude,', ',gpsd.fix.longitude,'  Time: ',gpsd.utc

          os.system('clear')

          latt = gpsd.fix.latitude




      #print gpsd.utc
          #print gpsd.fix.time
          print ' New Hope Valley Railway'
          print
          #print 'speed (m/s) ' , gpsd.fix.speed
```


```
        print  'Speed : '+ str (gpsd.fix.speed *2.236)[0:5] + ' mph'
        #speedd = str (gpsd.fix.speed *2.236)[0:5]
        speedd = str (round(gpsd.fix.speed *2.236,1))
        #print gpsd.utc
        print speedd
        #print str(dst(gpsd.utc))[1]
        #print str(dst(gpsd.utc))[0]



#########################################################################
#######
        # This section is for the Speedometer
    #####  from /home/pi/learn_pygame/pigame12.py
        ####
        mph = " MPH"

        if len(speedd)>3:
           #speedd =speedd[:2]
           speedd = str (round(gpsd.fix.speed *2.236))[0:2] # rounds the speed

        pygame.display.set_caption(str(script)+' '+ ver)  # Title of window
        #ball = pygame.image.load('/home/pi/learn_pygame/logo.jpg').convert()
        ball = pygame.image.load('/home/pi/speedometer/logoTFT.jpg').convert()
        # The next line clears the speed before it is updated
        clr_speed = pygame.image.load('/home/pi/speedometer/clrscreen.jpg').convert()
        #clr_speed = pygame.image.load('/home/pi/learn_pygame/clrscreen.jpg').convert()
        pygame.display.set_mode((480,320),0,32) ## sets full screen
        screen.blit(ball,(10,10))
        speedlimit = pygame.image.load('/home/pi/speedometer/speed_limit_40x50.jpg').convert()

        def show_date(message):
           font = pygame.font.Font(None,56)
           text = font.render(message, 1,(200,200,200))
           #screen.blit(text, (75,325))
           screen.blit(text, (10,200))

        def show_mph(message):
```


```
    font = pygame.font.Font(None,75)
    text = font.render(message, 1,(200,200,200))
    #screen.blit(text, (400,200))
    screen.blit(text, (350,100))

def display(message):   ##### show speed


    font = pygame.font.Font(None,200)
    text = font.render(message, 1,(0,250,250))
    screen.blit(text, (85,60))  # Column, Row


def disclaimer(message):
    font = pygame.font.Font(None,20)
    text = font.render(message, 1,(200,200,200))
    #screen.blit(text,(200,120))
    screen.blit(text,(50,60))

def bad_data(message):
    font = pygame.font.Font(None,35)
    text = font.render(message, 1,(200,200,200))
    screen.blit(text, (15,100))



screen.fill(BLACK)
screen.blit(ball,(10,10))
screen.blit(clr_speed,(320,200))
if gpsd.fix.speed *2.236 > 15:
    screen.blit(speedlimit,(20,100))



if speedd == 'nan':
    bad_data('Waiting for good data')

else:
    display('  '+ speedd)
```


```
        show_mph('MPH')
        disclaimer('Accurate to + or - 1 mph with a settling time of 2 sec')

        if speedd !='nan':  ## When system is started data is no good
            if update_timetag == 0:

                update_timetag=1
                timetag = dst(gpsd.utc)[0]
                daytag = dst(gpsd.utc)[1]
                if timetag =="DST":
                    off_set = 4
                else:
                    off_set = 5




            show_date(str(EDST(gpsd.utc)))



        pygame.display.update()
```

```
##########################################################################
###
    ########################################


#####################################
```


```
    time.sleep(.2) #set to whatever

  except (KeyboardInterrupt, SystemExit): #when you press ctrl+c
   print "\nTEDDDDD"

   print "    ",gpsd.fix.latitude
   print "    ",gpsd.fix.longitude

print'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%'
   file = open('/home/pi/gps_2017/datafile.txt', 'a')
   file.write('   '+str(gpsd.fix.latitude)+',')
   file.write('   '+str(gpsd.fix.longitude)+',')
   GPIO.output(11,1)
   print usbnum  ### The number of active USB ports
```