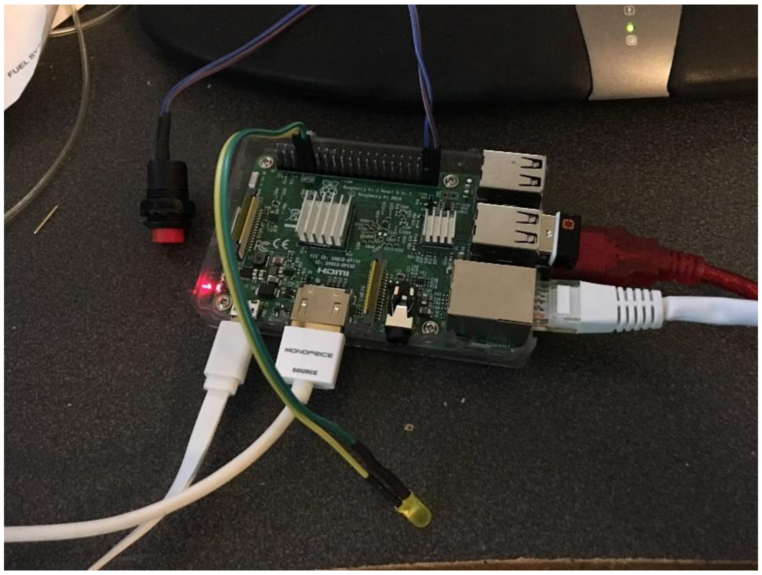
MEGR3092 Advanced Motorsports Instrumentation. Raspberry pi project.

V0.38 10/4/16



Logger build guide:

You will likely find this easier with a HDMI cord and a usb keyboard.

If those are not available, you should be able to login via SSH using a program called Putty. The Pi supports bonjour by default. This means you can access it by typing the hostname.local from FTP, SSH or other client. Your hostname is likely obdpi.local if you have modified it. If you have not, it will default to raspberrypi.local

For Example, use filezilla to download you logs over a network connection. Hostname:Sftp://obdpi.local User:pi Pw:gofast

You will need to know the IP address or the hostname of the pi (as discussed above) to continue and login to your pi via putty over Ethernet.

Install raspbian first.

Download Raspbian (HIGHLY RECOMMENDED OVER NOOBS) NOOBS CAN TAKE HOURS TO INSTALL

Or Download Noobs https://www.raspberrypi.org/downloads/noobs/

If installing raspbian directly, copy to micro sd card using the windows program win32diskimager https://sourceforge.net/projects/win32diskimager/

Connect pi to hdmi, micro usb power and attach keyboard

Install raspbian with pixel if you installed noobs.

(20-30 minutes)

Login the first time.

Open terminal. You now will see a “console”

Run “sudo raspi-config”

Change password. For the class images will have a password of “gofast”

“sudo passwd root”

“Sudo passwd pi”

Using sudo raspi-config under advanced, Change hostname to “obdpi”

Change keyboard to US, English.

Turn on rdp (remote desktop)

Sudo raspi-config and open advanced settings. Turn on RDP

Sudo apt-get install cutecom

Use this tutorial and install the software

http://www.cowfishstudios.com/blog/obd-pi-raspberry-pi-displaying-car-diagnostics-obd-ii-data-on-an-aftermarket-head-unit

Please use This for the git repository instead!

https://github.com/macsboost/pyobd-pi.git

from the pi directory run this command to give write permissions and access to every user

```
# sudo chmod 777 pyobd-pi
```

```
# cd pyobd-pi
```

Run with a monitor attached or remote desktop in.

```
# python pyobd
```

select configure and choose the USBtty0 as the device.

** baud rate in obd_io.py was changed to 115200 for our hardware. Other hardware may need 38400

To test a connection, open up a terminal and run

```
:#  cd pyobd-pi

#  sudo su

#  python obd_gui.py
```

Use the Left and Right arrow key to cycle through the gauge display.

Note: Left and Right mouse click will also work

To exit the program just press Control and C or Alt and Esc.

If you would like to log your data run:

```
#  cd pyobd-pi

#  python obdlog.py
```

To exit the program just press Control and C or Alt and Esc.

The logged data file will be saved under:

/home/username/pyobd-pi/log/

data can be analyzed with megalogviewer

NOTE! If you installed xrdp on an raspbian Jessie(most likely)

You will need to uninstall it to upgrade to raspbian with Pixel and chromium.

Do not UPDATE AT CLASS WED. It will take hours. The software will work on the old raspbian version without pixel just fine.

However, this is how you do it.

```
sudo apt-get purge xrdp
sudo apt-get update
sudo apt-get dist-upgrade
sudo apt-get install -y rpi-chromium-mods
sudo apt-get install -y realvnc-vnc-server
sudo raspi-config
```

go to advanced settings and turn on the RDP server.

Now you can connect via RDP to the pi again. Use the bonjour address of raspberrypi.local or obdpi.local

More details:

https://www.raspberrypi.org/blog/introducing-pixel/

OBD-Pi: Raspberry Pi Displaying Car Diagnostics (OBD-II) Data On An Aftermarket Head Unit

In this tutorial you will learn how to connect your Raspberry Pi to a Bluetooth OBD-II adapter and display realtime engine data to your cars aftermarket head unit.

Hardware Required:
1. Raspberry Pi
2. Aftermarket head unit (Note: Must support Auxiliary input)
3. Plugable USB Bluetooth 4.0 Low Energy Micro Adapter
4. 2A Car Supply / Switch or Micro USB Car Charger
5. ELM327 Bluetooth Adapter or ELM327 USB Cable
6. RCA cable
7. Keyboard (*optional)

What is OBD-II?
OBD stands for On-Board Diagnostics, and this standard connector has been mandated in the US since 1996. Now you can think of OBD-II as an on-board computer system that is responsible for monitoring your vehicle's engine, transmission, and emissions control components.

Vehicles that comply with the OBD-II standards will have a data connector within about 2 feet of the steering wheel. The OBD connector is officially called a SAE J1962 Diagnostic Connector, but is also known by DLC, OBD Port, or OBD connector. It has positions for 16 pins.

pyOBD?
pyOBD (aka pyOBD-II or pyOBD2) is an open source OBD-II (SAE-J1979) compliant scantool software written entirely in Python. It is designed to interface with low-cost ELM 32x OBD-II diagnostic interfaces such as ELM-USB. It will basically allow you to talk to your car's ECU, display fault codes, display measured values, read status tests, etc.

I took a fork of pyOBD's software from their GitHub repository, https://github.com/peterh/pyobd, and used this as the basis for my program.

The program will connect through the OBD-II interface, display the gauges available dependent on the particular vehicle and display realtime engine data to the cars aftermarket head unit in an interactive GUI.
Software Installation
Before you start you will need a working install of Raspbian with network access.

We'll be doing this from a console cable connection, but you can just as easily do it from the direct HDMI/TV console or by SSH'ing in. Whatever gets you to a shell will work!

Note: For the following command line instructions, do not type the '#', that is only to indicate that it is a command to enter.

Before proceeding, run:
```
#  sudo apt-get update
#  sudo apt-get upgrade
#  sudo apt-get autoremove
#  sudo reboot
```

```
Install these components using the command:
#  sudo apt-get install python-serial
#  sudo apt-get install bluetooth (bluez-utils)? blueman
#  sudo apt-get install python-wxgtk2.8 python-wxtools wx2.8-i18n libwxgtk2.8-dev
#  sudo apt-get install git-core
#  sudo reboot

Next, download the OBD-Pi Software direct from GitHub
(https://github.com/Pbartek/pyobd-pi.git)

Or using the command:
#  cd ~
#  git clone https://github.com/Pbartek/pyobd-pi.git

Vehicle Installation
The vehicle installation is quite simple.

1. Insert the USB Bluetooth dongle into the Raspberry Pi along with the SD card.

2. Insert the OBD-II Bluetooth adapter into the SAE J196216 (OBD Port) connector.

3. Connect you RCA cable to the back of your aftermarket head unit and plug the other
end into your Raspberry Pi.

4. Install your 2A Car Supply / Switch or Micro USB Car Charger.

5. Finally turn your key to the ON position and navigate your head unit to Auxiliary
input.

6. Enter your login credentials and run:
#  startx

7. Launch BlueZ, the Bluetooth stack for Linux. Pair + Trust your ELM327 Bluetooth
Adapter and Connect To: SPP Dev. You should see the Notification "Serial port
connected to /dev/rfcomm0"

Note: Click the Bluetooth icon, bottom right (Desktop) to configure your device.
Right click on your Bluetooth device to bring up Connect To: SPP Dev.

8. Open up Terminal and run:
#  cd pyobd-pi
#  sudo su
#  python obd_gui.py

Use the Left and Right arrow key to cycle through the gauge display.
Note: Left and Right mouse click will also work

To exit the program just press Control and C or Alt and Esc.
Update:
Data Logging
If you would like to log your data run:
#  cd pyobd-pi
#  python obd_recorder.py

The logged data file will be saved under:
```

```
/home/username/pyobd-pi/log/

Enjoy and drive safe!
```

Helpful info:

http://www.makeuseof.com/tag/15-useful-commands-every-raspberry-pi-user-should-know/

http://www.cowfishstudios.com/blog1.html

http://www.cowfishstudios.com/blog/obd-pi-raspberry-pi-displaying-car-diagnostics-obd-ii-data-on-an-aftermarket-head-unit

to start pi gui # startx

to configure options: # sudo raspi-config

the software is installed in "/home/pi/pyobd-pi"

Our USB adapters show up as "ttyUSB0" under the root folder /dev

Change directory # cd pyobd-pi

To go back a directory # cd ..

To go back to your user home director # cd ~

To make a directory # mkdir

To remove a directory # rmdir

To close a program <Ctl C>

To start the gui # startx

Use "sudo" to increase your permission level to allow a program to execute. i.e. # sudo reboot

Display raspi IP address:

# ifconfig eth0

Or

# ip a

Iphone tools:

Fing – network scanner

Or, use raspberrypi.local or obdpi.local, depending on your hostname.

Webssh – ssh terminal

`/boot/config.txt` Remoter pro $4.99 for graphical login over wifi to the xrdp server.

Connect your pi to your phone and use your phone as a remote control.

GIT:

To update our software, go Into the pyobd-pi folder and execute

```
# sudo git pull
```

To configure git,

```
#sudo git config –global user.email email@uncc.edu
```

To save your changes:

```
# sudo git commit
```

When you are ready to push to the git server

```
# sudo git commit
```

To add a file to the repository

```
# sudo git add filenametoadd.example
```

Getting the python script to auto run on boot:

I have created a new script that responds to GPIO input to trigger logging.

This script is called obdlog.py. This does not require a logging switch. I recommend plugging in the OBD2 adapter to the car, boot the pi, then run this python script to log.

Alternatively, the latest script that uses a switch is called obdgpslog2.py The obdgpslog2 script can be used without a gps. You must use a logging switch to start logging with this version.

(follow the steps in this guide for auto load on boot, but look at the line below for the actual crontab command)

http://www.instructables.com/id/Raspberry-Pi-Launch-Python-script-on-startup/step3/Add-logs-directory/

To edit the crontab script use the following command

```
# sudo crontab -e
```

add one of the following lines to crontab script

Without GPS:

```
@reboot sleep 20 & sh /home/pi/pyobd-pi/launcher.sh >/home/pi/logs/cronlog 2>&1
```

With GPS:

```
@reboot sleep 20 & sh /home/pi/pyobd-pi/launchergps.sh >/home/pi/logs/cronlog 2>&1
```

You will need to create a /logs directory under /home/pi using the mkdir logs command.

** the sleep command is required to allow adequate time for devices to connect. It prevents the program from generating an exception if started too early in the boot process.

To stop your progam after boot, should you need to, open the terminal and type in

```
# sudo killall python
```

You may want to do this to kill the program, modify it so that you can reload it without a reboot.

When auto started, you can view the terminal output by pressing ctl-alt-F2. You can get back to the gui by pressing ctl-alt-F7

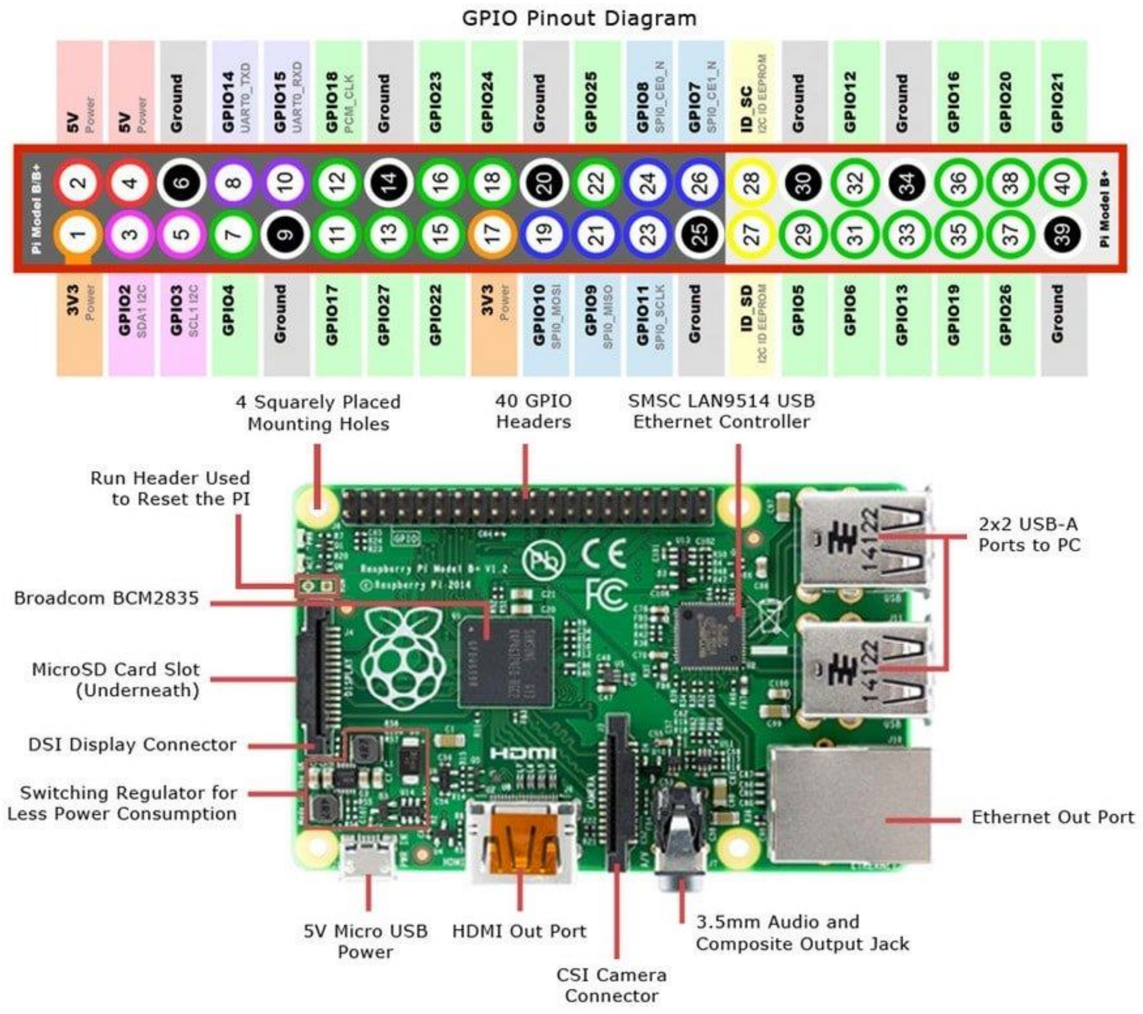
Logging triggers:

The obdlog.py will log on start by default. It can be changed by changing the bootup log variable to False.

The obdgpslog2.py script will by default log when pins 37 and 39 on the pi are shorted. You need an SPST switch if you want to use a switch. Wire each leg of the switch to each pin. The pin is setup with an internal pull up. The switch shorts out the two pins and pulls the sense pin to ground. The script waits for the transition and then starts logging. If the system boots with the switch pressed in it will not log. It requires being unswitched and then rearmed.

If you wish to add an LED for indicating the python script is logging, it can be added between pins 33, 34 along with the appropriate resistor. . A 220-500 ohm resistor should work.

The positive side of the LED should point to pin 33.



Default behavior can be changed for either script by editing the script files with a text editor.

If a HDMI monitor is not attached to the pi at boot, it defaults to the RCA or headphone jack video output.

To force the HDMI on at boot you can modify this file

/boot/config.txt

and add these two lines:

```
hdmi_force_hotplug=1

hdmi_drive=2
```

To have permissions to write this file use

```
# cd /

# cd boot

# sudo nano config.txt
```

Use PS to determine if your program is running

```
# sudo ps axg | grep python
```

You will see a list of 3 processes, two with python if it is running properly.

Adding a GPS Receiver

https://blog.retep.org/2012/06/18/getting-gps-to-work-on-a-raspberry-pi/

setup the port correctly

http://catb.org/gpsd/installation.html

For UBLOX ttyACM0

Fixes

https://learn.adafruit.com/adafruit-ultimate-gps-on-the-raspberry-pi/setting-everything-up

```
sudo dpkg-reconfigure gpsd
```

```
sudo service ntp restart
```

add to autostart

gpsd /dev/ttyACM0

GPS Fixes:

by amrbekhit » Wed Apr 16, 2014 11:10 am

I know this is quite an old post, but it shows up frequently in search results related to gpsd problems on bootup.

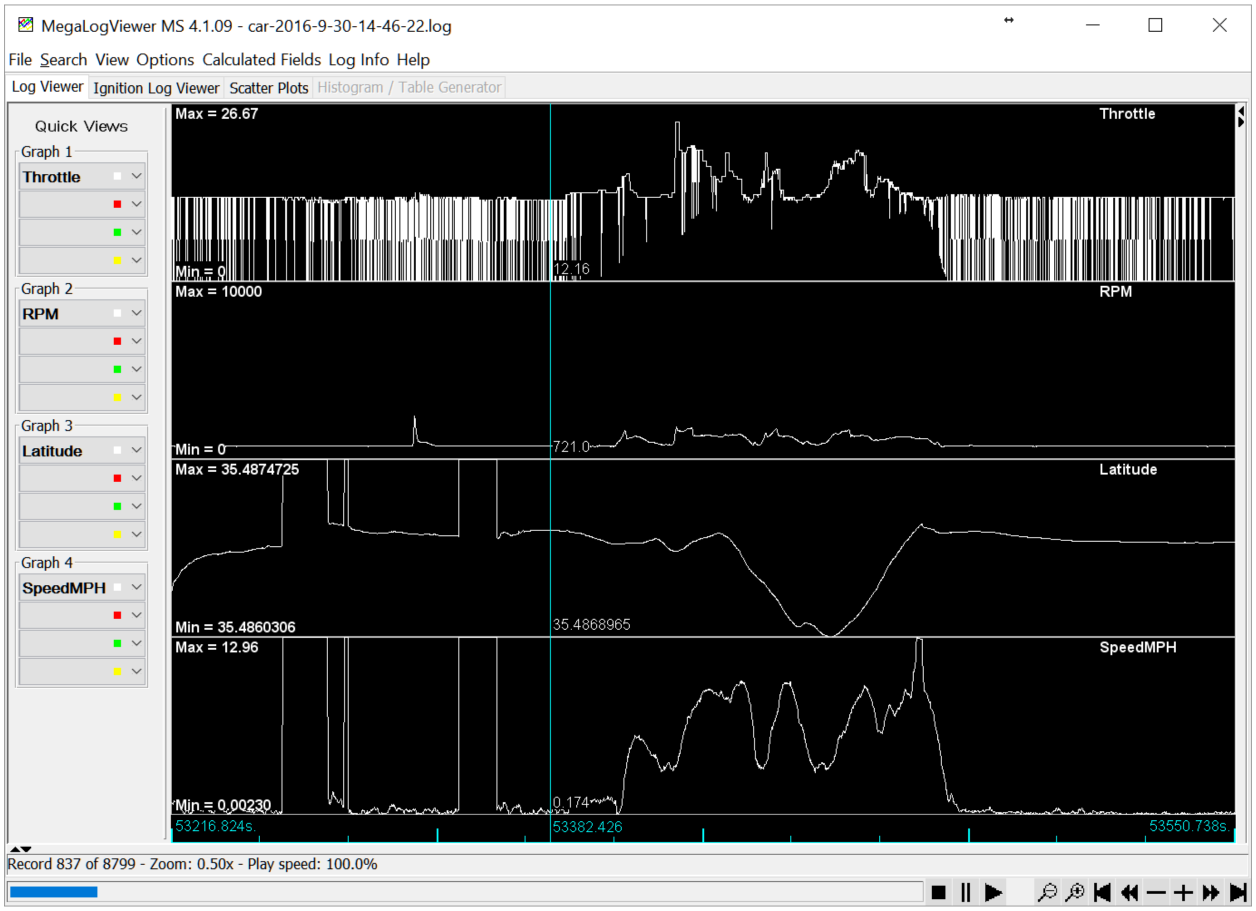
Here's a simpler solution than messing around with udev.

As brodieh rightly pointed out, running `ps aux | grep gps` shows that gpsd is not being started up with the correct paramters, despite the /etc/defaults/gpsd file being set up "correctly" via `dpkg-reconfigure gpsd`.

A simple way to fix that is to modify /etc/default/gpsd so that rather than the device name being placed under the DEVICES option, place it under the GPSD_OPTIONS option instead. For example, do this:

CODE: SELECT ALL

```
# Default settings for gpsd.
# Please do not edit this file directly - use `dpkg-reconfigure gpsd' to
# change the options.
START_DAEMON="true"
GPSD_OPTIONS="/dev/ttyACM0"
DEVICES=""
USBAUTO="true"
GPSD_SOCKET="/var/run/gpsd.sock"
```



Files can be opened by the megalogviewer

Copy with a usb stick, or SFTP Into the pi and get the log. You use this windows viewer to look at the data.