This application can be used with or without an ECU connection and a Raspberry Pi. I have implemented a debug mode for this very reason, allowing the application to be ran directly on any system that has python installed. This debug mode can be accessed in the config module of the application. Setting the debug mode flag to true will enable debug mode in the application and the application will not attempt to make any connection to the ECU on start up. This flag also tells the application to read the values stored in the debug log csv file instead of from the ECU. This debug mode allows the user to test the application and the GUI.

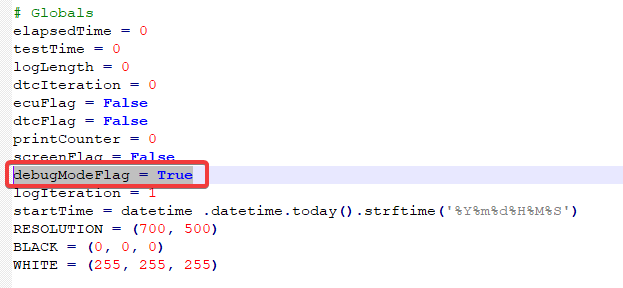


Figure 25 - DebugModeFlag in config.py

The user must ensure that Python is installed on either the raspberry pi or on the machine they are debugging from. This allows the application to be run on virtually any machine that supports the python libraries. To install Python on the Raspberry Pi the below commands must be used:



Figure 26 - Installing Python on RPi

You can check whether Python has been installed successfully by entering the command:

**python –version**

If Python is correctly installed and added to your PATH, this command will return the current version of Python installed, which will be Python 3.6.

To install Python on a windows machine, the latest version installer can be downloaded from the link below:

<https://www.python.org/downloads/windows/>

The windows installer will also add Python to your PATH allowing Python to be run from anywhere on your system. The install can be verified using the same command as above.

Once Python has successfully been installed on your system, pip and the dependencies must also be installed. If you are using a Raspberry Pi, this can be done using the following command:

**sudo apt-get install python-pip**

This installs pip, which allows us to install the other required python libraries with ease using the following commands:

**pip install obd**

**pip install numpy**

**pip install pygame**

This installs the Python OBD library, the numpy library and the PyGame library, which is responsible for drawing the GUI.

pip should be included with the windows install of Python, if not, it can be downloaded by executing the script in the link below:

<https://pip.pypa.io/en/stable/installing/>

The other dependencies can be installed using pip with the same commands once it has been successfully installed on your system.

If you are using a windows machine, you are now ready to use the application. Navigate to the root directory of the application, \Piagnostics, and execute piagnostics.py using the command line command below:

**python piagnostics.py**

The application should run, and the GUI should be displayed with the debug log values. The window can be clicked, and the GUI will display the DTCs menu, where any DTCs will be shown and logged.

If you are running from the Raspberry Pi, there are some additional steps. Before you take the Raspberry Pi to the vehicle, there are some steps we can take now to make our life easier down the line.

Flash the latest version of Raspbian Lite using an SD card and SD card reader. The image and steps on this can be found here:

<https://www.raspberrypi.org/documentation/installation/installing-images/>

Once Raspbian Lite has been flashed and is running, we must store our network details for the Wi-Fi connection in **/etc/wpa\_supplicant/wpa\_supplicant.conf.**

This will allow us the SSH into the Raspberry Pi for ease of use and to send commands without the need to connect a display to the Raspberry Pi.

SSH can be enabled in the raspi-config file located in the home directory. The hostname and password can also be set here.

Finally, we must transfer the application to the Raspberry Pi and make the application run on start-up of the Raspberry Pi, this will remove the need to manually run the application each time the vehicle is started. Once the files have been transferred, insert the code below into **/etc/rc.local.** . I keep the files in my /home directory, but you can append the code to match your directory:

**python /home/Piagnostics/piagnostics.py**

You are now ready to connect the Raspberry Pi to a power source within the vehicle, for example a USB port. When the vehicle is started the Pi will boot and execute the application. Plug the OBD-II ELM327 adapter into the OBD-II port of your vehicle. Check your owner’s manual if you cannot locate the port in your vehicle.

Once the OBD-II adapter is plugged into your vehicle, you can start the vehicle. The Pi should bootup and the OBD-II adapter should start flashing.

Next, we must configure the OBD-II adapter to connect with the Raspberry Pi via Bluetooth and make the ECU connection. Once the LED’s on both the Raspberry Pi and the OBD-II adapter have turned green they are both fully initialized and ready to be used. SSH into the Raspberry Pi. This can be done using a number of terminal emulators. My SSH client of choice is Putty, which can be downloaded from here:

<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

Once you have successfully SSH into the Raspberry Pi, there are a number of commands that must be ran to connect the OBD-II adapter to the Raspberry Pi correctly. They can be seen below:

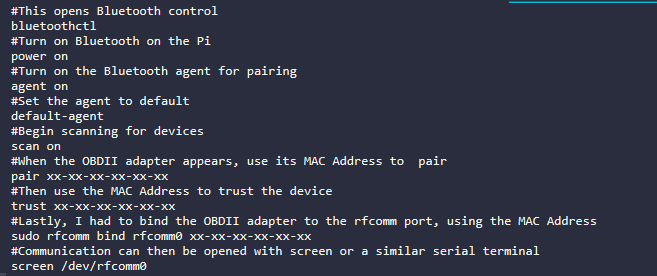


Figure 27 - Commands to configure OBD-II ELM327 adapter

If executed correctly, a serial terminal will open with the connection to /dev/rfcomm0 established. This means the OBD-II adapter has been successfully paired with the Raspberry Pi and will now be paired automatically each time the OBD-II adapter and the Raspberry Pi are powered on and plugged in.

Restart the Raspberry Pi and cycles the engine of your vehicle with a restart. The application will now be running and logging data. These logs will be stored in the /logs folder in the /Piagnostics directory.