Preparing your Raspberry Pi for the NMEA2000 library

Linux has included CAN (Control Area Network) support via socketCAN. Utilizing this interface to enable the NMEA2000 library on a Raspberry Pi involves three steps:
1. Selecting and installing CAN interface hardware
2. Configuring and starting Linux socketCAN
3. Using the NMEA2000 library

More can be learned about socketCAN and Linux at the following links:
https://www.kernel.org/doc/Documentation/networking/can.txt
https://www.can-cia.org/fileadmin/resources/documents/proceedings/2012_kleine-budde.pdf
http://elinux.org/CAN_Bus

1. CAN Hardware Interface

The quickest and easiest CAN hardware for the RPi is the PiCAN-2 series of devices by Spang electronics:
http://skpang.co.uk/catalog/pican2-canbus-board-for-raspberry-pi-23-p-1475.html
http://copperhilltech.com/pican-2-can-interface-for-raspberry-pi/

Based on the MCP2515 CAN controller chip there are several options including dual CAN controllers, and even isolated ones. Follow the installation guide for these boards to install the hardware as well as bring up the socketCAN interface on the Raspberry Pi:
www.skpang.co.uk/catalog/images/raspberrypi/pi_2/PICAN2_jessie_2016-05-10.pdf
www.copperhilltech.com/content/PICAN2_jessie_2016-05-10.pdf

Note on RPi verion used and other non RPi Linux system

SocketCAN is a widely supported capability available on almost all current releases of Linux. The links provided in step #1 above are specific to the 2017 'jessey' release of Raspberry Pi software. If you are using a prior release for your RPi make sure to locate the corresponding instructions from the CAN hardware provider.

Once the hardware is installed and the socketCAN enabled steps 2 & 3 are universally applied to almost all releases of RPi software.

In a like way, steps 2 & 3 may also be applied to almost any Linux system - once appropriate CAN hardware has been installed and the socketCAN ability enabled. A likely approach would be through a CAN <–> USB adapter. Searching for 'CAN USB adapter socketCAN' will show several options. See below in section *'Alternative CAN hardware'* for some links.

2. Configuring and starting socketCAN

After following the driver installation instructions above, you can next test your CAN hardware before proceeding. The simplest way is to install Linux `can-utils` , connect your can hardware to some other known CAN device which is broadcasting information, and then do a can dump.

'Start up' the CAN port using:

```
$ sudo /sbin/ip link set can0 up type can bitrate 250000
```

Connect up another CAN device which sends out messages and do the following on your RPi. You should see the CAN packets displayed as they come in:

```
# Skip these two steps if can-utils are already installed
$ sudo apt-get update
$ sudo apt-get install can-utils


 $ candump can0
  can0  19FFFD80   [8]  01 78 0D 01 AB 91 35 77
  can0  19FFFC80   [7]  01 78 FF 7F FF FF FF
  can0  19FEC880   [8]  01 78 B0 34 00 00 FF FF
  can0  19FFFD80   [8]  01 78 0D 01 B2 8E 35 77
  can0  19FFFC80   [7]  01 78 FF 7F FF FF FF
  can0  19FEC880   [8]  01 78 B6 34 00 00 FF FF
  can0  19F21280   [8]  00 09 49 01 00 FF FF FF
  can0  19F21280   [8]  01 FF FF FF 00 00 00 00
  can0  19F21480   [8]  01 46 05 F9 FF 00 00 49
  can0  19F21980   [8]  01 D0 01 DC 05 00 FF 00
  can0  19FFFD80   [8]  01 78 0E 01 5D 93 35 77
  can0  19FFFC80   [7]  01 78 FF 7F FF FF FF
  can0  19FEC880   [8]  01 78 BE 34 00 00 FF FF
  can0  19FFFD80   [8]  01 78 0E 01 73 8A 35 77
  can0  19FFFC80   [7]  01 78 FF 7F FF FF FF
  can0  19FEC880   [8]  01 78 BE 34 00 00 FF FF
```

This will verify the hardware and the socketCAN software is functioning correctly.
Hint: place the 'Start-up' command in your `/etc/rc.local` file so the CAN port will be started each time the RPi boots:

```
$sudo nano /etc/rc.local
```

and add the following line(s) at the end of this file:

```
# Initiate the CAN ports
sudo /sbin/ip link set can0 up type can bitrate 250000
sudo /sbin/ip link set can1 up type can bitrate 250000
```

(Only add the last line if you have a dual-port CAN adapter).

3. Using the NMEA2000 library

Once you have verified the hardware and socketCAN is functioning import the following two libraries into your development environment**:

https://github.com/ttlappalainen/NMEA2000
https://github.com/thomasonw/NMEA2000_socketCAN

Then use the built-in examples to test your new CAN enabled Raspberry Pi - or the following simple program to emulate `canDump` above:

```
/*
 * File:   main.cpp
 * Author: al
 *
 * Testing for CAN and RPI
 *
 * See: https://github.com/thomasonw/NMEA2000_socketCAN
 *
 * Created on February 12, 2017, 2:37 PM
 */

#include <cstdlib>
#include <stdio.h>
#include <iostream>
#include "NMEA2000_CAN.h"

using namespace std;

int main(void)
{
    cout << "Starting CAN watching" << endl;

    setvbuf (stdout, NULL, _IONBF, 0);                                          // No buffering on stdout, just send chars as they come.

    NMEA2000.SetCANPort("can0");                                                // Select the CAN port @ can0 (default)
                                                                                // This is new (optional) call to allow you to use a different socketCAN port


    NMEA2000.SetForwardStream(&serStream);                                      // Connect bridge function for streaming output.
    NMEA2000.SetDebugMode(tNMEA2000::dm_ClearText);                             // Lets us see the debug messages on the terminal
    NMEA2000.SetForwardType(tNMEA2000::fwdt_Text);                              // Show in clear text (for now)

    if (!NMEA2000.Open()) {
       cout << "Failed to open CAN port" << endl;
       return 1;
   }

    cout  << endl << "CAN started, going to watch it now" << endl;

     while(1) {
         NMEA2000.ParseMessages();                                               // Will send out CAN messages in open text
    }

    return 0;
}
```

If everything is working well, you should see an output like this:

```
Starting CAN watching
CAN device ready

CAN started, going to watch it now
Pri:6 PGN:131069 Source:128 Dest:255 Len:8 Data:1,78,c,1,ab,91,35,77
Pri:6 PGN:131068 Source:128 Dest:255 Len:7 Data:1,78,ff,7f,ff,ff,ff
Pri:6 PGN:130760 Source:128 Dest:255 Len:8 Data:1,78,60,34,0,0,ff,ff
Pri:6 PGN:127506 Source:128 Dest:255 Len:9 Data:e8,1,0,ff,ff,ff,ff,ff,ff
Pri:6 PGN:127508 Source:128 Dest:255 Len:8 Data:1,3d,5,3,0,0,0,e8
Pri:6 PGN:127513 Source:128 Dest:255 Len:8 Data:1,d0,1,dc,5,0,ff,0
Pri:6 PGN:131069 Source:128 Dest:255 Len:8 Data:1,78,c,1,dd,8f,35,77
Pri:6 PGN:131068 Source:128 Dest:255 Len:7 Data:1,78,ff,7f,ff,ff,ff
Pri:6 PGN:130760 Source:128 Dest:255 Len:8 Data:1,78,60,34,0,0,ff,ff
Pri:6 PGN:131069 Source:128 Dest:255 Len:8 Data:1,78,c,1,59,91,35,77
Pri:6 PGN:131068 Source:128 Dest:255 Len:7 Data:1,78,ff,7f,ff,ff,ff
Pri:6 PGN:130760 Source:128 Dest:255 Len:8 Data:1,78,61,34,0,0,ff,ff
Pri:6 PGN:127506 Source:128 Dest:255 Len:9 Data:e9,1,0,ff,ff,ff,ff,ff,ff
Pri:6 PGN:127508 Source:128 Dest:255 Len:8 Data:1,3d,5,f3,ff,0,0,e9
Pri:6 PGN:127513 Source:128 Dest:255 Len:8 Data:1,d0,1,dc,5,0,ff,0
. . .
```

** Note: I use the NetBeans IDE – using the remote SSH attachment to my RPi. Details here:
http://www.raspberry-projects.com/pi/programming-in-c/compilers-and-ides/netbeans-windows/installing-netbeans-for-c-remote-development-on-a-raspberry-pi
(Make sure to install and enable SSH services on your RPi before trying to use it!)

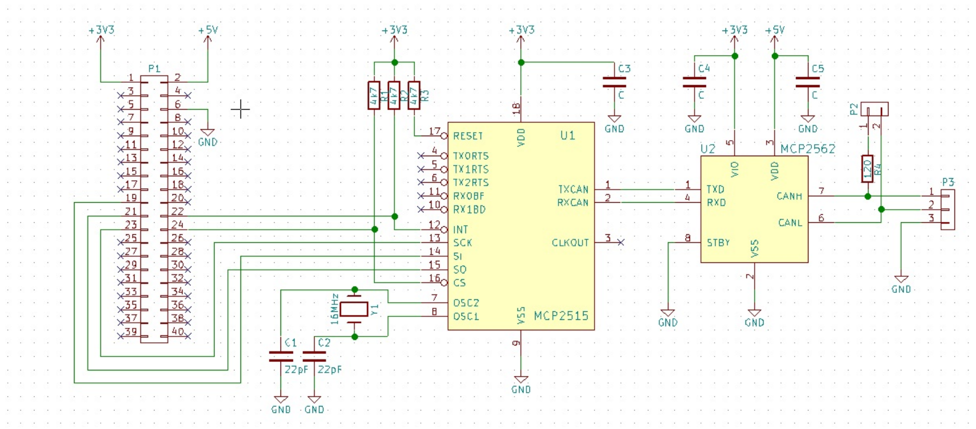
Alternative CAN hardware

In addition to the PiCAN-2 devices, one could use any CAN adapter which includes soketCAN support. An overview of those can be found here:
http://elinux.org/CAN_Bus#SocketCAN_Supported_Controllers

USB-CAN adapters are available, some for less than $20us. You will need to verify if they support socketCAN. Some examples which do include:
http://canable.io/
https://github.com/kahiroka/slcanuino
http://www.fischl.de/usbtin/

These would also be a likely choice for non PRi Linux systems. Install the USB <–> CAN adapter, socektCAN and then proceed to steps #2 and 3 above.

Another option is to use one of the widely available MCP2515 + MCP2562 adapter boards, often for around $2. But those come with one caution: The Raspberry Pi board is a 3.3v device, while the CAN bus specifics 5v levels. This is most easily solved by utilizing the MPC2562's separate IO and DRV supply pins, as shown here:



(design reference: http://lnxpps.de/rpie/)

Locating an adapter board which allows independent voltages is not simple. If you are careful one could modify a common PCB to isolate pin #3 on the MCP2562. Read more here:
https://www.raspberrypi.org/forums/viewtopic.php?f=44&t=141052
http://lnxpps.de/rpie/

Here is one board at a resonable cost with seperate IO and DRV supply pins: http://modtronix.com/im1can.html

---

(Original of this document may be found here: https://github.com/thomasonw/NMEA2000_socketCAN/blob/master/Raspberry-Pi-CAN.md)