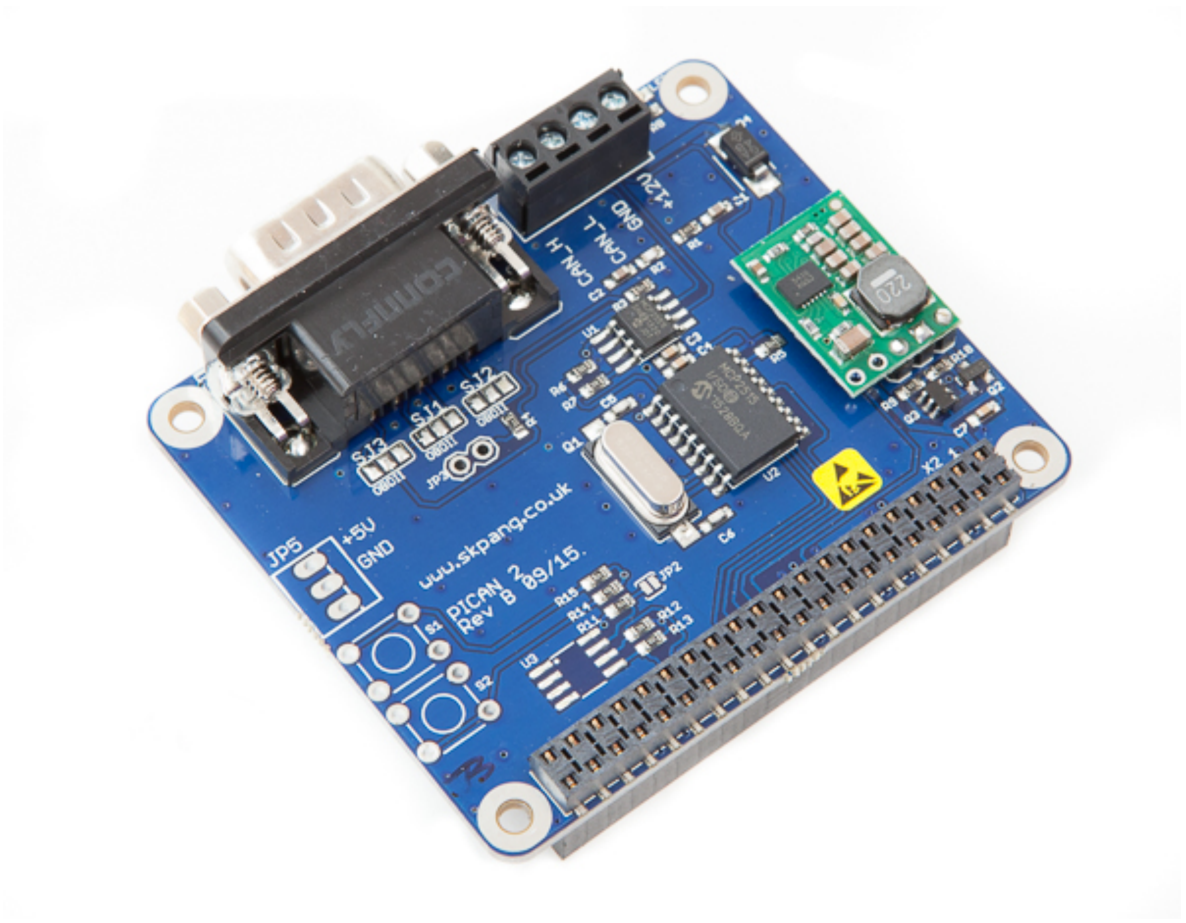


PiCAN 2 SMPS USER GUIDE

V1.3

Product name	PICAN CAN-Bus Board for Raspberry Pi
Model number	RSP-PICAN2SMPS
Manufacturer	SK Pang Electronics Ltd

Contents

Table of Contents

1. Introduction

This PiCAN board provide CAN-Bus capability for the Raspberry Pi 2. It uses the Microchip MCP2515 CAN controller with MCP2551 CAN transceiver. Connections are made via DB9 or 4 way screw terminal. This board has a 5v 1A SMPS that can power the Pi is well via the screw terminal or DB9 connector.

Easy to install SocketCAN driver. Programming can be done in C or Python.

1.1. Features

- CAN v2.0B at 1 Mb/s
- High speed SPI Interface (10 MHz)
- Standard and extended data and remote frames
- CAN connection via standard 9-way sub-D connector or screw terminal
- Compatible with OBDII cable
- Solder bridge to set different configuration for DB9 connector
- 120Ω terminator ready
- Serial LCD ready
- LED indicator
- Foot print for two mini push buttons
- Four fixing holes, comply with Pi Hat standard
- SocketCAN driver, appears as can0 to application
- Interrupt RX on GPIO25
- 5v 1A SMPS to power Raspberry Pi and accessories from DB9 or screw terminal
    - Reverse polarity protection
    - High efficiency switch mode design
    - 6v to 20v input range

2. Hardware Installation

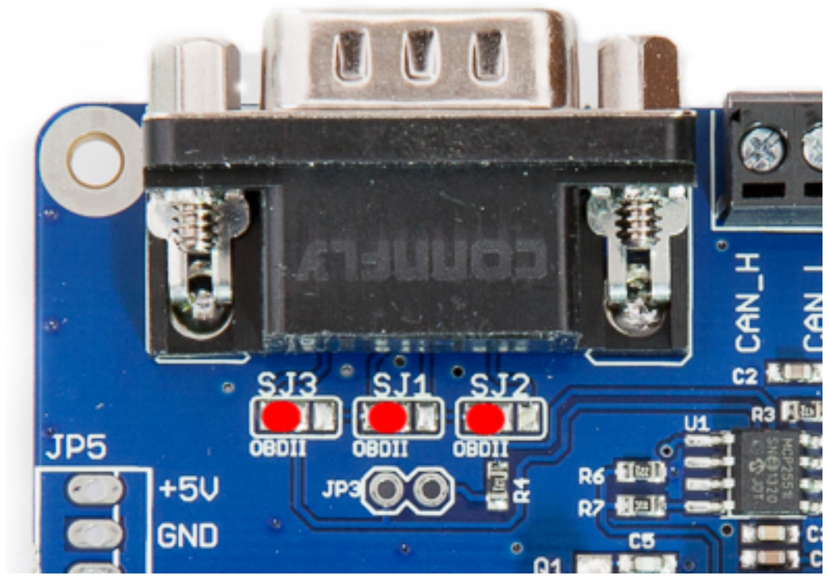
Before installing the board make sure the Raspberry is switched off. Carefully align the 40way connector on top of the Pi. Use spacer and screw (optional items) to secure the board.

1.2. Configuring DB9 Connector

The CAN connection can be made via the DB9 connector. The connector be configured for different pinout. Depend if you are using an OBDII cable or a CAN cable.

1.3. OBDII Cable

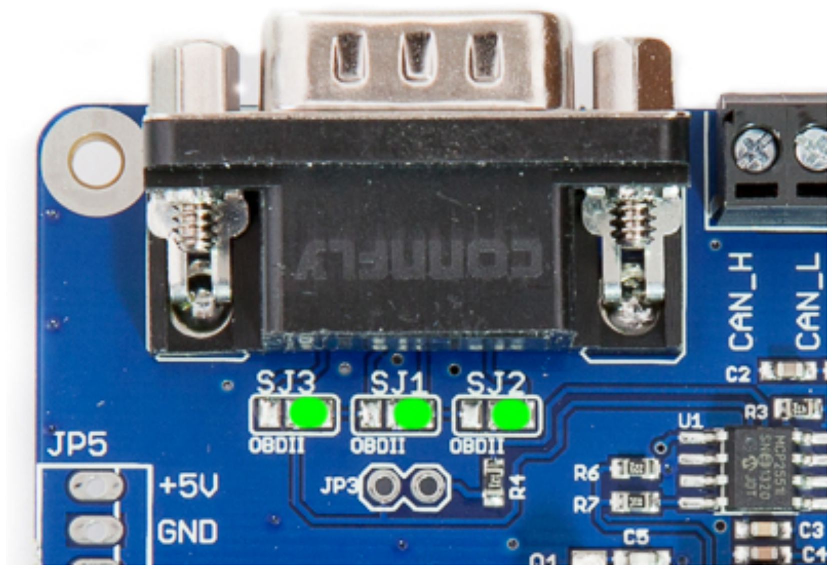
Close the solder bridges on the lefthand side on SJ1, SJ2 and SJ3 as shown with a red dot.



DB9 Pin number	Function
2	GND
3	CAN_H
5	CAN_L
9	+Vin

1.4. CAN Cable

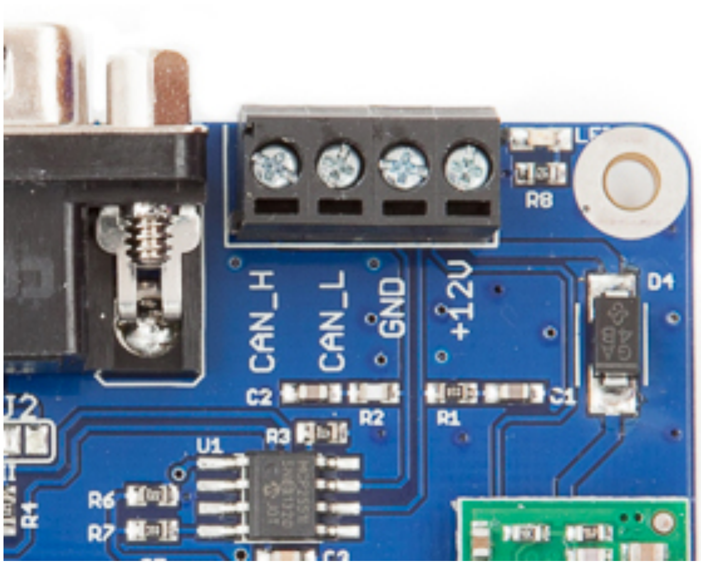
Close the solder bridges on the righthand side on SJ1, SJ2 and SJ3 as shown with a green dot.



DB9 Pin number	Function
3	GND
7	CAN_H
2	CAN_L
9	+Vin

1.5. Screw Terminal

The CAN connection can also be made via the 4 way screw terminal.

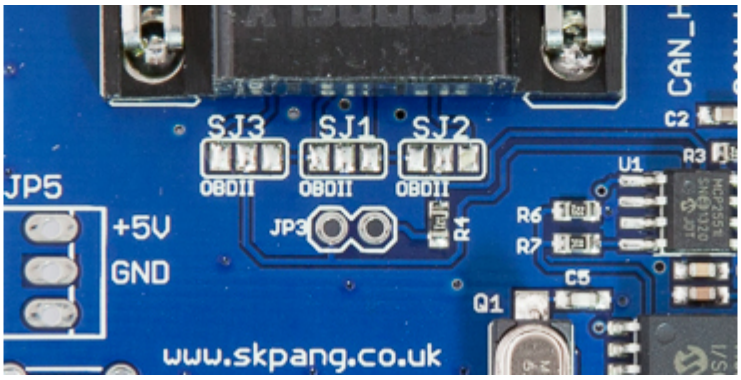


Pin number	Function
1	CAN_H
2	CAN_L
3	GND
4	+Vin

Note : The +Vin has an input voltage range 6v to 20v.

1.6. 120Ω Terminator

There is a 120Ω fitted to the board. To use the terminator solder a 2way header pin to JP3 then insert a jumper.



1.7. LED

There is a red LED fitted to the board. This is connected to GPIO22.

1.8. Not Fitted Items

The board has footprint for two mini push buttons S1 and S2, they are connected to GPIO24 and GPIO23 respectively.

JP5 can be use to power a serial LCD with data on TXD line from the Pi. There is also 5v supply on JP5.

U2 is a EEPROM for ID use.

3. Software Installation

It is best to start with a brand new Raspbian image. Download the latest from:

https://www.raspberrypi.org/downloads/raspbian/

After first time boot up, do an update and upgrade first.

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

```
sudo reboot
```

Add the overlays by:

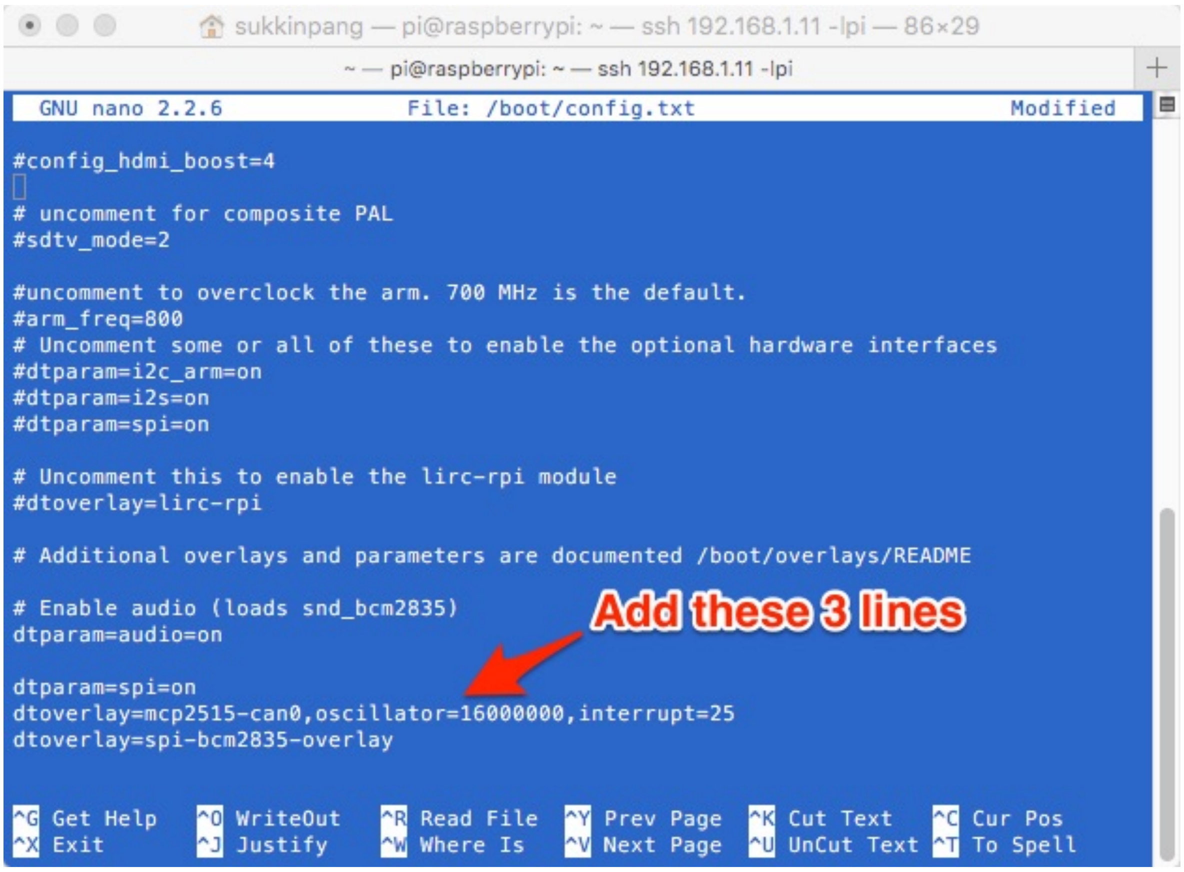
```
sudo nano /boot/config.txt
```

Add these 3 lines to the end of file:

```
dtparam=spi=on
```

```
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=25
```

```
dtoverlay=spi-bcm2835-overlay
```


sukkinpang — pi@raspberrypi: ~ — ssh 192.168.1.11 -lpi — 86×29
~ — pi@raspberrypi: ~ — ssh 192.168.1.11 -lpi
Add these 3 lines

```
  GNU nano 2.2.6            File: /boot/config.txt                         Modified

#config_hdmi_boost=4

# uncomment for composite PAL
#sdtv_mode=2

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800
# Uncomment some or all of these to enable the optional hardware interfaces
#dtparam=i2c_arm=on
#dtparam=i2s=on
#dtparam=spi=on

# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
dtparam=audio=on

dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=25
dtoverlay=spi-bcm2835-overlay


^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
```

Reboot Pi:

```
sudo reboot
```

1.9. Bring Up the Interface

You can now bring the CAN interface up:

```
sudo /sbin/ip link set can0 up type can bitrate 500000
```

Download and copy the CAN test programs to the Pi.

http://www.skpang.co.uk/dl/can-test_pi2.zip

Connect the PiCAN2 to your CAN network via screw terminal or DB9.

To send a CAN message use :

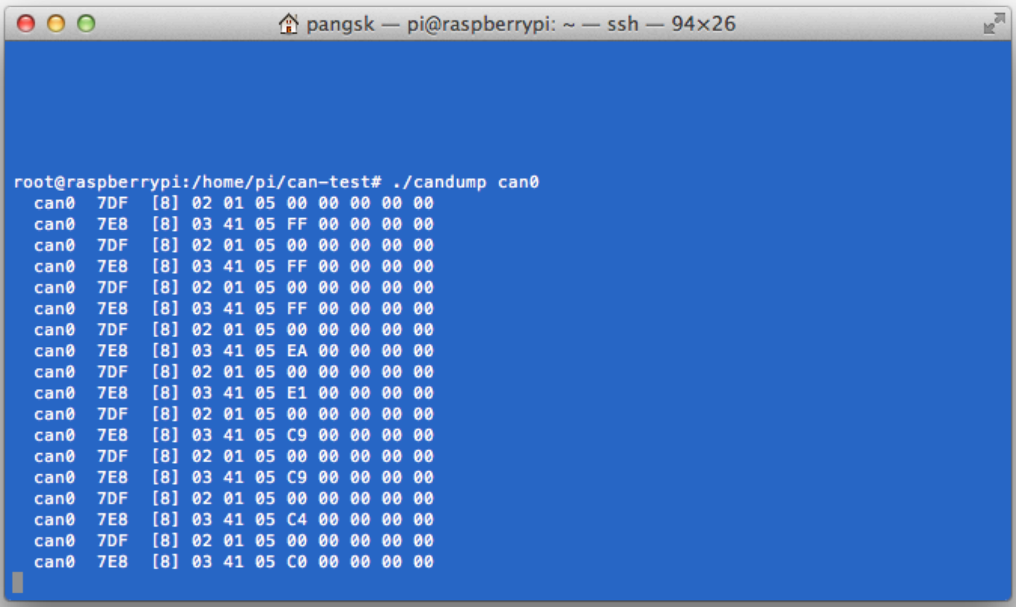
```
./cansend can0 7DF#0201050000000000
```

This will send a CAN ID of 7DF. Data 02 01 05 – coolant temperature request.

Connect the PiCAN to a CAN-bus network and monitor traffic by using command:

```
./candump can0
```

You should see something like this:



```
root@raspberrypi:/home/pi/can-test# ./candump can0
  can0  7DF  [8] 02 01 05 00 00 00 00 00
  can0  7E8  [8] 03 41 05 FF 00 00 00 00
  can0  7DF  [8] 02 01 05 00 00 00 00 00
  can0  7E8  [8] 03 41 05 FF 00 00 00 00
  can0  7DF  [8] 02 01 05 00 00 00 00 00
  can0  7E8  [8] 03 41 05 FF 00 00 00 00
  can0  7DF  [8] 02 01 05 00 00 00 00 00
  can0  7E8  [8] 03 41 05 EA 00 00 00 00
  can0  7DF  [8] 02 01 05 00 00 00 00 00
  can0  7E8  [8] 03 41 05 E1 00 00 00 00
  can0  7DF  [8] 02 01 05 00 00 00 00 00
  can0  7E8  [8] 03 41 05 C9 00 00 00 00
  can0  7DF  [8] 02 01 05 00 00 00 00 00
  can0  7E8  [8] 03 41 05 C9 00 00 00 00
  can0  7DF  [8] 02 01 05 00 00 00 00 00
  can0  7E8  [8] 03 41 05 C4 00 00 00 00
  can0  7DF  [8] 02 01 05 00 00 00 00 00
  can0  7E8  [8] 03 41 05 C0 00 00 00 00
```

4. Writing Your Own Software

You can write your own application software in either C or Python.

1.10. Application in Python

Download the Python-CAN files from:

```
https://bitbucket.org/hardbyte/python-can/get/4085cffd2519.zip
```

Unzip and install by

```
sudo python3 setup.py install
```

Bring the CAN interface up if it is not already done:

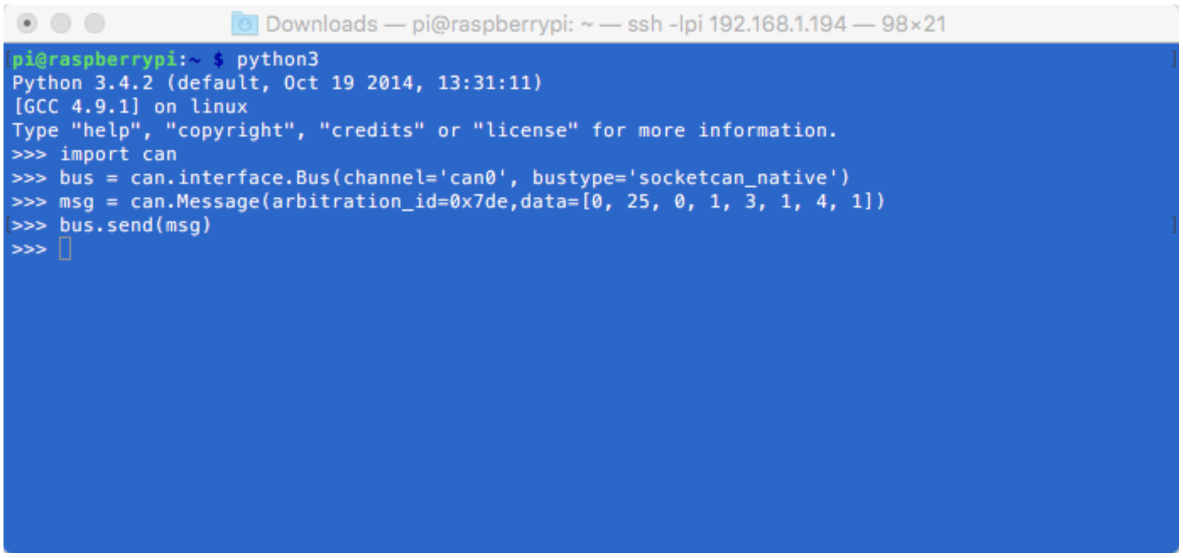
```
sudo /sbin/ip link set can0 up type can bitrate 500000
```

Now start python3

```
python3
```

To sent a message out type the following lines:

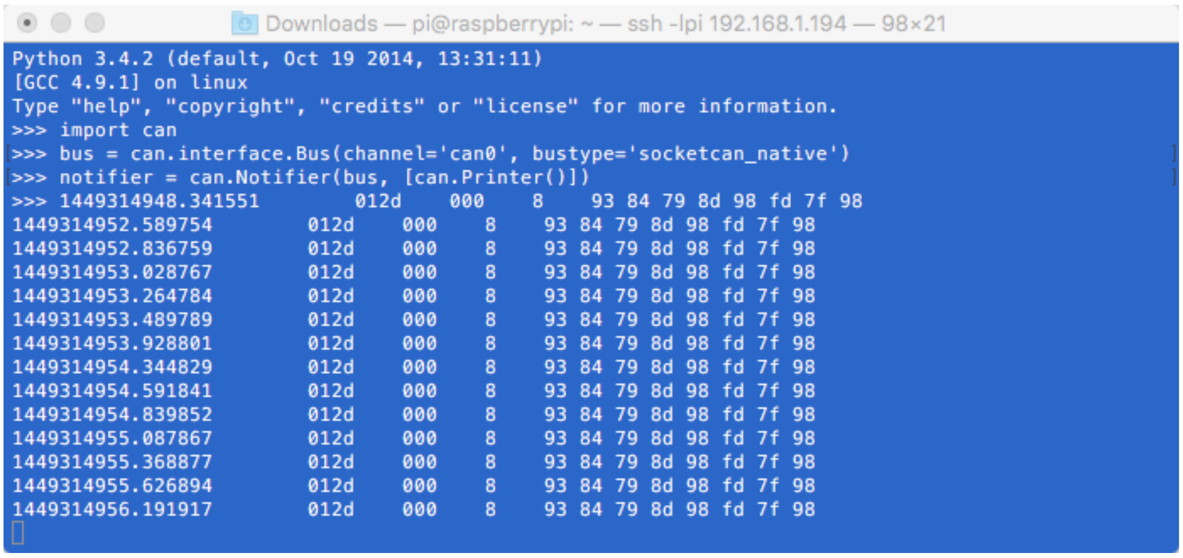
```
import can

bus = can.interface.Bus(channel='can0', bustype='socketcan_native')

msg = can.Message(arbitration_id=0x7de,

                  data=[0, 25, 0, 1, 3, 1, 4, 1],

                  extended_id=False)

bus.send(msg)
```


Downloads — pi@raspberrypi: ~ — ssh -lpi 192.168.1.194 — 98×21

```
pi@raspberrypi:~ $ python3
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import can
>>> bus = can.interface.Bus(channel='can0', bustype='socketcan_native')
>>> msg = can.Message(arbitration_id=0x7de,data=[0, 25, 0, 1, 3, 1, 4, 1])
>>> bus.send(msg)
>>> 
```

To received messages and display on screen type:

```
notifier = can.Notifier(bus, [can.Printer()])
```


Downloads — pi@raspberrypi: ~ — ssh -lpi 192.168.1.194 — 98×21

```
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import can
>>> bus = can.interface.Bus(channel='can0', bustype='socketcan_native')
>>> notifier = can.Notifier(bus, [can.Printer()])
>>> 1449314948.341551       012d    000    8    93 84 79 8d 98 fd 7f 98
1449314952.589754       012d    000    8    93 84 79 8d 98 fd 7f 98
1449314952.836759       012d    000    8    93 84 79 8d 98 fd 7f 98
1449314953.028767       012d    000    8    93 84 79 8d 98 fd 7f 98
1449314953.264784       012d    000    8    93 84 79 8d 98 fd 7f 98
1449314953.489789       012d    000    8    93 84 79 8d 98 fd 7f 98
1449314953.928801       012d    000    8    93 84 79 8d 98 fd 7f 98
1449314954.344829       012d    000    8    93 84 79 8d 98 fd 7f 98
1449314954.591841       012d    000    8    93 84 79 8d 98 fd 7f 98
1449314954.839852       012d    000    8    93 84 79 8d 98 fd 7f 98
1449314955.087867       012d    000    8    93 84 79 8d 98 fd 7f 98
1449314955.368877       012d    000    8    93 84 79 8d 98 fd 7f 98
1449314955.626894       012d    000    8    93 84 79 8d 98 fd 7f 98
1449314956.191917       012d    000    8    93 84 79 8d 98 fd 7f 98
```

1.11. Application in C

Bring the CAN interface up if it is not already done:

```
sudo /sbin/ip link set can0 up type can bitrate 500000
```

Download the source code and example files by typing the following in the command prompt:

```
wget http://skpang.co.uk/dl/cantest.tar
```

Unpack the tar file and change into directory by:

```
tar xf cantest.tar
cd linux-can-utils
```

The example file is called cantest.c to edit this file, type the following in the command prompt:

```
nano cantest.c
```

Line 77 is the CAN message to be sent out.

```
unsigned char buff[] = "7DF#0201050000000000";
```

7DF is the message ID and 0201050000000000 is the data. Change the data to suit.
Press CTRL-X to exit.
To compile the program type:

```
make
```

Check there are no errors. To run the program type:

```
./cantest
```