

PiCAN GPS USER GUIDE

V1.0

Product name	PiCAN GPS CAN-Bus Board for Raspberry Pi
Model number	RSP-PICANGPS
Manufacturer	SK Pang Electronics Ltd

Contents

Table of Contents

1. Introduction	3
1.1. Features	3
2. Hardware Installation	3
1.2. Screw Terminal.....	4
1.3. 120Ω Terminator	4
1.4. LED1	4
1.5. LED2	4
1.6. Not Fitted Items.....	4
1.7. External GPS Antenna and Backup Battery	4
3. Software Installation	5
1.8. Basic GPS Test	6
1.9. Using GPSD	7
1.10. External GPS Antenna	8
1.11. Bring Up the CAN Interface.....	9
4. Writing Your Own Software.....	10
1.12. Application in Python.....	10
1.13. Application in C.....	11

1. Introduction

This PiCAN GPS board provides CAN-Bus capability for the Raspberry Pi 2 and 3. It uses the Microchip MCP2515 CAN controller. CAN connections are made via 4 way screw terminal plug. GPS is provided by a 66 channels MTK3339 chipset module. An onboard battery holder for a CR1220 cell. The backup power is for the real time clock and help to reduce fix time. The GPS module has built in patch antenna but an external active antenna can also be use via the uFL connector.

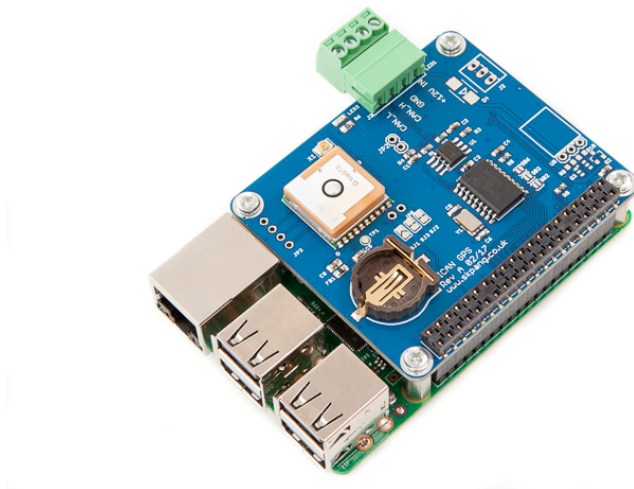
Easy to install SocketCAN driver. Programming can be done in C or Python.

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 screw terminal
- 120 Ω terminator ready
- Serial LCD ready
- LED indicator
- Four fixing holes, comply with Pi Hat standard
- SocketCAN driver, appears as can0 to application
- Interrupt RX on GPIO25
- MTK3339 chipset
- -165 dBm sensitivity, 10 Hz updates, 66 channels
- RTC battery holder
- Fix status LED
- On board patch antenna
- uFL connector for external active antenna

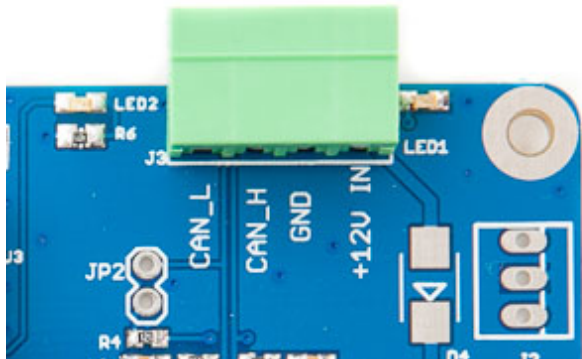
2. Hardware Installation

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. Screw Terminal

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



CAN0 (J3)	
Pin number	Function
1	n/c
2	GND
3	CAN_H
4	CAN_L

1.3. 120Ω Terminator

There is a 120Ω fitted to the board. To use the terminator solder a 2way header pin to JP2 then insert a jumper.

1.4. LED1

There is a red LEDs fitted to the board. This is connected to GPIO04. It can use as a general purpose indicator.

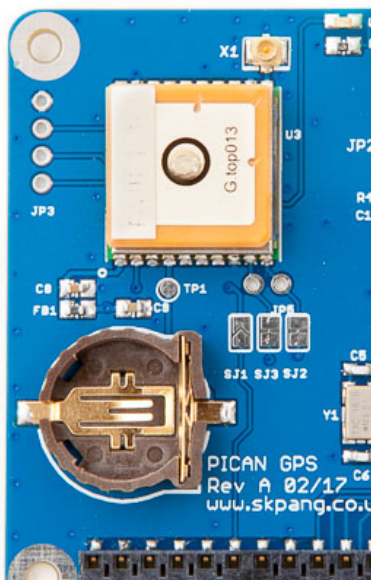
1.5. LED2

The LED blinks at about 1Hz while it's searching for satellites and blinks once every 15 seconds when a fix is found.

1.6. Not Fitted Items

J2 can be use to power a serial LCD with data on TXD line from the Pi. There is also 5v supply on J2.

1.7. External GPS Antenna and Backup Battery



An external GPS antenna can be fitted via connector X1. This is an uFL connector. A uFL to SMA cable adapter is normally required.

A backup battery can be fitted for the RTC and help to reduce fix time.

3. Software Installation

The following procedure is for the Raspberry Pi 3.

It is best to start with a brand new Raspbian Jessie 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
```

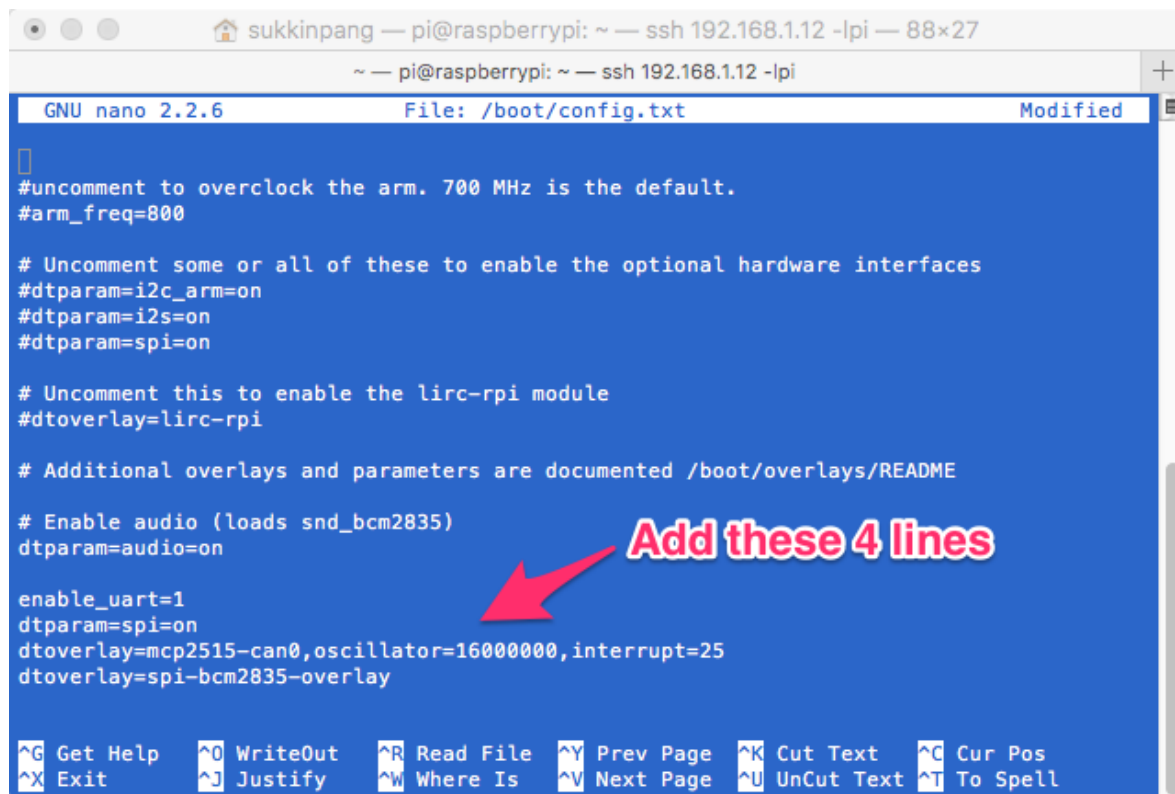
Add these 4 lines to the end of file:

```
dtparam=spi=on
```

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

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

```
enable_uart=1
```



```
sukkinpang — pi@raspberrypi: ~ — ssh 192.168.1.12 -lpi — 88x27
~ — pi@raspberrypi: ~ — ssh 192.168.1.12 -lpi
GNU nano 2.2.6 File: /boot/config.txt Modified
#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

enable_uart=1
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
```

Edit the cmdline.txt by:

```
sudo nano /boot/cmdline.txt
```

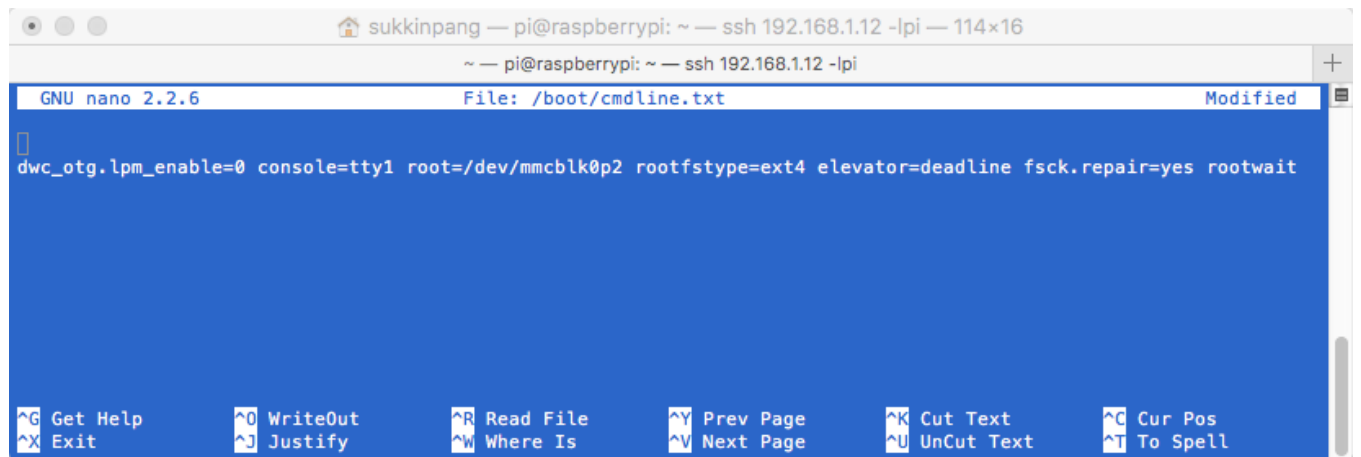
Change

```
dwc_otg.lpm_enable=0 console=serial0,115200 console=tty1  
root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline  
fsck.repair=yes rootwait
```

to

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2  
rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait
```

That is remove `console=serial0,115200`



```
sukkinpang — pi@raspberrypi: ~ — ssh 192.168.1.12 -lpi — 114x16  
~ — pi@raspberrypi: ~ — ssh 192.168.1.12 -lpi  
GNU nano 2.2.6 File: /boot/cmdline.txt Modified  
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait  
^G Get Help      ^O WriteOut      ^R Read File     ^V Prev Page     ^K Cut Text       ^C Cur Pos  
^X Exit          ^J Justify       ^W Where Is     ^N Next Page     ^U UnCut Text    ^T To Spell
```

Reboot Pi:

```
sudo reboot
```

1.8. Basic GPS Test

The raw data can be read by:

```
stty -F /dev/ttyS0 raw 9600 cs8 clocal -cstopb
```

Now read the data

```
cat /dev/ttyS0
```

If you see the data with lots of commas then it means the module has not got a fix yet. Make sure the GPS module has a clear view of the sky. Alternatively connect an external GPS antenna to the board and place the antenna outdoor.

You should see something like this:

```

~/Downloads — pi@raspberrypi: ~ — ssh 192.168.1.12 -lpi
$GPGSV,3,2,12,18,29,128,13,20,26,056,19,07,13,328,37,29,11,085,25*74
$GPGSV,3,3,12,10,11,156,15,08,10,272,33,31,01,191,,46,,,*4B
$GPRMC,215921.000,A,5145.2752,N,00004.5914,E,0.03,313.71,060317,,,A*6F
$GPZDA,215921.000,06,03,2017,,*5B
$GPGGA,215922.000,5145.2752,N,00004.5914,E,1,09,1.08,71.0,M,47.0,M,,*5C
$GPGSA,A,3,29,21,26,18,08,16,20,27,07,,,,,1.63,1.08,1.21*07
$GPRMC,215922.000,A,5145.2752,N,00004.5914,E,0.01,266.10,060317,,,A*6A
$GPZDA,215922.000,06,03,2017,,*58
$GPGGA,215923.000,5145.2751,N,00004.5914,E,1,09,1.08,71.0,M,47.0,M,,*5E
$GPGSA,A,3,29,21,26,18,08,16,20,27,07,,,,,1.63,1.08,1.21*07
$GPRMC,215923.000,A,5145.2751,N,00004.5914,E,0.01,256.38,060317,,,A*61
$GPZDA,215923.000,06,03,2017,,*59
$GPGGA,215924.000,5145.2751,N,00004.5914,E,1,09,1.08,71.0,M,47.0,M,,*59
$GPGSA,A,3,29,21,26,18,08,16,20,27,07,,,,,1.63,1.08,1.21*07
$GPRMC,215924.000,A,5145.2751,N,00004.5914,E,0.01,228.21,060317,,,A*67
$GPZDA,215924.000,06,03,2017,,*5E
$GPGGA,215925.000,5145.2751,N,00004.5914,E,1,09,1.08,71.0,M,47.0,M,,*58
$GPGSA,A,3,29,21,26,18,08,16,20,27,07,,,,,1.63,1.08,1.21*07
$GPRMC,215925.000,A,5145.2751,N,00004.5914,E,0.03,188.21,060317,,,A*6D
$GPZDA,215925.000,06,03,2017,,*5F
$GPGGA,215926.000,5145.2751,N,00004.5914,E,1,09,1.08,71.0,M,47.0,M,,*5B
$GPGSA,A,3,29,21,26,18,08,16,20,27,07,,,,,1.63,1.08,1.21*07
$GPGSV,3,1,12,16,76,265,25,21,65,080,25,26,63,169,28,27,46,276,21*75
$GPGSV,3,2,12,18,29,128,14,20,26,056,18,07,13,328,37,29,11,085,25*72
$GPGSV,3,3,12,10,11,156,14,08,10,272,34,31,01,191,,47,,,*4C
$GPRMC,21

```

1.9. Using GPSD

A program called GPSD can be use to view the data in a formatted way.

To install GPSD, type:

```
sudo apt-get install gpsd gpsd-clients python-gps
```

You need first to disable systemd service that GPSD installs:

```
sudo systemctl stop gpsd.socket
```

```
sudo systemctl disable gpsd.socket
```

To start GPSD, type:

```
sudo gpsd /dev/ttyS0 -F /var/run/gpsd.sock
```

```
gpsmon
```

You should see this screen:

```

sukkinpang — pi@raspberrypi: ~ — ssh 192.168.1.12 -lpi — 89x36
~ — pi@raspberrypi: ~ — ssh 192.168.1.12 -lpi
tcp://localhost:2947      NMEA0183>
Time: 2017-03-07T20:35:07.000Z Lat: 51 45' 16.727" N Lon: 0 04' 35.202" E
Cooked PVT

GPGGA GPGSA GPRMC GPVTG PMTK705 PMTK001 GPZDA GPGSV
Sentences

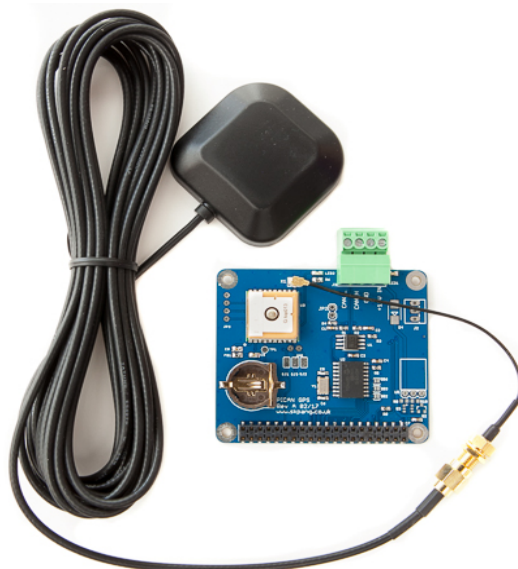
Ch PRN Az El S/N Time: 203507.000 Time: 203507.000
0 26 286 73 28 Latitude: 5145.2788 N Latitude: 5145.2788
1 21 158 57 23 Longitude: 00004.5867 E Longitude: 00004.5867
2 29 68 43 31 Speed: 0.52 Altitude: 70.6
3 16 298 43 37 Course: 120.35 Quality: 1 Sats: 08
4 31 202 35 28 Status: A FAA: A HDOP: 1.00
5 20 93 22 21 MagVar: RMC Geoid: 47.0
6 25 123 15 18 GGA
7 5 42 14 22
8 27 255 14 40
9 23 298 7 42
10 9 332 7 20
11 123 0 0 0 Mode: A 3 UTC: RMS:
Sats: 26 31 23 5 16 29 21 2 MAJ: MIN:
DOP: H=1.00 V=0.82 P=1.30 ORI: LAT:
PPS offset: LON: ALT:
GSA + PPS GST
(35) $GPZDA,203504.000,07,03,2017,,*56\x0d\x0a
(73) $GPGGA,203505.000,5145.2789,N,00004.5868,E,1,08,1.00,70.7,M,47.0,M,,*51\x0d\x0a
(58) $GPGSA,A,3,26,31,23,05,16,29,21,27,,,,,1.30,1.00,0.82*03\x0d\x0a
(72) $GPRMC,203505.000,A,5145.2789,N,00004.5868,E,0.52,103.44,070317,,A*6E\x0d\x0a
(35) $GPZDA,203505.000,07,03,2017,,*57\x0d\x0a
(73) $GPGGA,203506.000,5145.2788,N,00004.5867,E,1,09,0.87,70.6,M,47.0,M,,*52\x0d\x0a
(60) $GPGSA,A,3,26,31,23,05,16,29,25,21,27,,,,,1.16,0.87,0.77*04\x0d\x0a
(72) $GPRMC,203506.000,A,5145.2788,N,00004.5867,E,0.57,120.35,070317,,A*61\x0d\x0a
(35) $GPZDA,203506.000,07,03,2017,,*54\x0d\x0a
(73) $GPGGA,203507.000,5145.2788,N,00004.5867,E,1,08,1.00,70.6,M,47.0,M,,*5C\x0d\x0a
(58) $GPGSA,A,3,26,31,23,05,16,29,21,27,,,,,1.30,1.00,0.82*03\x0d\x0a
(72) $GPRMC,203507.000,A,5145.2788,N,00004.5867,E,0.52,120.35,070317,,A*65\x0d\x0a
(35) $GPZDA,203507.000,07,03,2017,,*55\x0d\x0a

```

1.10. External GPS Antenna

The PiCAN-GPS board has built in patch antenna which provides a -165dBm sensitivity. However if your project is in a box and does not have a clear view of the sky then an external antenna is more suitable.

The optional external antenna has a SMA connector and an uFL to SMA cable adapter is required.



1.11. Bring Up the CAN Interface

You can now bring the CAN interfaces 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 .

To send a CAN message on can0 (CAN B J4) use :

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

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

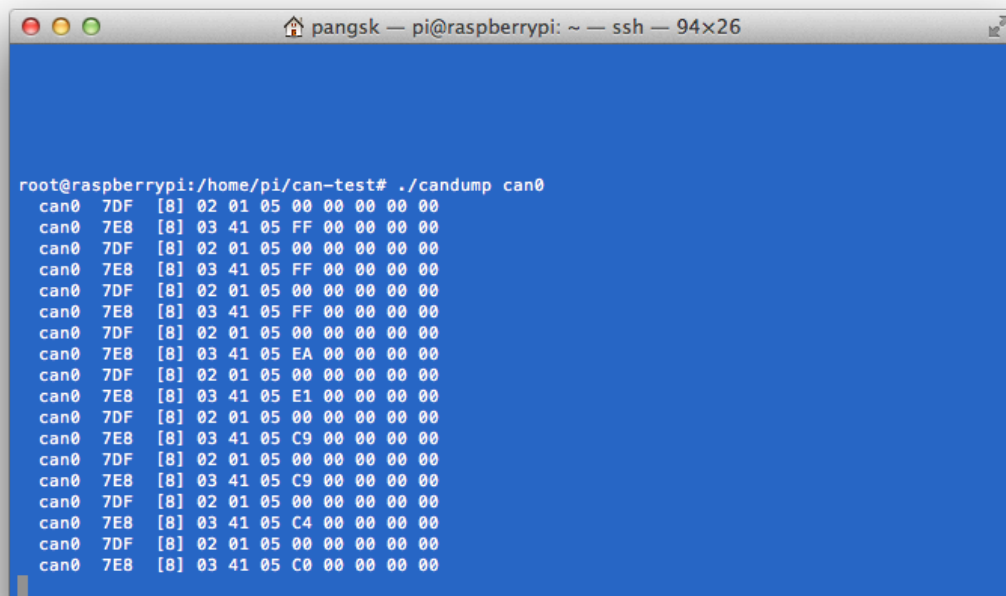
To send a CAN message on can1 (CAN A J3) use :

```
./cansend can1 7DF#0201050000000000
```

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

```
./candump can0
```

You should see something like this:

A screenshot of a terminal window titled 'pangsk — pi@raspberrypi: ~ — ssh — 94x26'. The terminal shows the command 'root@raspberrypi:/home/pi/can-test# ./candump can0' and its output. The output consists of 20 lines of CAN bus traffic, alternating between CAN ID 7DF and 7E8. Each line shows the CAN ID, a bracketed number [8], and a hexadecimal data field. The data fields for 7DF are 02 01 05 00 00 00 00 00 and for 7E8 are 03 41 05 FF 00 00 00 00, except for one 7E8 line which has 03 41 05 EA 00 00 00 00. The terminal background is blue.

```
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.

1.12. 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:

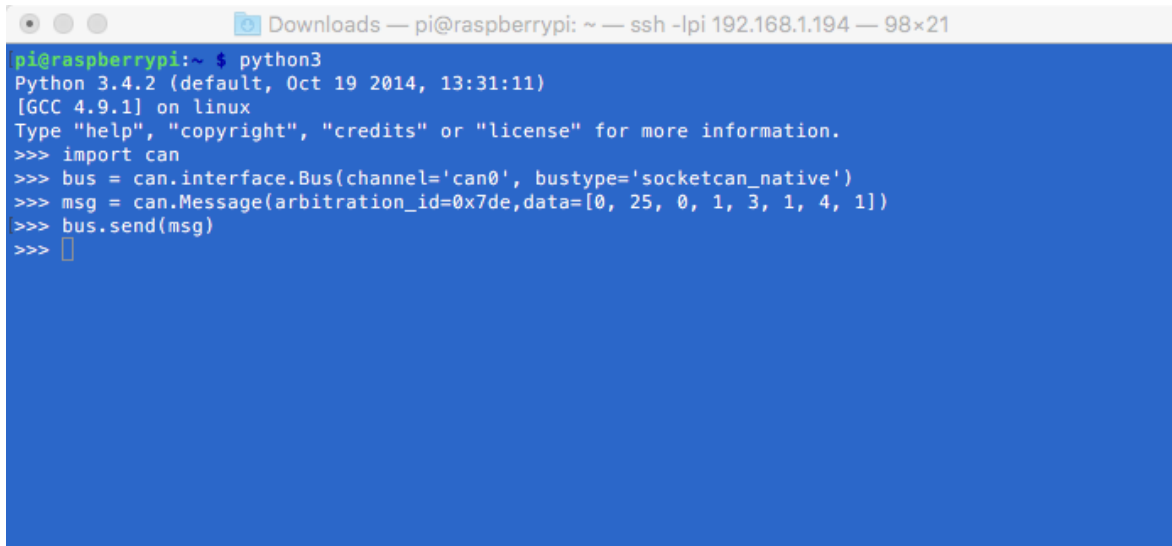
```
sudo /sbin/ip link set can0 up type can bitrate 500000
```

Now start python3

```
python3
```

To sent a message out type the following lines:

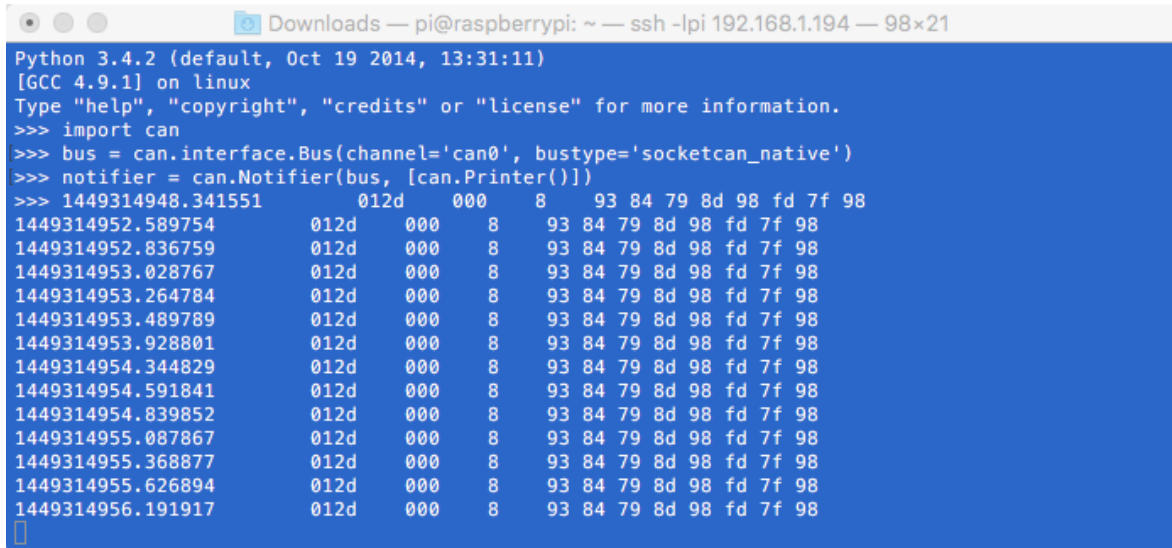
```
import can
bus = can.interface.Bus(channel='can0', bustype='socketcan_native')
msg = can.Message(arbitration_id=0x7de,
                  data=[0, 25, 0, 1, 3, 1, 4, 1],
                  extended_id=False)
bus.send(msg)
```

A screenshot of a terminal window titled "Downloads — pi@raspberrypi: ~ — ssh -lpi 192.168.1.194 — 98x21". The terminal shows the following commands and output:

```
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()])
```



```

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

```

1.13. 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
```