Raspberry Pi Internet Radio

Constructors Manual

A guide to building Internet radios using the Raspberry Pi

Bob Rathbone Computer Consultancy

www.bobrathbone.com

15th of November 2016

Version 5.6

Contents

Introduction 14

Examples 15

Vintage Radio Conversion 18

Building in a IR sensor and remote control 19

Hardware 20

Raspberry PI computer 20

Using the Raspberry PI model B+ an Model 2 and 3 20

Raspberry Pi Zero 22

The HD44780 LCD display 22

Raspberry PI 3 Model B 23

Raspberry Pi 3 - Model B Technical Specification 23

Raspberry Pi 3 - Model B Features 23

Radio variants 24

Housing the radio 25

Wiring 26

Version 2, 3 or model B+ boards 29

Version 1 boards (early boards) 29

Rotary encoder wiring 30

LCD Module Wiring 31

Power supply considerations 33

Preventing electrical interference 34

Using a clip on ferrite core on the +5 volt cable 34

Fit a mains filter 34

Try a decoupling capacitors 34

Preventing ground loops 35

GPIO Hardware Notes 36

Parts List 37

Construction HD44780 LCD 38

Building the LCD and pushbuttons interface board 38

Construction using an Adafruit LCD plate 40

Introduction 40

Using other switches 41
Using the Adafruit LCD plate with the model B+, 2B and 3B 41
Construction using an I2C LCD backpack 42
Adafruit I2C Backpack 42
Arduino PCF8574 I2C backpacks 43
Creating the interface board for the I2C back pack 43
Construction using the PiFace CAD 44
Installing an IR sensor and remote control 45
IR Sensor 45
Remote control 45
Remote Control Activity LED 46
Construction using a HiFiBerry DAC 47
HiFiBerry DAC using the P5 connector 47
The HifBerry DAC Plus using the 40 Pin Connector 48
Construction using IQAudio products 49
Conventions used in this tutorial 51
System Software installation 52
SD card creation 52
Installation or upgrade on Debian Wheezy 52
Log into the system 52
Install sysvinit-core 52
Online update and upgrade of the Operating System 52
Disable booting to the desktop environment 52
Setting the time zone 53
Changing the system hostname and password 55
Installing the radio Software 57
Music Player Daemon Installation 57
Install the Radio Daemon 58
Options 1, 2, 3, 4 or 9 59
Options 5, 6 or 7 60
Option 8 PiFace CAD 61
Configure the audio output 62
Reboot to enable the software 64
Setting the mixer volume 65

Configuring other sound devices 66
Configuring a USB sound devices 66
Configuring a HiFiBerry products 68
Configuring IQAudio sound cards 70
Testing the Music Player Daemon MPD 71
Manually configuring sound cards 72
Installing the Infra Red sensor software 73
Testing the remote control 76
Disabling the repeat on the volume control 77
Configuring GPIO outputs 77
Switches and rotary encoders GPIO assignments 78
LCD display GPIO assignments 78
Configuring the remote control activity LED 78
Testing the remote control activity LED 79
Changing the date format 79
Configuring the Adafruit LCD backlight colours 79
Configuring the playlist number 79
Configuring startup mode for Radio or Media player 80
Configuring the volume range 80
Operation 81
Starting the program 81
Buttons 83
Rotary encoder operation 84
Mute function 85
Playing MP3 85
Playing music from a USB stick 85
Playing music from the SD card 85
Playing music from a Network Attached Storage (NAS) 85
Organising the music files 85
MPD Logging 85
Radio program logging 86
Configuration and status files 86
Displaying an RSS feed 87
Using the Timer and Alarm functions 87

Setting the Timer (Snooze) 87
Setting the Alarm 87
Using the Alarm and Timer functions together 88
Music Player Clients 88
Using the MPC client 88
Adafruit RGB Plate changing colours 89
Shutting down the radio 89
Creating and Maintaining Playlist files 90
Creating playlists 90
The stationlist file 91
Radio stream resources on the Internet 92
Overview of media stream URLs 93
PLS file format 93
M3U Files 93
ASX file 94
Direct stream URLs 94
Listening to live Air Traffic Control (ATC) 95
Installing the Web interface 98
Install Apache 98
Test the Apache web browser 98
Install the Web Browser server pages 99
Start the radio web interface 99
Changing the Web Interface Radio photo 100
Mounting a network drive 101
Finding the IP address of the network drive 101
The CIFS mount command 101
Older NAS drives sec security option 102
The NFS mount command 102
Display the share directory 102
Un-mounting the /share directory 103
Copy the mount command to the configuration 103
Load the music library 103
Update the playlists for the new share 103
Disabling the share 103

Further information 104
Source files 105
The LCD Class 105
The Radio Daemon 105
The Adafruit Radio daemon 105
The LCD with I2C backpack 105
The Daemon class 106
The Radio class 106
The Rotary class 106
The alternative Rotary class 106
The Log class 106
The Configuration Class 106
The RSS class 106
The Translate class 106
LCD test programs 106
Switch test programs 106
The create_m3u program 107
The display_current program 107
The display_model script 107
The select_daemon.sh script 107
The select_audio.sh script 107
The remote control daemon 107
The UDP network communications class 107
The language class 107
The Status LED class 108
The Menu Switch class 108
Downloading the source from github 108
Contributors code 109
Miscellaneous 110
Simple tone regulator 110
Using the Adafruit backlit RGB LCD display 111
Using a 4 line by 16 character LCD 112
Troubleshooting 113
The Raspberry Pi will not boot 113

Trouble shooting problems with MPD 113
LCD screen not working 114
The LCD only displays hieroglyphics 114
The LCD displays hieroglyphics or goes blank occasionally 114
LCD backlight not working 114
LCD only displays dark blocks on the first line 114
MPD fails to install 115
Music Player Daemon won't start 115
The MPD program may display a socket error 115
The LCD displays the message "No playlists" 115
Constant alternate display of Station Name and Volume 115
The MPD daemon complains about the avahi daemon 116
Buttons seem to be pressing themselves 116
Radio daemon doesn't start or hangs 116
Stream decode problems 116
Cannot mount remote network drive 117
Button or Rotary encoder problems 117
Rotary encoders not working 117
Volume control not working with DAC or USB speakers 118
Noisy interference on the radio 118
Humming sound on the radio 118
Music is first heard at boot time then stops and restarts 118
USB device won't play 118
Unexpected message during an upgrade 119
Reboot hangs if radiod running 119
Missing logrotate or other configuration files 119
IR remote control problems 120
The irrecord program complains that lircd.conf already exists 120
The irrecord cannot open /dev/lirc0 120
HiFBerry DAC plus no sound 120
Using the diagnostic programs 121
The test_lcd, test_i2c_lcd and test_ada_lcd programs 122
The test_switches program 122
The test_rotary_class.py program 122

The remote_control program 122
The display_model program 122
The display_current program 123
Running the radio program in nodaemon mode 123
Creating a log file in DEBUG mode 124
Displaying information about the Raspberry Pi 124
Displaying information about the Operating system 124
Display the kernel details 125
Displaying the GPIO information 125
Configuring a wireless adaptor 126
Install the wireless adapter 126
Configure the adaptor 126
Explanation of the network fields 127
Operating the wireless interface 127
Troubleshooting the wireless adapter 128
Configuring a static IP address 128
Ethernet static IP configuration 129
Wireless LAN static IP configuration 130
Streaming to other devices using Icecast2 131
Inbuilt MPD HTTP streamer 131
Introduction to Icecast 131
Installing Icecast 131
Overclocking the Raspberry PI 132
Icecast2 Operation 132
Switching on streaming 133
Playing the Icecast stream on a Windows 7 134
Playing the Icecast stream on a Windows 10 135
Playing the Icecast2 stream on an Apple IPad 136
Playing the Icecast2 stream on an Android device 136
Visual streaming indicator 137
Administration mode 137
Troubleshooting Icecast2 140
Problem - Icecast streaming page says it can’t be displayed 140
Problem – No Mount Point displayed 140

Problem - Cannot play the stream on my Android device 140
Problem – Music keeps stopping or is intermittent 140
Configuring the speech facility 141
The /var/lib/radiod/voice file 141
Testing espeak 141
The language file 143
Suppressing an individual message 143
Creating a new language file 144
Speech Operation 144
Controlling the Music Player daemon from Mobile devices 145
Android devices 145
Apple devices 145
Frequently asked questions (FAQs) 146
What is the login name and password? 146
Why are the radio stations not in the order that they were defined? 146
Why are some station names not being displayed in the web interface? 146
Why doesn't the web interface display URLs until a station is selected? 147
Why are music tracks played randomly when loaded? 147
Why not display volume as blocks instead of Volume nn? 147
Why do I see the Station number displayed in brackets? 147
Why do I see a station number on LCD line 3? 147
Is it possible to change the date format? 148
Is there a pause & resume function? 148
Why do I see a different station name from the one in the playlist? 148
What Rotary Encoder can I use for this project? 148
Can this code or documentation be re-used in other projects? 148
Licences 149
Intellectual Property, Copyright, and Streaming Media 149
Disclaimer 150
Technical support 150
Acknowledgements 151
Glossary 152
Appendix A - System Files used by the Radio Program 154
A.1 Files added to the system 154

etc/radiod.conf 154
etc/logrotate.d/radiod 156
etc/init.d/radiod 156
etc/init.d/asound.conf 156
etc/init.d/pifacercd 157
etc/lirc/lircrc 157
A.2 System files modified by the installation 158
etc/inittab 158
boot/cmdline.txt 158
etc/modules 158
boot/config.txt 159
Appendix B – Wiring diagrams 160
B.1 Raspberry Pi Rotary Encoder version with backlight dimmer 160
Index 161

Figures

Figure 1 The completed classic style radio 15
Figure 3 Radio using the Adafruit LCD plate 15
Figure 4 Lego internet radio 15
Figure 5 Pi radio using rotary encoders 16
Figure 6 Old Zenith radio using rotary encoders 16
Figure 7 Zenith radio rear view 16
Figure 8 Zenith radio top view 16
Figure 9 PiFace CAD Radio with IR Remote Control 17
Figure 10 The Radio running on a Pi Zero 17
Figure 11 Boom Box radio front view 17
Figure 12 Boom Box Radio rear view 17
Figure 13 Philips BX490A (1949) Vintage Internet Radio 18
Figure 14 IR Sensor and Remote control 19
Figure 15 Adafruit and IR sensor and activity LED 19
Figure 16 Raspberry PI Model B Computer 20
Figure 17 Raspberry PI Model B+ 21
Figure 18 Raspberry PI B+ AV cable 21
Figure 19 Raspberry Pi Zero 22
Figure 20 USB Ethernet adapter 22
Figure 21 The HD44780 LCD display 22
Figure 22 OLED 4 x20 LCD display 22
Figure 23 Raspberry PI 3 Model B 23
Figure 24 Some examples of radio cases 25

Figure 25 Switch wiring version 2 boards 29
Figure 26 Switch Wiring version 1 boards 29
Figure 27 Rotary Encoder Diagram 30
Figure 28 Rotary encoder with push switch 30
Figure 29 Rotary encoder pin-outs 30
Figure 30 HD44780 LCD electrical circuit 32
Figure 31 Wire LCD pin 1 (GND) and 5 (RW) together 32
Figure 32 Clip on ferrite core 34
Figure 33 Loop +5V supply around the core 34
Figure 34 Various mains filters 34
Figure 35 Integrated mains socket and filter 34
Figure 36 0.1 uF decoupling capacitor 34
Figure 37 3.5mm Jack Ground Loop Isolator 35
Figure 38 GPIO Numbers 36
Figure 39 26 pin header extender 36
Figure 40 Radio parts 38
Figure 41 Interface board (Wiring side) 38
Figure 42 Interface board (Component side) 39
Figure 43 Radio rear inside view 39
Figure 44 Adafruit LCD plate 40
Figure 45 Adafruit LCD plate with ribbon cable adapter 40
Figure 46 Adafruit I2C Backpack 42
Figure 47 LCD connected to an Adafruit I2C backpack 42
Figure 48 Arduino I2C backpack 43
Figure 49 Ciseco Humble PI I2C interface board 43
Figure 50 The I2C backpack interface board 43
Figure 51 PiFace CAD and Rasberry PI 44
Figure 52 PiFace CAD in a case 44
Figure 53 TSOP38238 IR sensor 45
Figure 54 Soldering precautions 45
Figure 55 LED polarity 46
Figure 56 Adafruit plate with IR sensor and activity LED. 46
Figure 57 Raspberry PI P5 connector 47
Figure 58 HiFiBerry DAC plus Connector 47
Figure 59 Pi Radio with HiFiBerry DAC 47
Figure 60 HiFiBerry DAC Plus 48
Figure 61 HiFiBerry mounted on the Rasberry Pi 48
Figure 62 IQAudio DAC plus 49
Figure 63 Disabling the graphical desktop 53
Figure 64 Setting the time zone 53
Figure 65 Selecting the time zone 54
Figure 66 Saving the timezone 54
Figure 67 Changing the raspberry PI password 55
Figure 68 raspi-config advanced options 56
Figure 69 Changing the hostname 56

Figure 70 Radio type selection 59
Figure 71 The I2C bus display using the i2cdetect program 61
Figure 72 The audio output configuration program 63
Figure 73 Confirm audio selection screen 63
Figure 74 Reboot screen 64
Figure 75 Basic Alsa sound mixer 65
Figure 75 USB DAC selection 67
Figure 77 The USB PnP Alsa Mixer 67
Figure 78 Selecting HiFiBerry products 68
Figure 79 Set mixer analogue volume 69
Figure 80 Set mixer digital volume 69
Figure 81 IQAudio sound mixer 70
Figure 82 Live ATC web page 95
Figure 83 WinAmp playing ATC live feed 96
Figure 84 WinAmp station information 96
Figure 85 Radio web interface 99
Figure 86 Snoopy web interface 100
Figure 87 Simple tone control circuit 110
Figure 88 Dual 100K Linear potentiometer 110
Figure 89 Tone control board 110
Figure 90 IN4148 diode 111
Figure 91 Over-clocking the Raspberry PI 132
Figure 92 Icecast2 Status 134
Figure 93 Streaming in the Firefox Web Browser 135
Figure 94 Selecting Windows Media Player 135
Figure 95 Windows media player 136
Figure 96 Icecast admin login 137
Figure 97 Icecast Global Server Status 138
Figure 98 Icecast2 Mount point information 139
Figure 99 MPDroid set-up screen 145
Figure 100 MPDroid play screen 145
Figure 101 MPDroid play queue 145
Figure 102 Wiring Raspberry Pi Radio Rotary Encoder version 160

Tables

Table 1 Radio variants 24
Table 2 Radio wiring conflicts 26
Table 3 Controls and LCD wiring 26 pin version 27
Table 4 Radio and DAC devices 40 pin wiring 28
Table 5 LCD module wiring for 26 pin Raspberry Pi's 31
Table 6 LCD module wiring 40 pin Raspberry PI's 31
Table 7 Parts list 37
Table 8 Remote Control Activity LED 46
Table 9 IR Sensor Pin outs 74
Table 10 Remote Control Key names and functions 76
Table 11 Push Button Operation 83
Table 12 Rotary Encoder Knob Operation 84
Table 13 Playlist files and directories 90
Table 14 Adafruit backlit RGB display wiring 111

Introduction

This manual describes how to create an Internet Radio using the Raspberry PI educational computer. The source and basic construction details are available from the following web site:
http://www.bobrathbone.com/raspberrypi_radio.htm

This manual provides a detailed overview of construction and software. It contains instructions for building the radio using either the HDD44780 LCD directly wired to the Raspberry PI GPIO pins or alternatively using an either an Adafruit RGB-backlit LCD plate or the PiFace Control and Display (CAD) . An I2C backpack is also now supported. It can be constructed using either push buttons or rotary encoders. In version 5.3 onwards it also contains the software for to run with a converted vintage radio.

The features of the Raspberry PI internet radio are:

- Raspberry PI running standard Music Player Daemon (MPD)
- Four different LCDs are supported
  - 2 x 16 character LCD with HD44780 controller
  - 4 x 20 character LCD with HD44780 controller
  - Adafruit LCD plate with 5 push buttons (I2C interface)
  - PiFace Control and Display (CAD) with IR sensor and buttons
- The LCD can be directly interfaced via GPIO pins or using the an I2C or SPI interface
- Works with newer revisions of the Raspberry PI including the model B+, 2B, 3B and Pi Zero
- Can use an IR sensor and remote control
- Clock display or IP address display (for web interface)
- Five push button operation (Menu, Volume Up, Down, Channel Up, Down)
- As alternative to the above rotary encoder switches may be used
- Support for Digital sound cards such as **HiFiBerry** and **IQAudio**
- Vintage radio conversion to internet radio supported
- Timer (Snooze) and Alarm functions
- Artist and track scrolling search function
- Plays music from a USB stick, SD card or from a Network drive (NAS)
- Menu option to display a single RSS news feed
- Web interface using snoopy and others
- Control the radio from either an Android device or iPhone and iPad
- Plays Radio streams or MP3 and WMA tracks
- Output either using the analogue audio jack, USB DAC or HiFiBerry/IQAudio DAC.
- Play output on PC or on a mobile device using ICECAST streaming
- Playlist creation program using a list of URLs (M3U file)
- Fully integrated with mobile apps such as Android **MPDroid** or Apple **mPod**
- Speech for visually impaired and blind persons using **espeak**
- Support for European character sets (Limited by LCD capabilities)
- Easily installed using Debian packages
- This software runs on **Jessie** or **Jessie Lite**.

This design caters for both the complete novice and more advanced constructors. Do not be put off by the size of this manual as it shows a lot of different designs. Simply read through it and decide which one is the best for you. Some examples are shown in the following pages.

Examples

Various examples of the Raspberry PI internet radio were built using this design.

Figure 1 The completed classic style radio

This classic style Internet Radio is built into a wooden case. This is using two four inch speakers and audio amplifier stripped out from an old pair of PC speakers. It has five buttons in all. The centre square button is the menu selection.

Figure 2 Radio using the Adafruit LCD plate

Example of the PI internet radio using an Adafruit RGB-backlit LCD plate for Raspberry PI from AdaFruit industries. It has five push buttons and is one of the easiest options to construct. If you want to build this into a case then don't use the buttons supplied with the kit but use external buttons.

Figure 3 Lego internet radio

Example of a fun radio built using this design and Lego from Alan Broad (United Kingdom). This really puts the fun back into computing.

Figure 4 Pi radio using rotary encoders

The rotary encoder switch version of the radio consists of a Raspberry PI connected to an Adafruit 20 character x 4 line RGB LCD display housed. It is all housed in a LogiLink PC speaker set with two rotary encoders. The rotary encoders also have push buttons (Push the knob in). The left one is the *Mute* switch and the right one is the *Menu* switch. The blue glow in the sub-woofer opening comes from a bright blue LED.

Figure 5 Old Zenith radio using rotary encoders

Example of the PI radio from James Rydell built into an old Zenith valve radio case. The pictures below show the inside and top view respectively. The two original controls have been replaced by two rotary encoders. The old valve radio inside has been completely removed and replaced with the Raspberry PI and radio components. The LCD display has been built into the top so as not to spoil the original face of the radio. This is a fine example of what can be done with this project.

Figure 6 Zenith radio rear view

Figure 7 Zenith radio top view

More on the next page.

Figure 8 PiFace CAD Radio with IR Remote Control

The radio supports the PiFace Control and Display (CAD) board. This is a good choice for complete beginners. See: http://www.piface.org.uk/products/piface_control_and_display/

This has 5 push buutons, a reset button (not currently used) and an Infra Red (IR) sensor. This has the advantage that it has inbuilt support for a remote control. It has one drawback in that the push buttons are on the bottom of the unit. The PiFace CAD uses the Serial Peripheral Bus interface (SPI) on the Raspberry Pi.

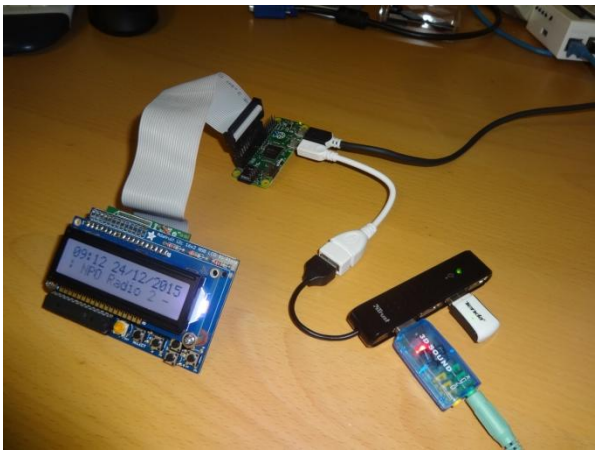


Figure 9 The Radio running on a Pi Zero

This is an example of the radio running on a Raspberry Pi Zero. In this example it uses a micro to standard USB adaptor to connect a simple USB hub. A USB sound dongle and Tenda wireless adapter are plugged into the USB hub. A USB to Ethernet adapter can also be used in place of the wireless adapter. The display used is the Adafruit LCD plate. Also note that the Pi Zero comes with an unpopulated 40 pin GPIO interface. You need to either directly solder wires to the GPIO interface (Not advised) or solder either a 26 or 40 pin male header (Advised).

This beautiful radio is a fine example of the latest version of the design. It is using a Raspberry PI model 2B and rotary encoders with inbuilt push button. The display is a 4 x 20 LCD. The sound system is a Velleman 30 Watt amplifier (bottom right) and two 5 ¼ inch 50 watt speakers. It has an IR sensor (Left speaker on the right side) and an activity LED (between the two knobs).

Figure 10 Boom Box radio front view

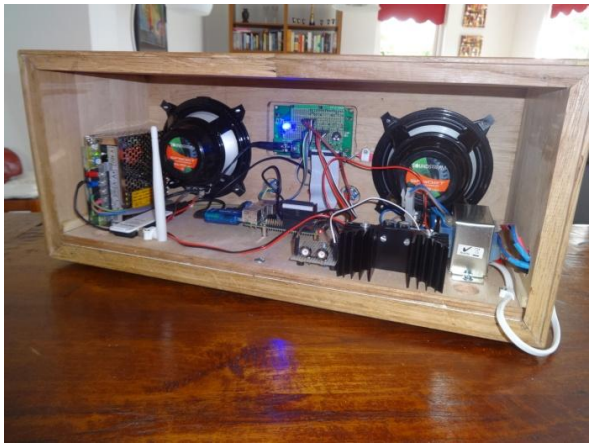


Figure 11 Boom Box Radio rear view

There are many more examples of what people have done to house their PI radio but unfortunately they can't all be shown in this manual. For alternative ideas see the constructor's page at: http://www.bobrathbone.com/pi_radio_constructors.htm

Vintage Radio Conversion

From version 5.3 onwards software for use with a vintage radio is included (retro_radio.py).

Figure 12 Philips BX490A (1949) Vintage Internet Radio

The radio is a Philips BX490A manufactured in the Netherlands in 1949. The purpose of this design is retain as much of the original look and feel of a vintage radio which has been converted to run as an Internet radio. It does not have any LCD display. In the above example the following controls are used.

- Far left switch - Simple tone control
- Middle left switch - Volume and mute switch
- Middle right switch – Radio channel (Tuner) or media track selection
- Far right switch – Menu switch (8 positions)
- Push button on right side (Not shown) - Standard menu switch

At the top left the so-called magic eye tuning indicator has been replaced with an Red Green Blue status LED. In the above picture the LED is glowing green (Normal operation). This window also contains the IR sensor and activity LED for a remote control. If the radio is busy (loading stations for example) it glows blue. For an error or shutdown the LED glows RED. The IR remote control also flashes red to indicate IR remote control activity.

The software allows espeak to be configured to 'speak' station and search information etc.

The details on how to construct a similar project is containd in the following document:

Raspberry Pi Vintage Radio
http://www.bobrathbone.com/raspberrypi/Raspberry%20PI%20Vintage%20Radio.pdf

Building in a IR sensor and remote control

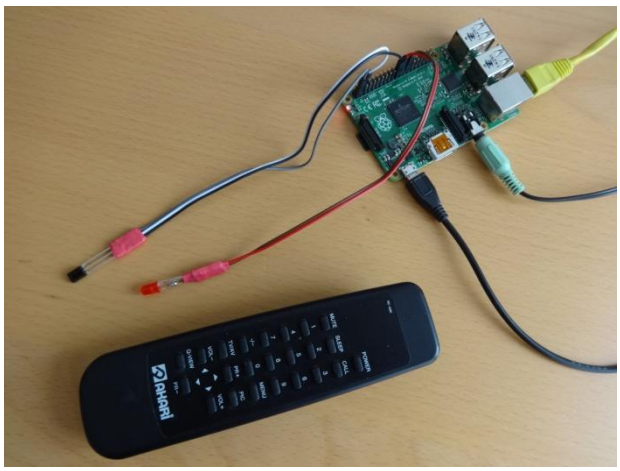


Figure 13 IR Sensor and Remote control

The radio can be built with an IR Sensor and remote control. Also included is an activity LED which flashes when the remote control is used
A TSOP382xx series IR Sensor is used in conjunction with almost any remote control. An activity LED can also be added which flashes every time remote control signal is detected. This facility uses software from PiFace. See http://www.piface.org.uk

The remote control provides the same functionality as the buttons or rotary encoders.

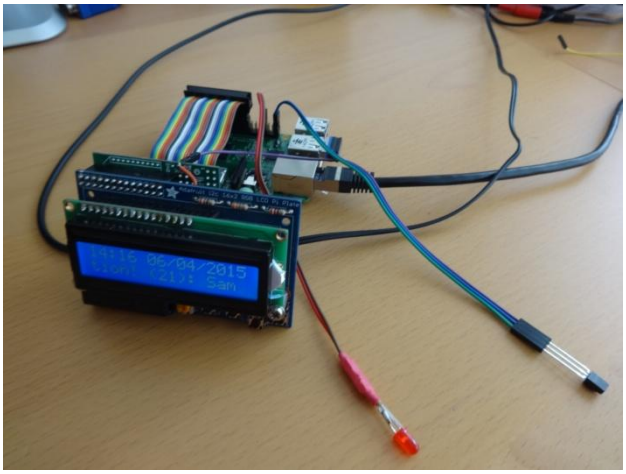


Figure 14 Adafruit and IR sensor and activity LED

The AdaFruit RGB plate can also be fitted with an IR sensor and activity LED but needs a model B+, 2B or 3B (40 GPIO pins) and 26 pin extender as shown in Figure 38 on page 36.

Note that a 40 pin Raspberry PI is needed as the Adafruit Plate occupies all 26 pins on the 26 pin versions of the Raspberry PI.

Hardware

The principal hardware required to build the radio consists of the following components:

- Current versions of the Raspberry PI computer (Version 1 boards no longer supported)
- An HD44780 LCD display or an Adafruit RGB-backlit LCD plate for the Raspberry PI
- LCD and switches interface board

Raspberry PI computer

The **Raspberry Pi** is a credit-card-sized single-board computer developed in the United Kingdom by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools.

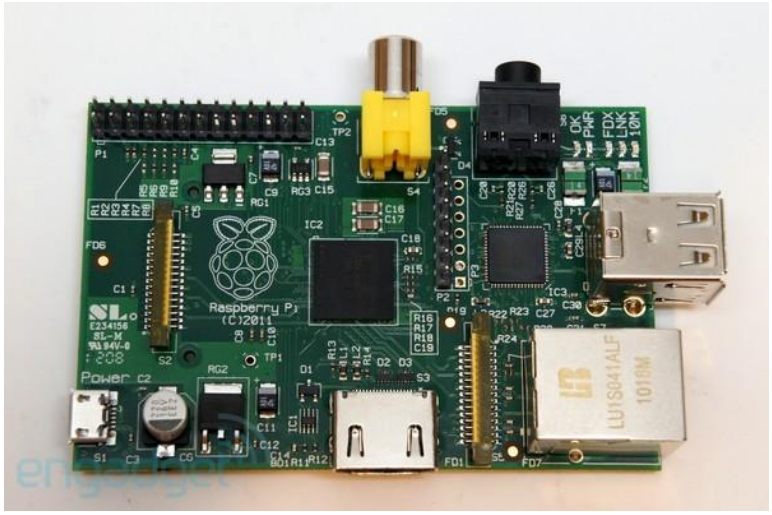


Figure 15 Raspberry PI Model B Computer

More information on the Raspberry PI computer may be found here:
http://en.wikipedia.org/wiki/Raspberry_Pi

If you are new to the Raspberry PI try the following beginners guide. http://elinux.org/RPi_Beginners

Using the Raspberry PI model B+ an Model 2 and 3

In July 2014 the Raspberry PI foundation introduced the model B+ board. This has a different physical layout to version 1 and 2 boards and uses a micro SD card for the operating system.

The main features are:

- The B+ CPU and memory (512 MB) unchanged.
- Model 2 Pi has a 4 core CPU (BCM2836) and 1GB memory
- Four USB ports instead of 2
- USB Power limiting – switchable from 600mA to 1.2A.
- 40 Pin GPIO header with 26 usable pins instead of 17 (21 on the Rev 2)
- First 26 GPIO pins are compatible with model B rev 2 boards
- There is no longer any P5 or P6 connectors that were present on the model B.
- Composite video routed via the 3.5mm jack used for audio (Audio jack plug compatible).

- Micro SD card for the operating system.
- Round corners on the PCB and four equally spaced mounting holes.
- Improved power supply – 2 amp polyfuse on the input and SMPS 3.3 and 1.8v generators to replace the linear ones on the existing Pi.
- Improved audio output saving up to 1 watt according to the PI foundation

Figure 16 Raspberry PI Model B+

The radio software works fine with the model B+ and model 2 and 3 boards. However these are some of the differences that you are likely to encounter:

- The new board will not normally fit into existing Raspberry PI cases
- Existing 26 pin interface and prototype cards may not fit without a 26 pin header
- You can not fit a 26 pin ribbon cable into the GPIO header without a 26 pin header (See Figure 38 26 pin header extender on page 36)
- If using an Adafruit LCD plate; this will require insulating tape on top the USB ports to prevent them shorting out components on the Adafruit interface board.

The new AV (Audio/Video) port combines the audio and video signals in a single jack. Instead of using a standard composite cable, this new connector requires a 4 pole 3.55mm AV cable. To complicate matters: not all of these cables are the same! However existing audio jack plugs are compatible with the new AV connector.

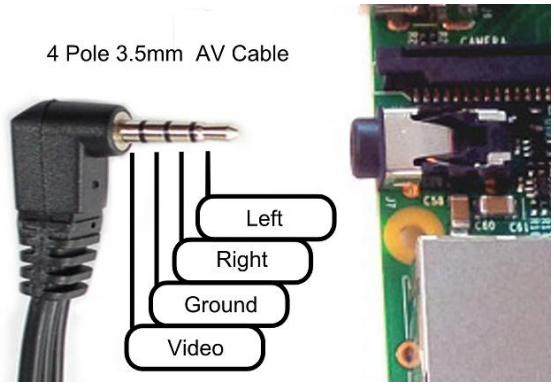


Figure 17 Raspberry PI B+ AV cable

When choosing a cable, seek an **iPod 4 pole AV** cable. This will however result in the left and right audio channels being reversed but otherwise provides the proper connections. Using other cables, such as a camcorder cable will be hit or miss. Typically camcorder cables have the wrong pin connections for Video and Ground.

This change also can cause some issues with shared grounding with audio speakers. If separate audio and composite AV connector is required, these can be split apart using the same jack inputs as for the model A and B.

Raspberry Pi Zero

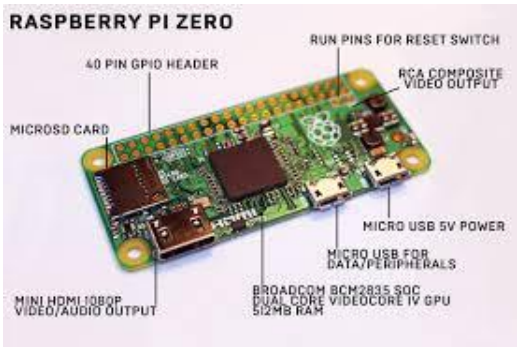


Figure 18 Raspberry Pi Zero

The Raspberry Pi Zero has had limited testing but seems to work fine. Network connection however is only possible with either a USB to Ethernet adapter or a Wi-Fi Dongle. Note that the USB is a Micro USB and will need an micro USB to standard USB adapter

Figure 19 USB Ethernet adapter

Note: The Pi Zero will not boot from standard Debian Wheezy distribution. Use Debian Jessie or Jessie Lite.

The HD44780 LCD display

Figure 20 The HD44780 LCD display

The HDD44780 LCD interface is an industry standard interface for a variety of LCD displays. These can come in various sizes but the two lines by 2x16 or 4x20 character displays are the most popular. The software for this Internet radio supports either display. Most of these modules compatible with the Hitachi HD44780 LCD controller so there is a wide choice of these displays.

Figure 21 OLED 4 x20 LCD display

The latest displays use OLED displays (Organic Light Emitting Diode) and give very good results.
See https://en.wikipedia.org/wiki/OLED

For pin-out details see LCD pin outs on page 30.

Raspberry PI 3 Model B

The Raspberry Pi 3 Model B was launched in February 2016. The main difference with previous models is that it has onboard WiFi and Bluetooth adaptors so it is no longer necessary to purchase these separately.

Note: To use the new onboard features it is necessary to install a Jessie (or Jessie Lite) image from 26 February 2016 or later.

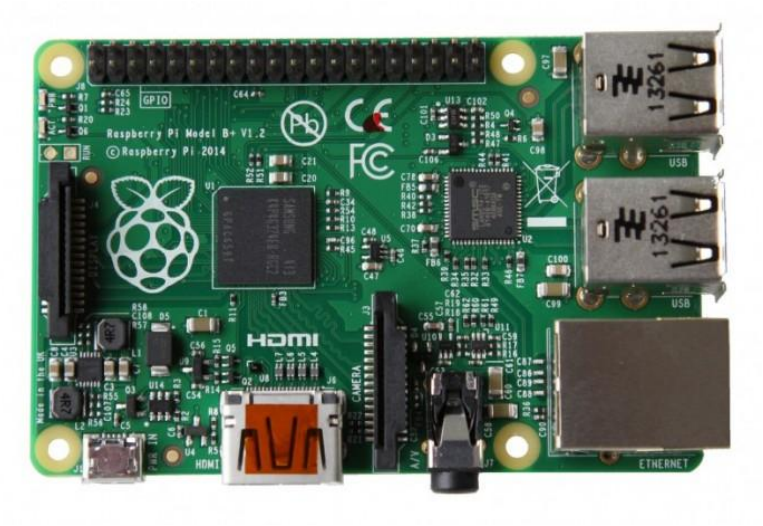


Figure 22 Raspberry PI 3 Model B

Raspberry Pi 3 - Model B Technical Specification

- Broadcom BCM2387 chipset
- 1.2GHz Quad-Core ARM Cortex-A53
- 802.11 bgn Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE (Low Energy))
- 1GB RAM
- 64 Bit CPU
- 4 x USB ports
- 4 pole Stereo output and Composite video port
- Full size HDMI
- 10/100 BaseT Ethernet socket
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Micro USB power source

Raspberry Pi 3 - Model B Features

- Broadcom BCM2387 ARM Cortex-A53 Quad Core Processor powered Single Board Computer running at 1.2GHz
- 1GB RAM so you can now run bigger and more powerful applications
- Fully HAT compatible
- 40pin extended GPIO to enhance your “real world” projects.
- Connect a Raspberry Pi camera and touch screen display (each sold separately)
- Stream and watch Hi-definition video output at 1080p
- Micro SD slot for storing information and loading your operating systems.
- 10/100 BaseT Ethernet socket to quickly connect the Raspberry Pi to the Internet

Radio variants

Before starting you need to make a choice which type of radio you are going to build. There are nine possible variants that can be constructed as shown in the following table.

Table 1 Radio variants

Variant	Description	Display Type	Controls	Program
1	Two line 16 character LCD with push buttons	Two line LCD	Five push buttons	radiod.py
2	Four line 20 character LCD with push buttons	Four line LCD	Five push buttons	radio4.py
3	Two line 16 character LCD with rotary encoders	Two line LCD	Two rotary encoders with push buttons	rradiod.py
4	Four line LCD with rotary encoders	Four line LCD	Two rotary encoders with push buttons	rradio4.py
5*	Adafruit LCD plate with push buttons	Two line LCD (via I2C interface)	Five push buttons (Via I2C interface)	ada_radio.py
6	Two line 16 character LCD with rotary encoders	Two line LCD (via I2C backpack interface)	Two rotary encoders with push buttons	rradiobp.py
7	Four line 20 character LCD with push buttons	Four line LCD (via I2C backpack interface)	Two rotary encoders with push buttons	rradiobp4.py
8**	PiFace Control and Display (CAD) using SPI interface	Two line LCD and Infra Red sensor	Five push buttons and a reset button.	radio_piface.py
9	Vintage radio conversion	No display	Two rotary encoders Menu rotary switch	retro_radio.py

* This is a good choice for a complete beginner but you will still need some soldering skills.
** If you have no soldering skills then this is the best choice as the board comes ready made.

All of these use a different program at present as shown in the last column of the above table. Which one to use is a matter of personal choice. Variant 5 and 8 are good choices for a complete beginner.

The AdaFruit LCD plate or PiFace CAD is without a doubt the easiest to construct but can only use push buttons and not rotary encoders. The others require more effort but are worth the trouble.

It is then a simple choice of which display (two or four line) and whether to use rotary encoders or push button switches. The rotary encoder options give the most natural feel for the radio as most conventional radios use knobs to control the volume and station tuning.

The Retro radio software (option 9) is specifically intended for converting an old radio to an Internet radio whilst retaining the original look and feel of the radio. It has no LCD display.

The four lines LCD can display more information. Options 6 and 7 interface using an I2C backpack from Adafruit industries and only uses two wires. The PiFace CAD has support for remote controls but tends to be more sluggish in operation than other variants.

Housing the radio

This manual describes a couple of ways of housing the radio. A few ideas are below:

- A custom built case as shown in this manual
- Old plastic boxes or food containers
- Construct a case using Lego
- Use a pair of speaker housings that have enough room
- Install in an old vintage radio (really cool)
- Use an old wooden wine box
- Use an old video recorder, CD player or desktop set
- Buy a PC speaker set with enough room to build in the radio.

Figure 23 Some examples of radio cases




Take a look at the constructor's gallery at http://www.bobrathbone.com/pi_radio_constructors.htm to get some ideas that other constructors have used.

Note: Don't forget to make sure that there is adequate airflow through the radio housing to allow cooling of the Raspberry PI and other components. If necessary, drill at least five or six holes at the top and bottom of the housing.

If you decide to use a metal case (not advised) you will need a WiFi dongle with an aerial mounted externally to the case. Also the case must be earthed at the main supply both for safety reasons and to prevent interference with sound and/or the LCD screen.

Wiring

Table 3 and Table 4 on the following pages 27 and 28 respectively show the interface wiring for both the push button and rotary encoder versions of the radio. There are two versions of the wiring, 26 and 40 pin versions (Table 3 and Table 4 respectively). The connections used by the radio are highlighted in yellow. The IQaudio and newer HiFiBerry DACs require 40 pin versions of the Raspberry Pi.

This is where it can get a little confusing. The radio components (LCD, buttons, rotary encoders etc.) can use either 26 pin or 40 pin wiring as shown in the two tables. The 26 pin version wiring can also be used on a 40 pin Raspberry Pi. The 40 pin version of the wiring in Table 4 guarantees that there will be no wiring conflicts with DAC components.

Things get more complicated with **HiFiBerry** or **IQAudio** Digital to Audio Converters (**DACs**). These devices give excellent audio output quality and naturally many constructors want to use these. However they conflict with two GPIO pins that are used for the original radio wiring scheme.

Table 2 Radio wiring conflicts

Pin	GPIO	Radio Function	Conflicts with	Use pin	GPIO	Note
12	GPIO18	Down switch	DAC	19	GPIO10	26 or 40 pin Rpi
15	GPIO15	LCD data 5	IQAudio Amp Mute	31	GPIO6	40 pin only Rpi

Colour Legend | Radio | Conflict | Alternative wiring |

You are strongly advised to use the alternative wiring scheme so that IQAudio and **HiFiBerry** products and similar can be used either at the outset or at a later date.

The configuration for the radio is contained in a file called **/etc/radiod.conf**. By default and for backward compatibility this is configured for the original wiring scheme. From version 5.5 onwards every component of the radio is configurable in this configuration file.
So if you need to change the wiring to support DAC components then this needs to be reflected in the **radiod.conf** file.

```
down_switch=18
:
lcd_data5=22
```

Change to:

```
down_switch=10
:
lcd_data5=6
```

If using DAC products it may well worth considering using an LCD connected via the I2C interface. This will free up all the pins used by a directly connected LCD.

Table 3 Controls and LCD wiring 26 pin version

Pin	Description	Radio Function	Name	LCD pin	Push Buttons	Encoder (Tuner)	Encoder (Volume)
1	3V3	+3V supply			COMMON		
2	5V	5V for LCD		2,15			
3	GPIO2	I2C Data*	I2C Data				
4	5V						
5	GPIO3	I2C Clock*	I2C Clock				
6	GND	Zero volts		1,3*,5,16		Common	Common
7	GPIO 4	Mute volume					Knob Switch
8	GPIO 14	Volume down	UART TX		LEFT		Output A
9	GND	Zero Volts					
10	GPIO 15	Volume up	UART RX		RIGHT		Output B
11	GPIO 17	Channel Up			UP	Output A	
12	GPIO 18**	Channel Down	I2S Clock		DOWN	Output B	
13	GPIO 27	LCD Data 4		11			
14	GND	Zero Volts					
15	GPIO 22***	LCD Data 5	IQAudio Amp mute	12			
16	GPIO 23	LCD Data 6		13			
17	3V3	+3V supply					
18	GPIO 24	LCD Data 7		14			
19	GPIO 10**	Channel Down	SPI-MOSI		DOWN	Output B	
20	GND	Zero Volts					
21	GPIO 9	IR Sensor in (1)	SPI-MOSO				
22	GPIO 25	Menu Switch			MENU	Knob Switch	
23	GPIO 11	IR LED out (1)	SPI-SCLK				
24	GPIO 8	LCD E	SPI-CE0	6			
25	GND	Zero Volts					
26	GPIO 7	LCD RS	SPI-CE1	4			
33	GPIO 13	IR LED out (2)					
35	GPIO 19	HiFiBerry DAC+	I2S				
36	GPIO 16						
37	GPIO 26	IR Sensor (2)					
38	GPIO 20	HiFiBerry DAC+	I2S DIN				
40	GPIO 21	HiFiBerry DAC+	I2S DOUT				

Colour Legend | Radio | I2S | I2C (shared) | SPI Interface | IQAudio Amp mute |

* These pins are used for the I2C LCD backpack if used instead of the directly wired LCD to GPIO pins.
** Pin 12 is used by the HiFiberry or HiFiBerry DAC, Use GPIO10 (Pin 19) if using a DAC.
*** Pin 15 is used by the IqAudio amp mute function. Use GPIO

Table 4 Radio and DAC devices 40 pin wiring

Pin	Description	Radio Function	Name	IQAudio Function	Push Button	Encoder (Tuner)	Encoder (Volume)
1	3V3	+3V supply	+3V	+3V	+3V		
2	5V	5V for LCD	+5V	+5V			
3	GPIO2	I2C Data	I2C Data	I2C Data			
4	5V			+5V			
5	GPIO3	I2C Clock	I2C Clock	I2C Clock			
6	GND	Zero volts	0V	0V		Common	Common
7	GPIO 4	Mute volume				Knob Switch	
8	GPIO 14	Volume up	UART TX		UP	Output A	
9	GND	Zero Volts		0V			
10	GPIO 15	Volume down	UART RX		DOWN	Output B	
11	GPIO 17	Menu switch					Knob Switch
12	GPIO 18			I2S CLK			
13	GPIO 27						
14	GND	Zero Volts		0V			
15	GPIO 22			Mute			
16	GPIO 23	Channel down		Rotary enc A	LEFT		Output A
17	3V3	+3V supply		0V			
18	GPIO 24	Channel up		Rotary Enc B	RIGHT		Output B
19	GPIO 10		SPI-MOSI				
20	GND	Zero Volts					
21	GPIO9		SPI-MISO				
22	GPIO 25	IR Sensor		IR sensor			
23	GPIO 11		SPI-SCLK				
24	GPIO 8	LCD E	SPI-CE0				
25	GND	Zero Volts		0V			
26	GPIO 7	LCD RS	SPI-CE1				
27	DNC			PiDac+ Eprom			
28	DNC			PiDac+ Eprom			
29	GPIO5	LCD Data 4					
30	GND	Zero Volts					
31	GPIO6	LCD Data 5					
32	GPIO12	LCD Data 6					
33	GPIO 13	LCD Data 7					
34	GND	Zero Volts					
35	GPIO 19	IQAudio DAC+	I2S	I2S			
36	GPIO 16	IR LED out					
37	GPIO 26						
38	GPIO 20	IQAudio DAC+	I2S DIN	I2S DIN			
39	GND	Zero Volts					
40	GPIO 21	IQAudio DAC+	I2S DOUT	I2S DOUT			

Colour Legend	Radio	IQAudio	I2C (shared)	SPI Interface

Note: Make sure you are using the correct columns in the above table. Use column 6 (Push Buttons) for the push button version and the last two columns (Encoder Tuner/Volume) for the rotary encoder version.

Version 2, 3 or model B+ boards

Wire one side of the switches to the 3.3V pin. Wire the other side of each switch to the GPIO pin as shown in the last column of the above table. This is the normal option. Version 2 boards have internal pull up/down resistors and don't require external resistors. In fact including these can cause problems.

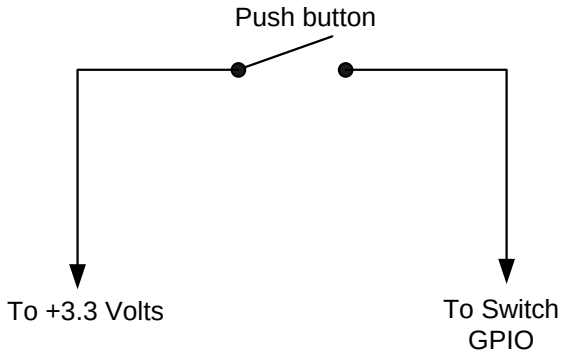


Figure 24 Switch wiring version 2 boards

Version 1 boards (early boards)

It is becoming increasingly difficult to support version 1.0 boards and you are advised to purchase a newer Raspberry Pi board for this project. However tips for using version 1 boards will be retained in this manual; however, if there is a problem, regrettably no support can be provided.

Wire one side of the switches to the 3.3V pin. Wire the other side of each switch to the GPIO pin as shown in the last column of the above table. Also wire this same side of the switch to the 0V pin via a 10KΩ resistor. See Figure 25 below.

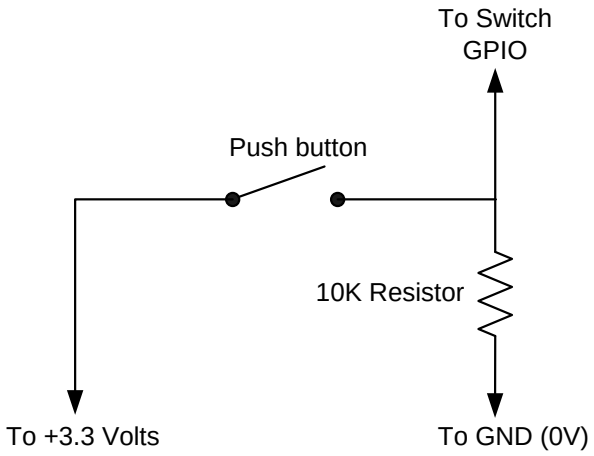


Figure 25 Switch Wiring version 1 boards

Rotary encoder wiring

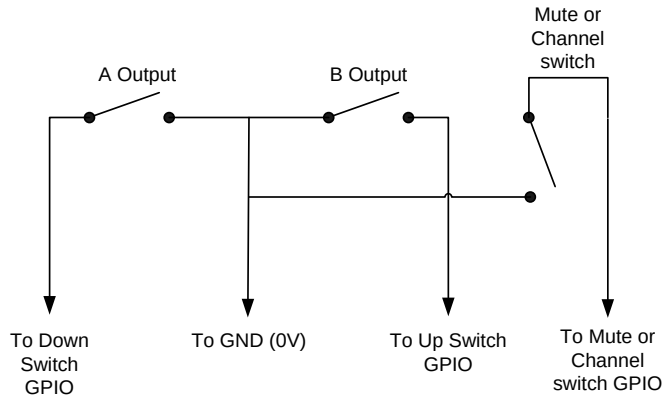


Figure 26 Rotary Encoder Diagram

Rotary encoders have three inputs namely Ground, Pin A and B as shown in the diagram on the left. Wire the encoders according that shown in Table 3 on page 27. If the encoder also has a push button knob then wire one side to ground and the other to the GPIO pin. In the case of the mute switch this will be pin 7 (GPIO 4). Version 1 boards are not supported but do work.

Warning: The push switches (if fitted) on the rotary encoder are wired differently from the push buttons in the push button versions of the radio. For these encoders one side of the push button is wired to GND (not 3.3V) and the other to the relevant GPIO.

If using a Revision 1 board it is necessary to use 10K pull up resistors connected between the GPIO inputs of the rotary encoder outputs and the 3.3 volt line. Do not add resistors if using revision 2 boards and onwards.

Figure 27 Rotary encoder with push switch

This project uses a COM-09117 12-step rotary encoder from Sparkfun.com. It also has a select switch (by pushing in on the knob). These are "Incremental Rotary Encoders". An incremental rotary encoder provides cyclical outputs (only) when the encoder is rotated. The other type is an absolute rotary encoder which maintains position information even when switched off (See Wikipedia article on rotary encoders). These tend to be bigger and more expensive due to extra electronics required Only incremental encoders are used in this project.

The rotary encoders used in this project are wired with the COMMON or GND pin in the middle and the A and B outputs either side. However some rotary encoders are wired with A and B as the first two pins and GND (COM) as the third pin. Also not all encoders come with a switch, so separate switches for the Menu and Mute button will need to be installed. Check the specification for your encoders first.

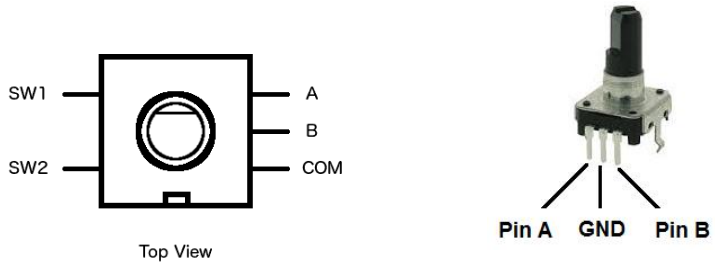


Figure 28 Rotary encoder pin-outs

Note: Not all manufacturers' rotary encoders will work with this project. If they work then fine if not regrettably you will need to purchase the recommended encoders.

LCD Module Wiring

The following shows the wiring for a directly wired HD44780 LCD controller. It has 16 or 18 pins. There are two ways of wiring the LCD data lines.

Table 5 LCD module wiring for 26 pin Raspberry Pi's

LCD Pin	GPIO 26 pin	Pin 26 #	Description
1	n/a	6	Ground (0V) – Wire this directly to LCD pin 5
2	n/a	1	VCC +5V
3	n/a	6 or 9	Contrast adjustment (0V gives maximum contrast) [1]
4	GPIO7	26	Register Select (RS). RS=0: Command, RS=1: Data
5	n/a	6 or 9	Read/Write (RW). Very important this pin must be grounded! R/W=0 (GND): Write, R/W=1 (+5V): Read. Will damage the PI if not grounded (0V). Wire LCD pin 5 and 1 together
6	GPIO8	24	Enable (EN)
7			Data Bit 0 (Not required in 4-bit operation)
8			Data Bit 1 (Not required in 4-bit operation)
9			Data Bit 2 (Not required in 4-bit operation)
10			Data Bit 3 (Not required in 4-bit operation)
11	GPIO27	13	Data Bit 4 (D4)
12	GPIO22	15	Data Bit 5 (D5) Note if using IQAudio products this pin conflicts !!
13	GPIO23	16	Data Bit 6 (D6)
14	GPIO24	18	Data Bit 7 (D7)
15	n/a	1	LED Backlight Anode (+5V) or Red LED (Adafruit RGB plate) [2]
16	n/a	6 or 9	LED Backlight Cathode (GND)
17			Optional Green LED (Adafruit RGB plate) [2]
18			Optional Blue LED (Adafruit RGB plate) [2]

If using **IQAudio** products it is necessary to use the 40 pin version of the wiring (See Table 4). In particular the LCD data lines use pins 29,31,32 and 33 as shown below.

Table 6 LCD module wiring 40 pin Raspberry PI's

LCD Pin	GPIO 40 pin	Pin 26 #	Description
11	GPIO5	29	Data Bit 4 (D4)
12	GPIO6	31	Data Bit 5 (D5)
13	GPIO12	32	Data Bit 6 (D6)
14	GPIO13	33	Data Bit 7 (D7)

All other wiring is the same as the 26 pin version shown in Table 5.

Note 1: The contrast pin 3 (VE) should be connected to the centre pin of a 10K potentiometer. Connect the other two pins of the potentiometer to 5v (VDD) and 0v (VSS) respectively. Adjust the preset potentiometer for the best contrast. If you do not wish to use a contrast potentiometer then wire pin 3 (VE in the diagram below) to GND (0V). The following diagram (Courtesy protostack.com) shows the electrical connections for the standard 16 pin LCD.

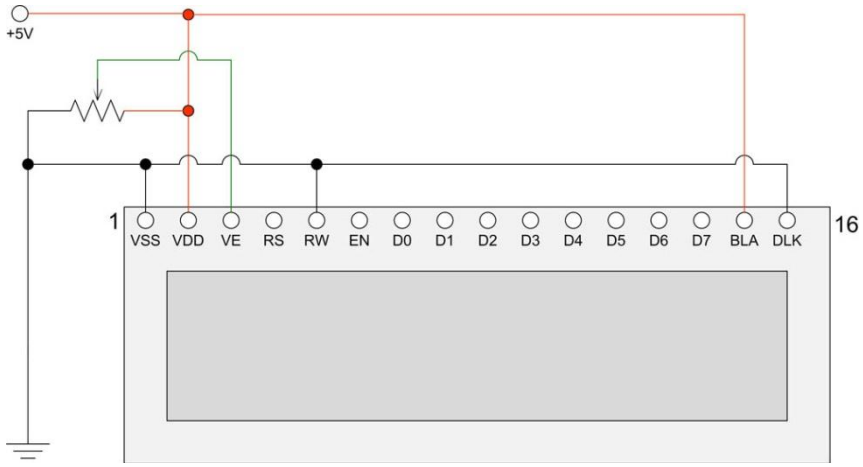


Figure 29 HD44780 LCD electrical circuit

The standard LCD comes with 16 pins. Adafruit supply an RGB backlit LCD with two extra pins namely pins 17 and 18. These are non-standard. These must be wired to ground (0 Volts) to work. For more information see the section *Using the Adafruit backlit RGB LCD* display on page 111.

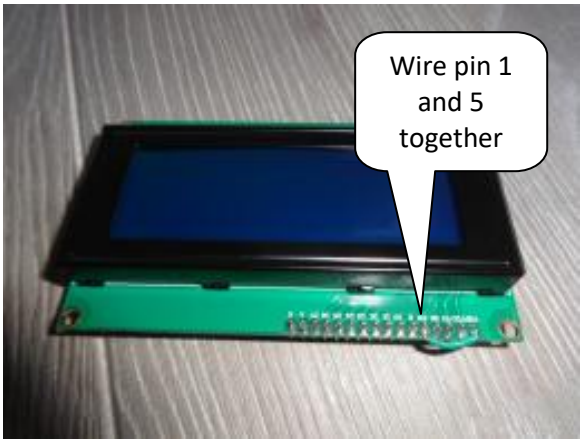


Figure 30 Wire LCD pin 1 (GND) and 5 (RW) together

The Read/Write (RW) pin 5 must be connected to pin 1 (0V). It is very important that this pin is grounded!

If pin 5 is not grounded it will damage the Raspberry PI. Always wire LCD pin 5 and 1 directly together. Do not rely on grounding pin 5 with a GND wire on the connector. If this wire drops off then the LCD data lines will be put into write mode putting +5V on the GPIO pins causing irreparable damage to the Raspberry Pi.

Warning – Some LCD displays such as Midas have a different voltage arrangement for Pin 15 and Pin 5 (Contrast). Pin 15 is an output which provides a negative voltage (VEE) which connects to one end of the 10K contrast potentiometer and the other end to +5V (VDD). Connecting +5 Volts to pin 15 will destroy the LCD device.
DO NOT USE THESE DEVICES UNLESS YOU KNOW WHAT YOU ARE DOING.

Power supply considerations

The Raspberry Pi uses a standard Micro USB (type B) power connector, which runs at 5 V. In general the Raspberry PI can draw up to 700mA. Many telephone adapters deliver less than that and can lead to problems. You also need to consider the LCD screen which can also need up to 20mA but depends on the type of backlight.

Try to find a power adapter that delivers at least 1.0 Ampere. Even better is 1.5 Amperes. See the following article. The newer RPi models can draw up to 1.5 Amps if USB peripherals are attached.

http://elinux.org/RPi_VerifiedPeripherals#Power_adapters

The Raspberry PI can be powered either the USB port or via the GPIO header (Pin 2). Some prototyping boards such as the Ciseco Humble PI can provide power in this way.
See http://shop.ciseco.co.uk/k001-humble-pi/
This interface board can be ordered with an optional 5 volt regulator. If using the Humble PI try with regulator use 6v to 7v as the power input to the Humble PI 5v regulator (Not the Raspberry PI). Trying to use 9v or more will mean that the 5 volt regulator will get far too hot.

If using an adaptor or separate 5 volt Power Supply try to use a switched-mode power supply adaptor. This take less current and generate less heat that a power dissipation device. If a power supply is designed to be earthed then use a 3 core cable with live, neutral and earth wires.

Things not to do:

- Do not try to tap off power from the Power supply or transformer used by the speaker's amplifier. This won't work (earth loops) and can cause damage to the PI and peripherals.
- Do not tap off (cascade) from the amplifier DC supply (12 volts for example) with another 5V voltage regulator. This will most likely cause interference.
- Do not feed power to the PI from two sources (USB hub and Power adapter). Try to use USB hubs that don't feed 5 volts back through the USB ports of the Raspberry PI
- Do not connect an untested power supply to the Raspberry PI without checking the voltage first.

Things to do:

- Use double pole mains switches for isolating the mains supply when switched off. A lot of European plugs can be reversed leaving the live wire un-switched if using a single pole switch.
- If using a metal case always earth it and use a three pin plug with earth pin.
- In general feed the 5 volt supply via the Raspberry Pi rather than via the GPIO header. This is because the Raspberry Pi is fitted with a so called poly-fuse for protection.

You should try to use a single power supply switch for the radio. One technique which is quite useful is to split open a USB power adaptor (Use a hacksaw very carefully). Connect the AC power supply of the adaptor to the mains switch. This switch can also provide the mains supply to the speaker amplifier. Also see the section on preventing electrical interference on page 34.

Always consider safety first and make sure that no-one including yourself can receive an electric shock from your project including when the case is open. Also see disclaimer on page 150.

Preventing electrical interference

One of the most irritating faults that one can have is the LCD screen occasionally either going blank or displaying hieroglyphics especially when switching on and off other apparatus or lights on the same circuit. This is due to Electromagnetic Interference (EMI).

See https://en.wikipedia.org/wiki/Electromagnetic_interference for more information.

EMI can be caused by any number of sources such as fluorescent lighting, switching on and off equipment on the same circuit as the radio or even electrical storms. If you are using a standard Raspberry PI USB power supply then you will probably not experience this problem as nearly all are fitted with a ferrite core (This is the big lump in the cable or may be built in). If you do experience this problem then try the following solutions one at a time in the order shown below. They can all be used together if required.

Using a clip on ferrite core on the +5 volt cable

Figure 31 Clip on ferrite core

One of the most effective solutions is to put a clip on ferrite core on the +5V cable going to both the Raspberry Pi and USB hub. Loop the wire through at least once. Even a single loop seems to be enough. Try this first!

Figure 32 Loop +5V supply around the core

Fit a mains filter

Figure 33 Various mains filters

Try using a mains filter. This has the advantage that it can prevent spikes coming in from the mains and protect against electrical storms. The picture on the right shows an integrated filter and panel mount mains socket.

Figure 34 Integrated mains socket and filter

Try a decoupling capacitors

Figure 35 0.1 uF decoupling capacitor

Try fitting 0.1 uF decoupling capacitors on so close as possible to the +5V supply coming into the LCD board. Connect between the +5V supply and GND (0V)

Preventing ground loops

Figure 36 3.5mm Jack Ground Loop Isolator

Avoid creating ground loops in the first place during construction. Ground loop issues usually cause a humming noise. Trying to tap off the Raspberry power supply from the power supply for the amplifier (if used) is one sure way to create a ground loop. If you experience such a problem then a 3.5mm Jack Ground Loop Isolator available from suppliers such as Kenable (http://www.kenable.co.uk) can prevent unwanted hum on the audio system. Place the isolator between the Audio output of the Rasberry Pi or sound card and the amplifier input.

For further information on ground loops:
https://en.wikipedia.org/wiki/Ground_loop_(electricity)

GPIO Hardware Notes

The following shows the pin outs for the GPIO pins on revision 1 and 2 boards. For more information see: http://elinux.org/RPi_Low-level_peripherals.

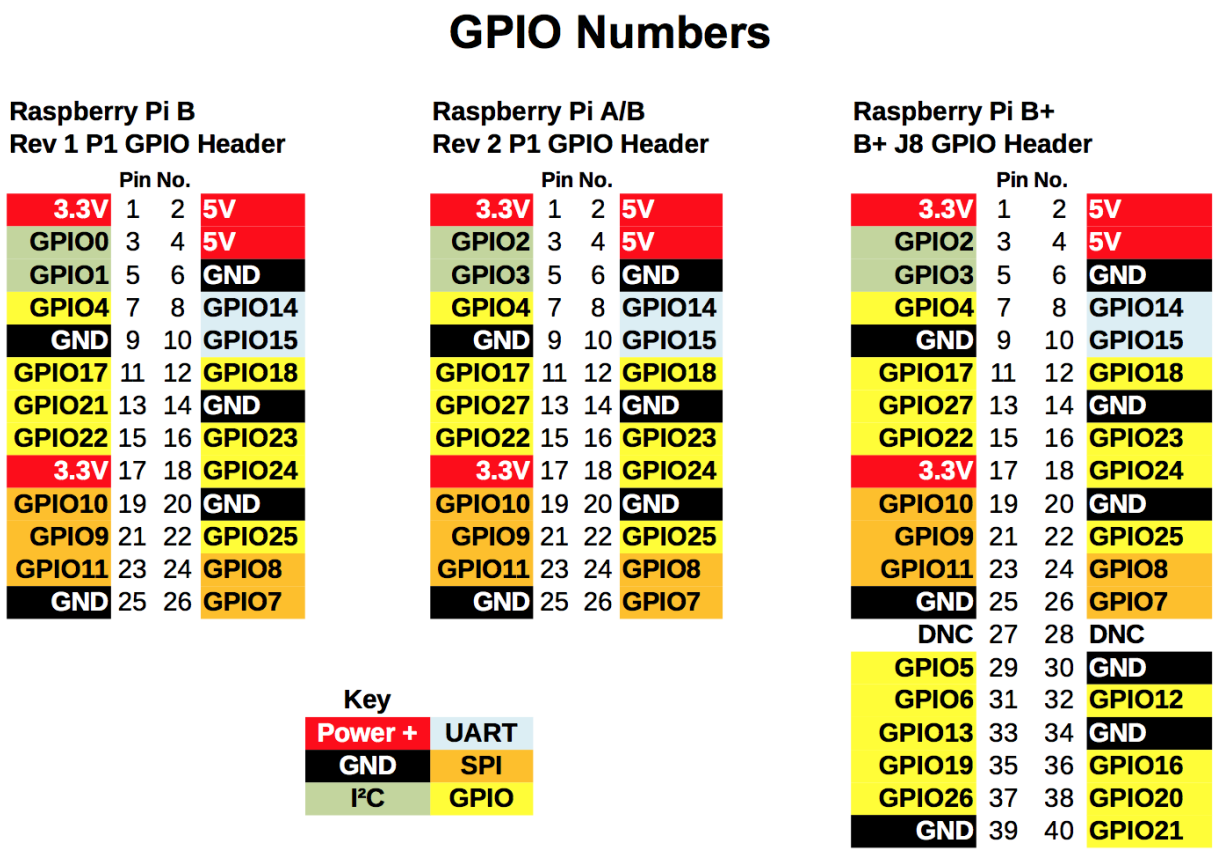
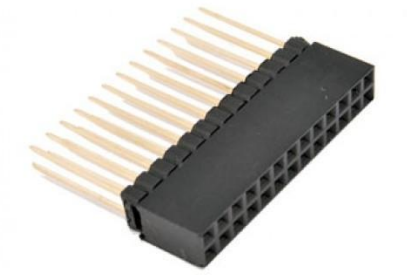


Figure 37 GPIO Numbers

Note: The B+, 2B and 3B have the same pin-outs.



If connecting any 40 pin interface board via a 26 way ribbon cable it will be necessary to fit a 26 pin header extender and plug the ribbon cable into it.

Figure 38 26 pin header extender

Parts List

The following table shows the parts list for the Raspberry PI Internet Radio. This list is for the version using the HD44780 LCD directly connected to the GPIO pins. If using the Adafruit five button LCD Plate then don't order the parts marked with an asterix (*)

Table 7 Parts list

Qty	Part	Supplier
1	Raspberry Pi Computer	Farnell Element 14
1	Clear Raspberry Case	RS Components
1	8GByte SD Card	Any PC or Photographic supplier
1	Wooden Radio Case	A good friend of mine
1	Raspbian Jessie OS	Raspberry Pi foundation downloads
2	Four inch loudspeakers	From set of old PC speakers
2	Four inch loudspeaker grills	Any electronics shop
1	Stereo Amplifier (3 to 5 watt)	From set of old PC speakers
1	Transformer for amplifier	From set of old PC speakers
1	LCD HD44780 2 x 16 Display *	Farnell Element 14
1	ModMyPi Slice of Pi *	Ciseco PLC
4	Round push buttons *	Any electronics shop
1	Square push button *	Any electronics shop
2	Rotary encoders if using this option * **	Sparkfun.com
1	26 way ribbon cable	Tandy or Farnell Element 14
5	10KΩ resistors * (Revision 1 boards only)	Tandy or Farnell Element 14
5	1K resistors * (Revision 1 boards only)	Tandy or Farnell Element 14
1	Four port USB hub (Revision 1 & 2 boards only)	Any PC supplier
1	External power supply for USB hub (1200 mA)	Any PC supplier
1	26 way PCB mount male connector	Any electronics shop
1	26 way GPIO extender (model B+ boards only)	ModMyPi and others
1	Mains cable	Hardware shop
1	Double pole mains switch with neon	Farnell Element 14
5	Male 2 pin PCB mount connectors	Any electronics shop
2	Female 4 pin PCB connectors	Any electronics shop
1	Female 2 pin PCB connectors	Any electronics shop
1	16 pin male in-line PCB mount connector	Any electronics shop
1	Stereo jack plug socket	Any electronics shop
1	Panel mount Ethernet socket	Any electronics shop
1	Adafruit I2C LCD interface Backpack ***	http://www.adafruit.com/
1	PiFace Control and Display (CAD) ****	See http://www.piface.org.uk
1	TSOP38238 IR Sensor	Adafruit industries and others
1	Red or Green LED and 220 Ohm resistor	Any electronics shop
	Shrink wrap and thin wire for PCB wiring	Any electronics shop

* These components are not required if using the Adafruit LCD plate.
** If using rotary encoders.
*** If using the Adafruit I2C back (Note: This only works in combination with rotary encoders)
**** Only if using the PiFace Control and display.

Construction HD44780 LCD

The following is for an HD44780 LCD display directly wired to the GPIO pins. If using the Adafruit LCD plate see section called *Construction using an Adafruit* LCD plate on page 40.

The following illustration shows the parts for the classic style radio.

Figure 39 Radio parts

The above photo shows the components before assembly. See from back and left to right. A wooden case, mains cable, Raspberry PI in a transparent plastic case, speaker grills, 5v power supply, 4 inch speakers, 2 x 16 LCD display, four port USB hub, stereo amplifier, five push button switches, 11.5v transformer for the stereo amplifier, ribbon cable and the LCD and push buttons interface board. Not shown is the (optional) USB wireless dongle.

Building the LCD and pushbuttons interface board

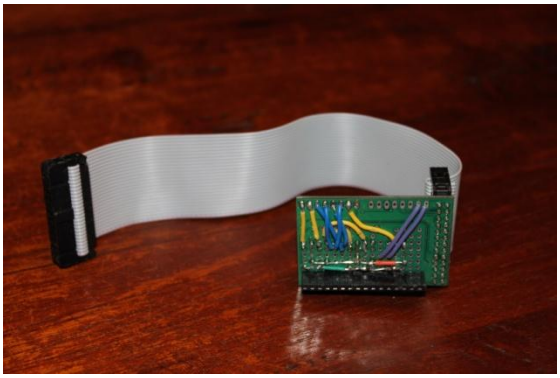


Figure 40 Interface board (Wiring side)

The LCD and push button interface was originally built using a ModMyPi Slice of PI from Ciseco PLC however this product has been discontinued however any suitable prototype board will do. The board was fitted with 1 female 16 pin in-line connector for LCD and a male 26 pin connector for the 26 way ribbon cable. If no ribbon cable is to be used then use a female 26 way connector to plug into the Raspberry PI GPIO interface

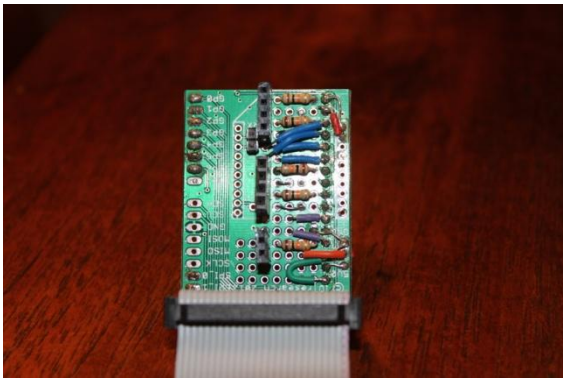


Figure 41 Interface board (Component side)

The component side of the LCD and push button shows the female connectors for five push button switches and the five 10KΩ pull down resistors. In this construction the 1KΩ resistors shown in Figure 25 Switch Wiring on page 20 weren't used but do provide some extra protection for GPIO inputs.

The following picture shows the components mounted in the wooden case.

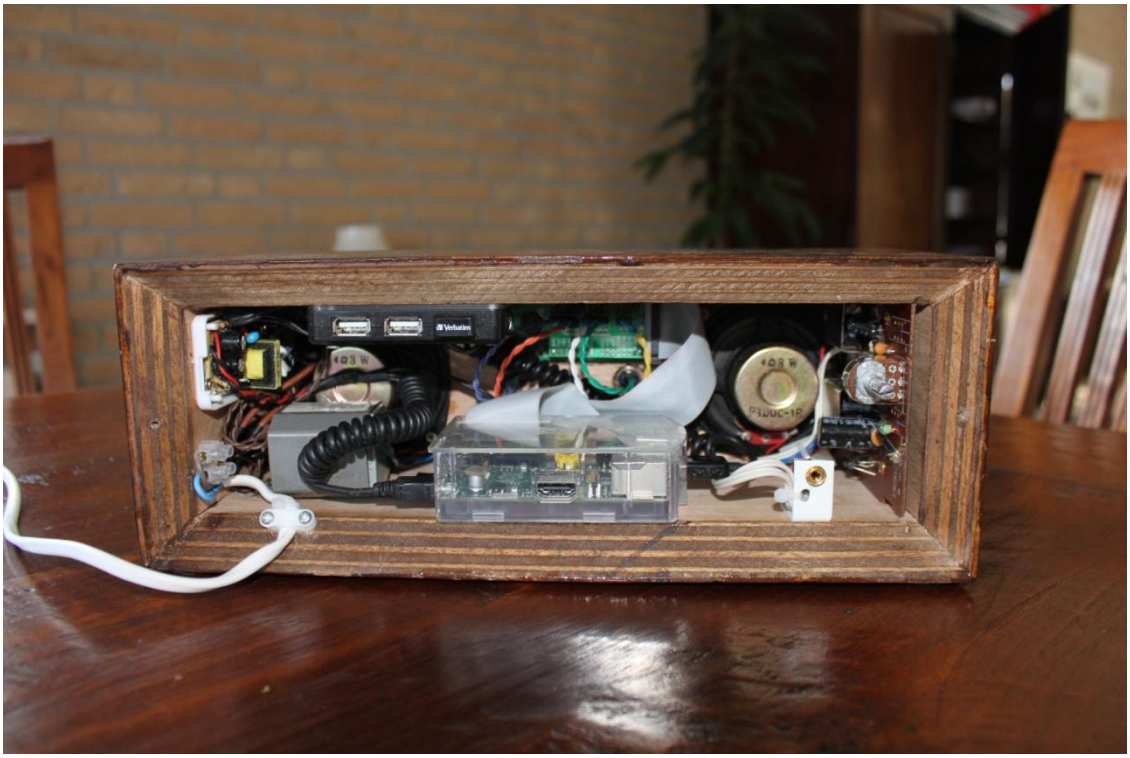


Figure 42 Radio rear inside view

Shown from top left to bottom right: 5V power supply (from a standard phone charger), four port USB with a memory stick for music files, LCD and Switch interface board (You can just see one of the switches connected with a twisted green wire), stereo amplifier (volume control now just a preset) and 4 inch speakers from a set of old PC speakers, mains input (mains switch behind this), mains transformer for the amplifier, Raspberry PI in a clear plastic case and finally the headphones socket.

Construction using an Adafruit LCD plate

Introduction

This section is for the radio using an Adafruit RGB-backlit LCD plate for Raspberry PI. The complete instructions for both ordering and building this product are available from the following web site: http://www.adafruit.com/products/1110 (See tutorials)

Note: Don't confuse this product (which has an I2C interface chip) with the two line and four line RGB LCDs which Adafruit also sell.

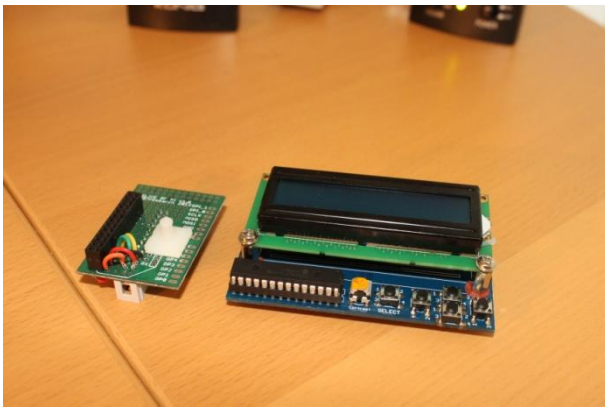


Figure 43 Adafruit LCD plate

The Adafruit LCD plate is designed to directly into the GPIO header on the Raspberry PI. These fit into a female 26 way header. If you want to connect the Adafruit LCD via a ribbon cable you will need to mount a 26 way male header instead of the female header and you will also need to construct a reversing board (Shown on the left of the picture on the left). Because ribbon cable swaps the two rows of pins over the reversing card is required to swap the two rows of pins back to their correct orientation.

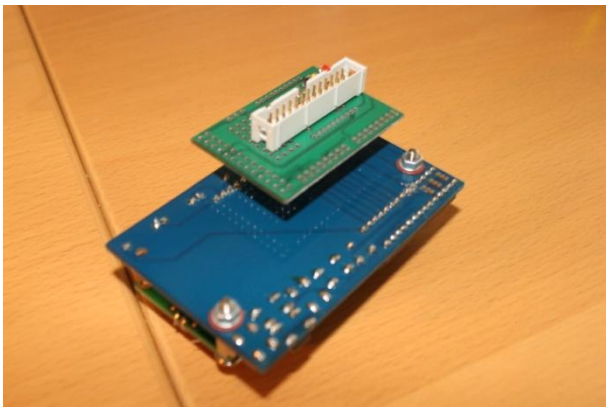


Figure 44 Adafruit LCD plate with ribbon cable adapter

Back view of the reversing board plugged into the Adafruit LCD plate.

The GPIO pins used are:

1. 3.3 Volts
2. 5.0 volts
3. SDA0
4. -
5. SCL0
6. Ground

Note 1: If you are going to plug the Adafruit LCD plate directly into the GPIO header on the Raspberry PI then you don't need the above reversing plate. Just follow the construction instructions on the tutorial on the Adafruit site.

Note 2: The "Select" button on the Adafruit plate is the "Menu" button for the radio.

Note 3: If you want to use an Adafruit display that allows setting different colours for the backlight then see section *Configuring the Adafruit LCD backlight colours* on page 79 for instructions on how to do this.

Using other switches

The Adafruit Plate comes with five 4 pin switches which are mounted on the interface board. You will almost certainly want to use other switches say mounted on a front panel. It doesn't matter which type of switch you use as long as it is a push to make type. The only reason that a four connector switch is used is for mechanical strength.

If you look closely you will see push button symbol between pins 2 and 4 and 1 or 3 on the component side for four of the switches. Either 2 and 4 and 1 or 3 should be connected to the switches.

It is advisable to solder two posts (male pins) for each switch on the reverse side of the board (The non-component side). Don't solder wires directly into the board. It is better to use push-on jumper wires connected to the switches to connect to the posts on the card.

Note: Rotary encoders cannot be used with the Adafruit Plate as these require thee connections and the Adafruit routines to utilise them are not supplied by Adafruit.

Using the Adafruit LCD plate with the model B+, 2B and 3B

The plate is designed for both Revision 1 and Revision 2 Raspberry Pi's. It uses the I2C (SDA/SCL) pins. Adafruit supply a special extra tall 26-pin header so the plate sits above the USB and Ethernet jacks. For Pi Model B+, the resistors sit right above the new set of USB ports. To keep them from shorting against the metal, a piece of electrical tape must be placed on the top of the USB ports.

Construction using an I2C LCD backpack

Skip this section if you are not using an I2C backpack. There are two versions of the backpack supported:

1. Adafruit I2C backpack using an MCP23017 port expander – address 0x20
2. Arduino I2C backpack using a PCF8574 port expander – address 0x27

This design supports both I2C and SPI interfaces. I2C is the default. This interface only requires two signals namely the I2C Data and Clock. This saves six GPIO pins when compared with the directly wired LCD interface. See https://www.adafruit.com/product/292.

The radio software also supports the more common PCF8574 chip based backpack popular with the Arduino hobby computer may also be used. See http://www.play-zone.ch/en/i2c-backpack-pcf8574t-fur-1602-lcds-5v.html for example.
This is configurable in the **/etc/radiod.conf** file.

```
# The i2cbackpack is either ADAFRUIT or PCF8574
# i2c_backpack=PCF8574
i2c_backpack=ADAFRUIT
```

Note: In previous versions the **i2c_backpack** parameter was incorrectly shown as PCF8475 instead of PCF8574. Check the **/etc/radiod.conf** file.

Adafruit I2C Backpack

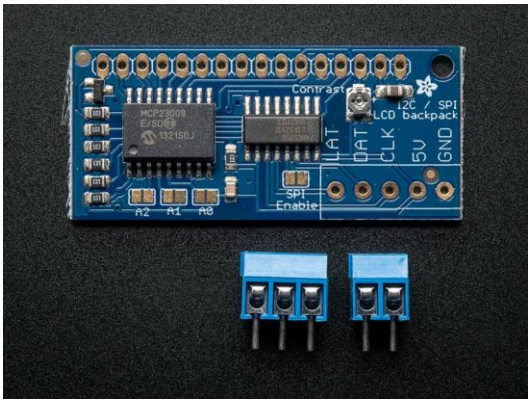


Figure 45 Adafruit I2C Backpack

The Adafruit I2C/SPI backpack interface is shipped as shown in the diagram opposite. There are no connectors shipped to connect to the LCD itself to this interface. These must be ordered separately.

Order a 16 in-line connector.

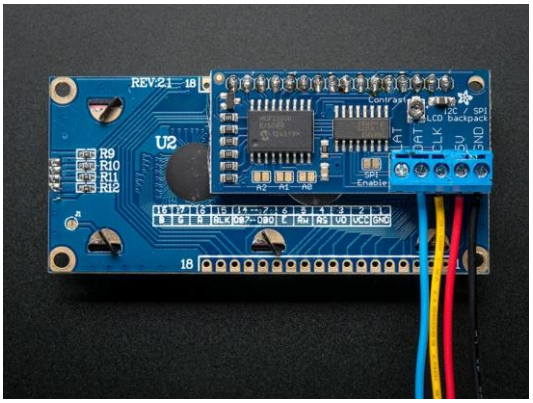


Figure 46 LCD connected to an Adafruit I2C backpack

The diagram shown on the left shows a 2x16 character LCD connected to the I2C backpack.
The wiring right to left is:

1. LAT (Unused)
2. Blue – I2C Data – GPIO pin 3
3. Yellow – I2C Clock – GPIO pin 5
4. Red - +5 volts – GPIO pin 2
5. Black - GND (0 volts) – GND pin 6

The I2C Data (DAT) connects to pin 3 on the Raspberry Pi GPIO header and the I2C Clock (CLK) to pin 5 on the GPIO header.

Arduino PCF8574 I2C backpacks

These types of backpack are popular with Arduino users. The device address is usually 0x27.

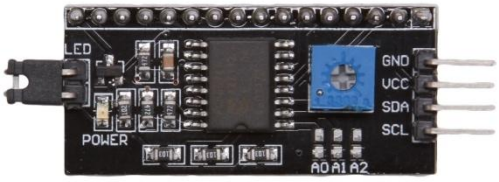


Figure 47 Arduino I2C backpack

The wiring From top to bottom is:

1. GND (0 volts) – GPIO pin 6
2. VCC +5 volts – GPIO pin 2
3. SDA I2C Data – GPIO pin 3
4. SCL I2C Clock – GPIO pin 5

The blue potentiometer on the right is the contrast adjustment.
To use this device amend the i2c_backpack parameter in **/etc/radiod.conf** (Comment out the ADAFRUIT line).

```
i2c_backpack=PCF8574
#i2c_backpack=ADAFRUIT
```

Creating the interface board for the I2C back pack

An interface board is recommended to connect the I2C backpack and rotary encoders etc. to the GPIO interface. The Ciseco Humble PI is an ideal way to construct the interface board.
See http://shop.ciseco.co.uk/k001-humble-pi/

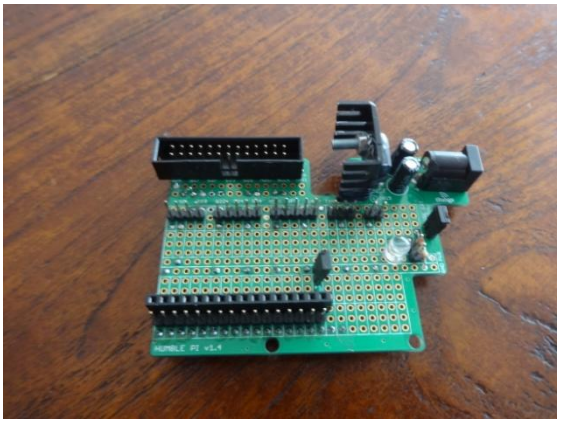


Figure 48 Ciseco Humble PI I2C interface board

Figure 49 The I2C backpack interface board

The above figure shows the I2C interface board using the Ciseco Humble PI. The header pins in the centre from left to right are, I2C interface connector (4 pins), Volume rotary encoder (5 pins), Channel rotary encoder (5 pins), IR sensor (3 pins) and front panel LED (2 pins). In this version there are two rows of 18 pins (male and female) to allow different I2C backpack to be connected. You will normally only need one or the other.

The above diagram shows the Adafruit I2C backpack connected to the interface board along with the rotary encoders. The 26 pin male header connects to the GPIO ribbon cable on the Raspberry PI. On the left is a 6V to 9V power input feeding a 5 volt regulator.

Construction using the PiFace CAD

Fortunately no soldering or construction is required with the PiFace CAD. Just plug it in and run the software. The main advantage of the PiFace CAD is that it has an IR sensor which means that it can be used with a remote control. It is however more sluggish in its operation when compared to other variants of the radio. It also has the disadvantage that the push buttons are on the bottom of the unit.

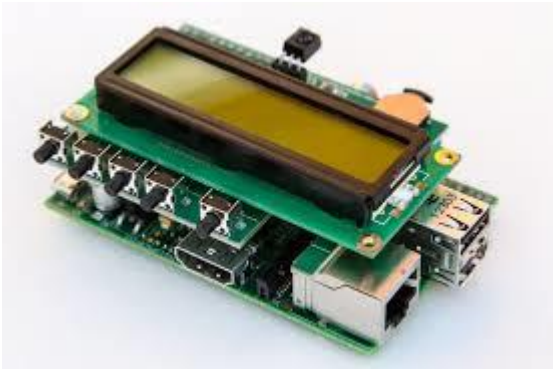


Figure 50 PiFace CAD and Rasberry PI

Figure 51 PiFace CAD in a case

Various ready-made cases are available from various suppliers. Warning: not all fit properly and might require some modification.

The PiFace CAD uses the SPI interface (from Motorola)
See http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus for further information on SPI.

Installing an IR sensor and remote control

IR Sensor

If you are using the PiFace CAD then the IR sensor is already built in and is connected via the PiFace CAD to GPIO 23 (Pin 16) so skip this section. If you wish to use an IR remote control with other variants of the radio then purchase an IR sensor TSOP38238 or similar. The output in this case will be connected to GPIO 11 (Pin 23) or GPIO 26 (pin 37) for LCD or Adafruit Plate respectively.

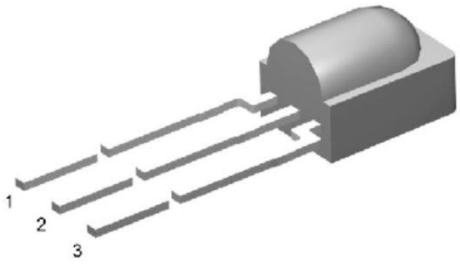


Figure 52 TSOP38238 IR sensor

The TSOP38xxx series works from 2.5 to 5.5 volts and is ideal for use the Raspberry PI.

IR sensor	Description	RPi
Pin 1	Signal Out	GPIO in *
Pin 2	Ground	Pin 6
Pin 3 **	Vs 3.3 Volts	Pin 1

* See Table 9 IR Sensor Pin outs

** **Caution** Do not connect accidently to 5 volts

There are equivalent devices on the market such as the TSOP4838 which operate on 3.3 volts only.

See http://www.vishay.com/docs/82491/tsop382.pdf for more information on these IR sensors.

Tip: These IR sensors are very prone to damage by heat when soldering them. It is a good idea to use a 3 pin female connector and push the legs of the IR detector into them. If you solder the IR detector directly into a circuit then take precautions by connecting a crocodile clip across each pin in turn whilst soldering it. See figure below:

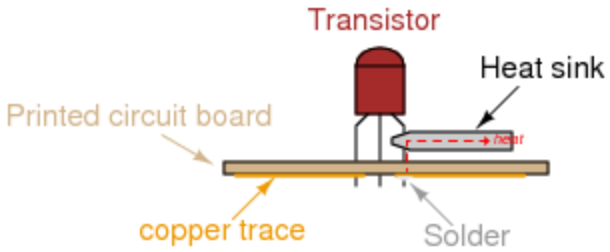


Figure 53 Soldering precautions

Remote control

Almost any surplus IR remote control can be used with this project. Later on it is explained how to set up the remote control with the radio software. You will need to install the software for IR sensor.

See the section called *Installing the Infra Red sensor software* on page 73.

Remote Control Activity LED

If wanted an activity LED can be connected to GPIO 11 or 13 depending on the type of radio. This flashes every remote control activity is detected. It is a good idea to include this.

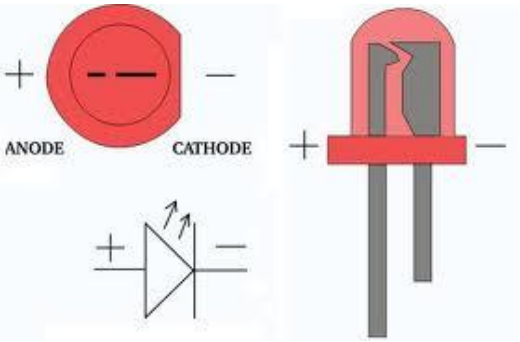


Figure 54 LED polarity

LEDs have polarity and must be wired correctly to work. The diagram shows the polarity of a typical LED. The longer lead is the positive (+) connection and connects to the Anode (The smaller terminal inside the LED)
Also the LED must be wired in series with a resistor to limit the current, typically 100 Ohms is OK. Failure to do this may cause the LED to burn brightly for a while then burn out.
Connect the cathode to GND (RPi Pin 6) and the Anode (+) to GPIO 11 (Pin 23) via a 100 Ohm resistor.

Sudo vi /et The following table shows the GPIO pin used for the LED connections.

Table 8 Remote Control Activity LED

Radio Type	Pin	GPIO	Type of Raspberry PI
Activity LED not fitted	none	n/a	Not applicable
Two or Four line LCD with Push Buttons	23	**11**	26 or 40 Pin version
Two or Four line LCD with Rotary encoders	23	**11**	26 or 40 Pin version
Two or Four line LCD with I2C backpack	23	**11**	26 or 40 Pin version
Pi Face CAD with push buttons	33	**13**	40 pin version only
Adafruit RGB plate with push buttons	33	**13**	40 pin version only
Vintage radio with no LCD display	16	**23**	26 or 40 Pin version
Designs using IQAudio sound boards	36	**16**	40 pin version only

How to configure the LED is shown on in the section called *Configuring the remote control activity LED* on page 78.

Warning: Configuring the LED on GPIO 11 (Pin 23) when using a PiFace CAD will cause the PiFace CAD to crash on start up. GPIO 11 is part of the SPI interface used by the PiFace CAD.

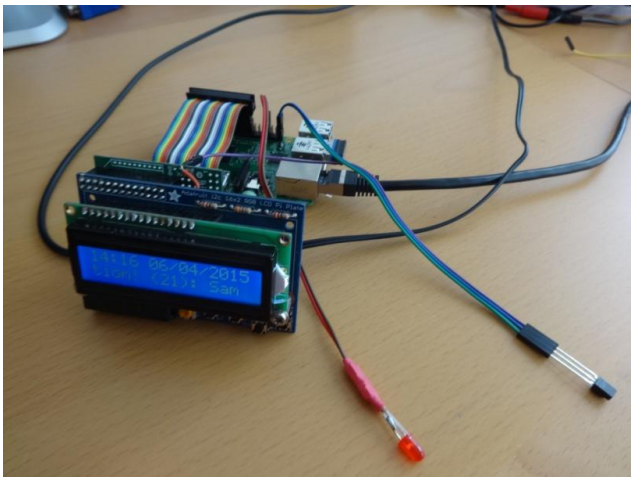


Figure 55 Adafruit plate with IR sensor and activity LED.

The illustration on the left shows an Adafruit RGB plate with IR sensor and activity LED.

The IR sensor picks up 3.3 volts from the reversing plate and connects the signal output to Pin 40 (GPIO 21) and GND (pin 39).

The LED connects to pin 33 (GPIO 13) and ground (pin 34).

Construction using a HiFiBerry DAC

This version supports the HiFiBerry DAC (Digital Analogue Converter) from HiFiBerry. See https://www.hifiberry.com. There are a lot of versions of the HiFiBerry DAC.

They split down into two main types:

1. The HiFi Berry DAC using the P5 connector (Older boards but not all)
2. The HiFi Berry DAC PLUS using the 40 pin connector (A+, B+ and Pi2 boards)

Choose the correct HiFiBerry DAC for your Raspberry PI.

HiFiBerry DAC using the P5 connector

The following example is using the version with a normal audio jack plug and a Raspberry B2. Solder an eight pin header into the Raspberry B. Connect the eight pin socket on the HiFiBerry to the eight pin P5 header on the Raspberry Pi.

See https://www.hifiberry.com/soldering-hifiberry/

Figure 56 Raspberry PI P5 connector

Pin layout of the P5 Connector:

1 - +5V
2 - +3V3
3 - GPIO28 PCM_CLK
4 - GPIO29 PCM_FS
5 - GPIO30 PCM_DIN
6 - GPIO31 PCM_DOUT
7 - Ground
8 - Ground
These GPIOs are not available on the A+/B+/Pi2

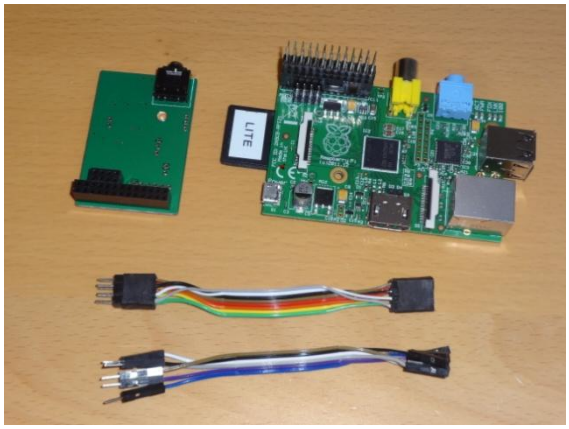


Figure 57 HiFiBerry DAC plus Connector

Figure 58 Pi Radio with HiFiBerry DAC

Create your own P5 connector using an eight Male/Female patch wires as shown in the above left figure. Although not essential it is a good idea to use shrink wrap to keep the patch pins neatly together.

The HifBerry DAC Plus using the 40 Pin Connector

The HiFiBery DAC PLUS uses the 40 pin connector and has an unpopulated 40 pin header to extend the GPIO pins on the HiFiBerry DAC to use with other cards.

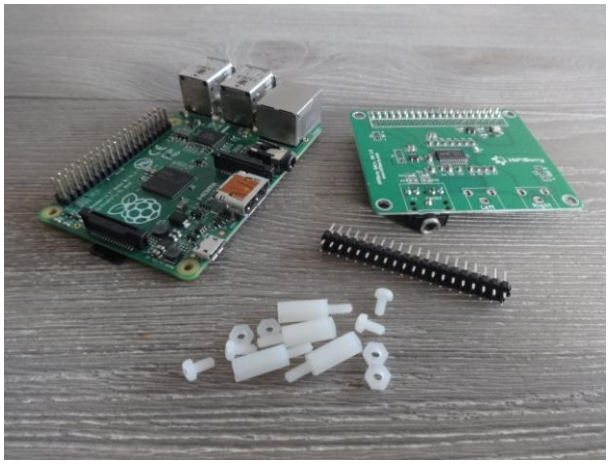


Figure 59 HiFiBerry DAC Plus

The DAC plus uses the 40 pin connector on new Raspberry PIs. A 40 pin dual in line male header is required (purchase separately).

Solder the 40 pin male header into the component side of the HiFiBerry DAC as shown Figure 60 below. The Radio interface card is then mounted on this header.

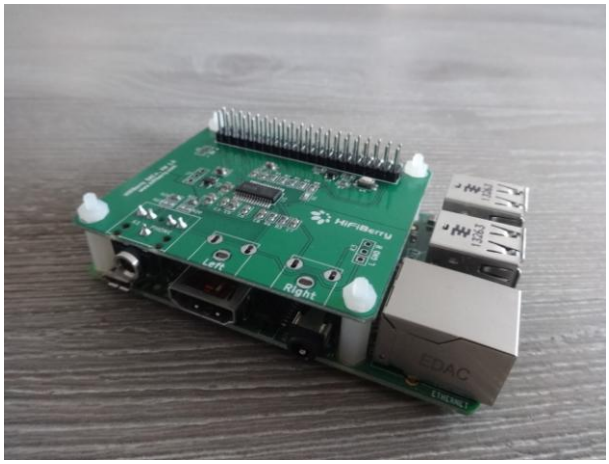


Figure 60 HiFiBerry mounted on the Rasberry Pi

The A+/B+/Pi2 uses the following pins supporting PCM.
Pin 12 GPIO18 PCM_CLK (Conflict!)
Pin 35 GPIO19 PCM_FS
Pin 38 GPIO20 PCM_DIN
Pin 40 GPIO21 PCM_DOUT
(All set to mode ALT0)

Pin 12 (GPIO) conflicts with the down switch on the radio. Wire the down switch to GPIO 10 (Pin 19) and configure the **down_switch=10** parameter in **/etc/radiod.conf**.

Construction using IQAudio products

IQAudio manufacture a comprehensive range of sound devices. See their web site at:
http://www.iqaudio.co.uk/

The wiring for the radio is completely different for these cards as the **IQAudio** products use several GPIOs currently used by the radio software. The radio must be wired as shown in *Table 4 Radio and DAC* devices 40 pin wiring and NOT the 26 pin wiring version shown in Table 3. The **/etc/radiod.conf** configuration file must also be configured to support these devices.

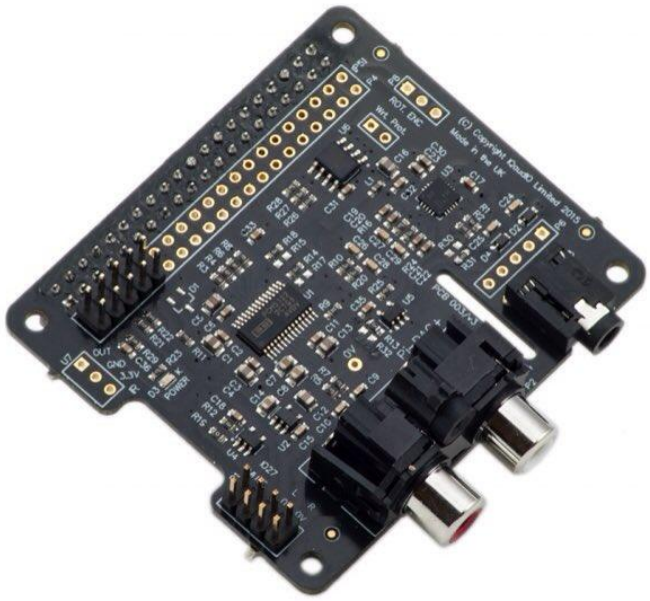


Figure 61 IQAudio DAC plus

The switch and LCD GPIO configuration in **/etc/radiod.conf** must also be changed to match the wiring shown in *Table 4 Radio and DAC devices 40 pin* wiring.

```
# Other switch settings
menu_switch=17
mute_switch=4
up_switch=15
down_switch=14
left_switch=23
right_switch=24

# LCD GPIO connections
lcd_select=7
lcd_enable=8
lcd_data4=5
lcd_data5=6
lcd_data6=12
lcd_data7=13
```

Note: The above configuration is released in a file called **radiod.conf.40_pin**. This can be copied to **/etc/radiod.conf**.

```
$ cd /usr/share/radio
$ sudo cp radiod.conf.40_pin /etc/radiod.conf
```

Conventions used in this tutorial

Installation of the radio program requires you to enter lines at the command line prompt. This requires you to log into the Raspberry PI as user '**pi**'. The default password is **raspberry**.

Note: Don't carry out any of the following commands just yet. They are just examples.

```
Raspberrypi login: pi
Password: raspberry
pi@raspberrypi:~$ Last login: Sun Apr  6 10:18:18 2014 from 192.168.2.100
pi@raspberrypi:~$
```

The prompt line is displayed ending with a **$** sign. The **pi@raspberrypi:~** string means user 'pi' on host machine called 'raspberrypi'. The ~ character means the user 'pi' home directory **/home/pi**. In this tutorial if you are required to do something as user **pi** then only the $ sign will be shown followed by the command as shown in the example below:

```
$ mpc status
```

Some of the commands need to be carried out as user 'root'.
To become root user type in the '**sudo bash**' command:

```
$ sudo bash
root@raspberrypi:/home/pi#
```

Again the prompt shows the username, hostname and current working directory. However only the # followed by the required command will be shown in this tutorial. For example:

```
# apt-get install mpd mpc python-mpd
```

Some commands produce output which does not need to be shown. In such a case a ':' is used to indicate that some output has been omitted.

```
$ aplay -l
**** List of PLAYBACK Hardware Devices ****
: {Omitted output}
card 0: ALSA [bcm2835 ALSA], device 1: bcm2835 ALSA [bcm2835 IEC958/HDMI]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: Device [USB PnP Sound Device], device 0: USB Audio [USB Audio]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
```

System Software installation

SD card creation

Use at least an 8 Gigabyte Card. Create an SD card running the latest version of **Raspbian Jessie** or **Jessie Lite**. This can be downloaded from http://www.raspberrypi.org/downloads. See the *Image Installation Guides* on the above site for instructions on how to install the **Raspbian Jessie** operating system software.

Installation or upgrade on Debian Wheezy

This is no longer supported. Create an SD card using **Rasbian Jessie**.

Log into the system

Boot up the Raspberry Pi with the new SD card with the **Debian Jessie** operating system. If you have used **Jessie** then a graphical desktop will be displayed. Start a terminal session by clicking on the black terminal icon on the top left of the screen. With **Jessie Lite** only a log in prompt will be displayed. In such a case log into the Raspberry Pi as user **pi** and using the password **raspberry**.

Install sysvinit-core

Raspian Jessie now uses **sytemd** for startup processes instead of the more traditional **System V** startup. However the **radiod** package is not yet compatible with **systemd**. For example the radio program causes the Raspberry Pi to hang during a reboot. This is cured by installing **sysvinit-core**. This also cures the incorrect display of the status command. The installation also modifies a file called **/etc/inittab** which is not present in the initial installation of Jessie and is installed with **sysvinit-core**.

```
$ sudo apt-get -y install sysvinit-core
```

See https://en.wikipedia.org/wiki/Systemd for more information about **systemd**.

Note: The **radiod** package is dependent on **sysvinit-core** and will not install or function properly if it is not present.

Reboot the system.

```
$ sudo reboot
```

Online update and upgrade of the Operating System

Once you have installed the operating system login to the system, run **raspi-config**.

```
$ sudo raspi-config
```

Disable booting to the desktop environment

The desktop environment is not required for the Radio and takes a lot of processing power. It is enabled by default in **Jessie** but is not installed with **Jessie Lite**. Unless you have a reason that you want to use it then disable it. Select option **3 Boot options**

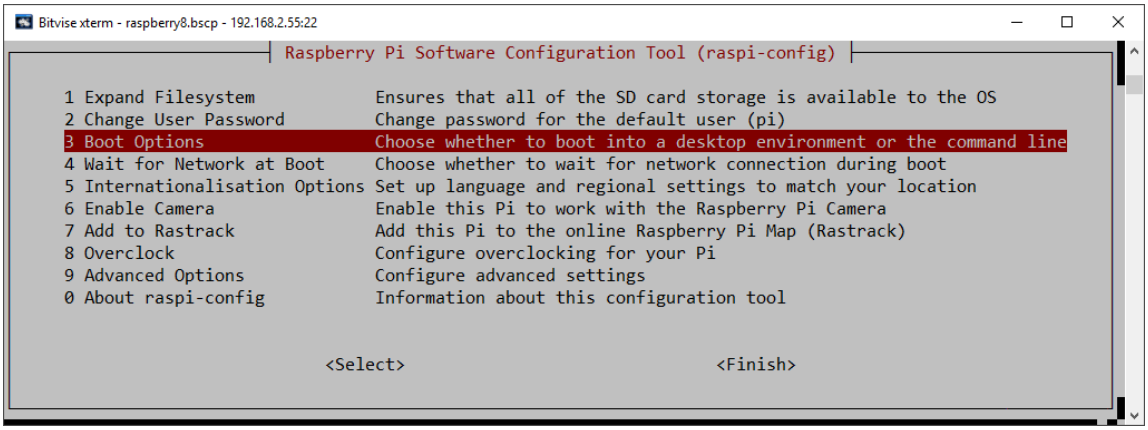
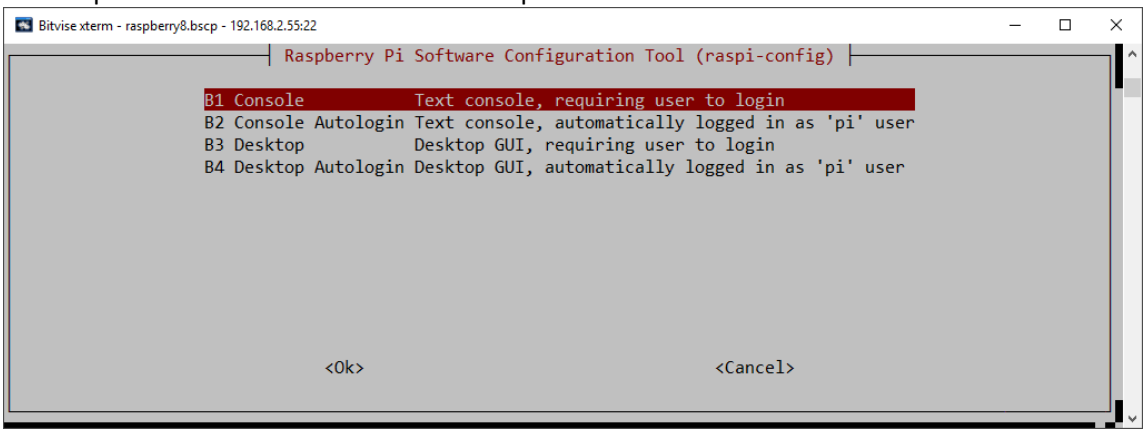


Figure 62 Disabling the graphical desktop

Select option **B1** or **B2** to disable the desktop and select OK



Tab to OK and enter.

Setting the time zone

The **Raspian Jessie** operating system is usually set to UK time. The easiest way to set the time zone for your country if you are in a different timezone is to use the **raspi-config** program.

```
$  sudo raspi-config
```

The following screen is displayed:

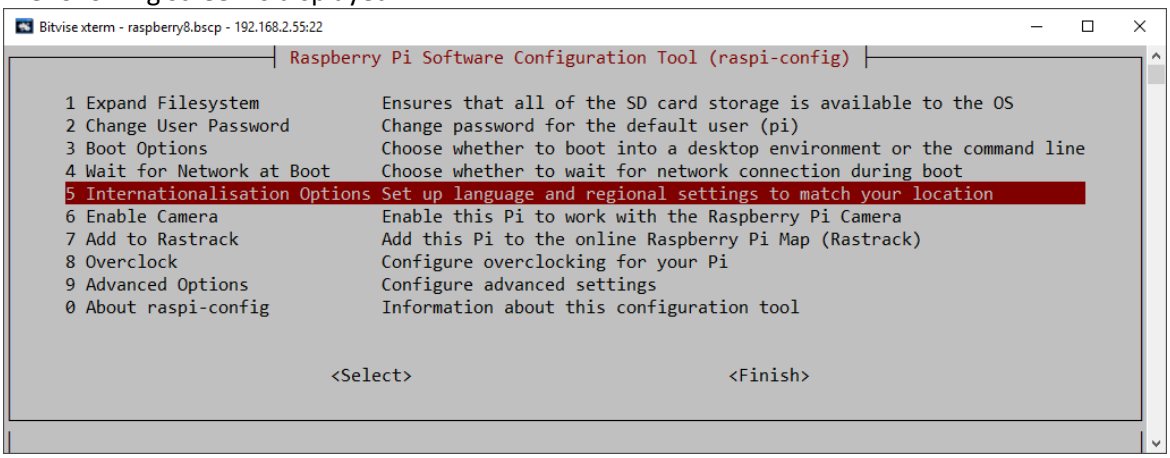


Figure 63 Setting the time zone

Select option 5 “Internationalisation Options”, Use the tab key to move to <Select> and press enter. The above screen is using the Bitvise SSH client program (See Putty on the web).

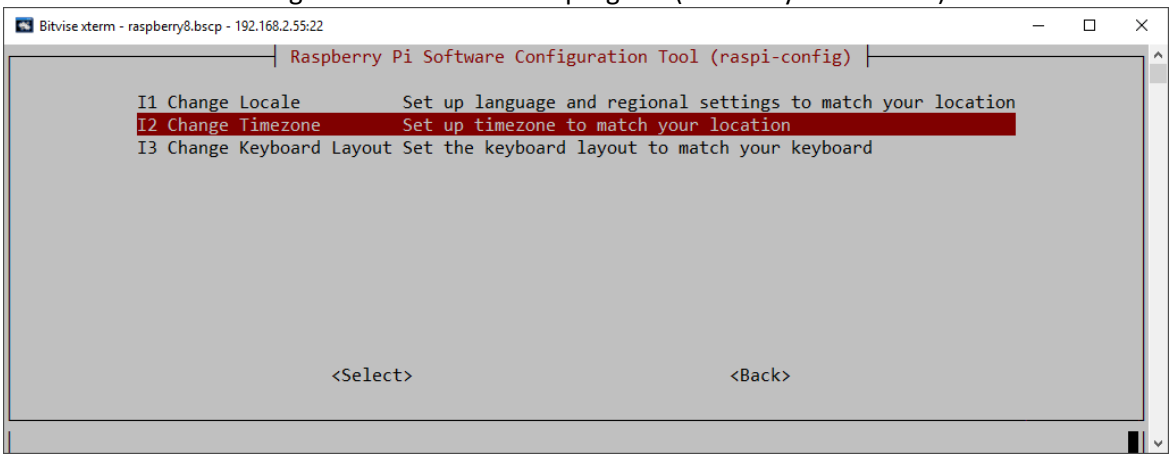


Figure 64 Selecting the time zone

Select the “Change Timezone” option. Again use the tab key to move to <Select> and press enter.

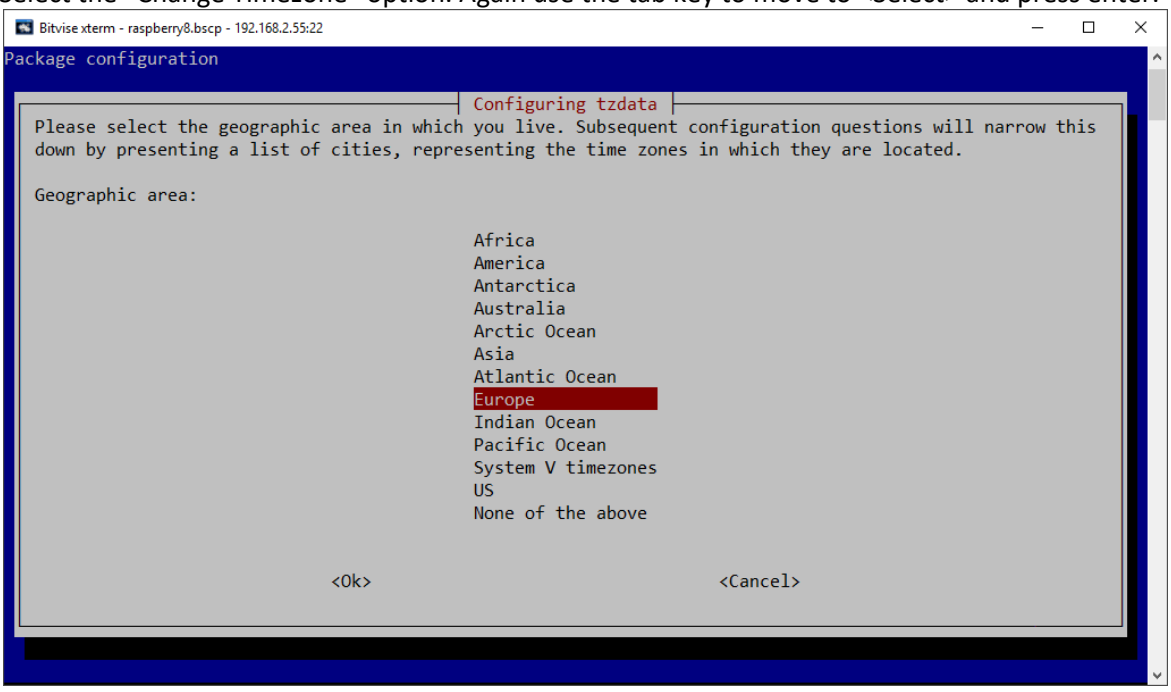
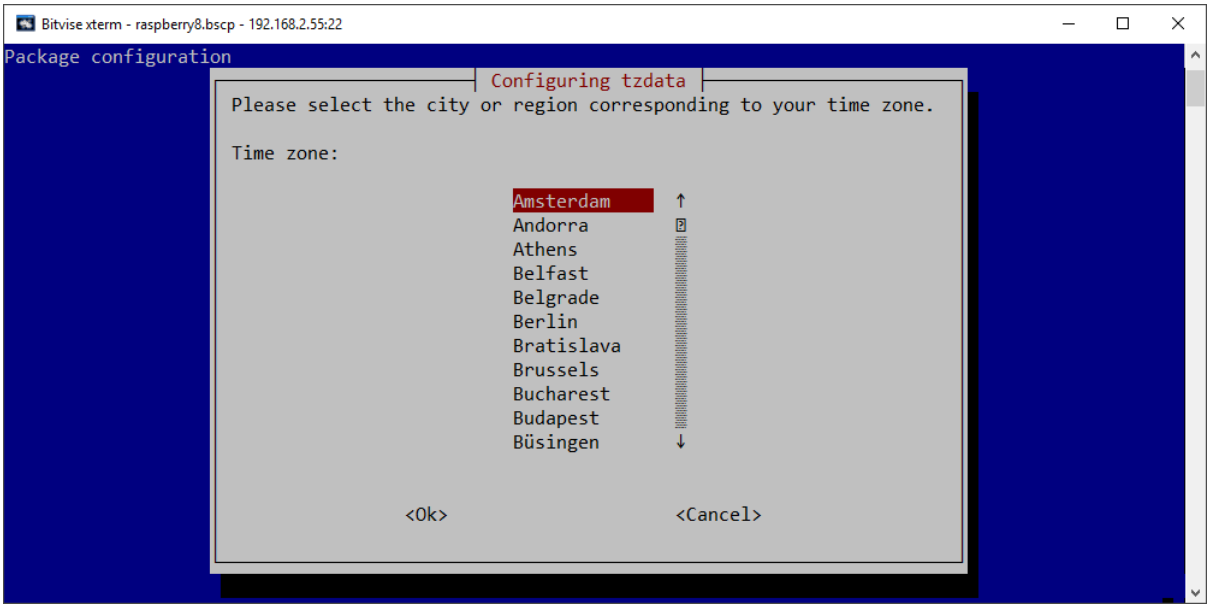


Figure 65 Saving the timezone

Select the region your country is in, Europe for example. Use the tab key to move to <OK> and press enter. The program will then display a list of timezones for the selected region.



Select the correct one and save it by tabbing to <ok> and pressing the enter key. The timezone will be updated. Exit the program once finished.

Changing the system hostname and password

It is a good idea to change the system password for security reasons especially if your raspberry PI is connected to a network with a wireless (WIFI) network. Changing the hostname is also a good idea as it makes identifying your radio on the network much easier. If you wish to do this, change the default hostname from 'raspberrypi' to something like 'piradio' or 'myradio'.
Both the password and hostname can be changed using the raspi-config program.

```
$  sudo raspi-config
```

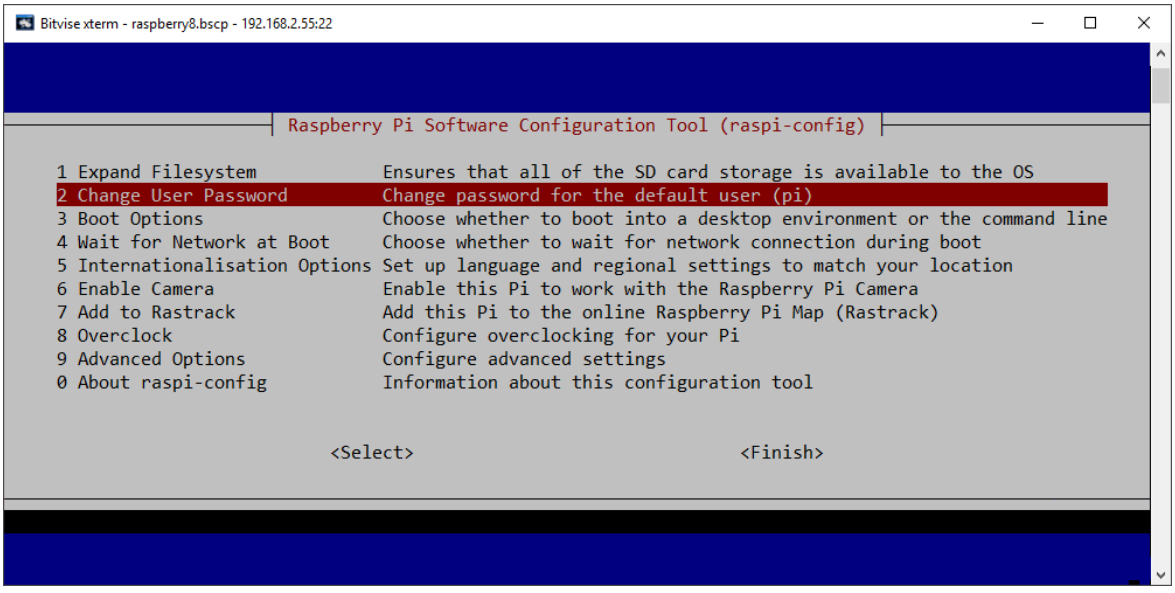


Figure 66 Changing the raspberry PI password

Option 2 is used to change the password. Make sure you record your new password somewhere safe (It is easy to forget it).

The hostname is changed in option 8:

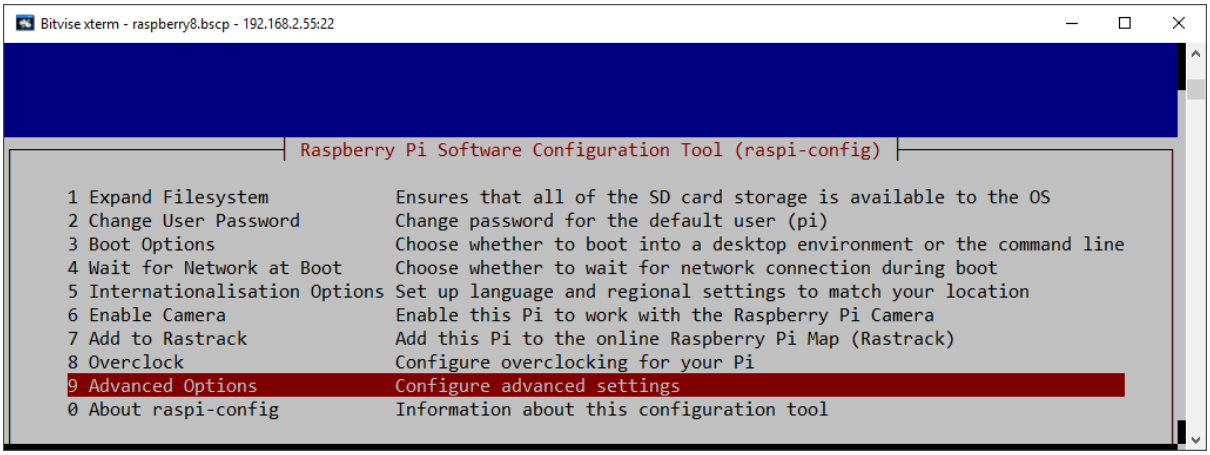


Figure 67 raspi-config advanced options

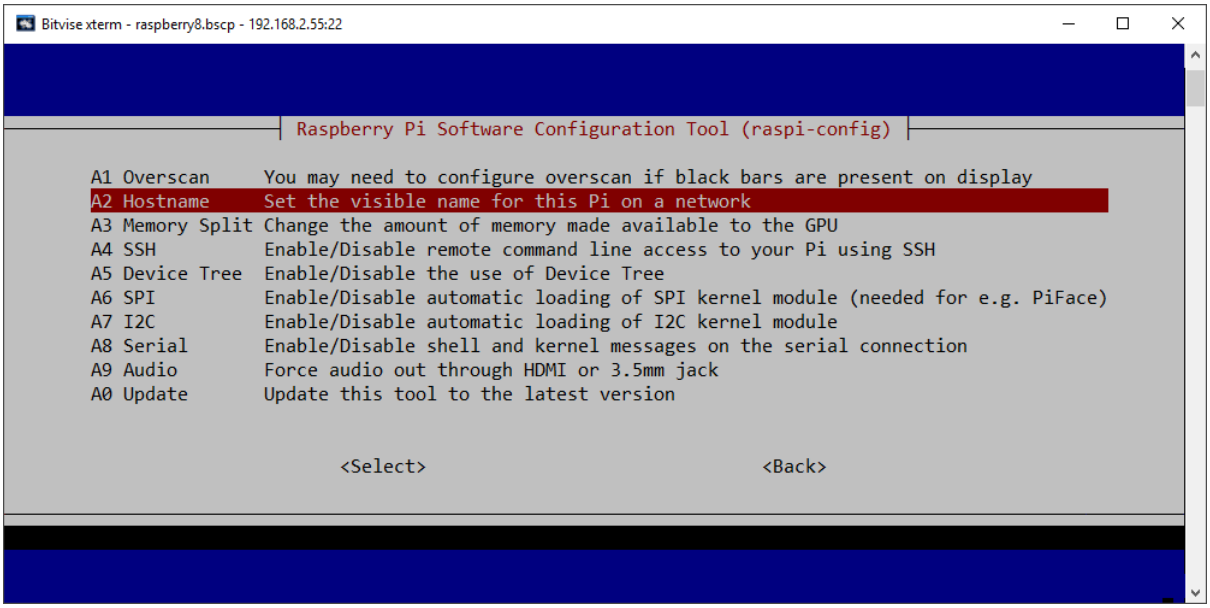


Figure 68 Changing the hostname

```
Enter new UNIX password:
Retype new UNIX password:
```

As the password is entered nothing is displayed. This is normal.

You will be asked if you wish to reboot the system. After you reboot the system you will see the new hostname at the login prompt.

```
pi@piradio:~$
```

In the above example the new hostname is **piradio**.

Installing the radio Software

Upgrading on Debian Wheezy is not supported. You are strongly advised to create a new SD card with the latest version of **Debian Jessie** or **Jessie Lite**. Version 5.0 onwards is a major release which has been heavily modified to work with **Jessie** or **Jessie Lite** and version 0.19 of the Music Player Daemon (MPD). Backward compatibility is not guaranteed in a major release. Also the Web interface has changed along with the way the sound system is now configured. This version is a minor update of 5.x.

If you are upgrading from radio version 4.x you must install everything from scratch.

1) Create a new SD card with **Debian Jessie** or **Jessie Lite** as detailed in the previous section
2) Install the radio package as described in this manual.

Music Player Daemon Installation

All commands should be carried out from the **/home/pi** directory.

If not already carried out do an update of the system. This will take some time.

```
$ sudo apt-get update
```

Note: This is a very important step and if not carried out the installation of MPD may fail.

Reboot the Raspberry PI to apply all of the updates.

```
$ sudo reboot
```

It is possible if the desktop is configured that you may see the following:

```
==== AUTHENTICATING FOR org.freedesktop.login1.reboot ===
Authentication is required for rebooting the system.
Multiple identities can be used for authentication:
 1.  ,,, (pi)
 2.  root
Choose identity to authenticate as (1-2): 1
```

If so choose option 1 and enter the password for user pi (default: raspberry).

After reboot install the Music Player Daemon (mpd) and its client (mpc) along with the Python MPD library.

```
$ sudo apt-get -y install mpd mpc python-mpd
```

Note: If installing on **Jessie Lite** this will take quite a long time as there are hundreds of software libraries to install.

The following errors can be ignored:

```
[....] Starting Music Player Daemon: mpdJun 30 10:07 : server_socket: bind
to '[::1]:6600' failed: Failed to create socket: Address family not
supported by protocol (continuing anyway, because binding to
'127.0.0.1:6600' succeeded)
Jun 30 10:07 : errno: Failed to open /var/lib/mpd/tag_cache: No such file or
directory
```

Note: At this stage there are no playlists configured so the music daemon won't play anything. The playlists are created when the **radiod** Debian Package is installed in the next section.

Install the Radio Daemon

The Raspberry PI Internet Radio software is distributed as a Debian package. This can be downloaded from http://www.bobrathbone.com/pi_radio_source.htm
Either download it to your PC or Macintosh and copy it to the **/home/pi** directory or get it directly using the **wget** facility.

To use the **wget** facility first copy the download link from the above page (Right click on the link). Log into the Raspberry PI. Now use **wget** to the software package:

```
$ wget http://www.bobrathbone.com/raspberrypi/packages/radiod_5.6_armhf.deb
```

Note that the version number shown above will change with later releases so check the download link. If installing for the first time then run **dpkg** to install the package.

```
$ sudo dpkg -i radiod_5.6_armhf.deb
```

Important: If upgrading or reinstalling the **radiod** package you must use the **--force-confmiss** flag.

```
$ sudo dpkg --install --force-confmiss radiod_5.6_armhf.deb
```

The following will be displayed:

```
Selecting previously unselected package radiod.
(Reading database ... 31544 files and directories currently installed.)
Preparing to unpack radiod_5.6_armhf.deb ...
Raspberry PI internet radio installation
Unpacking radiod (5.6) ...
Setting up radiod (5.6) ...
Executing post install script /var/lib/dpkg/info/radiod.postinst
```

The installation program will now ask which daemon should be installed:

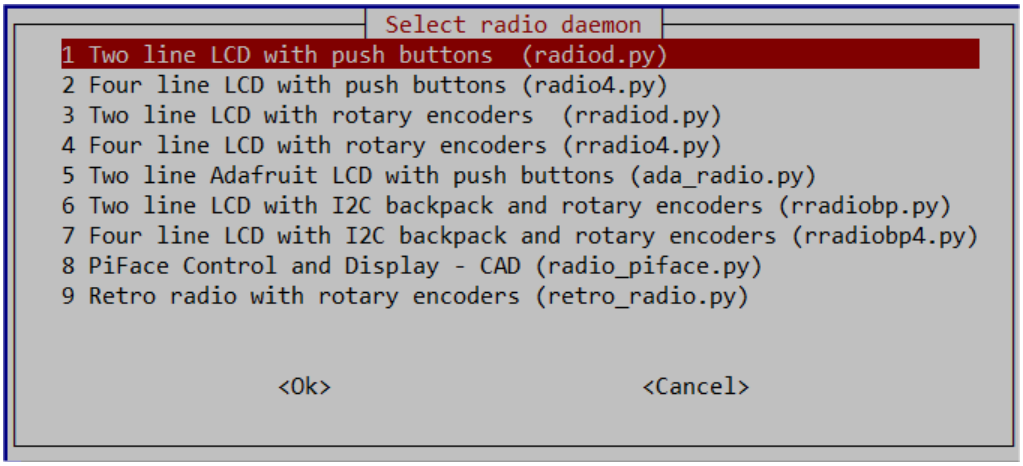


Figure 69 Radio type selection

Select the correct daemon for the radio that you are building. See *Table 1 Radio* variants on page 23. For example:

```
Select version: 5
Daemon ada_radio.py selected
```

```
update-rc.d: using dependency based boot sequencing
insserv: warning: current start runlevel(s) (empty) of script `mpd'
overrides LSB defaults (2 3 4 5).
insserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of script `mpd'
overrides LSB defaults (0 1 6).
insserv: warning: script 'mathkernel' missing LSB tags and overrides
```

Ignore the above **insserv** errors. The program now creates the playlists

```
Creating playlists:
This will take a few minutes
Done: See /usr/share/radio/playlists.log for information about playlists
created
```

The program amends the necessary configuration files. It now creates the playlists. This takes a few minutes. An initial set of playlists are created in the **/var/lib/mpd/playlists** directory. From here on the program output will vary upon the selection you make.

Go to the option you selected on the next pages.

Options 1, 2, 3, 4 or 9

If you selected options 1, 2, 3, 4 or 9 then no other special software is required to be installed.

```
Disabling serial interface in /etc/inittab
Disabling serial interface in /boot/cmdline.txt
Modifying /etc/mpd.conf
PI Radio software successfully installed
See /usr/share/doc/radiod/README for release information

PI Radio software successfully installed
See /usr/share/doc/radiod/README for release information
```

```
It is necessary to reboot the system to start the radio
```

Options 5, 6 or 7

If you selected option 5, 6 or 7 (Adafruit LCD plate or I2C backpack) then the following is displayed:

```
The installation will now run the raspi-config cofiguration program!
When it runs select Advanced options
and enable automatic loading of the I2C kernel module
Enter y to continue or x to exit: y
```

Enter 'y'. The **raspi-config** program now runs. Select 9 "Advanced options" then option A6.

```
      A6 I2C  Enable/Disable automatic loading of I2C kernel module
```

Exit the program. Do not select the option to reboot just yet!

The **raspi-config** program will then enable the I2C module. Finally the following is displayed.

```
It is necessary to install the I2C libraries
Carry out the following command:
   sudo apt-get install python-smbus
and reboot the system
```

After rebooting log back in and enter the following commands to add SMBus support (which includes I2C) to Python:

```
$ sudo apt-get -y install python-smbus
```

If you are using a revision 2 Raspberry Pi (New boards) carry out the following:

```
$ sudo i2cdetect -y 1
```

If you are using a revision 1 Raspberry Pi (Old boards) carry out the following:

```
$ sudo i2cdetect -y 0
```

This will search **/dev/i2c-0** or **/dev/i2c-1** for all address, and if correctly connected, it should show up at **0x20** for the Adafruit LCD Plate or normally **0x27** for the Arduino PCF8574 backpack but might be another address such as **0x3F.** See Figure 70 *The I2C bus display using the i2cdetect program*.

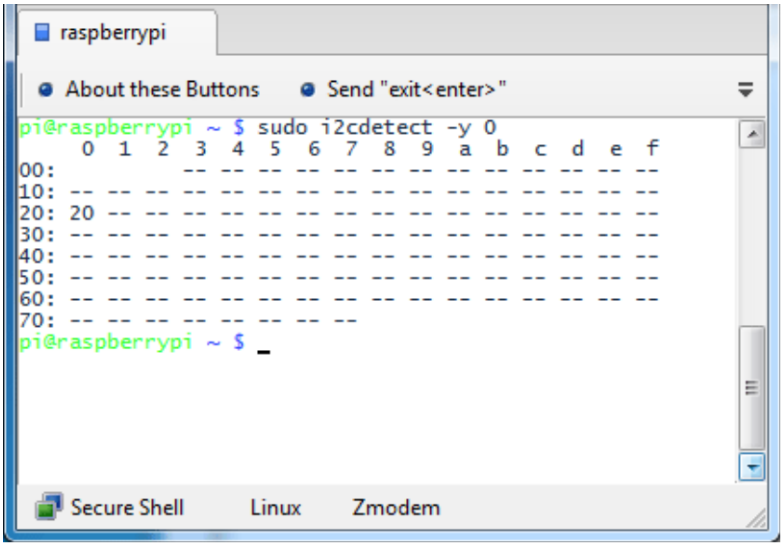


Figure 70 The I2C bus display using the i2cdetect program

Note: If the Arduino PCF8574 backpack is using another address other than **0x27** then you must modify the **i2c_address** parameter in **/etc/radiod.conf**. For example if the backpack is using the address **0x3F** then modify the **i2c_address** parameter to match this as shown in the example below:

```
# The i2c_address parameter overides the default i2c address. 0x00 = use default
# Some backpacks use other addresses such as 0x3F, then set i2c address=0x3F
i2c_address=0x3F
```

Once both of these packages have been installed, you have everything you need to get started accessing I2C and SMBus devices in Python. Now reboot the Raspberry PI.

```
$ sudo reboot
```

The Radio should start automatically. If not then go to the section called *Troubleshooting* on page 113.

Option 8 PiFace CAD

If option 8 was selected the following is displayed:

```
Configured to use PiFace CAD
The installation will now run the raspi-config cofiguration program!
When it runs select Advanced options
and enable automatic loading of the SPI kernel module
Enter y to continue or x to exit: y
```

Enter 'y'. The **raspi-config** program now runs. Select "Advanced options" then option A5 SPI Enable.

```
A5 SPI    Enable/Disable automatic loading of SPI kernel module (needed for
e.g. PiFace)
```

Answer 'yes' to all the questions about enabling and loading the SPI module.
The SPI module will be permanently enabled in the kernel.

If you haven't already done so update the operating system first.

```
$ sudo apt-get update
```

Install **pifacecad** (for Python 3 and 2) with the following command:

```
$ sudo apt-get install python{,3}-pifacecad
```

Now reboot the Raspberry PI.

```
$ sudo reboot
```

The Radio should start automatically.
The instruction following is for the PiFace CAD only. Once rebooted if the LCD doesn't display anything then test the display by running the **sysinfo.py** program. Stop the radio first:

```
$ python3 /usr/share/doc/python3-pifacecad/examples/sysinfo.py
```

This should display some system information on the LCD.

```
$ sudo reboot
```

Configure the audio output

From version 5.4 of the radio onwards the audio output is configured using the **select_audio.sh** utility. This must be run at this stage. This program does the following:

- Selects the audio output device and configures it
- Installs the **sysvinit-core** package if not already installed
- Installs the **alsa-utils** package (Required audio for mixer controls)
- Removes the **pulseaudio** package as it is not compatible with this software
- Configures the volume of the **Alsa** sound mixer for all audio devices
- Saves the **/etc/mpd.conf** configuration file to **/etc/mpd.conf.orig**
- Configure **/etc/mpd.conf** for the selected audio output
- If using a DAC it creates a file called **/etc/asound.conf** to configure a second sound card
- Optionally reboots the Raspberry Pi

Note: Even though the radio may be playing it is still necessary to run the **select_audio.sh** utility. Failure to do so will lead to problems such as no sound.

Run the **select_audio.sh** utility.

```
$ cd /usr/share/radio
$ ./select_audio.sh
```

The following is displayed:

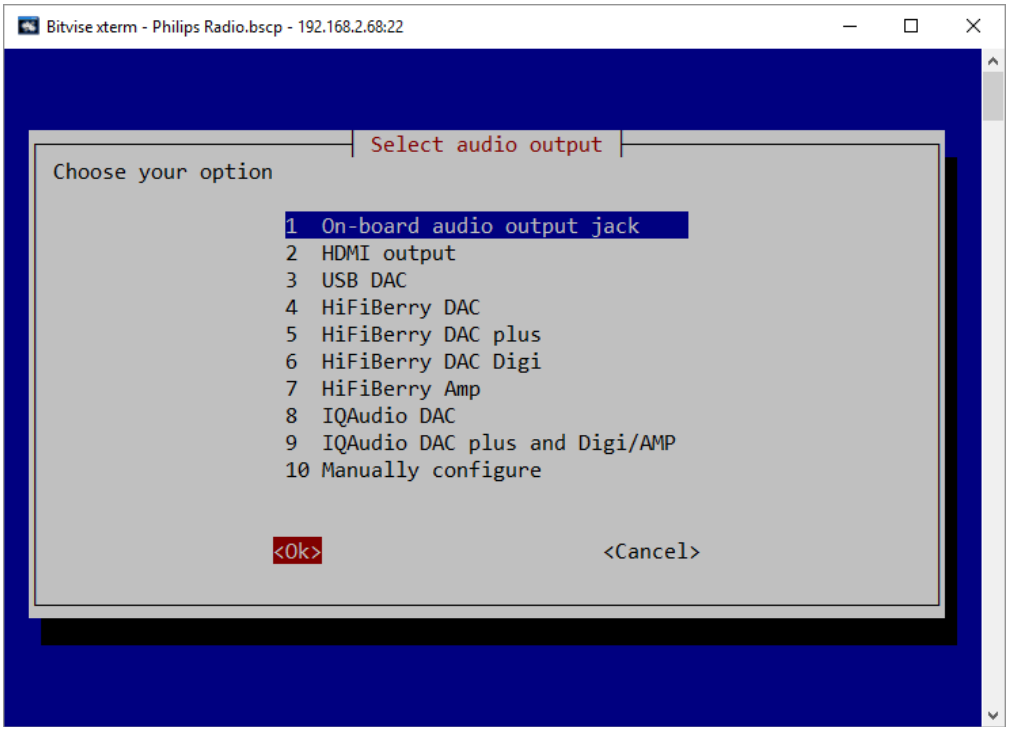


Figure 71 The audio output configuration program

Leave the select ion on option 1 On-board audio output and tab to <OK> and press enter.

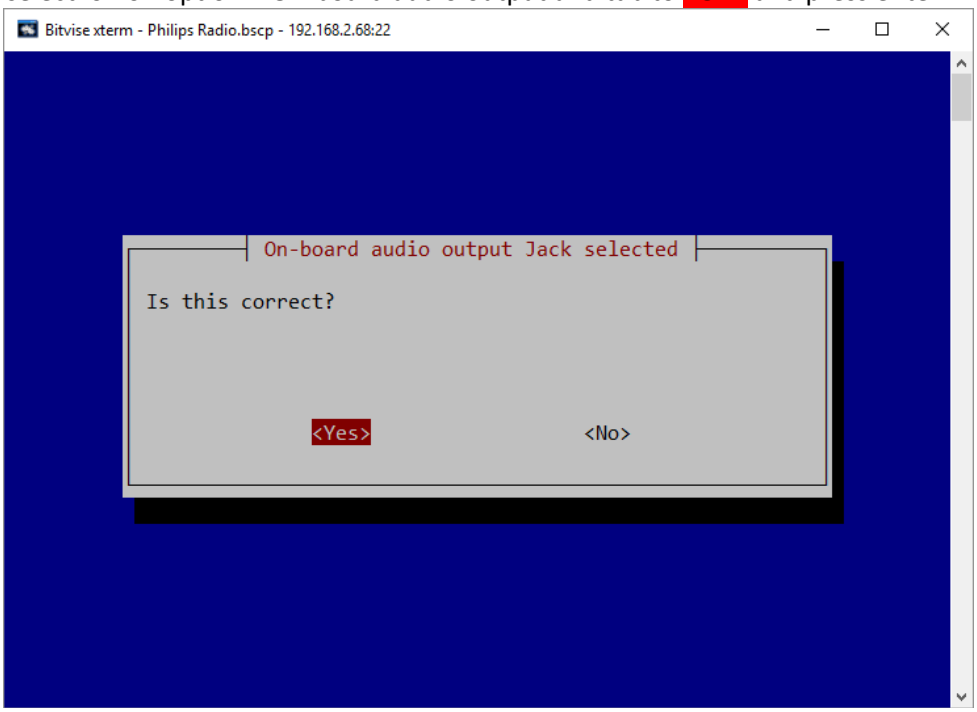


Figure 72 Confirm audio selection screen

Enter to continue. The following will be displayed:

```
On-board audio output Jack selected
:
{ A lot of installation messages}
:
Removing pulseaudio package
Removing redundant packages
```

```
On-board audio output Jack configured
```

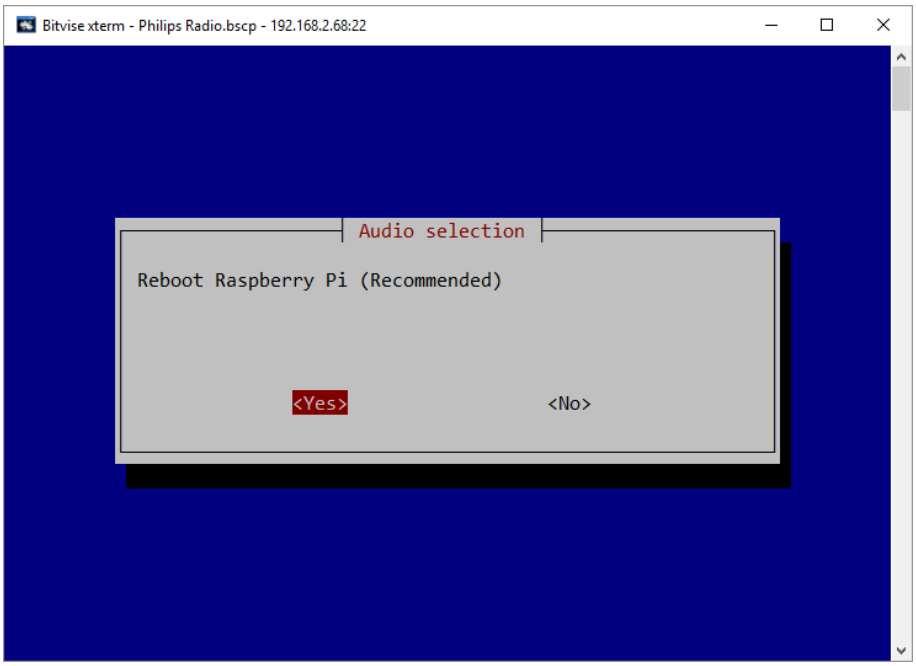


Figure 73 Reboot screen

Enter to reboot the system. If you choose <No> then the following will be displayed.

```
You chose not to reboot!
Changes made will not become active until the next reboot
```

Reboot to enable the software

The software is installed in the **/usr/share/radio** directory. Now reboot the Raspberry PI.

```
$ sudo reboot
```

Once rebooted the software should run and music should be heard out of the on-board audio jack. If not go to the section called Troubleshooting on page 113.

The radio daemon can be started and stopped with the **service** command:

```
$ sudo service radiod start
$ sudo service radiod stop
```

This will also stop and start the MPD daemon.

To prevent automatic start-up of the radio at boot time run the following command:

```
$ sudo update-rc.d radiod disable
```

To re-enable it:

```
$ sudo update-rc.d radiod enable
```

Setting the mixer volume

All sound goes through a mixer. Most sound cards have their own mixer. After rebooting the Raspberry Pi run the **alsamixer** program without any additional parameters:

```
$ alsamixer
```

The following screen is displayed:

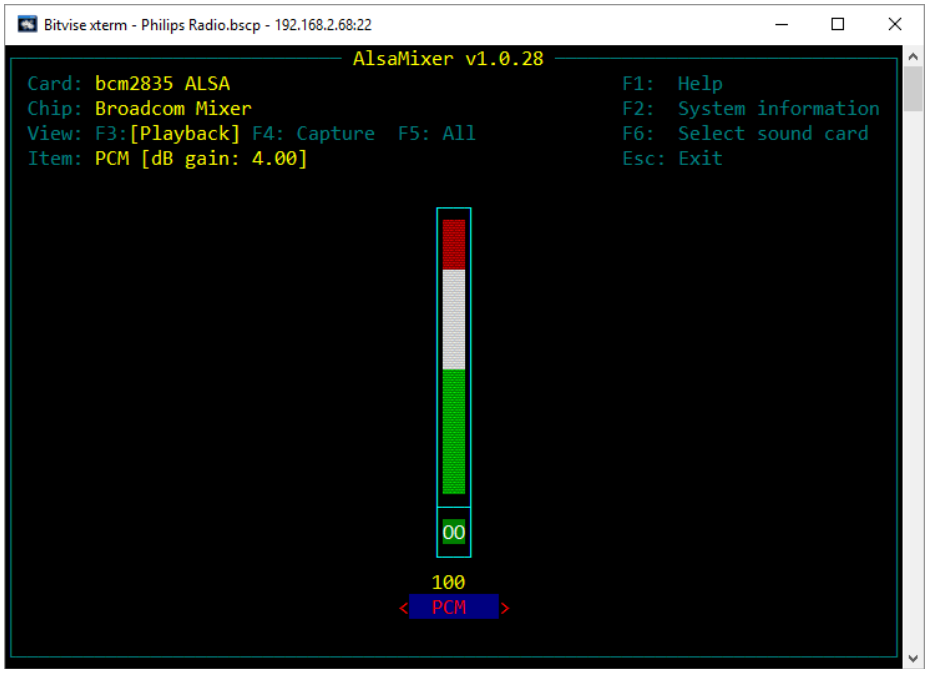


Figure 74 Basic Alsa sound mixer

The above illustration shows the **bcm2835** Alsa Mixer. There is only one mixer control called PCM (Pulse Code Modulated). Adjust the volume to 100% if not already set by using the Up and Down keys on the keyboard. Press the Esc key or Ctl Z to exit the program.

It is also possible to set the volume for the on-board mixer volume with the **amixer** program.

```
$ amixer cset numid=1 100%
numid=1,iface=MIXER,name='PCM Playback Volume'
  ; type=INTEGER,access=rw---R--,values=1,min=-10239,max=400,step=0
  : values=400
  | dBscale-min=-102.39dB,step=0.01dB,mute=1
```

Configuring other sound devices

Other sound devices can be used with the radio. Currently supported are the following devices:

- CMedia USB speakers or devices (See page 66)
- New HiFiBerry DAC and DAC+ products (See page 68)
- IQ Audio products (See page 70)

To check if the audio device is present run the **aplay** command.

```
$ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: ALSA [bcm2835 ALSA], device 0: bcm2835 ALSA [bcm2835 ALSA]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 0: ALSA [bcm2835 ALSA], device 1: bcm2835 ALSA [bcm2835 IEC958/HDMI]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: Device [USB PnP Sound Device], device 0: USB Audio [USB Audio]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
```

In the above example **Card 0** is the on-board audio output jack. **Card 1** is a USB PnP sound device.

To configure other sound device run the **select_audio.sh** utility.

```
$ cd /usr/share/radio
$ ./select_audio.sh
```

Configuring a USB sound devices

To configure a USB DAC sound devices such as CMedia speakers or sound dongles run the **select_audio.sh** utility.

To configure USB audio devices run the Run the **select_audio.sh** utility.

```
$ cd /usr/share/radio
$ ./select_audio.sh
```

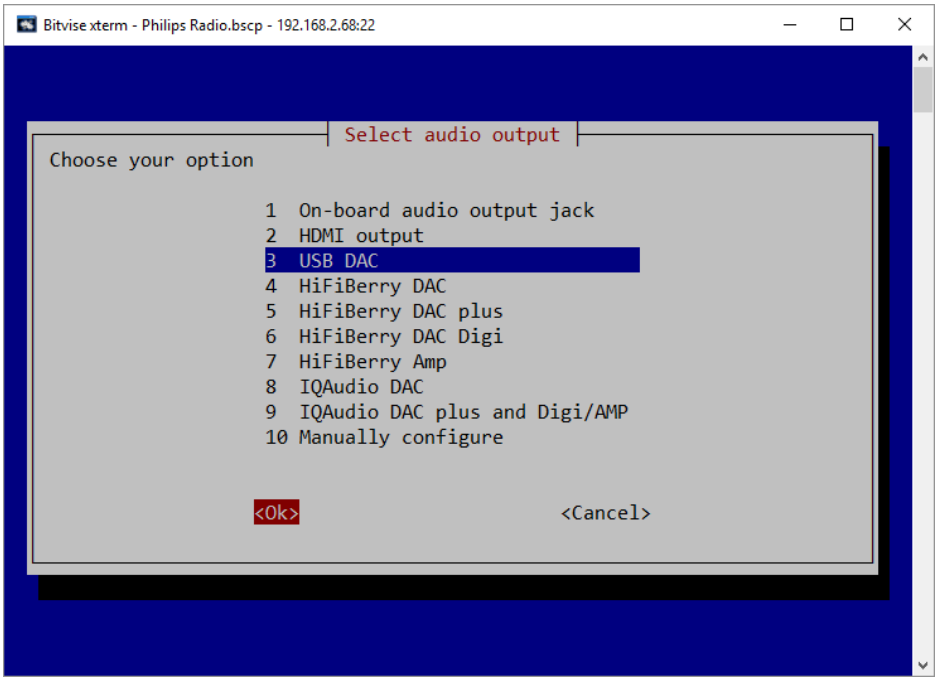


Figure 75 USB DAC selection

Select option 3 USB DAC. Reboot when prompted. After rebooting the Raspberry Pi run the **alsamixer** program specifying **Card 1** with the **-c1** parameter.

```
$ alsamixer -c1
```

The following screen is displayed:

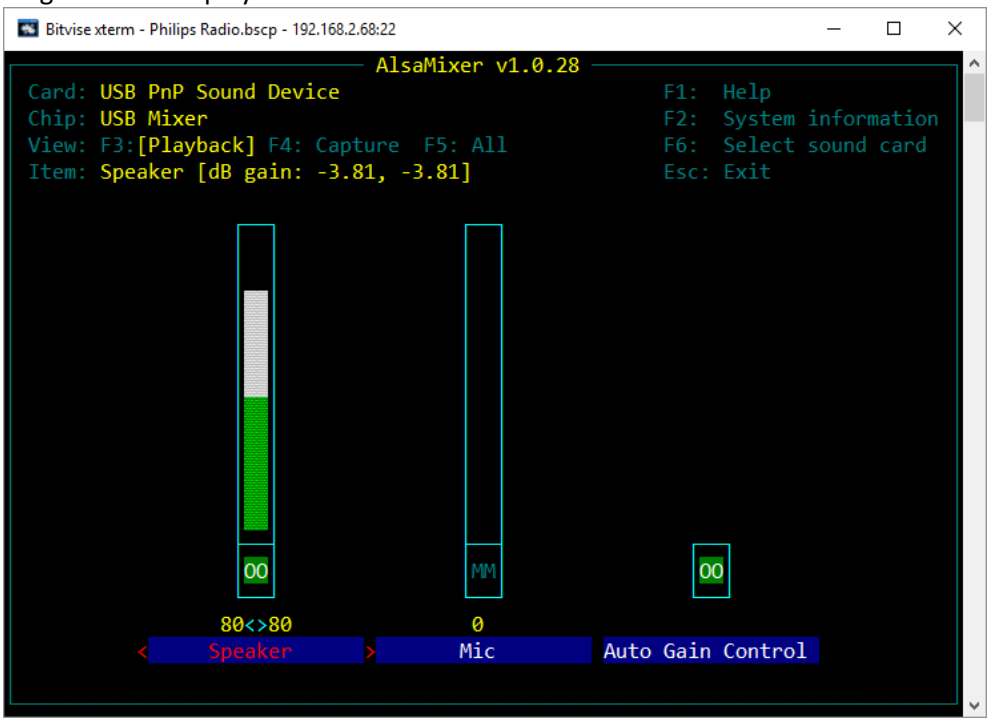


Figure 76 The USB PnP Alsa Mixer

Use the Left and Right keys to position on the 'Speaker field'. Adjust the sound level using the Up and Down keys (80% in the above example). Pres **Esc key** or **Ctl Z** key to exit.

Configuring a HiFiBerry products

Older versions of the HiFiBerry DAC that used the 26 pin GPIO header are not supported.
There are two basic versions of the DAC which have been tested with the Radio software:

- The DAC/DAC+ Light which uses the P5 connector
- The DAC+ standard/pro/amp which uses the 40 pin connector

To configure the HiFiBerry range of audio cards run the Run the **select_audio.sh** utility.

```
$ cd /usr/share/radio
$ ./select_audio.sh
```

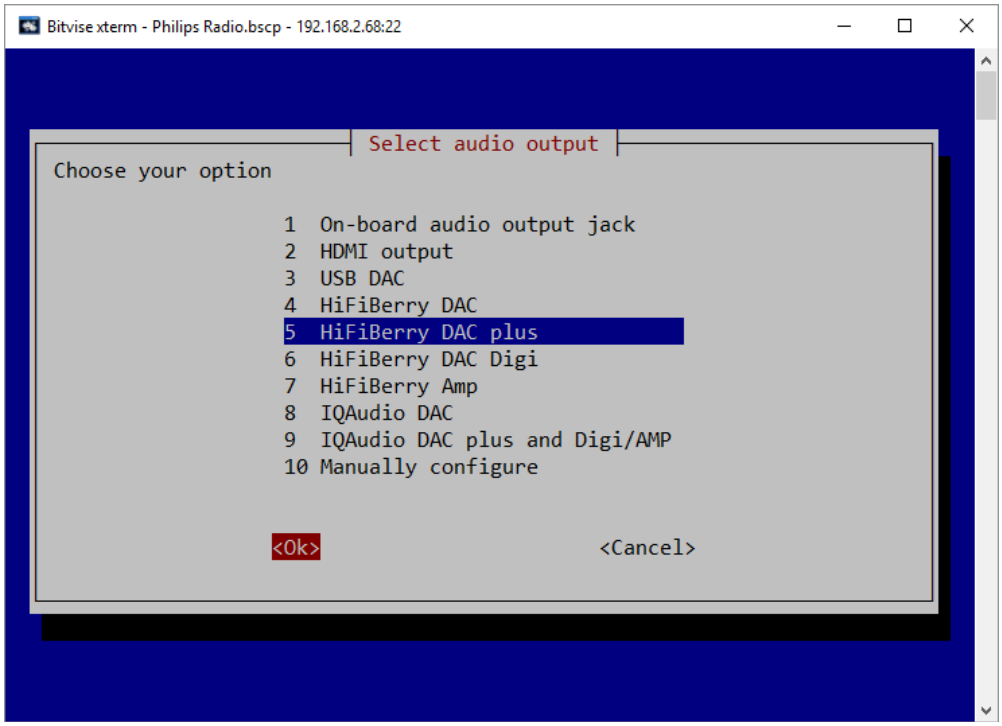


Figure 77 Selecting HiFiBerry products

Select option 4,5,6 or 7 depending upon the HiFiBerry device being used an press OK.

Reboot when prompted by the next screen.

After rebooting run the **alsamixer** program specifying **Card 1** with the **-c1** parameter to select card 1.

```
$ alsamixer -c1
```

Use the left and right keys to select the mixer control (Analogue) and use the up down keys to change the volume to 100%.

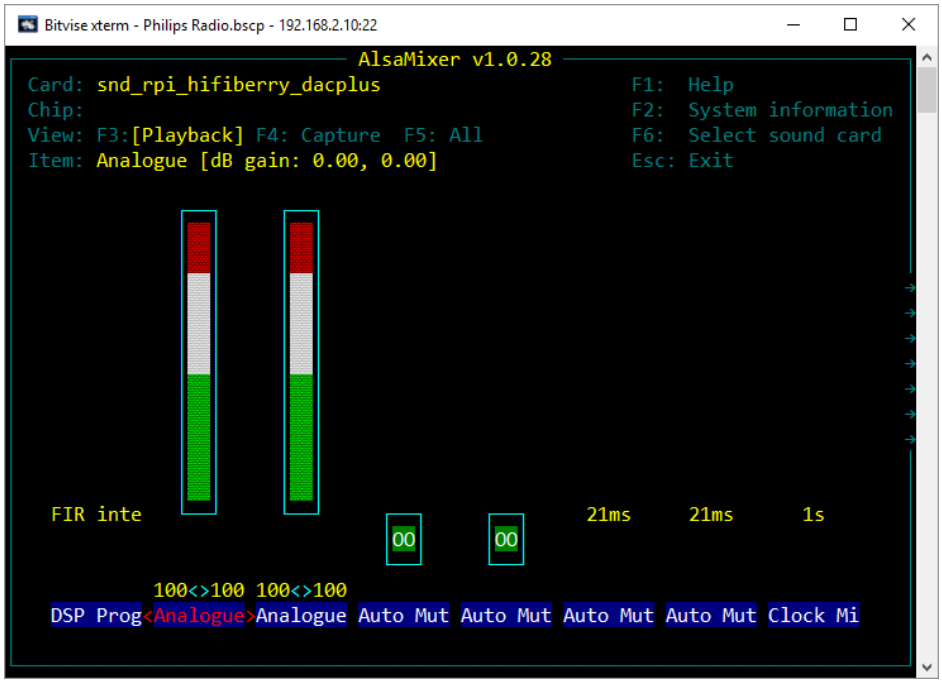


Figure 78 Set mixer analogue volume

Next use the right key to position on the “Digital” mixer control and use the up down keys to change the mixer volume:

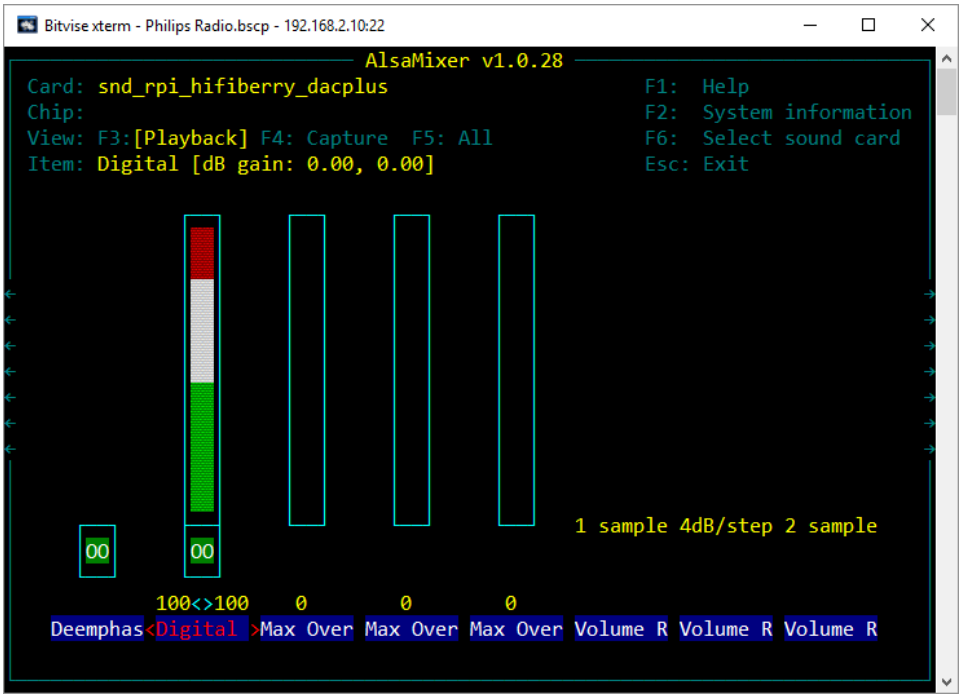
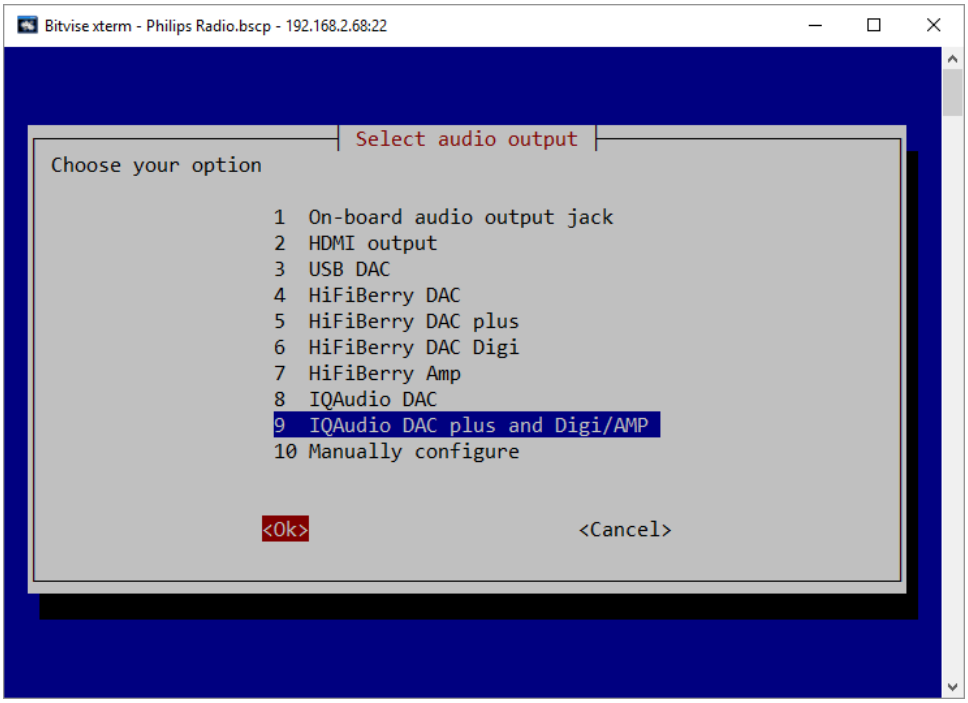


Figure 79 Set mixer digital volume

Configuring IQAudio sound cards

Run the select_audio.sh utility as previously shown:



Select option 8 or 9 depending upon the IQ product fitted. Reboot when prompted.
After rebooting run the **alsamixer** program specifying **Card 1** with the **-c1** parameter to select card 1.

```
$ alsamixer -c1
```

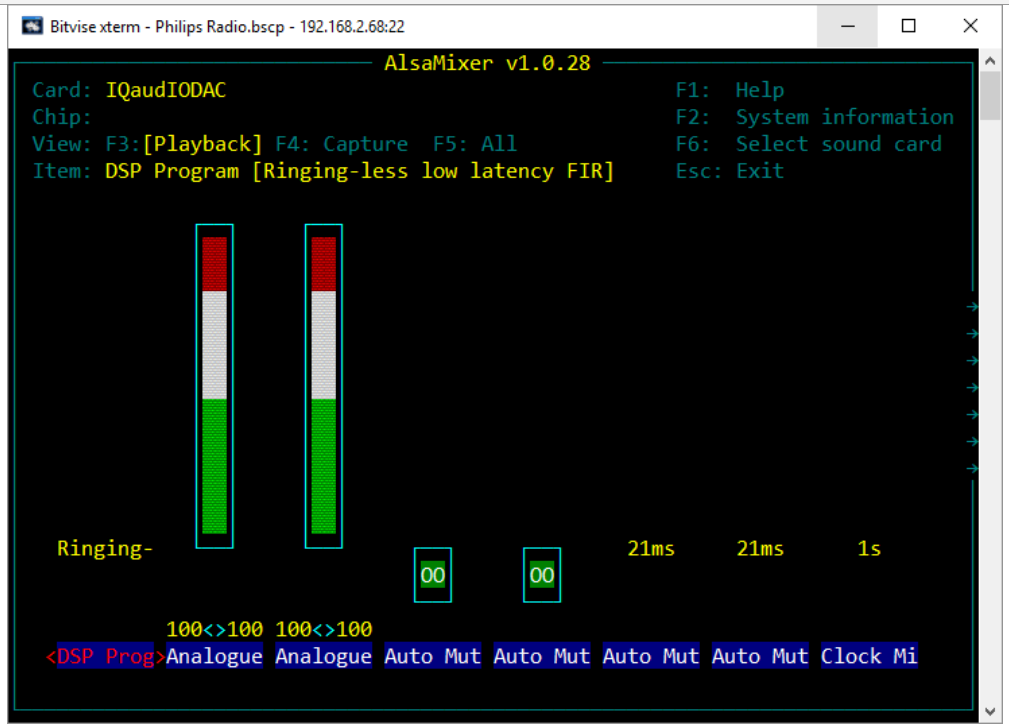
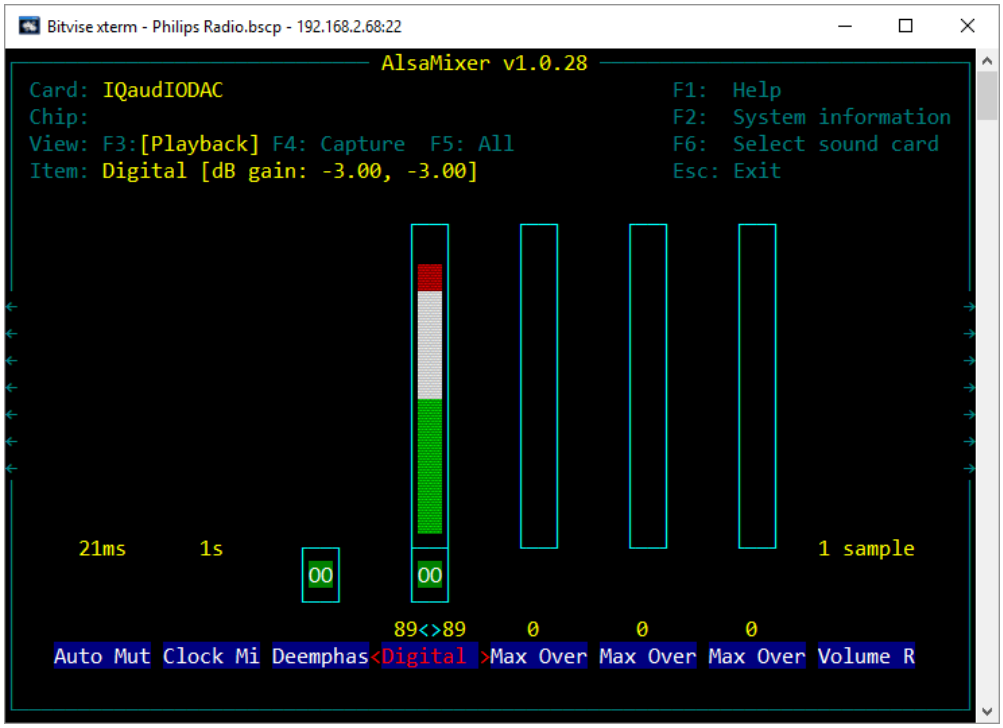


Figure 80 IQAudio sound mixer

Set the Analogue mixer control to 100%. Next use the right key to position on the “Digital” mixer control and use the up down keys to change the mixer volume for the Digital mixer control:



Testing the Music Player Daemon MPD

This section provides useful information on the operation of the Music Player Daemon (MPD) and its client (MPC) or diagnostics if no music is heard when the Radio is started.

If no music is being heard check the status of MPD:

```
$ sudo service mpd status
mpd is running.
```

If the following is seen:

```
$ sudo service mpd status
mpd is not running ... failed!
```

Start the MPD daemon.

```
$ sudo service mpd start
Starting Music Player Daemon: mpd.
```

If no music is heard check that there are playlists configured using the music player client **mpc playlist** command (sudo isn’t necessary):

```
$ mpc playlist
Nashville FM
RAI Radio Uno
```

```
RAI Radio Duo
Prima Radio Napoli
Radio 1 Nederland
:
```

If no playlists are shown run the **create_m3u.py** program as shown in the section called *Creating playlists* on 90.

However before doing so check that the **use_playlist_extension** option is correctly set in **/etc/radiod.conf**:

```
# MPD version 0.16 requires mpc load command to use the playlist extension
# so set to yes, MPD version 0.19 does not use the playlist extension
# so set to no
use_playlist_extensions=no
```

Manually configuring sound cards

Unless you have a need to manually configure some other sound card or need to troubleshoot a non-working card you can skip this section. Configuring a HiFiBerry DAC is shown in this example

Edit the **/boot/config.txt** and add the following line to the end of the file depending upon the version you are using.

```
dtoverlay=hifiberry-dacplus
```

See https://www.hifiberry.com/guides/configuring-linux-3-18-x/ for other devices.

Modify the **audio_output** section in **/etc/mpd.conf** to support the HiFiBerry DAC and software mixer.

```
audio_output {
   type       "alsa"
   name       "HiFiBerry DAC"
   device     "hw:1,0"
#  mixer_type    "hardware"
   mixer_type    "software"
#  mixer_device  "default"
#  mixer_control "PCM"
#  mixer_index=0

}
```

Reboot the Raspberry PI.

```
$ sudo reboot
```

If no music is heard run the **alsamixer** program and set the volume to at least 80% as shown in the previous section on **HiFiBerry** devices.

Installing the Infra Red sensor software

This section is applicable to both the PiFace CAD IR sensor or a separate IR and sensor and activity LED. You must first install the PiFace software as shown in the previous section.
You will need to configure the IR receiver yourself. More examples can be found in **/usr/share/doc/python3-pifacecad/examples/** (which may need unzipping using **gunzip**). The full instructions for setting up the IR software are available from:
http://piface.github.io/pifacecad/lirc.html#setting-up-the-infrared-receiver.
Or
http://www.piface.org.uk/guides/setting_up_pifacecad/setting_up_PiFace_CAD_to_use_a_remote/

If you haven't already done so update the operating system first.

```
$ sudo apt-get update
```

Install **pifacecad** (for Python 3 and 2) with the following command:

```
$ sudo apt-get -y install python{,3}-pifacecad
```

Run the **setup_pifacecad_lirc.sh** program

```
$ sudo bash
# cd /usr/share/radio
```

Now get the latest **piface** setup program and make it executable.

```
# mv setup pifacecad lirc.sh setup pifacecad lirc.sh.save
# wget https://raw.github.com/piface/pifacecad/master/bin/setup pifacecad lirc.sh
# chmod +x setup_pifacecad_lirc.sh
```

Now run it:

```
# ./setup_pifacecad_lirc.sh
This script will overwrite the following files:
/etc/modules
/boot/config.txt
Do you wish to continue?
1) Yes
2) No
```

Select option 1

```
Installing LIRC.
.
.
Now you must create an /etc/lirc/lircd.conf for your remote control.
Either download one from here:

    http://lirc.sourceforge.net/remotes/

Or generate one yourself with the following command:

    sudo irrecord -f -d /dev/lirc0 /etc/lirc/lircd.conf
```

If you are using the PiFace CAD software then go to the copy **lircrc.dist** file below:

Table 9 IR Sensor Pin outs

Radio Type	Pin	GPIO	Type of Raspberry PI
PI Face Cad (IR sensor already wired)	16	23	Any
Two or Four line LCD with Push Buttons	21	9	Any
Two or Four line LCD with Rotary encoders	21	9	Any
Two or Four line LCD with I2C backpack	21	9	Any
Adafruit RGB plate with push buttons	36	16	40 pin version only
All versions using IQAudio products	22	25	40 pin version only

If using the separate IR sensor and activity LED configure lirc-rpi in the **/boot/config.txt** file with the lirc-rpi device details. Edit the **/boot/config.txt** file and change the **gpio_in_pin** parameter from:

```
dtoverlay=lirc-rpi,gpio_in_pin=23,gpio_in_pull=high
```

to the GPIO number shown in Table 9 above.

For all radio versions except the Adafruit plate or PiFace CAD:

```
dtoverlay=lirc-rpi,gpio_in_pin=9,gpio_in_pull=high
```

Do this for the AdaFruit plate.

```
dtoverlay=lirc-rpi,gpio_in_pin=16,gpio_in_pull=high
```

For 40 pin versions using IQAudio products.

```
dtoverlay=lirc-rpi,gpio_in_pin=25,gpio_in_pull=high
```

Save the **/boot/config.txt**.
Now copy the **lircrc.dist** file to **/etc/lirc/lircrc**

```
$ cd /usr/share/radio
$ sudo cp lircrc.dist /etc/lirc/lircrc
```

Reboot the Raspberry Pi

```
$ sudo reboot
```

Now test your remote control with the mode2 command. You should see lots of codes when you press buttons on the remote control.

```
$ sudo mode2 -d /dev/lirc0
space 5358604
pulse 4591
space 4459
pulse 651
```

```
.
.
```

Check that the **lirc0** device exists. If it doesn't exist you cannot continue. Check that the **/boot/config.txt** file has been correctly configured as shown above.

```
$ ls -la /dev/lirc0
crw-rw---T 1 root video 246, 0 Jan  1  1970 /dev/lirc0
```

It is now necessary generate a configuration for the remote control then first move the distributed **/etc/lirc/lircd.conf** file out of the way.

```
$ sudo mv /etc/lirc/lircd.conf /etc/lirc/lircd.conf.org
```

Now run the configuration program.

```
$ sudo irrecord -f -d /dev/lirc0 /etc/lirc/lircd.conf
```

If you see the following

```
irrecord: could not open /dev/lirc0
irrecord: default_init(): Device or resource busy
irrecord: could not init hardware (lircd running ? --> close it, check
permissions)
```

Run the following command:

```
$ sudo service lirc stop
```

If the problem persists, make sure the /boot/config.txt file has been correctly set up as previously shown and that a reboot was carried out.

Follow the instructions in the **irrecord** program exactly!

The **irrecord** program will ask you for the names of the buttons that you want to configure. You may not make your own names up. You must use the names shown in the first column of the following table and which are defined in **/etc/lirc/lircrc.**

It is a good idea to just start with the basic keys for volume up and channel change and when you have the remote control working re-configure with all of the keys shown in *Table 10 Remote Control Key names and functions*.

Table 10 Remote Control Key names and functions

Key Names	Normal	Search	Source	Options
KEY_VOLUMEUP	Volume up	Volume up	Volume up	Volume up
KEY_VOLUMEDOWN	Volume down	Volume down	Volume down	Volume down
KEY_CHANNELUP	Channel up	Channel up	Channel up	Channel up
KEY_CHANNELDOWN	Channel down	Channel down	Channel down	Channel down
KEY_MUTE	Mute sound	Mute sound	Mute sound	Mute sound
KEY_MENU	Step menu	Play selected	Load tracks/stations	Next menu
KEY_UP	Not used	Previous artist	Toggle source	Previous option
KEY_DOWN	Not used	Next artist	Toggle source	Next option
KEY_LEFT	Not used	Track up	Not used	Toggle option
KEY_RIGHT	Not used	Track down	Not used	Toggle option
KEY_OK	Step menu	Play selected	Load tracks/stations	Next menu
KEY_LANGUAGE*	Voice on/off	Voice on/off	Voice on/off	Voice on/off
KEY_INFO*	Speak info	Speak info	Speak info	Speak info

* Only used if speech (espeak) is implemented for visually impaired persons.

Note: The **KEY_OK** and **KEY_MENU** do the same thing.

The actual list of available names that may be used can be displayed with the following command:

```
$ sudo irrecord --list-namespace
```

There are more than 440 key names but only use the ones defined in the list above.

Start the radio software if not already running.

```
$ sudo service radiod start
```

Configure the **pifacercd** daemon to start at boot time and start it

```
$ sudo update-rc.d -f pifacercd defaults
$ sudo service pifacercd start
```

The activity LED will flash a few times however it is necessary to reboot the system to enable the new IR remote configuration.

```
$ sudo reboot
```

Now test the remote control with your newly configured remote control.

Testing the remote control

If there are problems with the remote control you can test using **ircat** which will display the key codes that have been programmed. Stop the pifacercd daemon first and restart service lirc.

```
$ sudo service pifacercd stop
$ sudo service lirc start
```

```
Loading LIRC modules:.
Starting remote control daemon(s) : LIRC :.
Starting execution daemon: irexec:.
```

Now run **ircat** and press each key on the remote control in turn:

```
$ ircat piradio
KEY_VOLUMEUP
KEY_VOLUMEDOWN
KEY_CHANNELUP
KEY_CHANNELDOWN
KEY_MENU
KEY_MUTE
KEY_UP
KEY_RIGHT
KEY_LEFT
KEY_DOWN
KEY_LANGUAGE
KEY_INFO
```

Use Ctrl-C to exit. If keys are not responding repeat the previous Remote Control installation procedure.

Check that the radio is listening on UDP port 5100 (or as configured in /etc/radiod.conf).

```
$ sudo netstat -an | grep 5100
udp        0      0 127.0.0.1:5100          0.0.0.0:*
```

Disabling the repeat on the volume control

If you wish to disable the repeat on the volume control the edit the **/etc/lirc/lircrc** file, set **repeat = 0**, for KEY_VOLUMEUP and KEY_VOLUMEDOWN definitions.

```
begin
  prog = piradio
  button = KEY_VOLUMEUP
  config = KEY_VOLUMEUP
  repeat = 0
end

begin
  prog = piradio
  button = KEY_VOLUMEDOWN
  config = KEY_VOLUMEDOWN
  repeat = 0
end
```

Configuring GPIO outputs

Apart from changing the **down_switch** GPIO setting to be compatible with the **HiFiBerry DAC** it is not normally necessary to change the GPIO settings for the switches, rotary encoders or LCD display connections. The default settings match the wiring configuration shown in Table 3 on page 27. Unless here is a need to change the GPIO configuration skip this section.

From version 5.4 onwards the GPIO assignments for all switches, rotary encoders and LCD display settings are configurable in the **/etc/radiod.conf** file.

If the GPIO assignments are changed in the **/etc/radiod.conf** file then these must match the actual physical wiring for your radio project.

Switches and rotary encoders GPIO assignments

The default switch settings including the rotary encoders are shown below. As previously mentioned the only setting that normally might need to be changed is the **down_switch** setting.

```
# Down switch GPIO setting, if using a HiFiberry DAC set to 10
down_switch=18

# Other switch settings
menu_switch=25
mute_switch=4
up_switch=17
left_switch=14
right_switch=15
```

LCD display GPIO assignments

The default LCD settings are shown below. Normally there is no need to change these unless using **IQAudio** products. The lcd_data5 parameter conflicts with IqAudio Amp mute function. Change to use GPIO10 (lcd_data5=6).

```
# LCD GPIO connections
lcd_select=7
lcd_enable=8
lcd_data4=27
lcd_data5=22
lcd_data6=23
lcd_data7=24
```

Configuring the remote control activity LED

It is useful to have an activity LED which flashes every time the remote control is pressed. How to wire the activity LED is shown on page 46. Which pins you connect to will depend on the type of radio you are building. Table 8 on page 46 shows the required LED connections. Boards such as the **PiFace CAD** and the **AdaFruit RGB plate** will need a 40 pin Raspberry PI as all the first 26 pins are occupied but the plug in card.

Configure the LED in **/etc/radiod.conf** for to pin 11 GPIO 23 for all versions except the PiFace CAD, AdaFruit plate or Vintage radio. For PiFace CAD or Adafruit RGB plate configure either **remote_led=0** (No LED) or GPIO 13 (pin 33). For the vintage radio use GPIO 23 (Pin 16). See the section called *Remote* Control Activity LED on page 46.

```
# Output LED for remote control, default GPIO 11 (pin 23) or
# GPIO 13 (pin 33) for AdaFruit plate or PiFace CAD (40 pin RPi needed)
# remote_led=0 is no output LED
remote_led=11
```

Testing the remote control activity LED

It is possible to test the activity LED with the **remote_control.py** program.

```
$ cd /usr/share/radio/
$ sudo ./remote_control.py flash
```

Or use the service command

```
$ sudo service pifacercd flash
```

The **pifacercd** script calls the **remote_control.py** program. The activity LED should flash about six times. If not then check that the **remote_led** parameter in the **/etc/radiod.conf** configuration file is correctly set and that the activity LED is correctly wired (See LED wiring on page 46).

Changing the date format

The date is configured in the **/etc/radiod.conf** file using the dateformat parameter:

```
dateformat=%H:%M %d/%m/%Y
```

The default configuration is: **%H:%M %d/%m/%Y**
Where: %H = Hours, %M=Minutes, %d= Day of the month, %m=month, %Y=Year

It is possible to change the date format (for example for the United States) by changing the format. Some valid formats are:

%H:%M %m/%d/%Y	US format
%H:%M %d-%m-%Y	Minus sign as date separator
%d/%m/%Y %H:%M	Reverse date and time

Configuring the Adafruit LCD backlight colours

Some Adafruit displays such as the **rgb-negative Adafruit LCD** allow changing the colour of the backlight. This is configurable in the **/etc/radiod.conf** file. The colours that can be used are RED, GREEN, BLUE, YELLOW, TEAL, VIOLET and WHITE or OFF (No backlight).

The colour settings in the /etc/radiod.conf file

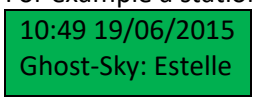
```
# Background colours (If supported) See Adafruit RGB plate
# options OFF, RED, GREEN, BLUE, YELLOW, TEAL, VIOLET, WHITE
bg_color=WHITE
mute_color=VIOLET
shutdown_color=TEAL
error_color=RED
search_color=GREEN
info_color=BLUE
menu_color=YELLOW
source_color=TEAL
sleep_color=OFF
```

Note: Always use the American spelling ‘color’ in all commands and not the British spelling ‘colour’.

Configuring the playlist number

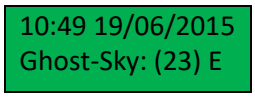
The radio can be con be configured to display the station number in brackets. To do this edit **/etc/radiod.conf** and change the **display_playlist_number=no** to yes.

For example a station is displayed as:



Change **display_playlist_number=no** to **display_playlist_number=yes**

```
display_playlist_number=yes
```



The playlist number is now displayed in brackets when the radio is restarted.

Configuring startup mode for Radio or Media player

The radio can be configured to start in either the default Radio mode or Media player mode. THe default is **RADIO**. To change this edit **/etc/radiod.conf** and change the **startup=RADIO** parameter to **MEDIA**.

```
# Startup option either RADIO or MEDIA (USB stick)
startup=RADIO
```

Configuring the volume range

This setting affects the volume control sensitivity.

The MPD daemon has a volume range from 0 to 100. The volume is incremented or decremented by one each time the volume button is pressed or rotary encoder is turned a notch. This means a lot of turns of the knob or pushes of the button to change the volume the full range. Also different devices are more sensitive than others.

For example the Adafruit plate version allows very rapid change of the volume and the default range of 0 to 100 is not a problem. The rotary encoder version of the radio requires a lot of twisting of the volume knob to get from 0 to 100.

This version allows you to set the volume range to increase the sensitivity of the volume control as shown below. For example if the volume range is set to 20 you will see the volume displayed from 0 to 20 however the MPD volume is incremented by 5.

Increment = 100 / Volume range. For example 100/20 = 5 or 100/50 = 2

So if the volume displayed on the LCD is 10 and the range is 20, then the MPD volume is 10 x 5 = 50.

The volume range is configured in **/etc/radiod.conf** configuration file using the **volume_range** parameter:

```
# Volume range 10, 20, 25, 50 or 100
volume_range=20
```

Ideally you should choose a volume range number that divides into 100 equally as shown above however other values will work.

Operation

This section assumes that the LCD screen is working correctly, the MPD daemon is installed and tested and that there is an Internet connection available. If you are using a 4x20 LCD then substitute *radiod.py* for *radio4.py* in all of the following commands. If using the Adafruit LCD plate then substitute *ada_radio.py* for all of the following commands. If using rotary encoders use *rradiod.py* and *rradiod.py* instead.

Starting the program

The program must either be run as root user or using sudo.
The basic operation of the program is:

```
$ sudo service radiod start|stop|restart|status|version
```

Where start: Start the radio program.
stop: Stop the radio program.
restart: Restart the radio program.
status: Show the status of the radio daemon.
version: Show the version number of the program

The advantage of using the above service commands is that they are the same for all versions of the radio as this is configured in the **radiod** service start/stop script. To start the radio:

```
$ sudo service radiod start
```

Alternatively the program may be started using the name of the particular program version you are using. Change to the **/usr/share/radio** directory and run the following command:

```
$ cd /usr/share/radio/
$ sudo ./radiod.py start
```

To stop the radio

```
$ sudo ./radiod.py stop
```

For the 4 x 20 character LED run:

```
$ sudo ./radio4.py start
```

To display the status either use the program directly or use the **service radiod status** command:

```
$ sudo ./radiod.py status
radiod running pid 2098
```

The above pid (Process ID) number will be different each time the program is run.

```
$ sudo service radiod status
```

```
radiod running pid 2098
```

If you see the following output when you run the status command this is a known problem with the **systemd** service daemon. This same problem causes a reboot of the Raspberry Pi to hang.

```
$ sudo service radiod status
● radiod.service - LSB: Raspberry PI Radio Daemon
   Loaded: loaded (/etc/init.d/radiod)
   Active: active (running) since Sat 2016-03-05 20:17:05 CET; 14h ago
  Process: 379 ExecStart=/etc/init.d/radiod start (code=exited,
status=0/SUCCESS)
   CGroup: /system.slice/radiod.service
           ├─602 python /usr/share/radio/ada_radio.py start
           └─8109 sh -c hostname -I

Mar 05 20:17:05 jessie systemd[1]: Started LSB: Raspberry PI Radio Daemon.
```

Install **sysvinit-core** as shown on page 52 to cure this problem.

To see what version of the software you are running:

```
$ sudo ./radiod.py version
Version 5.6
```

Buttons

There are five buttons, four function buttons and one menu button. The Menu button changes the display mode and the functions of the left and right hand buttons as shown in the following table. If using rotary encoders please see Table 12 on page 84.

Table 11 Push Button Operation

	Left hand buttons		Right hand buttons	
LCD Display Mode	**Left button**	**Right button**	**Left button**	**Right button**
Mode = TIME Line 1: Time Line 2: Station or Track	Volume Up	Volume Down	Station/Track up	Station/Track down
Mode = SEARCH If source = RADIO Line 1: Search: Line2: Radio Station	Volume Up	Volume Down	Scroll up radio station	Scroll down radio station
Mode = SEARCH If source = MUSIC LIBRARY Line 1: Search Line2: MusicTrack/Artist	Scroll up through artists	Scroll down through artists	Scroll up through track	Scroll down through track
Mode = SOURCE Line 1: Input Source: Line2: Internet Radio or Music Library	Volume Up Mute	Volume Down Mute	Toggle mode between Radio and Music Library	Toggle mode between Radio and Music Library
Mode = OPTIONS Line 1: Menu Selection Line 2: <option> Options are Random, Consume, Repeat, Reload Music, Timer, Alarm ,Alarm Time Set (Hours), Alarm Set (Minutes), Streaming:	Toggle selected mode on or off. Set timer and Alarm	Toggle selected mode on or off. Set timer and Alarm	Cycle through Random, Consume, Repeat, Reload Music, Timer, Alarm , Alarm Time Set and Streaming	Cycle through Random, Consume, Repeat, Reload Music, Timer, , Alarm Time Set and Streaming:
Mode = RSS (1) Line 1: Time Line 2: RSS feed	Volume Up Mute	Volume Down Mute	Station/Track up	Station/Track down
MODE = IP address Line 1: IP address Line 2: Station or Track	Volume Up Mute	Volume Down Mute	Scroll up through track or radio station	Scroll down through track or radio station

Note : If the **/var/lib/radiod/rss** file is missing then the RSS mode is skipped. If it contains an invalid RSS URL, this will be displayed on the LCD.

Rotary encoder operation

This option is for a radio with rotary encoders with push buttons. The volume knob when pushed in is the **Mute** sound function. Likewise the tuner knob when pushed in is the **Menu** switch.
The Menu button (Tuner knob depressed) changes the display mode and the functions of the clockwise and anti-clockwise operation of the knobs as shown in the following table.

Table 12 Rotary Encoder Knob Operation

	Volume knob		Tuner knob	
LCD Display Mode	**Clockwise**	**Anti-clockwise**	**Clockwise**	**Anti-clockwise**
Mode = TIME Line 1: Time Line 2: Station or Track	Volume Up	Volume Down	Station/Track up	Station/Track down
Mode = SEARCH If source = RADIO Line 1: Search: Line2: Radio Station	Volume Up	Volume Down	Scroll up radio station	Scroll down radio station
Mode = SEARCH If source = MUSIC LIBRARY Line 1: Search Line2: MusicTrack/Artist	Scroll up through artists	Scroll down through artists	Scroll up through track	Scroll down through track
Mode = SOURCE Line 1: Input Source: Line2: Internet Radio or Music Library	Volume Up Mute	Volume Down Mute	Toggle mode between Radio and Music Library	Toggle mode between Radio and Music Library
Mode = OPTIONS Line 1: Menu Selection Line 2: <option> Options are Random, Consume, Repeat, Reload Music, Timer, Alarm and Alarm Time(Hours), Alarm Time(Minutes) set and Change colour(1), Streaming on/off.	Toggle selected mode on or off. Set timer and Alarm	Toggle selected mode on or off. Set timer and Alarm	Cycle through Random, Consume, Repeat, Reload Music, Timer, Alarm Time Set, Streaming and Background colour(1)	Cycle through Random, Consume, Repeat, Reload Music, Timer, Alarm , Alarm Time Set, Streaming and Background colour(1)
Mode = RSS (2) Line 1: Time Line 2: RSS feed	Volume Up	Volume Down	Station/Track up	Station/Track down
MODE = IP address Line 1: IP address Line 2: Station or Track	Volume Up	Volume Down	Scroll up through track or radio station	Scroll down through track or radio station

Note 1: The colour change option is only available for the AdaFruit RGB plate (ada_radio.py).
Note 2: If the **/var/lib/radiod/rss** file is missing or contains an invalid RSS URL then the RSS mode is skipped.

Mute function

Pressing both volume buttons together or in the case of a rotary encoder with a push button (Volume) will mute the radio. If voice is enabled then then operation is slightly different (See section on espeak). Press either the volume up or down switch to un-mute the radio. If you change channel or use the menu switch the radio will also be un-muted. If the alarm is set then the radio will go into sleep mode.

Playing MP3 and WMA files

The radio software also allows you to play music from the following sources:

1. From a USB stick
2. From a music directory on the SD card (Create this yourself)
3. From a Network Attached Storage (NAS)

Playing music from a USB stick

Put your music tracks on a USB stick (MP3 and WMA files only) and insert it into the USB port of the Raspberry PI. Reboot the PI. Once the Radio program is running, push the Menu button until "Input source" is displayed. Press either the left or right button to change the source to "Music Library". Now press the Menu button again. The music on the USB stick will now be loaded.

Playing music from the SD card

With large (32GB) SD cards now available music can be stored on in one or more directories on the SD card. It is necessary to first create a directory in **/home/pi** as user **pi** and then link it in the **/var/lib/mpd/music/** directory. Carry out the following instructions as user **pi** to create **mymusic** for example:

```
$ mkdir /home/pi/mymusic
$ cd /var/lib/mpd/music/
$ sudo ln -s /home/pi/mymusic
```

Using FTP, copy the music from a PC to the **/home/pi/mymusic** directory and reload the library via the options menu.

Playing music from a Network Attached Storage (NAS)

This is a bit more involved to set up. See the section called *Mounting a network drive* on page *101*.

Organising the music files

For the search routines to work properly the music must be organised in a certain way. The files must be placed in the top level directory of USB stick. The search routines use the first directory as the artist's name and the music files themselves as the track name. For example:

Elvis Presley/The 50 Greatest Hits Disc 1/01 That's All Right.mp3

In the above example the first directory is set up with the artist name and this will appear in the search. The next directory "The 50 Greatest Hits Disc 1" is not used by the search routines. The file name "That's All Right.mp3" without the mp3 extension becomes the track name in this example.

MPD Logging

All logging for the MPD daemon is to the **/var/log/mpd/mpd.log** file by default.

Radio program logging

The Radio program logs to a file called **/var/log/radio.log.** See example log below:

```
2016-07-24 08:31:48,638 INFO Radio running pid 4307
2016-07-24 08:31:48,641 INFO Radio ['/usr/share/radio/rradio4.py, 'restart']
daemon version 5.6
2016-07-24 08:31:48,644 INFO GPIO version 0.5.11
2016-07-24 08:31:48,653 INFO Board revision 2
2016-07-24 08:31:50,565 INFO GPIO version 0.5.11
2016-07-24 08:31:50,599 INFO Board revision 2
2016-07-24 08:31:50,624 INFO OS release: Raspbian GNU/Linux 7 (wheezy)
2016-07-24 08:31:50,642 INFO Linux piboombox 3.18.11-v7+ #781 SMP PREEMPT
Tue Apr 21 18:07:59 BST 2015 armv7l GNU/Linux
2016-07-24 08:31:50,741 INFO Connected to MPD port 6600
2016-07-24 08:31:51,043 INFO Output 2 (PI Radio MPD Stream) is disabled
2016-07-24 08:31:51,047 INFO UDP Server listening on localhost port 5100
2016-07-24 08:31:51,056 INFO UDP listen:remote 0.0.0.0 port 5100
2016-07-24 08:31:51,058 INFO MPD started
2016-07-24 08:31:51,580 INFO mpd version: 0.16.0
2016-07-24 08:31:54,913 INFO Title:
2016-07-24 08:31:54,977 INFO Current ID = 35
2016-07-24 08:31:55,001 INFO Running
```

There are six levels of logging namely CRITICAL, ERROR, WARNING, INFO, DEBUG or NONE. This is configured in the **/etc/radiod.conf** file.

```
# loglevel is CRITICAL,ERROR,WARNING,INFO,DEBUG or NONE
loglevel=INFO
```

Configuration and status files

The main configuration file is **/etc/radiod.conf**. See

There are some other configuration and status files in the **/var/lib/radiod** directory. These are:

alarm	Alarm setting in t:hh:mm where t is the alarm type (t=0=off)
current_station	The current radio station
current_track	The current music track
rss	RSS feed URL
language	Espeak language definition file
share	The NAS share instruction
stationlist	The user list of radio station URLs
streaming	Icecast2 streaming on or off
timer	Timer (Snooze) value in minutes
volume	The volume setting
voice	The espeak voice file

It isn't normally necessary to change most of these files. However the **stationlist**, **share, language** and **rss** file will need to be edited as required. The other files are maintained by the program so that when it starts up the program uses the last settings, for example, the volume setting. All other user settings are maintained in the **/etc/radiod.conf** configuration file.

Displaying an RSS feed

To display an RSS feed it is necessary to create the **/var/lib/radiod/rss** file with a valid RSS URL. For example:

```
http://feeds.bbci.co.uk/news/uk/rss.xml?edition=int
```

The above is the RSS for the BBC news however any valid RSS feed may be used. If the **/var/lib/radiod/rss** is missing or contains an invalid RSS URL then this mode is skipped when stepping through the menu. The software comes with with a valid BBC RSS feed file in the **/var/lib/radio/rss** file . You can test the feed first by pasting it into your PC's web browser URL and pressing enter.

Using the Timer and Alarm functions

There is a timer (Snooze) and alarm function. The timer and alarm can operate individually or together. The timer when set will put the radio into Sleep Mode when the timer expires. The Alarm can be set to either On, Repeat or "Weekdays only".

Setting the Timer (Snooze)

Press the Menu button until the "Menu Selection" is displayed. Press either the channel UP or DOWN control until "Timer off" is displayed on line 2 of the LCD screen. Now push the volume UP button to set the timer. Use volume UP and DOWN to adjust the timer which will be displayed as "Timer hh:mm:ss" wher hh=hours, mm=minutes and ss=seconds. The Timer can be set up to 24 hours in increments of one minute. Once the timer is set, press the Menu button; the display will return to TIME mode.

On a four line LCD display the timer will be seen counting down after the Volume display on line 4. On a two line LCD display the timer count down will be displayed on line 1 after the time display.

When the timer expires (reaches zero) the radio will enter SLEEP mode. Sleep mode can only be exited by pressing the menu button.

To switch the timer off go back to the timer menu as described above and reduce the timer to 0 using the volume DOWN control. This will switch off the timer.

The timer function uses the **/var/lib/radiod/timer** file which will contain the value of the timer in minutes when it was successfully fired. You do not need to change the contents of this file.

Setting the Alarm

The Alarm menu has three settings:

- The alarm type (On, off, repeat etc)
- The Alarm Hours time (Pressing menu in this mode puts the radio into Sleep mode)
- The Alarm Minutes time (Pressing menu in this mode puts the radio into Sleep mode)

Press the Menu button until the "Menu Selection" is displayed. Press either the channel UP or DOWN (Or rotate rotary encoder) until "Alarm off" is displayed on line 2 of the LCD screen. Using the volume UP control cycle through the options which are

- Alarm off - The Alarm is switched off

- Alarm on – The Alarm is on for one time only. Once the alarm is fired it will return to off.
- Alarm repeat – The Alarm will be repeated every day and not switched off.
- Alarm weekdays only – The Alarm will only fire Monday through Friday. It is not reset.

Now move to "Set alarm time:" using the channel UP control. The current alarm time will be displayed on line 2 of the display. Using the volume UP and DOWN control adjust the alarm time (Hours or Minutes) to the required setting. If you do not wish to put the radio into sleep mode at this stage then use the channel UP/DOWN control to move away from the "Set alarm time:" option and press the Menu button. If you press the Menu button whilst in the "Set alarm time:" option and the Alarm is set to anything except off then the radio will enter Sleep mode and display the alarm on line 2 for a two-line LCD or on line 4 for a four-line LCD.

Note: Sleep mode can only be exited by pressing the Menu button.

The alarm function uses the **/var/lib/radiod/alarm** file which will contain the current alarm type and time. The format is **t:hh:mm** where t is type (0=off, 1=on, 2=repeat, 3=weekdays only) and hh:mm is hours and minutes (24 hour clock). You do not need to change the contents of this file.

PLEASE NOTE THAT THE ALARM RELIES UPON THE SELECTED RADIO STREAM TO BE AVAILABLE WHEN THE ALARM WAKES UP. THIS CANNOT BE GUARANTEED AS THE STATION FEED MAY BE OFF AIR OR THERE IS A PROBLEM WITH THE INTERNET CONNECTION. YOU SHOULD NOT THERFORE RELY SOLEY ON THIS ALARM FUNCTION IF YOU HAVE AN IMPORTANT APPOINTMENT OR A PLANE OR TRAIN TO CATCH FOR EXAMPLE. ALSO SEE DISCLAIMER ON PAGE 150.

Using the Alarm and Timer functions together

The Alarm and Timer functions can be used together. For example you want to set your radio to a 30 minute snooze time before going to sleep and to sound the alarm in the morning. Simple set the Timer to the required elapse time and then set the alarm as described in the previous section. Press the Menu button and the timer will be seen counting down followed by the alarm time on line 4 or line 1 for the four-line and two-line LCD respectively.

Music Player Clients

MPD is designed around a client/server architecture, where the clients and server (MPD is the server) interact over a network. A large number of graphical and web based clients are available for MPD and are to numerous to mention here. Please see the following link for further information on MPD clients: http://mpd.wikia.com/wiki/Clients. The main client used in this project is MPC.

Using the MPC client

Everything you should normally wish to do can be done using the radio. However there may be occasions that you wish to test or control music selection, volume etc. using MPC. It is also useful for diagnosing Music Player Daemon problems.

Log into the Raspberry PI using the console or SSH login. To start playing music run:

```
$ mpc play
```

To see a list of all available commands, run:

```
$ mpc help
```

Here are some frequently used **mpc** commands:

MPC command	Description
mpc	Displays status (**mpc status** also does the same)
mpc current	Displays currently playing station or track
mpc next	Play next song
mpc prev	Play previous song
mpc play n	Play station or track where n is the track or station number
mpc volume 75	Set volume to 75%
mpc stop	Stop playing
mpc random <on|off>	Toggle shuffling of songs on or off
mpc repeat <on|off>	Toggle repeating of the playlist
mpc clear	Clear the playlist
mpc consume <on|off>	When playing tracks remove from the playlist once played
mpc playlist	List loaded radio stations or streams
mpc listall	List all songs in the music directory

Adafruit RGB Plate changing colours

This section is only relevant for the Adafruit RGB plate. The **ada_radio.py** program has an option to change the colour of the display. Push the menu button until “Menu selection”. Push the channel button until “Select color” is displayed. Now push the volume button to cycle through the colours. The available colours are red, green, blue, yellow, teal, violet, white or Off (No backlight).

Shutting down the radio

You can simply switch the power off. This doesn’t appear to harm the PI at all. However if you want a more orderly shutdown then press the menu button for at least three seconds. This will stop the MPD daemon and issue a shutdown request to the Raspberry PI. Wait at least another ten seconds and then power off the Radio.

Creating and Maintaining Playlist files

Creating playlists

The *create_m3u.py* program is used to create playlists in the **/var/lib/mpd/playlists** directory. If you wish to understand more about playlist files see the section called *Overview of media stream URLs* on page 93.

The directories and files used by the *create_m3u.py* program are shown in the following table:

Table 13 Playlist files and directories

Name	Type	Description
/usr/share/radio/station.urls	File	Initial distribution file containing sample stations
/var/lib/radiod/stationlist	File	The file containing the users list of radio stations
/usr/share/radio/playlists	Directory	The directory containing any user playlist files
/tmp/radio_stream_files	Directory	Temporary work directory used during processing
/var/lib/mpd/playlists	Directory	The Music Player Daemon Playlist directory

From a user point of view only the **/var/lib/radiod/stationlist** file and the **/usr/share/radio/playlists** directory are important**.**

During initial installation the *create_m3u.py* is run automatically. This powerful little program does the following:

1. Copies the /usr/share/radio**/station.urls** file to the **/var/lib/radiod/stationlist** file. It does this only the first time that it runs. The **station.urls** file will not normally be used again.
2. Copies all playlists it finds in the **/usr/share/radio/playlists** directory to the **/var/lib/mpd/playlists** directory (Only for backward compatibility with earlier versions).
3. It reads each entry in the **/var/lib/radiod/stationlist** file.
4. If it comes across a (<playlist name>) definition it creates a new playlist.
5. If the line read contains a station definition then it determines if the URL in the station entry is a direct radio stream (.mp3, AAC etc) or if it is a redirect URL (.pls, .asx or .m3u).
6. If it is a direct radio stream URL it creates a playlist in M3U format and in the **/tmp/radio_stream_files** directory.
7. If it is an ASX URL (.asx) it reads the contents of the ASX file and uses them to create a playlist in M3U format in the **/tmp/radio_stream_files** directory.
8. If it is a redirect URL (.pls or .m3u) gets the file pointed to in the URL and from the contents of this file it creates an entry in M3U format in the **/tmp/radio_stream_files** directory.
9. It prompts you if you wish to remove all removes all the old playlists from the **/var/lib/mpd/playlists** directory.
10. It finally copies all files in the **/tmp/radio_stream_files** directory to the **/var/lib/mpd/playlists** directory.

The program itself is very easy to use. Just run it as sudo or root user in the **/usr/share/radio** directory:

```
$ cd /usr/share/radio
$ sudo ./create_m3u.py
```

This will create a set of playlist files in the **/var/lib/mpd/playlists** directory. The **/var/lib/radiod/stationlist** file contains a list of entries describing the radio station to be loaded.

To create a log file of the program run the following:

```
$ sudo ./create_m3u.py | tee playlist.log
```

You can examine the playlist.log file to see what actions the create_m3u.py program carried out and if there were any errors.

The program will ask if you wish to delete any old playlists:

```
There are 8 old playlist files in the /var/lib/mpd/playlists/ directory.
Do you wish to remove the old files y/n: y
```

Normally answer 'y' unless you don't wish to remove the old files. Note that old playlists files with the same name as the new ones will always be overwritten.

If you want to avoid the above prompt then there are two other parameters that you may use.

--delete_old Delete old playlist files in the MPD playlist directory
--no_delete Don't delete old playlist files in the MPD playlist directory

Example:

```
$ sudo ./create_m3u.py --no_delete
```

Finally there is a help parameter:

```
$ sudo ./create_m3u.py --help
```

The stationlist file

The **/var/lib/radiod/stationlist** file is the file that should be maintained by you to create playlists. When this *create_m3u.py* program is first run it copies the distribution file **station.urls** to the **/var/lib/radiod/stationlist** file. You may then modify the **/var/lib/radiod/stationlist** file.

The format is: **(<playlist name>)**
Example: **(BBC Radio Stations)**

The above will create a playlist called **BBC_Radio_Stations.pl** and will contain the title and URLs for each station. Now add or remove radio station definitions in the **stationlist** file. The first statement in the station definition is the name of the playlist in brackets:

The format is: **[<title>] http://<url>**
Example: **[BBC Radio 4 extra] http://www.bbc.co.uk/radio/listen/live/r4x.asx**

After modifying the stationlist file run the *create_m3u.py* program to create the Music Player Daemon playlists.

The *create_m3u.py* program creates file names using the title so in the above example the file will be called *BBC_Radio_4_extra.m3u* (The AS format will be converted to M3U format). If this file already

exists the new file will be renamed as *BBC_Radio_4_extra[1].M3U.* If that file also exists the program will keep incrementing the number in brackets until the name is unique. This is one of the reasons the files are created in a temporary directory first. Try to create unique titles in the **stationlist** file to avoid this.
If no title is found then the site name will be used with dots turned into underscores. For example if the following is defined without a title and no title is found:

[]http://bbc.co.uk/radio/listen/live/r1.asx

This will create a playlist file called *bbc_co_uk.m3u*. If this already exists it will be renamed to *bbc_co_uk[1].m3u.* If that file also exists the program will keep incrementing the number in brackets until the name is unique. It is therefore always better to define a unique title.

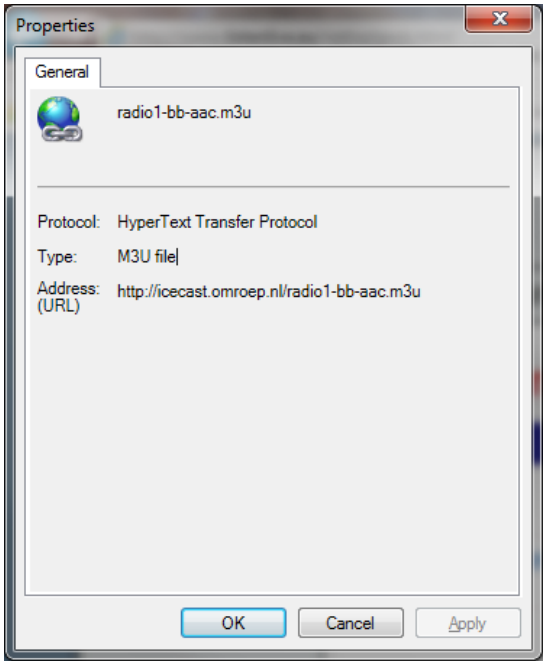
Radio stream resources on the Internet

There are a lot of resources on the Internet how to find PLS and M3U files so simply search for "PLS or M3U files" through the search machine of your choice. Below are some good sources of radio streams around the world.

http://www.listenlive.eu
http://www.radio-locator.com
http://bbcstreams.com/

For UK and Irish listeners
http://www.radiofeeds.co.uk/

To copy a URL open the web page in any browser on a PC and right click on the URL. Select properties from the drop down list. For internet explorer will show a window similar to the following will be displayed:



Copy and paste the URL into the **/var/lib/radiod/stationlist** file. Add the title in square brackets as shown in the previous section. Other browsers may provide options such as 'copy link'or 'save link as'. This is browser dependant.

Overview of media stream URLs

A deep understanding of this section is not necessary but can help in creating playlists. This section is provided for background information only. At first the whole business of how music streams are provided can be quite confusing. The URLs on a radio station web page can be of different types, for example:

1. A URL pointing to a PLS playlist file(Shoutcast Play List)
2. A URL pointing to a M3U playlist file (MPEG3 URL)
3. A URL pointing to an ASX playlist file (Advanced Stream Redirector)
4. A URL which is an actual stream such as MP3 (MPEG 3) or AAC (Advanced Audio Coding)

1, 2 and 3 are so called redirector URLs and point to a playlist file containing one or more URLs to the radio stream(s) itself. The *create_m3u.py* program tries to figure out what type of URL that it is and create a playlist from it. This is the facility you should use rather than trying to create your own playlists which can be quite time consuming.

PLS file format

A good place to start is the following Wikipedia article:

http://en.wikipedia.org/wiki/PLS_(file_format)

A PLS playlist file does not contain any music files itself, but rather points to music files stored elsewhere. The PLS file format is often used to play Internet radio streams, for example, if you want to play a radio stream from Shoutcast, you can copy the PLS file URL of the station from the site and play it in a desktop media player like Winamp. A PLS file will be similar to below

```
[playlist]
NumberOfEntries=2
Version=2
File1=http://206.217.213.16:8430
Title1=Blues Radio UK
Length1=-1
File2=http://205.164.62.13:8030
Title2=Absolute Blues Hits
Length2=-1
```

The PLS file must always start with the **[playlist]** statement. The **NumberOfEntries** statement must match the number of streams defined in the PLS file (Two in the above example). Set the **Version** number always to 2.

There must be a File*n*, Title*n* and Length*n* where *n* is the entry number.

M3U Files

M3U stands for MPEG3 URL. The following Wikipedia article explains the M3U file format:

http://en.wikipedia.org/wiki/M3U

An example M3U file is shown below:

radio10.m3u playlist file

```
#EXTM3U
#EXTINF:-1, Radio 10 Gold NL
http://icecast.streaming.castor.nl:80/radio10
```

These playlist files must have the m3u file extension. i.e. **<filename>.m3u**

The first line is the header and must be #EXTM3U. The second line is #EXTINF: and is information about the radio stream. The -1 means unlimited play length. This is followed by a comma and then the name of the radio station (*Radio 10 Gold* in this case). The third line is the URL (icecast in this case) for the radio stream. More than one radio stream may be defined in the m3u file. Simple add extra #EXTINF and URL lines for each radio stream.

ASX file

The Advanced Stream Redirector (ASX) format is a type of XML metafile designed to store a playlist of Windows Media files for a multimedia presentation. An example ASX file is shown below.

```
<ASX version="3.0">

<ABSTRACT>http://www.bbc.co.uk/iplayer/radio/bbc_radio_bristol/</ABSTRACT>
        <TITLE>BBC Bristol</TITLE>
        <AUTHOR>BBC</AUTHOR>
        <COPYRIGHT>(c) British Broadcasting Corporation</COPYRIGHT>
        <MOREINFO
HREF="http://www.bbc.co.uk/iplayer/radio/bbc_radio_bristol/" />
        <PARAM NAME="HTMLView"
VALUE="http://www.bbc.co.uk/iplayer/radio/bbc_radio_bristol/" />
        <Entry>
                <ref href="mms://wmlive-
nonacl.bbc.net.uk/wms/england/lrbristol?BBC-
UID=1523a2f20f858eb0ba0a4385918e6433df886bb650e021b4446ff486f8204e3a&amp;SSO
2-UID=" />
        </Entry>
</ASX>
```

Direct stream URLs

These URLs tend to end with .mp3 or _SC or AAC etc. However there are others. For example:

http://mp3.streampower.be/radio1-high.mp3
http://7639.live.streamtheworld.com:80/977_MIX_SC

You can determine if a URL is a direct radio stream by using the **wget** program.

```
# cd /tmp
# wget http://mp3.streampower.be/radio1-high.mp3
--2014-03-14 13:08:10--  http://mp3.streampower.be/radio1-high.mp3
Resolving mp3.streampower.be (mp3.streampower.be)... 80.200.255.61
Connecting to mp3.streampower.be (mp3.streampower.be)|80.200.255.61|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [audio/mpeg]
Saving to: `radio1-high.mp3'
```

```
    [                                                      <=>                 ]
365,281      15.8K/s
```

If **wget** doesn't exit and you see the <=> characters moving backwards and forwards then it is a URL to the radio stream itself. You will also see *Length: unspecified* in the output. Press control -C to exit **wget**. Remove the file that **wget** created (/tmp/radio-high.mp3 in this case).

Listening to live Air Traffic Control (ATC)

For those interested in aviation this is a fascinating use of the radio. Live ATC net provide streaming of live ATC transmissions from airports the world over. Their web site is http://www.liveatc.net/

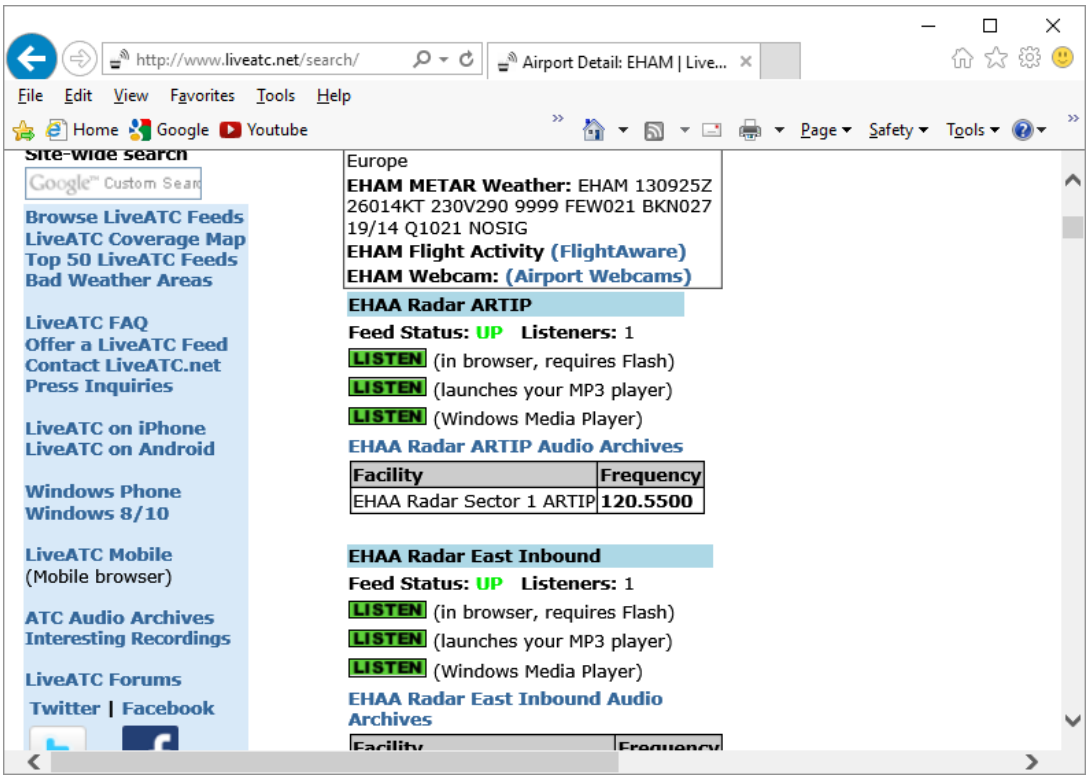


Figure 81 Live ATC web page

Not only do these streams provide the live ATC conversations but also the in the station information ATIS (Aerodrome Terminal Information Service). This consists of coded weather information which all pilots can understand.

One way to add these stations to a radio playlist is to install **WinAmp** on a PC. Enter either the **ICAO** or **IATA** code of the station in the search box on the Live **ATC** web site, for example **EHAM** or **AMS** (Schiphol, Amsterdam, the Netherlands) .

Click on the MP3 player **LISTEN** option. The station will be loaded and shortly **WinAmp** will start playing the stream.

Figure 82 WinAmp playing ATC live feed

Right click in the top left box (Display elapsed time of 1:05 in this example). The station information will be displayed.

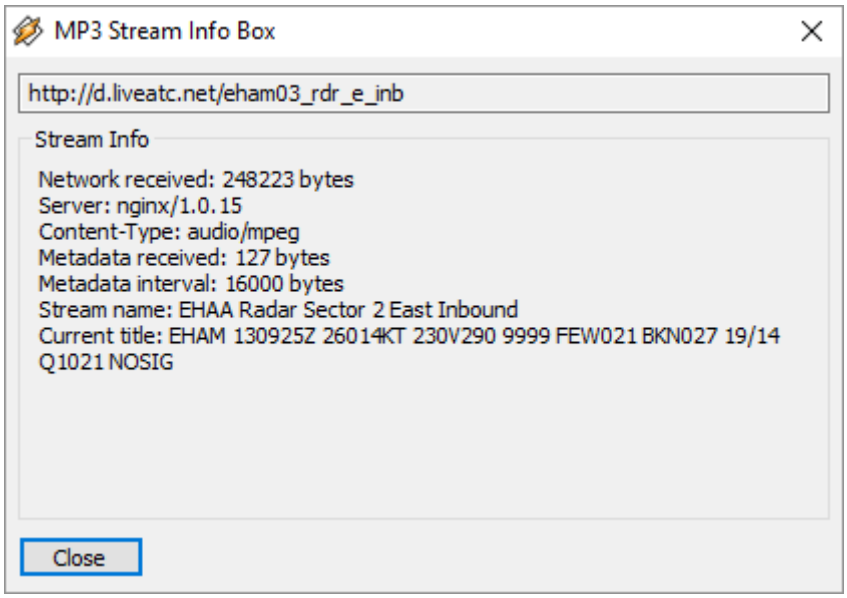


Figure 83 WinAmp station information

The URL for the stream is shown in the top box. **http://d.liveatc.net/eham03_rdr_e_inb**
The stream name is: EHAA Radar Sector 2 East Inbound
The station Title shows the ATIS information.
EHAM 130955Z 25016KT 9999 FEW020 BKN024 19/14 Q1021 NOSIG

Using the URL shown create a playlist to **/var/lib/radiod/stationlist** for the live traffic ATC as shown in the following example.

```
(_ATC live)
[EHAA Approach Departures] http://d.liveatc.net/eham4
[EHAA Radar Sector 2 East Inbound] http://d.liveatc.net/eham03_rdr_e_inb
[EHAA Radar SW]  http://d.liveatc.net/eham02_rdr_sw
[EHAA] Radar 3 South] http://d.liveatc.net/eham02_rdr_s
[EHEH Approach] http://d.liveatc.net/eheh2_app
```

In the above example the playlist name starts with an _ to put it at the end of all the other stations.

Now run the **create_m3u.py** program to create the playlists.

```
$ cd /usr/share/radio
$ sudo ./create_m3u.py
```

Now restart the radio or use the menu to reload the radio stations (Select source option):

```
$ sudo service radiod restart
```

Finally select the new ATC station(s).

Finding out ICAO and IATA airport codes

Try sites such as https://en.wikipedia.org/wiki/List_of_airports_by_ICAO_code:_A

Decoding ATIS information

See site http://www.met.tamu.edu/class/metar/quick-metar.html or search for ATIS decode.

In the following example:

EHAM 130955Z 25016KT 9999 FEW020 BKN024 19/14 Q1021 NOSIG

EHAM	Amsterdam Schiphol, the Netherlands
130955Z	13th of the current month. Time 09:55 Zulu (UTC)
25016KT	Wind 250 degrees at 16 Knots
9999	Visibility 10 Kilometres or greater
FEW020	Few clouds at 2000 feet
BKN024	Broken cloud at 2400 feet
19/14	Temperature 19 degrees Celsius. Due point 14 degrees Celsius
Q1021	Barometric pressure 1021 Millibars (Will be given in Inches Mg in US airports)
NOSIG	No significant weather.

Installing the Web interface

MPD has several web clients. See the following link: http://mpd.wikia.com/wiki/Clients. The one used in this example is called snoopy is used in this version the radio software.

Install Apache

Install Apache the web server. Make sure that the system is up to date with the following command otherwise installation of Apache will fail.

```
$ sudo apt-get update
```

Now install Apache and the PHP libraries for Apache as user root.

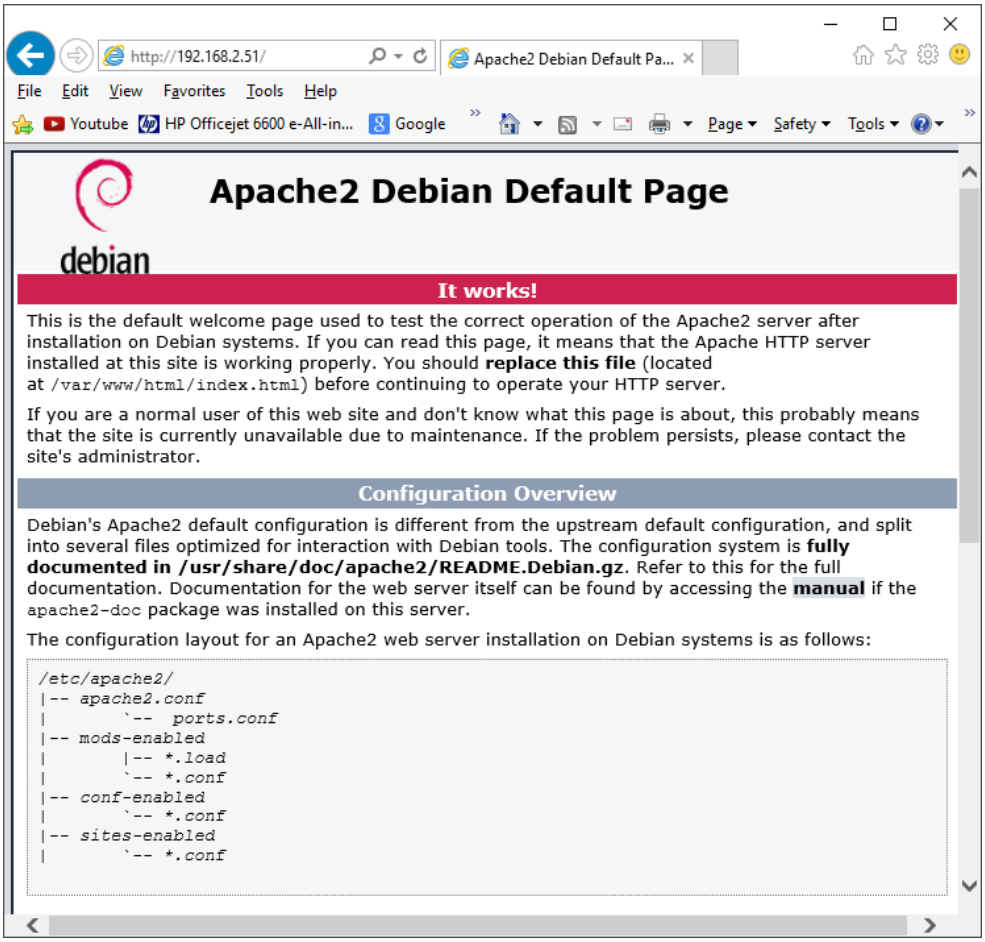
```
$ sudo bash
# apt-get install apache2 php5 libapache2-mod-php5
```

This will take some time. If the above fails run the following command and re-run the installation:

```
# apt-get -f install
```

Test the Apache web browser

Point your web browser at the IP address of the Raspberry PI. For example: http://192.168.2.51. You should see the following display.



Install the Web Browser server pages

It is now necessary to install the web pages for the Radio. Download the radio web pages Debian package from the Bob Rathbone web site:

```
$ wget http://www.bobrathbone.com/raspberrypi/packages/radiodweb_1.1_armhf.deb
```

Now run:

```
$ sudo dpkg -i radiodweb_1.1_armhf.deb
```

This package will install the radio web pages in the **/var/www/html** directory and the CGI scripts in **/usr/lib/cgi-bin** directory. It will also enable the CGI scripts module.

Start the radio web interface

Point your web browser at the IP address of the Raspberry PI. For example: http://192.168.2.11 .
You should see the following display:

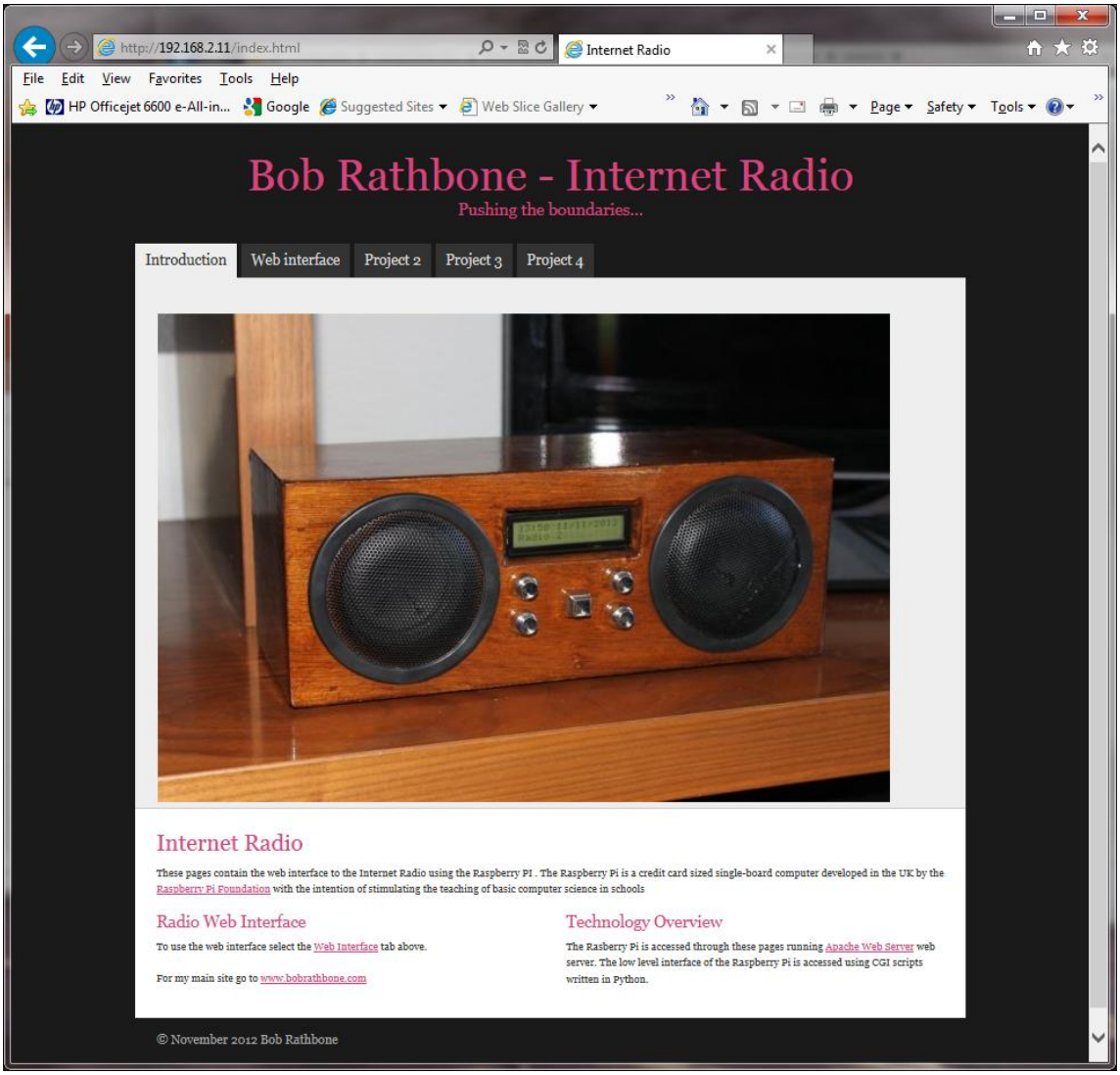


Figure 84 Radio web interface

Now click on the 'Web interface tab'. If the radio software is running you will see the following:

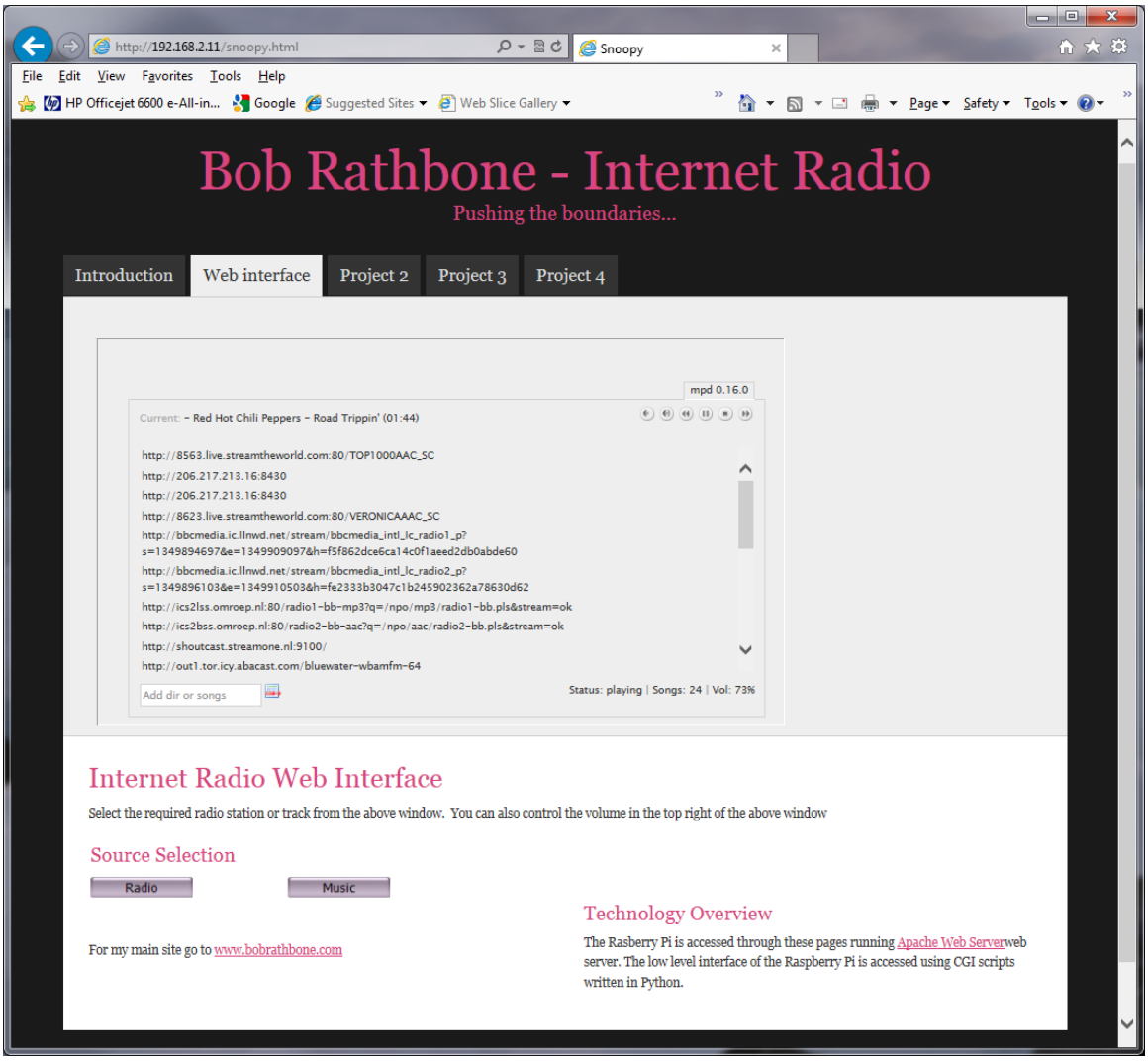


Figure 85 Snoopy web interface

Click on any station on the list to select a station. The Radio and Music buttons select the source.

Changing the Web Interface Radio photo

If you want to change the photo displayed by the web interface, then replace the **jpeg** photo file at **/var/www/html/images/radio.jpg**. Try to adjust the size on disk to about 50K using a suitable photo editor such as Photo Shop.

If the new image looks stretched then it may also be necessary to change image proportions in the <img..> statement in **/var/www/html/index.html** file. Find the following line in the index file and adjust the width/height values to display the photo with the correct proportions.

```
<img border="0" src="images/radio.jpg" width="864" height="576"></td>
```

This also applies to the project2, 3 and 4 html files.

Mounting a network drive

It is very likely that you may have your music on a shared network drive and want to play the music through the radio. There are two main types of network drive protocols used by Raspbian Jessie on the Raspberry Pi namely:

- CIFS – Common Internet File System
- NFS – Network File System

The protocol used for CIFs is SMB (Server Message Block – Microsoft). Previously connections to SMB was via a product called SAMBA but has been largely replaced by the mount using the CIFS option in the Linux. The steps to mount the network drive are as follows:

1. Find out the IP address of your network drive.
2. Create and test the mount command using either NFS or CIFS.
3. Copy the mount command to **/var/lib/radiod/share** file.
4. In the Radio menu select "Music Library" as the source and press "Menu" again to load
5. Update the playlists to include the files on the new share (Network drive).

This procedure assumes that you already have your Network Drive configured and working with your PC and can play music via the PC. In the examples below a Synology Network Drive was used with a volume called Volume1 with a directory called "music". The IP address for the Synology Network drive used was 192.168.2.6.

First stop the Radio software when creating and testing the mount command.

Don't configure **/etc/fstab** to do the mount of the network drive. Although this is the usual way of mounting shares however the radio program needs total control of the mount and un-mount process.

The general syntax for the mount command is as follows:

```
mount -t <type> -o option1,option2,… <remote IP address and directory>
<mount point>
```

Where: <type> is either **nfs** or **cifs**.

-o option1,option2 are the mount options.

<remote IP address and directory> Is the IP address and music directory path

<mount point> This will always be /share for this program

Finding the IP address of the network drive

Only general guidance can be given here. Nearly all network drives have a web interface. The IP address was almost certainly provided from DHCP in your home router. The IP address will be the IP address of the Web Interface. Look at your network drive documentation for further information.

The CIFS mount command

The following example mount command assumes that you have a guest user configured with password 'guest'. Adapt the command as required.

```
mount -t cifs -o username=guest,password=guest,uid=pi,gid=pi  //192.168.2.6/music
/share
```

The above command is all on one line. The **uid** and **gid** parameters set the ownership of the music files to user **pi**. The share directory is created when you first run the Radio program so there is no need to create it. If the command was successful you should be able to display the music from the network drive. Go to section called *Display the share directory* on page 102.

Older NAS drives sec security option

Older NAS drives may also require the **sec=ntlm** option to the **-o** line. The **sec** option is the authentication protocol and determines how passwords are encrypted between the server and client. Security mode **ntlm** used to be the default authentication method but that is now become **ntlmssp**. If you are accessing a network drive which doesn't support **ntlmssp** you have to add **sec=ntlm** to the options as shown below:

```
-o username=guest,password=guest,uid=pi,gid=pi,sec=ntlm
```

Many NAS devices use older technology so they often only use **ntlm** authentication. There are other authentication methods such as **ntlmv2** but most are not currently supported with the Raspberry Pi OS.

The NFS mount command

The following NFS mount example assumes the NFS protocol has been configured for the music directory.

```
mount -t nfs -o ro,nolock 192.168.2.6:/volume1/music /share
```

A few things to note here; the NFS mount command uses the volume name (Volume1 – can vary), The CIFS mount command doesn't. The second thing is that the IP address and remote directory are separated by a colon (:). If the command was successful you should be able to display the music from the network drive.

Display the share directory

If the mount was successful using either CIFS or NFS you should be able to display the **/share** directory with the ls command.

```
# ls -la /share
total 4
drwxrwxrwx 85 pi   pi      0 May 10 14:18 .
drwxr-xr-x 23 root root 4096 Jul 15 17:57 ..
drwxrwxrwx  4 pi   pi      0 May 10 14:16 Albert Hammond
drwxrwxrwx  3 pi   pi      0 May 10 14:16 Alexander Curly
drwxrwxrwx  3 pi   pi      0 May 10 14:16 Allen Price & Georgie Fame
drwxrwxrwx  3 pi   pi      0 May 10 14:16 Al Martino
drwxrwxrwx  3 pi   pi      0 May 10 14:16 Animals
drwxrwxrwx  4 pi   pi      0 May 10 14:16 Aretha Franklin
drwxrwxrwx  3 pi   pi      0 May 10 14:16 Armand
```

The important thing apart from seeing the files is that you should see that the files are owned by pi and group pi.

Un-mounting the /share directory

To un-mount the share directory use the **umount** command (not unmount).

```
# umount /share
```

Copy the mount command to the configuration

Once the mount command is working copy it to the **/var/lib/radiod/share** file.
For example for the CIFS mount command.

```
# echo "mount -t cifs -o username=guest,password=guest,uid=pi,gid=pi
//192.168.2.6/music /share" > /var/lib/radiod/share
```

The above command is all on one line.

Note: If you decide to directly edit the **/var/lib/radiod/share** file instead of using the above command then do not include quotations marks around the command.

Load the music library

Now run the radio program. The radio stations will be loaded. Cycle through the menu until **Input Source**: is displayed. Press the channel up or down buttons to select **Music Library**.

Now press the **Menu** button. The program loads whatever playlists it has in its database, and will most likely be only those from the USB stick if installed. However the *playlist* for the new share files are not yet in the MPD database. The playlist needs to be updated in the following section.

Update the playlists for the new share

Select Music Library Now cycle through the menu until **Menu Selection:** is displayed. Press the channel up or down buttons until the **Update list:No** is displayed. Use the Volume buttons to toggle the display to **Update list:Yes**.

Now press the Menu button. This will cause the MPD database to be cleared and updated from all the files loaded in the **/var/lib/mpd/music** directory including the new share. This can take some time (Several minutes) if the Network Drive contains a large amount of music files. During this process the Radio program will ignore any button depressions and you will see the first **Initialising** (Library) and then **Updating** (Library).

Disabling the share

To disable the share simply put a hash character (#) at the beginning of the line in the **/var/lib/radiod/share** file as shown in the example below. Alternatively remove the share file altogether.

```
# mount -t cifs -o username=guest,password=guest  //192.168.2.6/music /share
```

Further information

For your information if you display the **/var/lib/mpd/music** directory you will see two soft links to the **/share** and **/media** directories for the network drive and USB stick respectively.

```
# ls -la /var/lib/mpd/music/
total 8
drwxr-xr-x 2 root root  4096 May 19 11:17 .
drwxr-xr-x 4 mpd  audio 4096 May 16 19:02 ..
lrwxrwxrwx 1 root root     6 May 19 11:17 media -> /media
lrwxrwxrwx 1 root root     6 May 19 11:17 share -> /share
```

These links are created automatically by the Radio program. If these are missing they can be re-created with the ln -s command.

```
# cd /var/lib/mpd/music
# ln -s /media
# ln -s /share
```

This shouldn't normally be necessary as the links are created by the program when it creates the media and share mount points.

Source files

The source consists of several source modules all written in Python using Object Orientated techniques. The source will be visible in the **/usr/share/radio** directory once the Radio package has been installed. The radio Debian package is available at http://www.bobrathbone.com/pi_radio_source.htm .

For those who want to develop their own product all source is also available from Github. See *Downloading the source from* github on page 108.

The LCD Class

The LCD *lcd_class.py* class handles all of the LCD display routines. It contains simple commands to display and scroll lines of text on the HDD44780 2 x 16 LCD or 4 x 20 characters LCD. It is a useful standalone class that can be used in other projects. It is based on the routines from Matt Hawkins.

The Radio Daemon

There are four versions of this program for the HDD44780 LCD (directly wired to the GPIO pins). See

The *radiod.py* source provides the logic for operating the radio. It reads the push button switches and configuration files, loads the music files and radio stations. It is used with a 16 character two line LCD.

The *radio4.py* source provides the same logic for operating the radio but for a 20 character four line LCD.

The *rradiod.py* source is the version used with rotary encoder switches. It is used with a 16 character two line LCD.

The *rradio4.py* source is the version used with rotary encoder switches. It is used with a 20 character four line LCD.

The *retro_radio.py* source is specifically for converting a vintage radio to an internet radio. It supports rotary encoders and does not have an LCD dislplay.

The Adafruit Radio daemon

If you are an Adafruit RGB-backlit LCD plate for Raspberry Pi then the following programs are used:

ada_radio.py	The radio daemon for the Adafruit LCD plate.
ada_lcd_class.py	The LCD class using an I2C interface (Also interfaces the switches).
i2c_class.py	The IC2 class courtesy of Adafruit Industries (renamed).
test_ada_lcd.py	Test Adafruit LCD and switches.

The LCD with I2C backpack

The radio supports the Adafruit LCD backpack using the I2C interface. See the section called *Construction using an I2C* LCD backpack on page 42 It is used by the rradiobp.py 2 line LCD) and rradiobp4.py (4 line LCD) programs only.

The Daemon class

The *radio_daemon.py* code allows the radio program to run as a background daemon. It allows start, stop, restart, version and status commands.

The Radio class

The *radio_class.py* contains the actual commands that interface to the Music Player Daemon (MPD).

The Rotary class

The *rotary_class.py* configures and handles the interrupts (events) for the rotary encoders. It is used by the *rradiod.py* and *rradio4.py* programs. Also see the alternative below.

The alternative Rotary class

The *rotary_class_alternative.py* file is an alternative rotary encoder class. Certain Rotary Encoders will not work with the current version of the Rotary class, for example those from **TT Electronics**. To use this alternative rotary class modify the **rotary_class** parameter in the **/etc/radiod.conf** file from **"standard"** to **"alternative"**. Save the file and restart the radio.

```
# Some rotary encoders do not work well with the standard rotary class
# Rotary encoder driver. Set to "alternative" to use the alternative rotary
encoder class
rotary_class=standard
```

The Log class

The *log_class.py* routine provides logging of events to **/var/log/radio.log** file.

The Configuration Class

The **config_class.py** reads and stores the radio configuration from the **/etc/radiod.conf** file

The RSS class

The *rss_class.py* routines allow sequential gets from an RSS feed. These feeds are provided from news providers such as the BBC. This class gets the RSS feed defined in the **/var/lib/radiod/rss** file.

The Translate class

The *translate_class.py* is used to convert special international character sets (particularly from RSS feeds). It does this by first converting them to escape sequences and then to displayable ascii characters (These will show up in DEBUG logging). These *ascii* characters are then passed to the LCD class where they may are converted again to a valid character in the standard LCD character set.

LCD test programs

The *lcd_test.py* program provides some simple code to test the LCD. The *test_ada_lcd.py* program is used to test the LCD if you are using an Adafruit RGB-backlit LCD plate for the Raspberry PI.

Switch test programs

The *test_rotary_class.py* program can be used to check the rotary encoders.
The *test_switches.py* program is used to test the push button wiring.

Make sure you use the correct one for the radio that you have built.

The create_m3u program

The *create_m3u.py* program creates playlist files in the **/var/lib/mpd** directory using a list of web links (URLs) with titles as input. The original *create_playlists.py* program created PLS files but the new standard playlists with MPD is to use M3U files. The operation of the *create_m3u.py* program is covered in detail in the section on managing playlist files on page 90.

The display_current program

The *display_current.py* program is a small diagnostic program which displays the information for the current radio station or track. It is only used for trouble-shooting and it will not normally be used.

The display_model script

The *display_model.py* program displays the revision, cpu, memory and maker (If known) of the board. It is only used for trouble-shooting and it will not normally be used.

The select_daemon.sh script

The *select_daemon.sh* script is normally called during installation of the Radio Debian package but may be run by the user at any time. It selects the correct board revision and radio program variant.

The select_audio.sh script

The *select_audio.sh* script selects and configures the Audio output. It currently supports selection of the on-board audio jack, HDMI output, USB DAC, HiFiBerry and IQAudio DACs.

The remote control daemon

The remote control daemon consists of the *remote_control.py* and the *rc_daemon.py* program files. The *remote_control.py* program is for use with the LCD version of the radio and the *piface_remote.py* program is used with the PiFace CAD board.

There is a service start stop script called */etc/init.d/pifacercd*. This is configured for the correct program by the *select_daemon.sh* program during installation. The *udp_server_class.py* program is used for communication between the remote control daemon and the radio program.

The UDP network communications class

The remote control daemon uses the *udp_server_class.py* program which communicates over the local TCP/IP network using UDP port 5100 as the default; however the port is configurable in **/etc/radiod.conf**. When a button is pressed on the remote control this program sends the button identity (See Table 10 Remote Control Key names) to a UDP server running in the radio program.

Button press → IR remote control daemon → UDP message over network → Radio program.

The language class

The **language_class.py** provides the text and interface for the espeak package. It reads the **/var/lib/radiod/language** file (if present) and uses the espeak package to speak sation/track information, menu's and time.

The Status LED class

The status_led_class.py is called in the **retro_radio.py** software for use with a vintage radio. A Red Blue Green LED is driven to indicate status of the radio as there is no LCD screen. See the Raspberry Pi Vintage Radio supplement.

The Menu Switch class

The **menu_switch_class.py** code supports a 8 position rotary switch (Not encoder) as an alternative method of operating a simple menu system. It is used with the **retro_radio.py** software for use with a vintage radio.

Downloading the source from github

This is only of interest if you wish develop your own version of the Raspberry PI radio based upon the mainstream source code. Otherwise simply install the Install the Radio Daemon the radio software as shown on page 58. You can view the Raspberry PI source at https://github.com/bobrathbone/piradio

Note: This may be out of date compared to the latest version.

Before you can download the source from **Github** it is necessary to install **git**. For more information on **git** see http://en.wikipedia.org/wiki/Git_(software)

Install **git** with the following command:

```
$ sudo apt-get install git
```

Make a development directory and change to it:

```
$ mkdir /home/pi/develop
$ cd /home/pi/develop
```

Now clone the github piradio repository:

```
$ git clone git://github.com/bobrathbone/piradio
Cloning into 'piradio'...
remote: Counting objects: 71, done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 71 (delta 13), reused 64 (delta 9)
Receiving objects: 100% (71/71), 185.33 KiB | 334 KiB/s, done.
Resolving deltas: 100% (13/13), done.
```

This will create a sub-directory called 'piradio' which will contain the entire source. Also in the **/home/pi/develop/piradio** directory you will also see a directory called **.git** (dot-git). This is the control directory for **git**.

Note: Don't forget that if you use the **service radiod stop|start** commands that this will start and stop the software in contained in **/usr/share/radio** (If you installed from the package).

You will not necessarily need to use **git** any further unless you wish to save your changes under **git** control. To find out more about **git** and for general support and documentation see http://git-scm.com

The files to build the packages are contained in compressed tar files. These are *piradio_build.tar.gz* and *piradio_web_build.tar.gz* for the radio and web software respectively.

Contributors code

The **contributors** directory contains software contributed by other constructors. The code in these sub-directories is provided "as is" and without warranties. Neither can any guarantees be given that the software in these sub-directories will be compatible with future releases of the main-stream software.

Note: Absolutely no support is provided for this code. However a useful README file is included each sub-directory.

Miscellaneous

Simple tone regulator

It may be that you wish to fit a tone regulator to the radio. Below is one option.
The following diagram and modified text came from Jack Orman at:
http://www.muzique.com/lab/swtc.htm

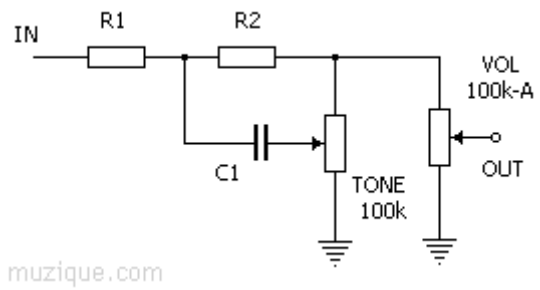


Figure 86 Simple tone control circuit

This tone control circuit that has a response that can be altered from high cut to high boost as the knob is turned. The output resistance is constant so the volume does not vary as the tone control is adjusted.

Suggested values for beginning experimentation with are R1=10k, R2=47k, C1=0.022uF and 100k for the tone and volume pots.

Figure 87 Dual 100K Linear potentiometer

Remember that the above circuit needs to be duplicated for right and left audio channels. This also means purchasing a dual linear 100K potentiometer for the tone control.

Note that the above circuit has a lot of attenuation of the audio output so using the onboard audio output of the Raspberry Pi might result in a disappointing level of volume. It is recommended to use a sound output DAC or USB sound dongle.

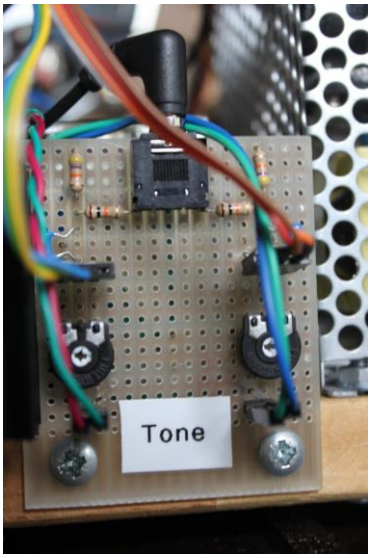


Figure 88 Tone control board

The illustration on the left shows a simple passive tone regulator board using the above circuit.

The audio output from the Raspberry Pi or DAC is fed into the board via a standard Audio socket.

Below the input are the connections to the tone regulator potentiometer mounted on the front panel of the radio.

Below the potentiometer connections are the two 100K presets for adjusting the output level to the Audio Amplifier.

Below these the Left and Right audio outputs connect to thA Amplifier.

Using the Adafruit backlit RGB LCD display

The Adafruit backlit RGB LCD has three LED backlights (Red, Blue and Green) which can either be switched on individually or in various combinations together as shown in the table below:

Table 14 Adafruit backlit RGB display wiring

Switch pin	Red (Pin 16)	Green (Pin 17)	Blue (Pin 18)	Colour	Diodes required
1	0	0	0	Off	0
2	0	0	1	Blue	0
3	0	1	0	Green	0
4	0	1	1	Light Blue	2
5	1	0	0	Red	0
6	1	0	1	Purple	2
7	1	1	0	Yellow	2
8	1	1	1	White	3
Common	GND			Total diodes	9

Figure 89 IN4148 diode

The diodes used are any low voltage low current diodes such as the IN4148. So to use all of the above combinations would require a single pole 8 way rotary switch and logic and nine diodes. The first switch position is off.

- Do not wire anything to position 1.
- Wire pin 16 (Blue) of the LCD backlight to switch position 2.
- Wire pin 17 (Green) of the LCD to switch position 3
- Wire pin 18 (Red) of the LCD to switch position 5
- Wire pin 17 and 18 via two diodes to pin 4 to give the colour light blue
- Do the same for the other two colour combinations
- Wire pin 16, 17 and 18 to pin 8 via three diodes to give the colour white
- Wire the centre pin of the switch to 0v (GND)

Using a 4 line by 16 character LCD

There are various LCDs on the market including a 16 character four line version.
To use this type of display modify the **lcd_class.py** program file as shown below:

```
LCD_LINE_3 = 0x94 # LCD RAM address for the 3rd line
LCD_LINE_4 = 0xD4 # LCD RAM address for the 4th line
# Some LCDs use different addresses (16 x 4 line LCDs)
# Comment out the above two lines and uncomment the two lines below
# LCD_LINE_3 = 0x90 # LCD RAM address for the 3rd line
# LCD_LINE_4 = 0xD0 # LCD RAM address for the 4th line
```

So the new lines are:

```
LCD_LINE_3 = 0x90 # LCD RAM address for the 3rd line
LCD_LINE_4 = 0xD0 # LCD RAM address for the 4th line
```

Also edit either the **radio4.py** or **rradio4.py** program depending on which you are using. Find the lcd.setWidth(20) line and amend it to:

```
lcd.setWidth(16)
```

Troubleshooting

Also see the section called *Using the diagnostic programs* on page 121. If you need to create a log file in DEBUG mode see the procedure for doing this on page 124.

The Raspberry Pi will not boot

This is always worrying if this happens but doesn't always mean that the situation is irrecoverable. It often indicates a SD card corruption problem. Connect the HDMI output of the Raspberry Pi to the HDMI input of a TV set. Reboot the Pi. If you see the following:

```
[....] An automatic file system check (fsck) of the root filesystem failed.
A manual fsck must be performed, then the system restarted. The fsck should
be performed in maintenance mode with the root filesystem mounted in read-on
[FAIL] ... failed!
[warn] The root filesystem is currently mounted in read-only mode. A
maintenance shell will now be started. After performing system maintenance,
please CONTROL-D to terminate the maintenance shell and restart the system.
... (warning).
sulogin: root account is locked, starting shell
root@raspberrypi:~#
```

Then run the following command:

```
# fsck -y /dev/mmcblk0p2
```

This will, with luck, correct the file system. When **fsck** has finished reboot the system. Once rebooted use **vi** or **nano** to modify **/etc/default/rcS**. Add the following line to **/etc/default/rcS**.

```
# automatically repair filesystems with inconsistencies during boot
#FSCKFIX=no
FSCKFIX=yes
```

Finally reboot the Raspberry Pi

```
$ sudo reboot
```

Trouble shooting problems with MPD

Most problems are due to the following:

- Incompatibility with the **pulseaudio** package
- Sound mixer volume set to zero or very low volume
- Incorrect setup of the **/etc/mpd/mpd.conf** file
- If using a sound card, no driver loaded in **/boot/config.txt**
- The Raspberry Pi audio is configured to use the HDMI output

Remove the **pulseaudio** package. This should have bee removed already by the select_audio.sh program.

```
$sudo apt-get remove pulseaudio
```

Check the audio jack cable first before doing anything else.

The **/var/log/mpd/mpd.log** file is the place to look first if all other things seem normal.

If using the onboard audio output there should not be a problem. The standard **/etc/mpd.conf** settings should be OK. Only if pulse audio is not removed or the Alsa mixer volume is not set can it lead to lack of sound. See *Music Player Daemon Installation* on page 57.

If using a USB or HiFiBerry DAC:

1. Check to see if the DAC is visible using **aplay**
2. Check that the **/etc/mpd.conf** is correctly configured.
3. Check that the mixer volume is correctly set.
4. For HiFiBerry DAC ensure **/boot/config.txt** contains the correct **dtoverlay** statement.

See *Configuring a USB sound devices* on page 66 and *Configuring a HiFiBerry* on page 68

LCD screen not working

Check that the wiring conforms to the *wiring list* on page 23. Make sure that pin 3 is grounded (0V) to give maximum contrast or if a contrast potentiometer is fitted then make sure it is at the maximum setting. If you are using the Adafruit LCD plate the make sure that you are running the **ada_radio.py** program and not one of the other programs (See Table 1 on page 24).

Run the *lcd_test.py* program to see if the LCD displays anything. This runs independently of any other software and can be used stand alone.

The LCD only displays hieroglyphics

This can be caused either by incorrect wiring of the LCD or the incorrect selection of the board revision during installation. This problem has also been experienced with faulty LCD hardware particularly when re-booting the Raspberry PI.

Check the wiring conforms to the *wiring list* on page 23. In particular check the data lines to pins 11, 12, 13 and 14 (See LCD wiring on page 30). Retest the LCD using the *lcd_test.py* program.
If the wiring is correct run the *select_daemon.sh* script to select the correct revision of the board and restart the program.

The LCD displays hieroglyphics or goes blank occasionally

If the LCD is normally working OK but goes wrong when switching on and off lights this is due to Electromagnetic Interference (EMI). See the section called *Preventing electrical interference* on page 34.

LCD backlight not working

Check that pins 15 and 16 of the LCD display have +5V and 0V(GND) respectively.
See on page 30.

LCD only displays dark blocks on the first line

This is normal when the raspberry PI starts up. The display should work with the *lcd_test.py* program. If the *lcd_test.py* program still doesn't display anything then check that the wiring conforms to the *wiring list* on page 27. If you are using the Adafruit LCD plate the make sure that you are running the **ada_radio.py** program and not one of the other programs (See Table 1 on page 24). *select_daemon.sh* script to select the correct radio daemon.

MPD fails to install

During installation of MPD some files return a 404 error (Not found) the following message is seen.

```
Unable to fetch some archives, maybe run apt-get update or try with –fix-
missing?
```

This is due to that an update was not previously carried out as shown in the section called *SD card creation* on page 52. Perform the update and upgrade as shown and re-install MPD and MPC.

Music Player Daemon won't start

The MPD daemon logs to the **/var/log/mpd/mpd.log** file. Examine this file for errors. The MPD daemon is dependant on good M3U files so check that these are correct as described in the section called *Creating and Maintaining* Playlist files on page 90.

The MPD program may display a socket error

When starting the MPD daemon the following message is seen:

```
Starting Music Player Daemon: mpdlisten: bind to '[::1]:6600' failed: Failed
to create socket: Address family not supported by protocol (continuing
anyway, because binding to '127.0.0.1:6600' succeeded)
```

If this message is seen in the MPD log file this is simply because IP version 6 (IPv6) isn't installed so the message doesn't affect operation of the MPD.

To prevent it from happening configure the *bind_to_address* parameter in the **/etc/mpd.conf** file to "any". The installation procedure should normally set this anyhow.

The LCD displays the message "No playlists"

Cause: There are no playlists found in **/var/lib/mpd/playlists**. Check to see if there are any playlists in **/var/lib/mpd/playlists.** You should see files with the **m3u** extension.

```
$ ls  /var/lib/mpd/playlists/
BBC_stations.m3u   Dutch_stations.m3u  German_Stations.m3u
Jazz_stations.m3u
Country_music.m3u  France.m3u          Italian_Stations.m3u
```

If no m3u playlist files are present then run the **create_m3u.py** program.

```
$ sudo ./create_m3u.py –delete_old
:
:
New radio playlist files will be found in /var/lib/mpd/playlists/
```

Restart the radio.

Constant alternate display of Station Name and Volume

Problem: The LCD screen continually switches between displaying station name and volume. Also the radio log file displays "ERROR radio._setVolume error vol=nn" where nn is the volume level. This is due an incompatibility with the **pulseaudio** package.

Solution: Remove the **pulseaudio** package.

```
$ sudo apt-get remove pulseaudio
```

The MPD daemon complains about the avahi daemon

The following message is seen in the **/var/log/mpd/mpd.log** file

```
Apr 10 15:37 : avahi: Failed to create client: Daemon not running.
```

Change the *zeroconf_enabled* parameter in the **/etc/mpd.conf** file to "no" . This is normally set in the radio package installation procedure. The *avahi* daemon is used to configure systems without a network connection but is not enabled by default. It is not required for this design.

Buttons seem to be pressing themselves

The symptoms are that it looks like buttons are generating their own signals i.e. they appear to being continually pressed although they are not being operated. In particular the MENU button displays this problem. This is because the inputs are "floating". All inputs for the button operated radios (Not Adafruit plate) need to be pulled down to ground using a 10K resistor for version 1 boards. Newer version 2.0 boards have inbuilt pull-up/pull-down resistors. The radio software enables the internal pull-down resistors so doesn't require external resistors. Use the very latest version of the software to eliminate this problem. Use the **test_switches.py** software to test the buttons.

Radio daemon doesn't start or hangs

This is almost certainly a problem with either the MPD daemon or failed internet connection."Check the network connection and run installation tests on the MPD daemon. Occasionally a bad playlist file can also cause this problem. You can check that your Raspberry PI has an internet connection with the *ip addr* command. The example below shows interface *eth0* connected as IP 192.168.2.22.

```
# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
    link/ether b8:27:eb:fc:46:15 brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.22/24 brd 192.168.2.255 scope global eth0
```

Stream decode problems

The radio may display a message similar to the following:

```
ERROR: problems decoding http://173.244.194.212:8078
```

This is due to an invalid URL (In the above example this is http://173.244.194.212:8078) in one of the M3U files.

Locate the offending URL in the play list file in the **/var/lib/mpd/playlists** directory. Either correct the radio stream URL or remove it all together.

Also check that the file URL is not the pointer to the playlist file (See section Creating and Maintaining Playlist files on page 90.

Cannot mount remote network drive

There are just too many possibilities to cover all of these here. However a few common problems are covered here:

Error: mount error(115): Operation now in progress
Cause: Most likely an incorrect IP address

Error: NFS mount hangs
Cause: Most likely an incorrect IP address

Error: mount.nfs: access denied by server while mounting <ip address>:/music
Cause: The volume name is missing – for example /volume1/music

Error: mount error(16): Device or resource busy
Cause: The share mount directory is in use because a mount has already been done. Run the umount command.

Error: mount error(2): No such file or directory
Cause: The path specified in the mount doesn't exist

Error:
mount.nfs: rpc.statd is not running but is required for remote locking.
mount.nfs: Either use '-o nolock' to keep locks local, or start statd.
mount.nfs: an incorrect mount option was specified
Cause:
You need to include the "–o noclock" option.
If the error isn't in the above list then search the web for suggestions.

Button or Rotary encoder problems

Use the *test_switches.py* or test*_rotary_class.py* to test the push buttons or rotary encoders respectively.

Rotary encoders not working

Check wiring in particular the common pin must be connected to ground (and not 3.3 volts). Run the **test_rotary_class.py** to test the switches.
If you see a message similar to the following then you are using the wrong GPIO libraries most likely because you haven't installed the correct version of Raspbian Jessie on the SD card as shown on page 52:

```
Traceback (most recent call last):
  File "test_rotary_class.py", line 38, in <module>
    volumeknob =
RotaryEncoder(LEFT_SWITCH,RIGHT_SWITCH,MUTE_SWITCH,callback)
  File "/home/pi/Projects/radio/rotary_class.py", line 39, in __init__
    GPIO.add_event_detect(self.pinA, GPIO.FALLING,
callback=self.switch_event)
AttributeError: 'module' object has no attribute 'add_event_detect'
```

Install the latest version of Raspbian Jessie as shown on page 52.

If you are running an older version Raspbian Jessie you can connect to the Internet and run the two following commands as user root.

```
# apt-get update
# apt-get upgrade
```

This can take some time. Reboot once the upgrade is finished.

Volume control not working with DAC or USB speakers

This is hardware dependant. Not all USB hardware and drivers work with mixer type “hardware”. If this problem is being experienced try setting the **mixer_type** parameter to “software”. Edit the /etc/mpd.conf file and change the mixer type to software.

```
mixer_type    "software"
```

Remove the # at the beginning of the line to enable software mixing and save the file. Restart the radio software.

Note: This solution was provided by one of the constructors and is untested by the author.

Noisy interference on the radio

If there is noise interference when playing the radio and this is still present even when the radio is muted this can be for several reasons. This can happen with a wired Ethernet connection and a Wi-Fi dongle are connected to the Raspberry Pi and the Ethernet activity is being picked up by the Wi-Fi dongle. This can be cured by using either a wireless adapter or the Ethernet connection and not both. Another common cause can be an inadequate power supply. See on Power supply considerations on page 33. Unfortunately the later 40 pin versions of the Raspberry Pi seam to be more prone to interference. The recommendation is to use a USB DAC or a HiFiBerry DAC plus.

Humming sound on the radio

This is usually due to a ground loop somewhere in the design. See the section called Preventing ground loops on page 35.

Music is first heard at boot time then stops and restarts

This only happens if the MPD daemon has been enabled to start at boot time. The reason that music is initially and then stops is because the MPD daemon is started at boot time and restarted when the radio software starts. To disable this behaviour use the following:

```
$ sudo update-rc.d mpd disable
```

This will stop MPD starting at boot time. Starting of the MPD daemon is completely controlled by the radio software. The radio package installation procedure now automatically disables the Music Player Daemon at boot time.

USB device won’t play

The **/var/log/mpd/mpd.log** file shows the following message:

```
<date> : mixer: Failed to set mixer for 'My ALSA Device': failed to set ALSA
volume: Invalid argument
```

This can happen with certain USB devices. The radio may start but stops almost immediately and displays the “Radio stopped” message on the LCD screen. The MPD daemon if run on its own plays OK but the volume can’t be changed using the **mpc volume** command.

If problems are experienced with your USB device (Tenx Technology for example) then add the **mixer_type “software”** parameter to the **/etc/mpd.conf** file.

```
audio_output {
        type            "alsa"
        name            "MyUSB DAC"
        device          "hw:1,0"        # optional
        format          "44100:16:2"    # optional
        #mixer_device    "default"       # optional
        #mixer_control   "PCM"           # optional
        #mixer_index     "0"             # optional
        mixer_type      "software"      # Add this line for USB devices
}
```

Unexpected message during an upgrade

It is possible one of the files has been changed in a new package. For example:

```
 Configuration file `/etc/logrotate.d/radiod'
 ==> Deleted (by you or by a script) since installation.
 ==> Package distributor has shipped an updated version.
   What would you like to do about it ?  Your options are:
    Y or I  : install the package maintainer's version
    N or O  : keep your currently-installed version
      D     : show the differences between the versions
      Z     : start a shell to examine the situation
 The default action is to keep your current version.
*** radiod (Y/I/N/O/D/Z) [default=N] ? Y
Installing new version of config file /etc/logrotate.d/radiod ...
Executing post install script /var/lib/dpkg/info/radiod.postinst
update-rc.d: using dependency based boot sequencing
```

If you see this enter a Y to install the new file unless you have a good reason not to do so.

Reboot hangs if radiod running

This is a known problem in Debian Jessie. This is because of the radio software is not currently compatable ith **systemd**. The **systemd** daemon is responsible for running start stop scripts including **radiod**. To solve this problem install **sysvinit-core**.

```
$ sudo apt-get install sysvinit-core
```

and reboot the Raspberry Pi. This bug also affects the status function.

Missing logrotate or other configuration files

If upgrading or reinstalling the radiod package you must use the --force-confmiss flag.

```
$ sudo dpkg --install --force-confmiss radiod_5.6_armhf.deb
```

This will re-install any missing configuration files.

IR remote control problems

The irrecord program complains that lircd.conf already exists

When running the following command:

```
$ sudo irrecord -f -d /dev/lirc0 /etc/lirc/lircd.conf
```

The following message is displayed:

```
irrecord: file "/etc/lirc/lircd.conf" already exists
irrecord: you cannot use the --force option together with a template file
```

This is because the existing /etc/lirc/lircd.conf has not been moved out the way. Run the following command:

```
$ sudo mv /etc/lirc/lircd.conf /etc/lirc/lircd.conf.org
```

See the instructions in the section called *Installing the Infra Red sensor software* on page 73.

The irrecord cannot open /dev/lirc0

If the following is seen when running the **irrecord** program:

```
irrecord: could not open /dev/lirc0
irrecord: default_init(): Device or resource busy
irrecord: could not init hardware (lircd running ? --> close it, check
permissions)
```

Run the following command and retry the command:

```
$ sudo service lirc stop
```

HiFBerry DAC plus no sound

Check first if the card is visible using the **aplay** command. Card 1 should be visible.

```
# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: ALSA [bcm2835 ALSA], device 0: bcm2835 ALSA [bcm2835 ALSA]
:
card 0: ALSA [bcm2835 ALSA], device 1: bcm2835 ALSA [bcm2835 IEC958/HDMI]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: sndrpihifiberry [snd_rpi_hifiberry_dacplus], device 0: HiFiBerry
DAC+ HiFi pcm512x-hifi-0 []
  Subdevices: 0/1
```

```
  Subdevice #0: subdevice #0
```

If card 1 is not visible check the following:
Check that the B output of the channel rotary encoder is wired to GPIO 10 (pin 19) and not GPIO 18 (pin 12). GPIO18 is used by the DAC plus. Also check that the **down_switch** parameter in **/etc/radiod.conf** is set to 10 (Comment out down_switch=18) to reflect the actual wiring.

```
#down_switch=18
down_switch=10
```

Make sure that the /boot/config.txt file contains the following line.

```
dtoverlay=hifiberry-dacplus
```

Finally modify the **audio_output** section in **/etc/mpd.conf**.

The Device parameter should point to card 1

```
audio_output {
   type      "alsa"
   name      "DAC"
   device    "hw:1,0"
   #format "44100:16:2" # optional
   mixer_device  "PCM"
   mixer_control "PCM"
   mixer_type   "software"
}
```

Reboot the Raspberry Pi and retest.

Using the diagnostic programs

A number of diagnostic programs are supplied to help troubleshoot problems or provide extra system information. These are:

- **lcd_test.py** Test the LCD screen (Directly wired to the GPIO)
- **test_ada_lcd.py** Test the Adafruit LCD plate and buttons
- **test_i2c_lcd.py** Test Adafruit LCD with I2C backpack or PCF8745 backpack
- **test_switches.py** Test push button switches (Directly wired to the GPIO)
- **test_rotary_class.py** Test rotary encoders (Directly wired to the GPIO)
- **display_model.py** Display Raspberry Pi model information
- **display_current.py** Display current station or track details

All diagnostic programs are supplied in the **/usr/share/radio** directory. Change to this directory first!

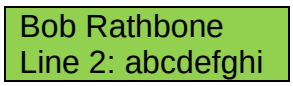
```
$ cd /usr/share/radio
```

All programs require the **./** characters in front of the name to execute. All of those using the GPIO interface also require to be called with **sudo**.

The test_lcd, test_i2c_lcd and test_ada_lcd programs

```
$ sudo ./test_lcd.py
```

The above program will display the following text on the LCD:



Line 2 scrolls the alphabet followed by 0 through 9.
The *test_ada_lcd.py* program does the same except it also prints a message on the console screen when a button is pressed.

The test_switches program

Test switches directly wired to the GPIO pins.

```
$ sudo ./test_switches.py
down_switch
up_switch
right_switch
left_switch
menu_switch
```

Pressing the switches should show on the screen as shown in the above example.

The test_rotary_class.py program

This program does a simple test of the switches. It uses the rotary_class.py code.

```
$ sudo ./test_rotary_class.py
Are you using an old revision 1 board y/n: n
Use Ctl-C to exit
Volume anticlockwise 3
Volume clockwise 1
Volume button down 3
Volume button up 4
Tuner clockwise 1
Tuner anticlockwise 3
Tuner button down 3
Tuner button up 4
^C
Exit
```

The remote_control program

The **remote_control.py** program listens on the IR interface for commands from the Remote Control. The actual IR interface makes use of PiFace CAD software, see http://piface.github.io/pifacecad/. The **remote_control.py** program is started from the **pifacercd** service. It then passes commands to the radio program. The **remote control** program provides complete control of the radio and can change menu options, do searches etc. just as the same as the knobs or buttons. It normally communicates with the radio program using UDP port 5100 on the local network interface.

The display_model program

This program displays the Raspberry PI model details.

```
$ ./display_model.py
000e: Model B, Revision 2.0, RAM: 512 MB, Maker: Sony
```

In this example 000e=Manufacturers revision, B=Model, 2.0=Revision, 512MB RAM, Maker=Sony. If you are unsure of the model or revision of the Raspberry PI use this program to find this out.

The display_current program

This is a useful diagnostic that prints out the raw information available from the MPD daemon. The radio daemon uses the same libraries as this test program.

```
$ ./display_current.py

id: 32
pos: 7
name: Blues Radio UK
file: http://206.217.213.16:8430
title: 04 Billy Jones Blues - I'm A Bluesman
current_id 8

Status
songid: 32
playlistlength: 25
playlist: 31
repeat: 0
consume: 0
mixrampdb: 0.000000
random: 0
state: play
xfade: 0
volume: 75
single: 0
mixrampdelay: nan
nextsong: 8
time: 22:0
song: 7
elapsed: 22.400
bitrate: 96
nextsongid: 33
audio: 32000:24:2

uptime: 28
db_update: 1400354144
artists: 228
playtime: 23
albums: 132
db_playtime: 302046
songs: 1297
```

To find out the exact meaning of all these fields please refer to the standard **python-mpd** documentation at https://pypi.python.org/pypi/python-mpd/ or http://pythonhosted.org/python-mpd2/topics/getting-started.html .

Running the radio program in nodaemon mode

If for some reason the radio program stops or crashes without explanation (particularly if you have modified the code), it can be extremely difficult to see what is happening as the radio software runs as a so called system daemon. There is a way to run the software in foreground mode. In this case stop the radio and change to **/usr/share/radio** directory and run the radio program with the

nodaemon option. Use the name of the radio program you are using. For example if the variant is Adafruit the run:

```
$ cd /usr/share/radio
$ sudo ./ada_radio.py nodaemon
```

If the program crashes it will display a stack trace which will give the file name and line numbers where the program crashed. Use **Control-C** to exit **nodaemon** mode (This will also display a stack trace which is normal and is not an error).

Creating a log file in DEBUG mode

You may be asked to supply a log file in DEBUG mode for support purposes.
Stop the radio and remote control if running:

```
$ sudo service radiod stop
$ sudo service pifacercd stop
```

Edit the **/etc/radiod.conf** file and switch on DEBUG mode as shown below.

```
# loglevel is CRITICAL,ERROR,WARNING,INFO,DEBUG or NONE
loglevel=DEBUG
```

Remove the old log file

```
$ sudo rm /var/log/radio.log
```

Start the radio and remote control if required:

```
$ sudo service radiod start
$ sudo service pifacercd start
```

Operate the radio including the operation you are having with.
Send the **/var/log/radio.log** file to bob@bobrathbone.com

Switch off DEBUG mode by editing the **/etc/radiod.conf** file and as shown below.

```
loglevel=INFO
```

Displaying information about the Raspberry Pi

There are a number of standard facilities to provide information about the Raspberry Pi which may be useful when diagnosing a problem.

Displaying information about the Operating system

Display **/etc/os.release** using the **cat** command

```
$ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 8 (jessie)"
NAME="Raspbian GNU/Linux"
VERSION_ID="8"
VERSION="8 (jessie)"
ID=raspbian
ID_LIKE=debian
```

```
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL=http://www.raspbian.org/RaspbianBugs
```

Display the kernel details

Display the kernel version using **uname**.

```
$ uname -a
Linux raspberry 4.1.19+ #858 Tue Mar 15 15:52:03 GMT 2016 armv6l GNU/Linux
```

Displaying the GPIO information

The **gpio readall** command (Jessie only) can be used to display the GPIO configuration.

```
 +-----+-----+---------+------+---+-Model B2-+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM | Switch
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 0 | IN   | TxD     | 15  | 14  | Left
 |     |     |      0v |      |   |  9 || 10 | 0 | IN   | RxD     | 16  | 15  | Right
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |Up/Down
 |  27 |   2 | GPIO. 2 |  OUT | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |  OUT | 0 | 15 || 16 | 0 | OUT  | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | OUT  | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     | Down*
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  | Menu
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 0 | OUT  | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | OUT  | CE1     | 11  | 7   |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |  28 |  17 | GPIO.17 | ALT2 | 0 | 51 || 52 | 0 | ALT2 | GPIO.18 | 18  | 29  |
 |  30 |  19 | GPIO.19 | ALT2 | 0 | 53 || 54 | 0 | ALT2 | GPIO.20 | 20  | 31  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+-Model B2-+---+------+---------+-----+-----+
```

* Alternative down switch for HiFiBerry DAC+ compatibility.

The physical pins are shown in the center numbered 1 to 26 and 51 through 54 (for a model B2) in this example. The above output is for a radio using a directly wired LCD and push buttons.

For example:
Physical pin 8 is BCM GPIO 14 and mode is configured as an input for the left switch and is currently low (V column is 0).

This command only works by default on **Raspbian Jessie**. Also the output of this command will be different for each variant of the radio.

To install the **gpio** utility on Raspian Wheezy see:
https://projects.drogon.net/raspberry-pi/wiringpi/download-and-install/

Configuring a wireless adaptor

You will almost certainly want to configure a wireless adaptor for the radio instead of a wired network connection. Choose a wireless adapter that has been approved for the Raspberry PI or use the model 3B with in-built WiFi adapter. See the following link for approved Raspberry PI peripherals: http://elinux.org/RPi_VerifiedPeripherals

Note: The model 3B WiFi adapter requires Jessie or Jessie Lite from the 26th of February onwards.

Install the wireless adapter

Switch off the Raspberry PI and plug in the adaptor into one of the USB ports. Power the PI back on and log in. Check to see if your wireless adapter has been recognised by running the **lsusb** command.

```
pi@raspberrypi:~$ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 148f:5370 Ralink Technology, Corp. RT5370 Wireless
Adapter
```

The above shows a Ralink (Tenda) wireless adaptor but this will vary depending on the adapter that has been installed.

Configure the adaptor

The configuration is contained in the **/etc/network/interfaces** file as shown below

```
$ cat /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

iface eth0 inet manual

allow-hotplug wlan0
iface wlan0 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug wlan1
iface wlan1 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

You should not need to change this file unless you wish to configure a static IP address (See *Configuring a static IP address* on page 128). The file to be amended is shown on the line beginning with wpa-roam and is **/etc/wpa_supplicant/wpa_supplicant.conf**. Edit this file.
It will only contain a couple of lines.

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
```

Add the following after the above lines:

```
 network={
    ssid="YOUR_SSID"
    scan_ssid=1
    psk="YOUR_KEY"
    proto=RSN
    key_mgmt=WPA-PSK
    pairwise=CCMP
    auth_alg=OPEN
}
```

Substitute YOUR_SSID and YOUR_KEY with the actual SSID and key for your wireless router. The above configuration is for a router using WPA encryption. If your router is using the older WEP encryption then you will need to adapt the configuration to use WEP. See next section.

Explanation of the network fields

Field	Description
ssid	your WIFI (SSID) name
scan_ssid	A value of 1 means broadcast and value of 2 means a hidden SSID (Normally enter a value of 1)
psk	Your WIFI password
proto	Your choice of RSN or WPA. RSN is WP2 and WPA is WPA1. (most configurations are RSN)
key_mgmt	Either WPA-PSK or WPA-EAP (pre-shared or enterprise respectively)
pairwise	Either CCMP or TKIP (WPA2 or WPA1 respectively)
auth_alg	OPEN option is required for WPA and WPA2 (other option, SHARED & LEAP)

The only problem with the above configuration is that the **psk** key is in plain text and can be read by anyone who has access to the Raspberry PI. It is possible to increase security by generating a so-called passphrase with the **wpa_passphrase** command. For example if your **ssid** is *mywlan* and the WIFI password is *abcdef1234* then use the following command to generate the passphrase.

```
# wpa_passphrase mywlan abcdef1234
network={
        ssid="mywlan"
        #psk="abcdef1234"
        psk=53a566e0ccf03ec40b46e6ef4fc48b836e428fb0fd5e0df95187ba96e60ce7ce
}
```

Copy and paste the passphrase into the psk parameter into the **/etc/wpa_supplicant/wpa_supplicant.conf** file. Do not include any quotes around it. Remove the comment line which shows the original key (#psk=”abcdef1234”)

Operating the wireless interface

If configured correctly the wireless adapter will start up when the Raspberry PI is rebooted.

The adaptor can be started and stopped with the following commands:

```
root@raspberrypi:/home/pi# ifup wlan0
```

and

```
root@raspberrypi:/home/pi# ifdown wlan0
```

To see what **SSIDs** are available run the **iwlist** command as shown in the following example:

```
root@raspberrypi:/home/pi# iwlist wlan0 scanning | grep ESSID
                    ESSID:"mywlan"
                    ESSID:"VGV751926F4B9"
                    ESSID:"prime"
                    ESSID:"Sitecom6A212C"
```

Troubleshooting the wireless adapter

Problem – Starting the wireless adapter gives the following message:

```
# ifup wlan0
wpa_supplicant: /sbin/wpa_supplicant daemon failed to start
run-parts: /etc/network/if-pre-up.d/wpasupplicant exited with return code 1
Failed to connect to wpa_supplicant - wpa_ctrl_open: No such file or
directory
wpa_supplicant: /sbin/wpa_cli daemon failed to start
run-parts: /etc/network/if-up.d/wpasupplicant exited with return code 1
```

This is due to an incorrect **/etc/wpa_supplicant/wpa_supplicant.conf** file. The problem is due to an incorrect configuration. For example a space after the **ssid=** directive as shown below.

```
network={
    ssid= "homelan"
    scan_ssid=1
    psk="d762c954df"
    proto=RSN
    key_mgmt=WPA-PSK
    pairwise=CCMP
    auth_alg=OPEN
}
```

Solution: Correct the error and run the **ifup wlan0** command.

Problem: The following is seen:

```
root@raspberrypi:/home/pi# ifup wlan0
ifup: interface wlan0 already configured
```

Solution: This isn't actually an error. Just run the **ifdown wlan0** command and retry the **ifup wlan0** command. It should then work.

Configuring a static IP address

The Raspberry PI Ethernet network interface comes configured out of the box to use DHCP (Dynamic Host Configuration Protocol) when using Raspbian Jessie or a similar operating system. This means it is given an IP address by the (home) router for a particular lease period. This also means that if the Raspberry PI is switched off for any length of time it may get a different IP address from the one it had previously. This may not be very convenient especially if you are using the web interface or a mobile app to control the radio. There are two possible choices here:

1. Configure the home router so that DHCP delivers a fixed address based upon the Raspberry Pi's MAC address. This is called DHCP IP address reservation.
2. Configure the Raspberry Pi with a static IP address.

It is beyond the scope of this manual to show how to reserve DHCP IP addresses and will vary anyway between routers. Also it isn't always possible configure the router. Take a look at the following link for a good example of how to do this: http://lifehacker.com/5822605/how-to-set-up-dhcp-reservations-so-you-never-have-to-check-an-ip-address-again
Alternatively search the internet with the name of your router and the term "DHCP IP address reservation".

If DHCP IP address reservation isn't an option then it is possible to configure a so-called static IP address which will not change between reboots. To do this you must find out the IP address of your router (gateway) and netmask for your network. Do this with the **netstat** command as shown below:

```
$ netstat -r -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.1.254   0.0.0.0         UG        0 0          0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

In the above example the IP address of the gateway is 192.168.1.254. Your router's IP address will almost certainly be different. Take a note of the gateway IP address and the subnet (Genmask) of your network. In this case that is 255.255.255.0. You will need to use these values to configure the interface. The next thing you need is a free IP address in your network. There will be lots of them but approximately 50 to 100 of them will be claimed for the DHCP pool. The only way to now what DHCP is using is to log into your router and look at the configuration. If this isn't possible or you are unsure of what to do then pick one somewhere in the middle of the network range. For example in the above network you could choose 192.168.1.125. Check that it isn't already in use somewhere else in your network. Check it with the **ping** command. If you see 100% packet loss then the IP address you have chosen isn't in use in your system so you can use it.

```
$ ping -c 4 192.168.1.125
PING 192.168.1.125 (192.168.1.125) 56(84) bytes of data.
From 192.168.1.8 icmp_seq=1 Destination Host Unreachable
From 192.168.1.8 icmp_seq=2 Destination Host Unreachable
From 192.168.1.8 icmp_seq=3 Destination Host Unreachable
From 192.168.1.8 icmp_seq=4 Destination Host Unreachable

--- 192.168.1.125 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3012ms
```

Now edit the **/etc/network/interfaces** file and comment out the existing line for the eth0 interface configuration (dhcp) and add a new definition under it as shown in the following examples:

Ethernet static IP configuration

For a wired Ethernet connection use a configuration as shown below using your new static IP address.

```
iface lo inet loopback
#iface eth0 inet dhcp
iface eth0 inet static
    address 192.168.1.125
```

```
    netmask 255.255.255.0
    gateway 192.168.1.254
```

Save the file and reboot the system. After reboot; log into the Raspberry PI using the new IP address. If unable to log in connect a keyboard and screen and reboot. Log in and troubleshoot the problem.

Wireless LAN static IP configuration

Likewise it is possible to configure the wireless LAN with a static IP, but with one important difference in that **WPA roaming** no longer makes sense and must be disabled by commenting it out. Configure the **wpa-ssid** and **wpa-psk** parameters for your network in the **/etc/network/interfaces** file instead of the **/etc/wpa_supplicant/wpa_supplicant.conf** file.

.

```
#iface default inet dhcp
auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.1.126
    netmask 255.255.255.0
    broadcast 192.168.1.255
#wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
wpa-ssid "<Your-SSID>"
wpa-psk "<Your-Passphrase>"
```

To display the IP address of the Wireless Adapter run the **ip addr** command:

```
# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
    link/ether b8:27:eb:fc:46:15 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.126/24 brd 192.168.2.255 scope global eth0
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen
1000
    link/ether c8:3a:35:c8:64:cd brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.126/24 brd 192.168.2.255 scope global wlan0
```

The above command also displays the MAC address of the Ethernet and WiFi connections. For example: **c8:3a:35:c8:64:cd**

You will need this information if you wish to do DHCP IP address reservation as mentioned previously in this section.

Streaming to other devices using Icecast2

Inbuilt MPD HTTP streamer

The MPD daemon can be configured to use its own inbuilt streamer. However this requires a special MPD client such as **gmpc** on the PC. It cannot be easily accessed from a web browser. If you wish to use the inbuilt streamer see the following URL:
http://mpd.wikia.com/wiki/Built-in_HTTP_streaming_part_2

Introduction to Icecast

You may wish to play the output of the Radio through your PC speakers or a mobile device such as a tablet or telephone. This is possible to do this using **Icecast**. For more information on **Icecast** see the following Wikipedia article: http://en.wikipedia.org/wiki/Icecast .

Please also refer to *Intellectual Property, Copyright, and Streaming Media* on page 149.

Installing Icecast

Install **icecast2** using the **install_streaming.sh** script as user root user.

```
$ cd /usr/share/radio
$ sudo bash
# ./install_streaming.sh
Starting Icecast2 integration with the Music Player Daemon
The Icecast2 installation program will ask if you wish to configure
Icecast2.
Answer 'yes' to this. Configure Icecast as follows:
Icecast2 hostname: localhost
Icecast2 source password: mympd
Icecast2 relay password: mympd
Icecast2 administration password: mympd
Continue y/n: y
```

Enter 'y' to continue:
The Icecast2 installation program will ask if you wish to configure Icecast2.
Answer '**yes**' to this. Configure Icecast as follows:

Icecast2 hostname: **localhost**
Icecast2 source password: **mympd**
Icecast2 relay password: **mympd**
Icecast2 administration password: **mympd**

It is important that you replace the default password 'hackme' with 'mympd' and that you leave the Icecast2 hostname as 'localhost'. The installation program continues configuration. The icecast2 server will be started:

```
Done Configuring icecast2..
insserv: warning: script 'ggpiod' missing LSB tags and overrides
Starting icecast2: Starting icecast2
Detaching from the console
icecast2.
```

Ignore the **innserv** warning message.

Check that the PI Radio stream is enabled

```
# mpc outputs
Output 1 (My ALSA Device) is enabled
Output 2 (PI Radio MPD Stream) is enabled
```

Check that MPD has established a connection with the icecast2 server

```
# netstat -tn | grep :8000
tcp        0      0 127.0.0.1:8000      127.0.0.1:38639         ESTABLISHED
tcp        0      0 127.0.0.1:38639     127.0.0.1:8000          ESTABLISHED
```

This completes the installation of Icecast2 however you may need to configure the clock speed.

Overclocking the Raspberry PI

With older versions of the Raspberry Pi it will almost certainly be necessary to over-clock to handle Icecast2 streaming using the **raspi-config** program. Medium over-clocking seems to be sufficient.

Run **raspi-config**. Select option 7 'Overclock'. After a warning screen about over-clocking has been displayed, the following screen will be displayed:

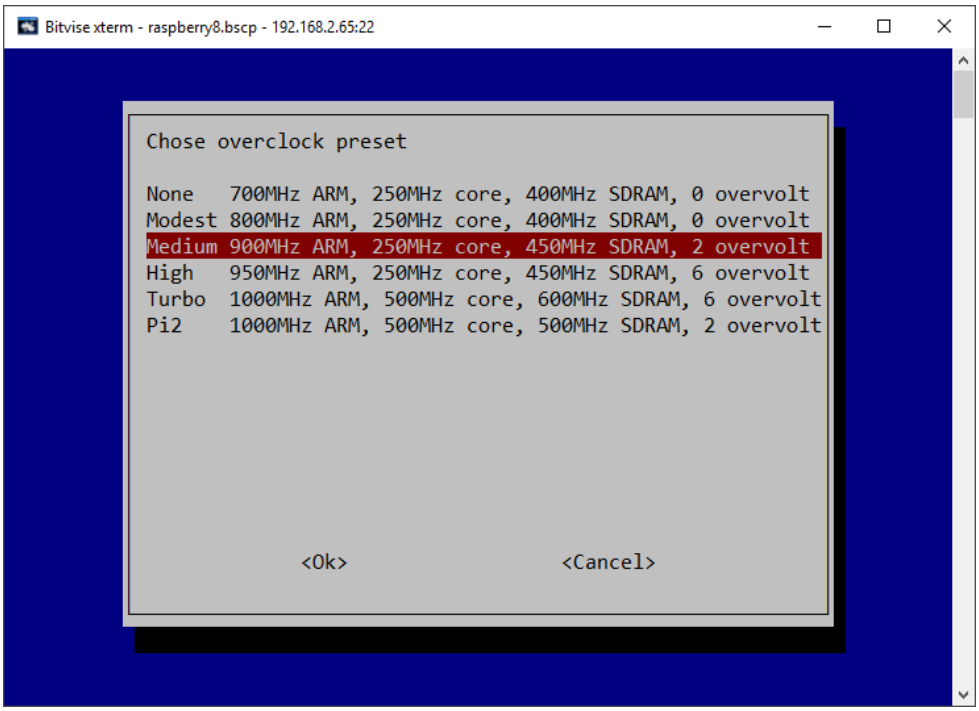


Figure 90 Over-clocking the Raspberry PI

Select 'Medium' to start with. Reboot the Raspberry PI when prompted.

Re-test the radio with streaming switched on.

Icecast2 Operation

The radiod daemon has full control over the Icecast2 service and stops and starts it as required. When the radio is first switched on the Icecast2 streaming service will not normally be enabled unless it was enabled as shown below by an earlier run of the radio software.

Switching on streaming

Before you can listen to the streaming on the PC or mobile device it is necessary to start the Icecast2 streaming daemon. It must be switched on first.

Use the options menu (Press menu button three times). Step through the menu option using the Channel up/down buttons until "**Streaming off**" is displayed in the LCD display (assuming Icecast is installed). Press either Volume button and after a short delay the text should change to "Streaming on" in the LCD display. Press the menu button again to exit the options menu.

This starts the Icecast2 service. It also writes the word "on" or "off" to a file called **/var/lib/radiod/streaming**. This is file is used to enable or disable the Icecast streaming function at boot time.

Starting Icecast2 manually

Use the following command:

```
$ sudo service icecast2 start
```

To stop it again:

```
$ sudo service icecast2 stop
```

Enabling Icecast2 at reboot time

It isn't necessary to enable the Icecast2 service at boot time as the radio program will start it depending on the contents of the **/var/lib/radiod/streaming** file. Should you wish to enable streaming at boot time then enable it with the following command:

```
$ sudo update-rc.d icecast2 enable
```

Playing the Icecast stream on a Windows 7

To play the Icecast2 radio stream on a PC point your web browser at the IP address of the radio on port 8000. In the following example the IP address of the radio is 192.168.2.11. So this would be:

http://192.168.2.11:8000

The following screen should be displayed. If not continue to the troubleshooting guide at the end of this chapter:

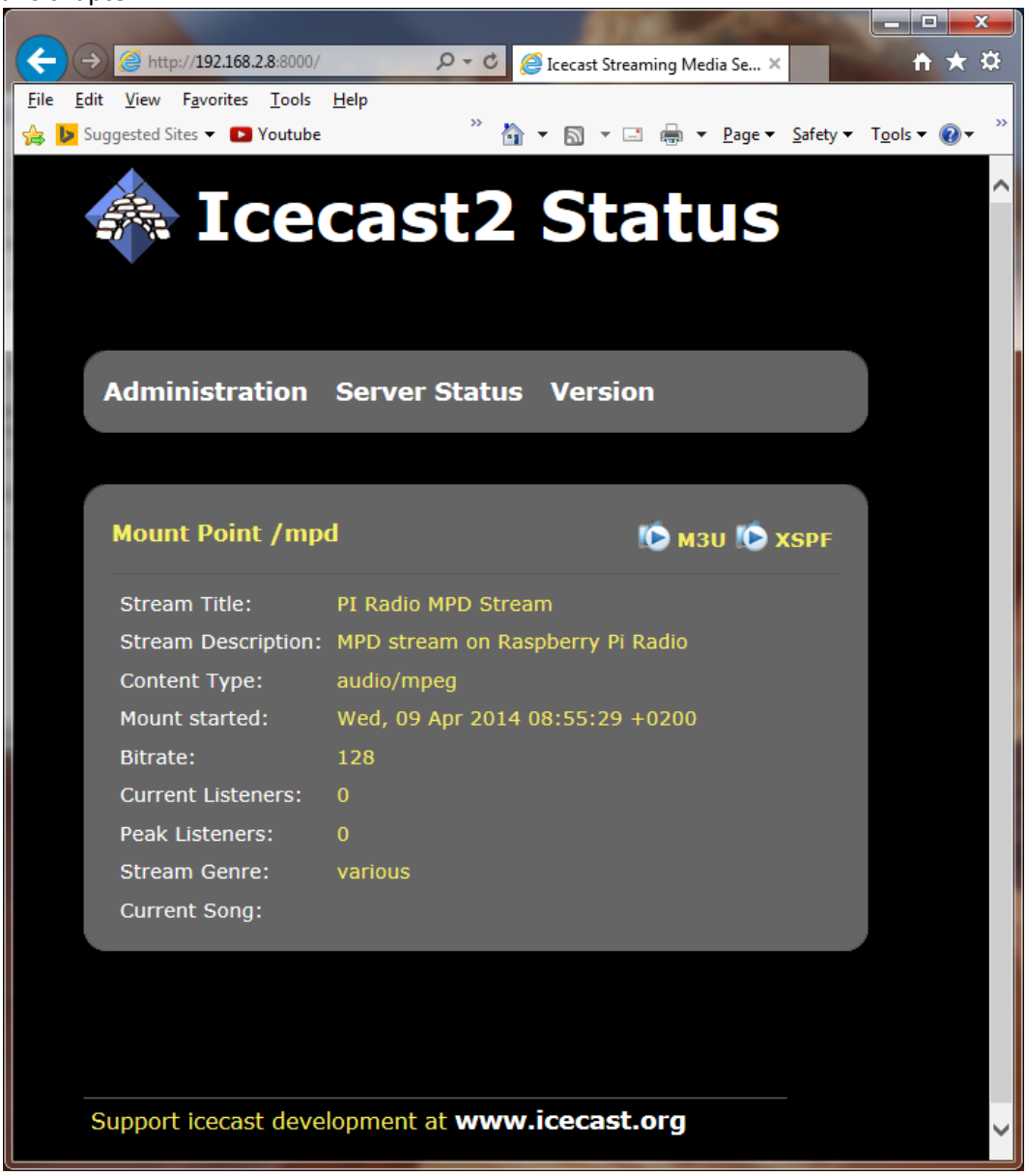


Figure 91 Icecast2 Status

Click on the **M3U** link to play the stream. This will launch your configured music player (For Windows PCs this is normally Windows Media Player.

Playing the Icecast stream on a Windows 10

The default player TWINUI does not appear to be able to find the M3U radio stream. Install Firefox Browser from https://www.mozilla.org and run the IP address of the radio on port 8000. The following is displayed. Click on **M3U**.

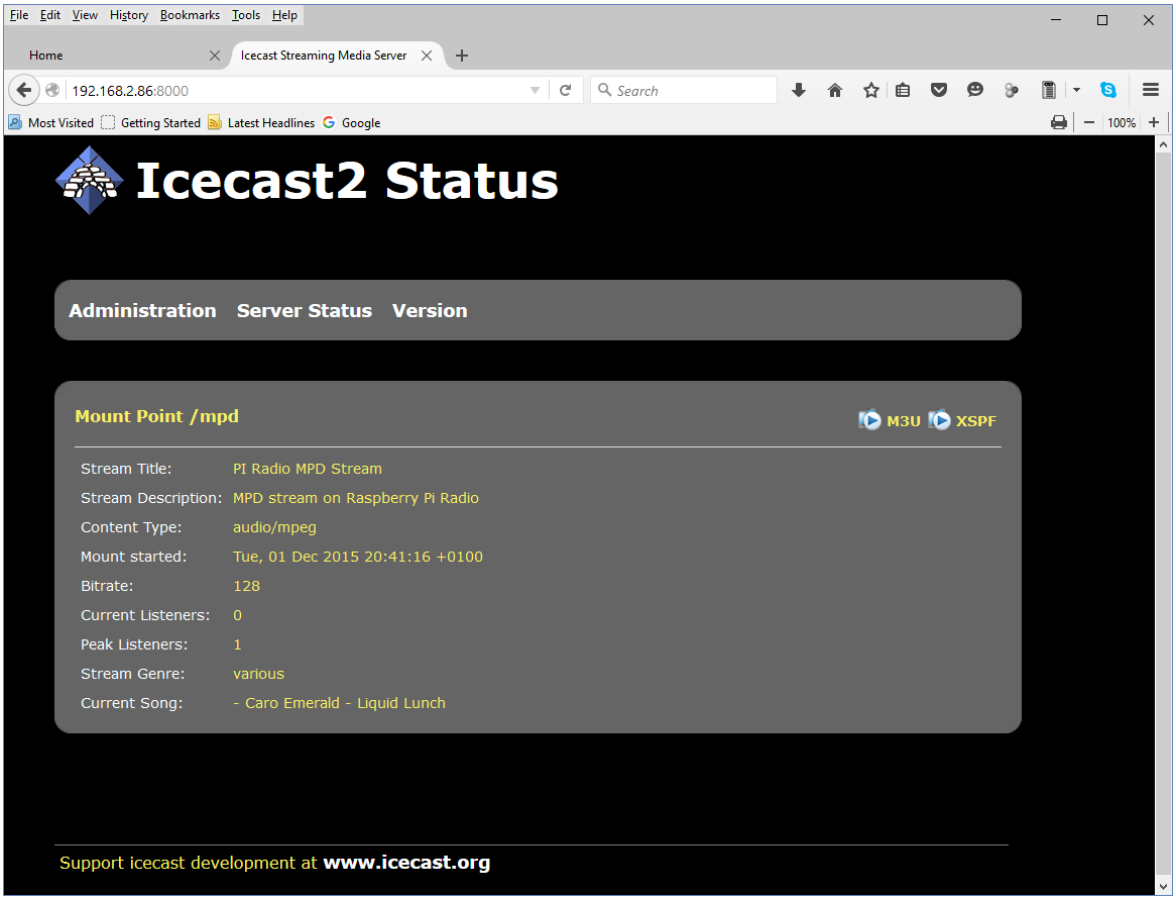
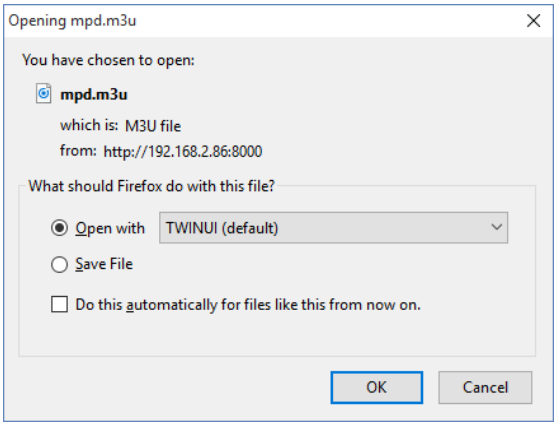


Figure 92 Streaming in the Firefox Web Browser

The following is displayed:



Change it to always open Windows media player:

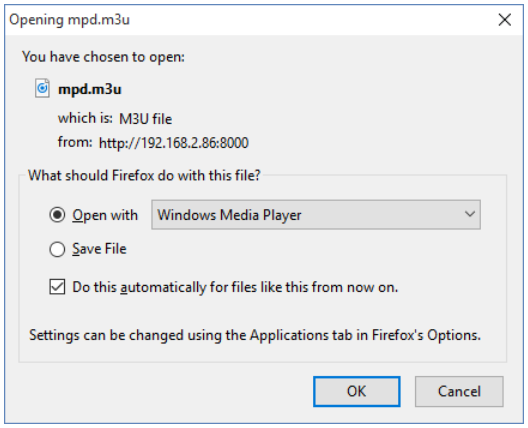


Figure 93 Selecting Windows Media Player

Press OK to continue.

Note: It is probably possible to configure Windows 10 Edge or Internet Explorer 11 to use Windows Media player instead of TWINUI.

The selected radio station or music track should be heard through the PC speakers.

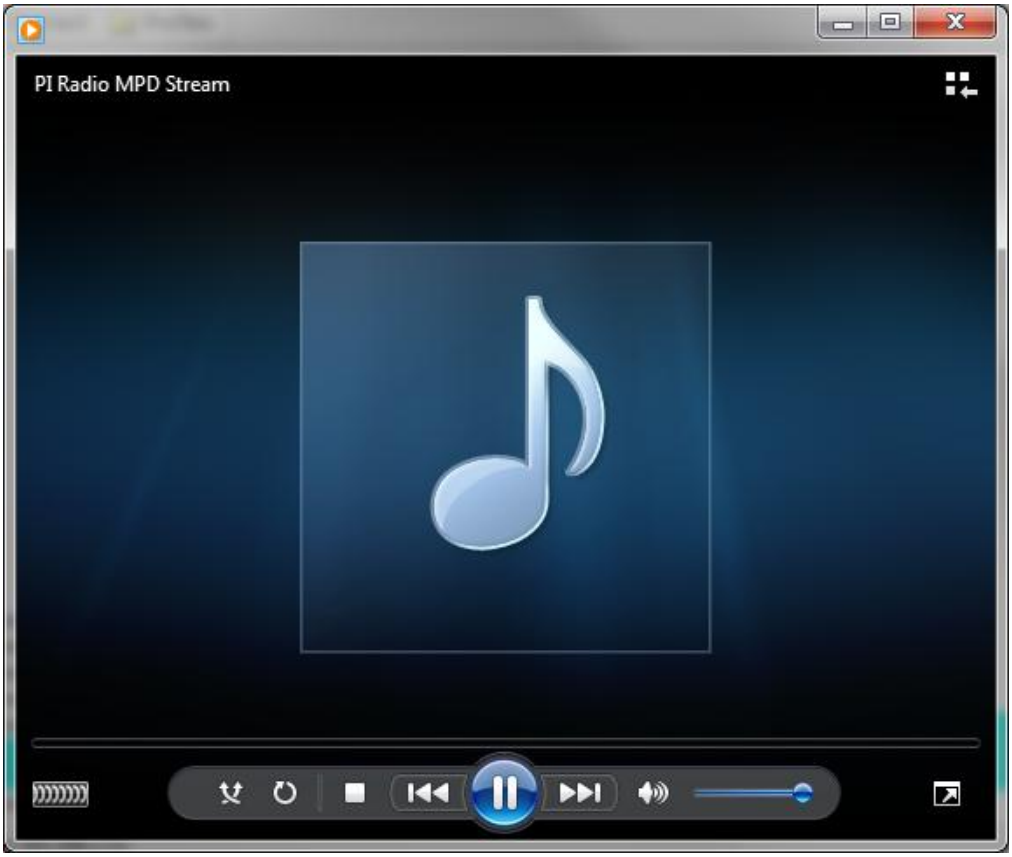


Figure 94 Windows media player

At this point you may wish to mute the sound from the radio itself. Simply reduce the volume to almost zero.

Note: If the mute function is used it will stop the **Icecast** stream.

Playing the Icecast2 stream on an Apple IPad

This is exactly the same as playing the Icecast2 stream on a Windows PC.

1. Open the Safari browser.
2. Type in the Icecast2 URL. For example **http://192.168.2.11:8000**
3. Click the M3U button

This should open the iTunes Player and after a short time should start playing the radio stream.

Playing the Icecast2 stream on an Android device

1. Open your web browser
2. Type in the Icecast2 URL. For example **http://192.168.2.11:8000/mpd** (don't include .m3u)
3. When asked to "Complete action with" select your *Android System* then *Music player*

The Icecast stream should start playing. It is important not to key in **mpd.m3u** at the end of the URL. It must be **mpd** only.

Visual streaming indicator

When streaming is switched on an asterix '*' character is displayed as a visual streaming indicator in the LCD display on the Raspberry PI radio. When the '*' character is displayed this indicates that the Icecast2 streaming is switched on.

For the four line 20 character display the visual indicator is displayed after the time on the first line.
09:26 02/05/2014 *

For the two line by 16 character display there isn't the room to do this so it is displayed after the Volume or Mute message on the second line.
Volume 75 * or Sound muted *

Administration mode

In the "Icecast Status" screen there is an Administration tab. Click on the Administration tab. The following screen will be displayed. Enter username 'admin' and the password 'mympd'.

Figure 95 Icecast admin login

The icecast2 administration page will be displayed.

It has two parts. The Global Server statistics and mount point information.

See next page:

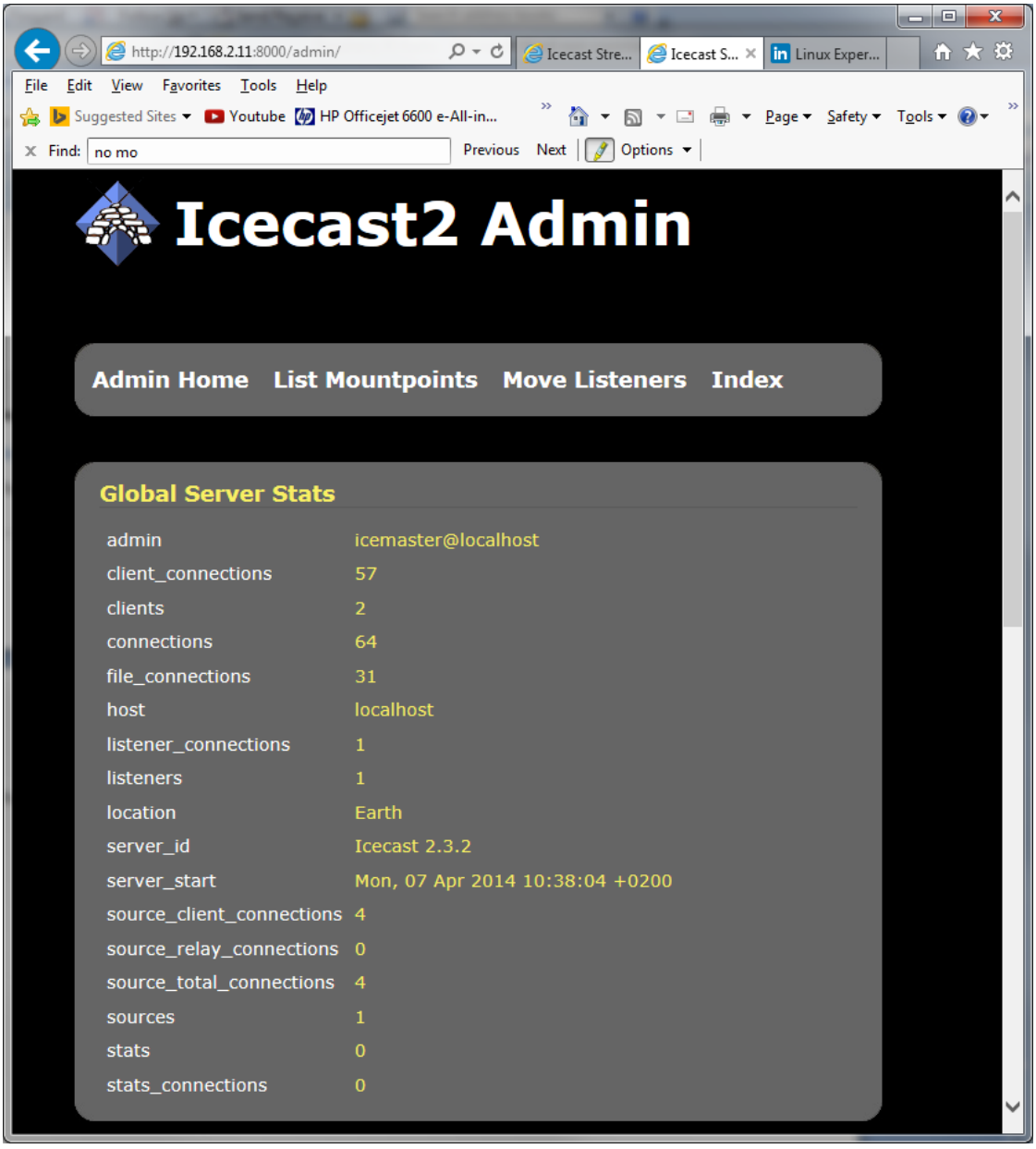


Figure 96 Icecast Global Server Status

It is beyond the scope of this manual to describe **Icecast2** administration. See the following site for further support. That said you should not normally need to change anything.

http://www.icecast.org/

The above site contains latest software updates, information, documentation and support forums.

Shown below is the second part (scroll down to display). This will display the **/mpd** mount point information and the currently playing track. This screen isn't automatically refreshed so you will need to refresh it if you want to display the name of any new track or station.

Figure 97 Icecast2 Mount point information

Troubleshooting Icecast2

Help for general problems with icecast2 can be found on the forums at http://www.icecast.org/

Icecast2 has two log files in the **/var/log/icecast2** directory namely **access.log** and **error.log**. The error log may give a clue as to the problem.

Below is a simulated error caused by mis-configuring the shoutcast entry in /etc/mpd.conf file. Here the hostname 'piradio' has been configured in the **/etc/mpd.conf** shoutcast entry instead of 'localhost'.

```
$ tail -f /var/log/mpd/mpd.log
Apr 07 10:43 : output: Failed to open "PI Radio MPD Stream" [shout]: problem
opening connection to shout server piradio:8000: Couldn't connect
```

Problem - Icecast streaming page says it can't be displayed.

Possible causes:

- The icecast service is not running on the radio.
  - Start it either from the Radio options menu (Streaming on) or run **sudo service icecast2 start** on the Rasperry PI and retry.
- Incorrect IP address or missing port number in the URL.
  - See Icecast2 *Operation* on page 132

Problem – No Mount Point displayed

Possible causes:

- This is mostly due to a mis-match in the MPD configuration and the Icecast2 configuration.
  - The icecast configuration is file is **/etc/icecast2/icecast.xml** . Make sure that all of the passwords are set to 'mympd'. The password 'hackme' will not work.
- There is no /mpd directory or the permissions are incorrect.
  - Check that the **/mpd** directory exists and that the permissions are set to 777. See Installing *Icecast* on page 131.

Problem - Cannot play the stream on my Android device

There are a number of Icecast players which can be downloaded onto Android and play Icecast2 streams across the network without problem. However the usual Android System Music player should work. The most likely cause of this problem is keying in an incorrect URL (Maybe adding .m3u to the end). See *Playing the Icecast2 stream on an Android device* on page 136.

Problem – Music keeps stopping or is intermittent

This is difficult to give a definitive answer to this problem. It must be remembered that running MPD and Icecast2 together on a Raspberry PI is pushing the Raspberry PI to its limits. It can also depend on your network or the PC you are using. Personal experience showed no problem playing a stream on PC with a wired network connection however a Laptop connected over a wireless network did not work well. Trying to play two or more devices on the MPD/Icast2 stream is also likely to result in poor results.

Try over-clocking the Raspberry PI using the **raspi-config** program. Medium over-clocking seems to be sufficient. See Overclocking the Raspberry PI on page 132. The icecast streaming facility is a fun thing to try out but if it doesn't work properly or is causing you stress; switch the streaming facility off.

Configuring the speech facility

It is possible to configure speech for visually impaired and blind persons who cannot read the display. As channels are changed or stepping through the menu the radio will "speak" to you. This excellent idea came from one of the project contributors, see *Acknowledgements* on page 151). This facility requires installation of the **espeak** package.
See http://elinux.org/RPi_Text_to_Speech_(Speech_Synthesis)

Install the **espeak** package:

```
$ sudo apt-get install espeak
```

Enable the speech facility in **/etc/radiod.conf** and restart the radio.

```
# Speech for visually impaired or blind listeners, yes or no
speech=yes
```

The verbose setting speaks the station or track details every time it is changed. However it can take a long time to move through the tracks or stations whilst speaking. Usually set this to no.

```
verbose = no
```

To get the right balance between speech volume and the normal radio volume adjust the **speech_volume** parameter percentage (10-100%)

```
speech_volume=30
```

The /var/lib/radiod/voice file

The **/var/lib/radiod/voice** file contains the **espeak** command (or part of it).

```
$ espeak -ven+f2 -k5 -s130 -a
```

Where -v is the voice (en+f2 = English female voice 2), -k is capitals emphasis, -s is the voice speed and -a is amplitude (0-200), the -a parameter is filled in by the radio program.

Testing espeak

You can test **espeak** with the following command.

```
$ espeak -ven+f2 -k5 -s130 -a20 "Hello Bob" --stdout | aplay
```

To see the capabilities of **espeak** see the website http://espeak.sourceforge.net/ or run:

```
$ espeak -h
```

If no sound is heard then then test using the aplay program. The **espeak** system will not work if **aplay** is not working. Test with aplay and a suitable wav file.

```
# mpc pause
# aplay /usr/share/scratch/Media/Sounds/Vocals/Singer2.wav
```

If still no sound check what devices are configured using aplay.

```
# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: ALSA [bcm2835 ALSA], device 0: bcm2835 ALSA [bcm2835 ALSA]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 0: ALSA [bcm2835 ALSA], device 1: bcm2835 ALSA [bcm2835 IEC958/HDMI]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: Device [Generic USB Audio Device], device 0: USB Audio [USB Audio]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
```

In the above example there are two devices namely *bcm2835 ALSA* (normal audio jack output) and *Generic USB Audio Device*. If using either a HiFiBerry DAC or IQAudio device then create the **/etc/asound.conf** file using nano:

```
# nano /etc/asound.conf
```

Add the following lines:

```
ctl.!default {
    type hw
    card 1
}

pcm.!default {
      type plug
      slave {
          pcm "plughw:1,0"
          format S32_LE
      }
}
```

The format S16_LE is an alternative format but does not work with HiFiBerry DAC. The above statements set up the default mixer and pcm sound device respectively to use card 1.

If using the USB (Card 1 device 0) then change the device definition in the above file.

```
pcm "plughw:1,0"
```

Retest with aplay (No need to reboot).

See https://www.hifiberry.com/upgrading-raspbian-to-jessie/

Note: This author does not provide support for **espeak**.

The language file

The language files are stored in **/home/pi/radio/language** directory. This contains the text that will be spoken. The default language **language.en** is file is copied to **/var/lib/radiod/language**. The language (if present) file is loaded during start-up of the radio. If not present the default English text is used.

The format of each entry in the language file is:

<label>:<text>

For example:

select_source: Select source

It is possible to display all the labels and text by running **language_class.py**.

```
$ cd /usr/share/radio
$ ./language_class.py
consume: Consume
rss_display:  RSS display
random: Random
loading_radio:  Loading radio stations
alarmminutes: Alarm minutes
sleeping:  Sleeping
main_display:  Main display
streaming: Streaming
sleep: sleep
yes: yes
options_menu:  Options menu
information:  Information display
select_source:  Select source
no: no
stopping_radio: Stopping radio
search_menu:  Search menu
source_media:  Music library
the_time:  The time is
loading_media:  Loading media library
repeat: Repeat
on: on
search:  Search
off: off
alarm: Alarm
source_radio:  Internet Radio
timer: Timer
reload: Reload
voice: voice
colour: Colour
alarmhours: Alarm hours
```

Suppressing an individual message

From version 5.3 onwards it is possible to suppress an individual message by adding an exclamation mark (!) to the beginning of the message string in the language file. For example if you do not wish to hear the time when speaking information then change **the_time** parameter by adding an ! to the beginning of the text to be spoken as shown in the example below:

```
the_time:  !The time is
```

Creating a new language file

To create a new language file by running the **language_class.py** program and redirecting the output to a file called **language.<new>** where **<new>** is the country code. For example to create a language file in Dutch, the country code is **nl**.

```
$ cd /usr/share/radio
$ sudo ./language_class.py > language/language.nl
```

Now edit the text (Not the labels) in the language/language.nl file. It isn't necessary to change every

```
# Nederlands text for uitspraak
main_display: Hoofd menu
search_menu: Zoek menu
select_source: Media selecteren
options_menu: Opties menu
rss_display: RSS beeld
information: Informatie beeld
the_time: De tijd is
loading_radio: Radio zenders laden
loading_media: Media laden
search: Zoek
source_radio: Internet Radio
source_media: Muziek selectie
sleeping: Slaapen
```

Finally copy the new language file to **/var/lib/radiod/language** (Omit the country code) and restart the radio.

```
$ sudo cp  language/language.nl  /var/lib/radiod/language
$ sudo service radiod restart
```

Speech Operation

At the moment the speech function is highly experimental and will be developed further if there is the demand. The best use is with the remote control which includes a button for toggling sound on and off and another button to speak information about the station or track as well as speaking the time. These buttons are set up in the section called *Installing the Infra Red sensor software* on page 73.

The Rotary encoder version of the radio is the best implemented. The MUTE switch is now the "Speak information" switch. To mute the radio, hold the button in for two seconds and release.

Controlling the Music Player daemon from Mobile devices

Android devices

There are a number of Android Apps capable controlling the Music Player Daemon from an Android such as a smart-phone or tablet. One of the most popular seems to be **MPDroid**. See the following link:

https://play.google.com/store/apps/details?id=com.namelessdev.mpdroid

MPDroid allows you to control a MPD server (Music Player Daemon) and stream from it. It is a fork from an earlier program called **Pmix** and adds various new features and streaming support. The radio daemon is completely integrated with MPD clients such as **mpc** and **MPDdroid**.

Load the MPDroid App use the Google Play Store on your device. Go to the settings menu and select **WLAN based connection**. Select **Host** and fill in the IP address of the radio and press OK. Set up the **Streaming url suffix** to **mpd.mp3**. All other settings can be left at their defaults.

Keep pressing the back button to exit and then re-start the MPDroid App. The play screen should be displayed as shown below. Volume, pause, fast forward/back can all be controlled from this screen.

To switch to the play list drag the play screen to the left. The current station list or play list will be displayed. Tap on the desired station or track to play it. Drag the play screen to the right to return to the play screen.

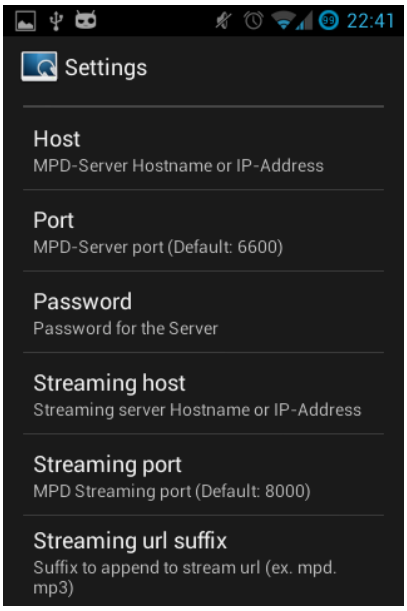


Figure 98 MPDroid set-up screen

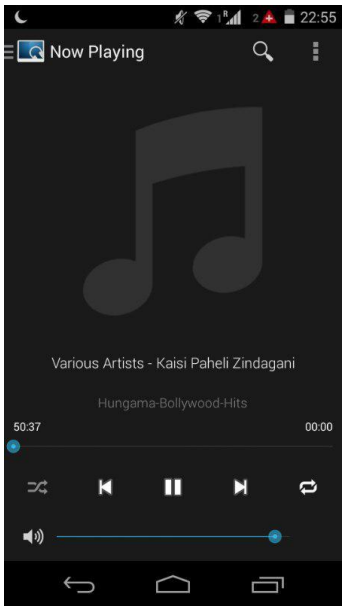


Figure 99 MPDroid play screen

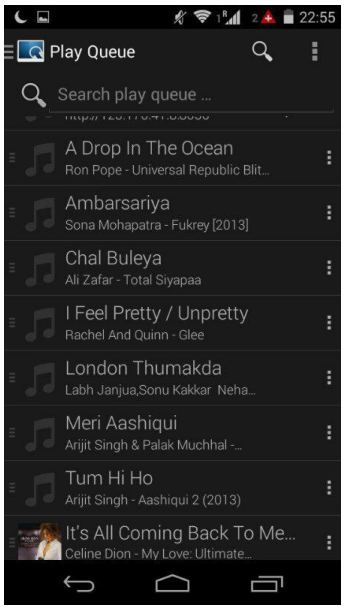


Figure 100 MPDroid play queue

Note: MPDroid is third party software and no support can be provided by bobrathbone.com.

Apple devices

Download **mPod – MPD Remote Control Software** from the Apple store or at following link: http://antipodesaudio.com/mpd.html . Run **mPod**, it will automatically find the Raspberry Pi running the Music Player daemon.

Frequently asked questions (FAQs)

What is the login name and password?

The default login name is: pi
The default password is: raspberry

Why are the radio stations not in the order that they were defined?

Playlists are loaded by the radio daemon in alphabetic order using the playlist name.
When loading an individual playlist, MPD loads the stations in the order that they are defined in each individual playlist. This has been improved by sorting the playlist.

It helps greatly to group stations of the same type into a single playlist. For example group all BBC radio stations into a single playlist.

The only way to get all of the radio stations in the order that you define them is to define a single playlist, for example **myplaylist**:

```
(myplaylist)
#
# United Kingdom
[BBC Radio 1] http://bbc.co.uk/radio/listen/live/r1.asx
[BBC Radio 2] http://bbc.co.uk/radio/listen/live/r2.asx
[BBC Radio 3] http://bbc.co.uk/radio/listen/live/r3.asx
:
:
[RAIradio3] http://www.listenlive.eu/rai3.m3u
```

This will produce a single playlist called **myplaylist.m3u** with the stations loading in the order that they have been defined in the **/var/lib/radiod/stationlist** file. Make sure there are no blank lines between station definitions otherwise this terminates the playlist. All remaining stations will end up in there own single playlist file.

Why are some station names not being displayed in the web interface?

The reason for this is that some stations don't send the name with the stream. If you run the **mpc playlists** command you will see that some radio stations shows only the station URL and not the name:

```
$ mpc playlist
RAIradio2
:
BBC Radio 4 extra
http://icestreaming.rai.it/1.mp3
BBC Radio 3
BBC Radio 6
BBC Radio 5 live
```

The only way around this is to complain directly to the radio station to ask them to amend their stream to include the station name and title details. The only way reason that the station name is seen with the radio program is that it picks up the names out of the station list file.
The snoopy web interface can't do this however.

Why doesn't the web interface display URLs until a station is selected?

When the Snoopy web interface is loaded it loads the playlists found in the **/var/lib/mpd/playlists/** directory. Snoopy displays the URLs but doesn't appear to use any titles defined in the playlists. It only displays the radio station information (if present) once it starts streaming from a particular radio station. Snoopy is third party software over which this author has no control.

Why are music tracks played randomly when loaded?

This is the default behaviour when the music library is loaded in version 5.2 or earlier. This has changed in version 5.3 onwards. Random always defaults to "off" when selecting the radio and to the value stored in the **/var/lib/radiod/random** file when the music library is selected.
This can be changed in the selection menu by selecting "Random off" after loading the music. However when the radio is restarted it will return to the default behaviour for the radio which is off.

Why not display volume as blocks instead of Volume nn?

This is a design choice. Volume is displayed as "Volume *nn*" where *nn* is 1 to 100. The reason the volume is not displayed as blocks is resolution. There are only 16 or 20 blocks available depending upon the display being used. This means worse case the volume would need to change by at least 7 before any visual clue would be given by a block display. Also the line on which the volume is displayed is also used to display timer and alarm functions and will further reduce the amount of blocks available even further.

Why do I see the Station number displayed in brackets?

If the **display_playlist_number** parameter is set to yes in **/etc/radiod.conf** then the the station name and play title will be displayed together the station number on the second line. The current title (if transmitted) is displayed third line for four line displays.

```
12:01 30/08/2014
WPJR Country (6)
Lonestar - Mr. Mom
Volume 75
```

For two line displays:

```
12:01 30/08/2014
WPJR Country (6)
```

To disable this feature set **display_playlist_number** parameter to **no** .

Why do I see a station number on LCD line 3?

For version 3.3 onwards if no song information is available then the station playlist number followed by the stream speed. In the following example Radio 1 is not transmitting any song information. It is number 37 in the play list. The speed from the stream is 96 Kilobit. The displayed stream speed can also continuously change for some radio stations where the stream speed is variable.

```
12:01 23/08/2015
Radio1
Station 37 96K
Volume 75
```

Is it possible to change the date format?

Yes. Please see the section called *Changing the date format* on page 79.

Is there a pause & resume function?

Yes but it is called mute and un-mute. The mute function also stops or pauses the MPD player. If playing a radio station a “stop” command is carried out. If playing a media track a “pause” is carried out. The reason these are different is that media may be safely paused but in the case of a radio station pausing causes buffering and jumping to the next station when the radio resumes normal playing. See the Mute function on page 85.

Why do I see a different station name from the one in the playlist?

The station information displayed comes from the stream itself. The name entered in the playlist definition is only used in the search function. This was a design decision because the station information is only available once a particular radio station is selected so only the playlist name can be initially used. If the station transmits the station title this is used instead.
Run the **display_current.py** program to see all the information that comes from the stream (It is quite interesting). This can be changed by setting **station_names=list** in **/etc/radiod.conf**

What Rotary Encoder can I use for this project?

The rotary encoders illustrated in this guide are COM-09117 12-step rotary encoders from sparkfun.com.

The radio uses so called “Incremental Rotary Encoder”. An incremental rotary encoder provides cyclical outputs (only) when the encoder is rotated. The other type is an absolute rotary encoder and maintains position information even when switched off (See Wikipedia article and my tutorial on rotary encoders).

The cheaper smaller rotary encoders are usually incremental encoders. Absolute rotary encoders are usually bigger and more expensive as they house more electronics.

If unfortunately the seller doesn't provide a specification then there is a small risk that they may not run with this software.

Note: Not all manufacturers’ rotary encoders will work with this project. If they work then fine if not regrettably you will need to purchase the recommended rotary encoders. You can also try the alternative rotary class which may just work with your encoders.

Can this code or documentation be re-used in other projects?

Yes it can. You can even use it commercially provided that you do so under the terms of the licence distributed with this package. The software and documentation for this project is released under the GNU General Public Licence and may be re-used it other projects. See Licences on page 149.

Licences

The software and documentation for this project is released under the GNU General Public Licence.

The GNU General Public License (GNU GPL or GPL) is the most widely used free software license, which guarantees end users (individuals, organizations, companies) the freedoms to use, study, share (copy), and modify the software. Software that ensures that these rights are retained is called free software. The license was originally written by Richard Stallman of the Free Software Foundation (FSF) for the GNU project.

The GPL grants the recipients of a computer program the rights of the Free Software Definition and uses *copyleft* to ensure the freedoms are preserved whenever the work is distributed, even when the work is changed or added to. The GPL is a *copyleft* license, which means that derived works can only be distributed under the same license terms. This is in distinction to permissive free software licenses, of which the BSD licenses are the standard examples. GPL was the first *copyleft* license for general use. This means that you may modify and distribute the software and documentation subject to the conditions of the licences.

See http://www.gnu.org/licenses for further information on the GNU General Public License.

The licences for the source and documentation for this project are:

GNU General Public License.	See http://www.gnu.org/licenses/gpl.html
GNU AFFERO General Public License.	See http://www.gnu.org/licenses/agpl.html
GNU Free Documentation License.	See http://www.gnu.org/licenses/fdl.html

Intellectual Property, Copyright, and Streaming Media

This is an unbelievably complex subject. The author is not a lawyer and can not offer any legal advice on this subject. If you decide to stream your music content or relay a radio station stream back out to the internet or within a public building then you should seek legal advice.
See also: http://en.wikipedia.org/wiki/Copyright_aspects_of_downloading_and_streaming

In general Radio stations are providing a stream to promote their radio station. As media providers they should have arrangements in place to make the content that they provide is legally streamed across the Internet but not all do. The question is it legal to listen (or view) such content is a complex one and subject to local and international laws and which vary considerably.

If you implement **Icecast** or any other streaming technology to re-stream content within your own home then provided that this is not streamed back out to the Internet or public location then one would think that you will not encounter any problems (but you never know).

If you stream music tracks or relay radio stations back out onto the internet or public space then almost certainly you will be infringing a copyright law or intellectual property rights somewhere. The penalties for such an infringement can be severe.

WARNING: YOU USE THE ICECAST STREAMING IN THIS PROJECT AT YOUR OWN RISK ESPECIALLY IF YOU MAKE THE STREAM CONTENT AVAILABLE ACROSS THE INTERNET OR PUBLIC SPACE, EVEN IF YOU ARE JUST RELAYING AN EXISTING MEDIA STREAM, LEGAL OR OTHERWISE.

Also see the Disclaimer on page 150.

Disclaimer

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BELIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE OR DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Technical support

Technical support is on a voluntary basis by e-mail only at bob@bobrathbone.com. Before asking for support, please first consult the troubleshooting section on page 113. I will always respond to e-mails requesting help and will never ignore them. I only ask that you do the same (i.e. Did my suggestions help or not?). Be sure to provide the following information:

- What have you built (Adafruit or normal LCD variants)?
- Which program and version are you running?
- A clear description of the fault.
- What you have already done to locate the problem?
- Is anything displayed on the LCD?
- Did you run the test programs and what was the result?
- Switch on DEBUG logging as described on page 85, run the program and include the **/var/log/radio.log** file.
- Did you vary from the procedure in the manual or add any other software?
- Please do not answer my questions with a question. Please supply the information requested.

Please note that support for general Raspberry PI problems is not provided. Only issues relating to the Radio software will be investigated.

For general Raspberry PI support see the following site:
http://www.raspberrypi.org/forums/

For support on Music Player Daemon issues see the help pages at the following link:
http://www.musicpd.org/

For issues relating to Icecast2 streaming see:
http://www.icecast.org

For those of you who want to amend the code to suit your own requirements please note: I am very happy to help people with their projects but my time is limited so I ask that you respect that. Please also appreciate that I cannot engage in long email conversations with every constructor to debug their code or to teach them Python.

Acknowledgements

My thanks to Matt Hawkins for the original LCD screen driver routines. It made the job of writing the *lcd_class.py* much easier.

Patrick Zacharias for his excellent work to implement the PiFace radio class.

The original instructions on how to use Rotary Encoders came from an excellent article by Guy Carpenter. See:
http://guy.carpenter.id.au/gaugette/2013/01/14/rotary-encoder-library-for-the-raspberry-pi/
His ideas were used in the *rotary_class.py* code.

To Adafruit Industries for their excellent LCD plate and I2C code. See http://www.adafruit.com.

To PiFace for their PiFace CAD and IR software. See http://www.piface.org.uk

To Steffen Müller for his article on Streaming audio with MPD and Icecast2 on Raspberry Pi.
See http://www.t3node.com/blog/streaming-audio-with-mpd-and-icecast2-on-raspberry-pi/

To contributors such as Alan Broad who supplied photos of the Lego example of the radio plus code contribution. Also to Mike Whittaker for his contribution on how to drive the USB speaker set. To other contributors such as James Rydell for his excellent implementation using an old Zenith radio.

Thanks to Michael Uhlmann for the work he did testing various Android Apps for MPD. Also Simon O'Niel who carried out configuration and testing of Cmedia sound dongle.

To Open Electronics Magazine for their excellent article on the Raspberry PI radio using the Adafruit LCD plate. See http://www.open-electronics.org/internet-radio-with-raspberry-pi/

To Joaquin Perez, Broadcast Engineer, Leeds for the backlight dimmer and circuit diagram.

To Luboš Ruckl for his work on the Rotary encoder class (adapted from code by Ben Buxton) and the PCF8574 LCD class (adapted from code by an unknown author but believed to be from the Aduino community).

To Béla Mucs from Hungary for his brilliant idea to support speech for visually impaired and blind persons. This facility uses the **espeak** package.

Gordon Henderson https://projects.drogon.net/ for the gpio wiring utility.

To all constructors of this project who have sent in photos of their radio's and their ideas for improvement and the many appreciative e-mails that I have received from them.

To http://www.allaboutcircuits.com for Figure 53 Soldering precautions.

Glossary

AAC	Advanced Audio Coding
ASX	Advanced Stream Redirector
ATC	Air Traffic Control
CGI	Common Gate Interface – Executable Server Side scripts
CAD	Control and Display (PiFace)
CIFS	Common Internet File System
DHCP	Dynamic Host Configuration Protocol
DSP	Digital Signal Processing/processor (In this context it is mixer control)
EMI	Electromagnetic Interference (For example fluorescent lighting etc.)
GPIO	General Purpose IO (On the Raspberry PI)
I2C	Industry standard serial interface (Philips now NXP) using data and clock signals
I2S	Inter-IC Sound (Used in DAC interface) from Philips (Now NXP)
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IATA	International Air Transport Association
ICAO	International Civil Aviation Organization
IR	Infra Red (sensor) for use with infra red devices such as remote controls
LCD	Liquid Crystal Display
LE	Low Energy – In this context Bluetooth LE (Bluetooth 4.0 core specification)
LIRC	Linux Remote Control software
M3U	MPEG3 URL
MAC	Media Access Control (address)
MPC	Command line client for MPD
MPEG	Moving Picture Experts Group
MPEG3	Music encoding standard from MPEG
NAS	Network Attached Storage

NFS	Network File System
NTP	Network Time Protocol
MPD	Music Player Daemon
OLED	Organic Light Emitting diode
OS	Operating system (Rasbian Jessie in this case)
PC	Personal Computer
PCM	Pulse-Code Modulation is a method used to digitally represent sampled analogue signals
PID	Process ID
PLS	MPEG Playlist File (as used by Winamp)
RSS	Really Simple Syndication – Web feed usually containing news items
SD	San Disk Memory Card commonly found in cameras and Smartphone's
SPI	Serial Peripheral Interface (Motorola)
SSID	Service Set Identifier. An SSID is the public name of a wireless network.
TCP/IP	The network protocol used by the Internet and computer networks.
UDP	Universal Datagram Protocol. A connectionless network protocol.
URL	Universal Resource Locator (A link to a Web page for example)
USB	Universal Serial Bus
WEP	Wired Equivalent Privacy (WEP) is a security algorithm considered less secure than WPA
WIFI	Wireless Network using the 802.11 Wireless Network protocol
WPA	Wi-Fi Protected Access (WPA)
WPA2	Wi-Fi Protected Access version II, an enhanced more secure version of WPA.
XML	Extensible Mark-up Language

Appendix A - System Files used by the Radio Program

A.1 Files added to the system

/etc/radiod.conf

This is the main configuration file for the radio program.

```
# Raspberry Pi Internet Radio Configuration File (26 pin version)
# $Id: radiod.conf,v 1.35 2016/10/21 09:35:55 bob Exp $

# Configuration file for version 5.4 onwards

[RADIOD]

# loglevel is CRITICAL,ERROR,WARNING,INFO,DEBUG or NONE
loglevel=INFO

# Startup option either RADIO or MEDIA (USB stick)
startup=RADIO

# Set date format, US format =  %H:%M %m/%d/%Y
dateformat=%H:%M %d/%m/%Y

# Volume range 10, 20, 25, 50 or 100
volume_range=20

# MPD port number (Do not change unless MPD reconfigured)
mpdport=6600

# Remote control communication  host and port Default localhost 5100
remote_control_host=localhost
remote_control_port=5100

# Remote control UDP server listen host either 0.0.0.0 (All interfaces) or
localhost
remote_listen_host=localhost

# Output LED for remote control, default GPIO 11 (pin 23) or
# GPIO 13 (pin 33) for AdaFruit plate or PiFace CAD (40 pin RPi needed)
# remote_led=0 is no output LED
remote_led=0

# Display playlist number in brackets yes or no
display_playlist_number=no

# Station name source, either from the stream (stream) or use playlist names
(list)
station_names=list

# Background colours (If supported) See Adafruit RGB plate
# options OFF, RED, GREEN, BLUE, YELLOW, TEAL, VIOLET, WHITE
bg_color=WHITE
mute_color=VIOLET
shutdown_color=TEAL
error_color=RED
search_color=GREEN
info_color=BLUE
menu_color=YELLOW
source_color=TEAL
sleep_color=OFF

# The i2cbackpack is either ADAFRUIT or PCF8574
#i2c_backpack=PCF8574
i2c_backpack=ADAFRUIT
```

```
# The i2c_address overides the default i2c address. 0x00 = use default
# Some backpacks use other addresses such as 0x3F, then set i2c_address=0x3F
i2c_address=0x00

# Speech for visually impaired or blind listeners, yes or no
# Needs espeak package - sudo apt-get install espeak
speech=no
# Speech volume as a percentage of the normal MPD volume
speech_volume=60
# Verbose - yes = each station change is spoken
verbose=no

# MPD version 0.16 requires the mpc load command to use the playlist
extension
# so set to yes, MPD version 0.19 does not use the playlist extension so
# set it to no
use_playlist_extensions=no

# Down switch GPIO setting, if using a HiFiberry/IQAudio DAC+ set to 10
down_switch=18
#down_switch=10

# Other switch settings
menu_switch=25
mute_switch=4
up_switch=17
left_switch=14
right_switch=15

# LCD GPIO connections
lcd_select=7
lcd_enable=8
lcd_data4=27
lcd_data5=22
lcd_data6=23
lcd_data7=24

# If using IQAudio set lcd_data5 to use another GPIO
#lcd_data5=6

# LCD screen width, 0 use program default. Usual settings 16 or 20
lcd_width=0

# Some rotary switches do not work well with the standard rotary class
# Rotary encoder driver. Set to "alternative" to use the alternative rotary
encoder class
rotary_class=standard
#rotary_class=alternative

# The following settings are only used by the retro_radio.py program
# Uncomment to use - change GPIO assignments as required
# RGB Status LED GPIO settings
#rgb_green=27
#rgb_red=23
#rgb_blue=22

# Menu rotary switch GPIO settings (retro_radio.py only)
# Uncomment to use - change GPIO assignments as required
#menu_switch_value_1=24
#menu_switch_value_2=8
#menu_switch_value_4=7
```

/etc/logrotate.d/radiod

This file causes the **/var/log/radio.log** to be rotated so that it doesn't continue to grow and fill the disk.

```
/var/log/radio.log {
      weekly
      missingok
      rotate 4
      compress
      notifempty
      maxsize 150000
      copytruncate
      create 600
}
```

Old log files are compressed and renamed, for example **/var/log/radio.log.1.gz**.

/etc/init.d/radiod

This is the start stop script for the radio daemon. The NAME parameter must be changed to point to the correct radio program. This is done by the **select_daemon.sh** shell script which is called during the installation procedure.

```
# Change NAME parameter this next line to the version of the daemon you are
using
# Choices are radiod.py, radio4.py, rradiod.py, rradio4.py, ada_radio.py,
# rradiobp4.py or rradiobp.py
# No spaces around the = character
NAME=radio4.py
```

This script is linked to the start stop run levels in the **/etc** directory.
For example:

```
ls -la /etc/rc2.d/S03radiod
lrwxrwxrwx 1 root root 16 Nov  8 14:28 /etc/rc3.d/S03radiod ->
../init.d/radiod
```

/etc/init.d/asound.conf

This file is only added if a DAC or USB sound device is added

```
# Set default mixer controls
ctl.!default {
    type hw
    card 1
}

# Set default PCM device
pcm.!default {
      type plug
      slave {
          pcm "plughw:1,0"
          format S32_LE
      }
}
```

/etc/init.d/pifacercd

This is the service start stop script for the PiFace remote control daemon. This starts and stops the /usr/share/radio/piface_remote.py program which handles the remote control for the PiFace Control and Display IR interface. This script is linked to the start stop run levels in the **/etc** directory.

/etc/lirc/lircrc

This file contains the button definitions for the remote control to Pi radio interface.

```
begin
  prog = piradio
  button = KEY_VOLUMEUP
  config = KEY_VOLUMEUP
  repeat = 1
end

begin
  prog = piradio
  button = KEY_VOLUMEDOWN
  config = KEY_VOLUMEDOWN
  repeat = 1
end

begin
  prog = piradio
  button = KEY_CHANNELUP
  config = KEY_CHANNELUP
end

begin
  prog = piradio
  button = KEY_CHANNELDOWN
  config = KEY_CHANNELDOWN
end

begin
  prog = piradio
  button = KEY_MUTE
  config = KEY_MUTE
end

begin
  prog = piradio
  button = KEY_MENU
  config = KEY_MENU
end

begin
  prog = piradio
  button = KEY_LEFT
  config = KEY_LEFT
end

begin
  prog = piradio
  button = KEY_RIGHT
  config = KEY_RIGHT
end

begin
  prog = piradio
  button = KEY_UP
  config = KEY_UP
end
```

```
begin
  prog = piradio
  button = KEY_DOWN
  config = KEY_DOWN
end

begin
  prog = piradio
  button = KEY_OK
  config = KEY_OK
end

begin
  prog = piradio
  button = KEY_LANGUAGE
  config = KEY_LANGUAGE
end

begin
  prog = piradio
  button = KEY_INFO
  config = KEY_INFO
end

# End
```

A.2 System files modified by the installation

All files to be modified by the installation process are first copied to <filename>.orig.

/etc/inittab

This file controls the initialisation of the Debian OS. The installation disables the serial line on the GPIO as this would create a conflict with the radio program.

```
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
# Original file stored as /etc/inittab.orig
```

/boot/cmdline.txt

This installation program disables the serial line start-up. The following line:

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2
rootfstype=ext4 elevator=deadline rootwait
```

is changed to:

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4
elevator=deadline rootwait
# Original file stored as /boot/cmdline.txt.orig
```

/etc/modules

If the i2C interface is installed then the i2c-dev module definition is added to this file. A reboot is required to load the module.

```
snd-bcm2835
```

```
i2c-bcm2708
i2c-dev
# Original file stored as /etc/modules.orig
```

/boot/config.txt

This is amended if installing the IR software by adding the lirc-rpi device definition. For example:

```
dtoverlay=lirc-rpi,gpio_in_pin=9
```

It may also be modified to support **HiFiBerry DAC** and **DAC+**. For example:

```
dtoverlay=hifiberry-dacplus
```

For **IQAudio** devices the relevant overlay will be specified.

```
dtoverlay=iqaudio-dacplus,unmute_amp
```

Appendix B – Wiring diagrams

B.1 Raspberry Pi Rotary Encoder version with backlight dimmer

The following wiring diagram was provided by Joaquin Perez, Broadcast Engineer, Leeds. He also shows the circuitry to dim the backlight using a BS170 Mosfet transistor (Software to support the LED dimmer to follow in a later release).

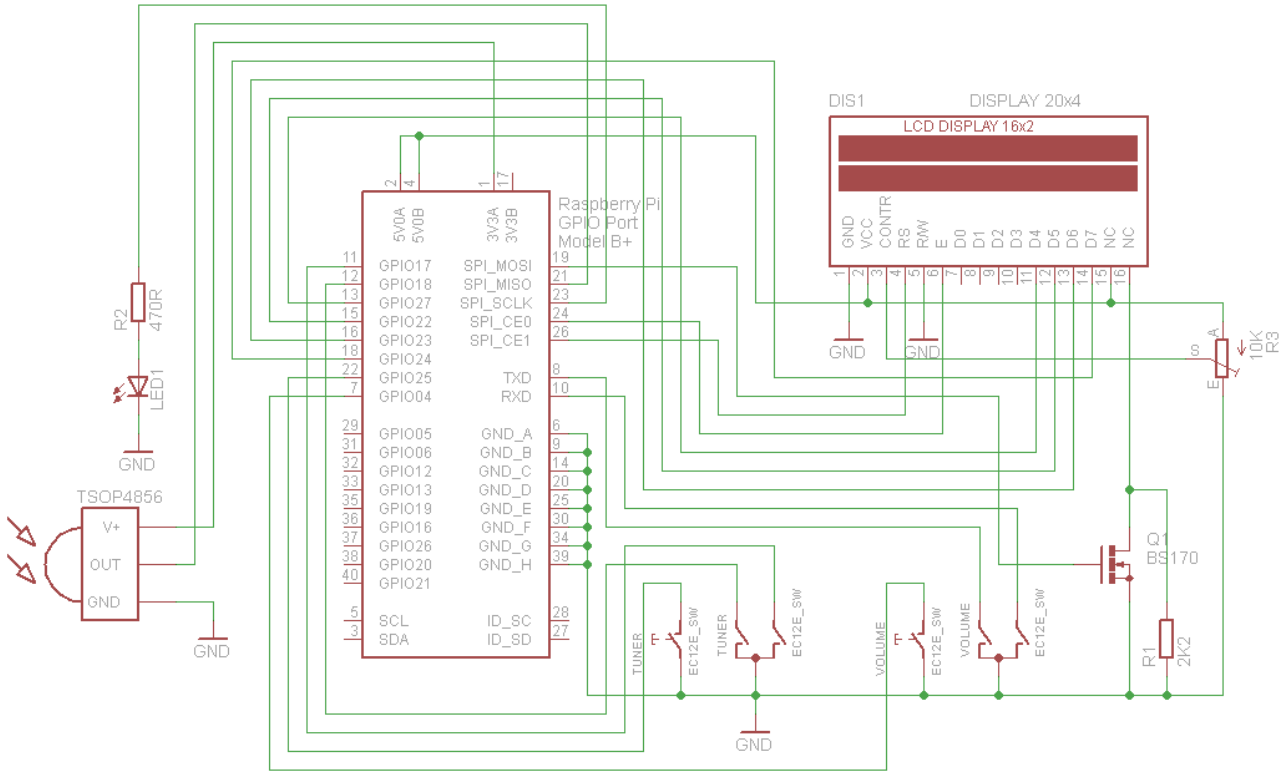


Figure 101 Wiring Raspberry Pi Radio Rotary Encoder version

Index

26 way ribbon cable, 36, 37, 38
802.11 bgn Wireless LAN, 23
AAC, 90, 93, 94, 152
activity LED, 17, 19, 46, 73, 74, 76, 78, 79
AdaFruit industries, 15
Adafruit LCD plate, 38, 81
AdaFruit RGB plate, 19, 78, 84
Adafruit RGB-backlit LCD, 14, 20, 106
airflow, 25
ASX, 90, 93, 94, 152
audio jack plug, 47
AV, 21, 22
Bluetooth, 23
Bluetooth 4.1, 23
Broadcom BCM2387 chipset, 23
CAD, 14, 17, 24, 37, 44, 45, 46, 61, 62, 73, 74, 78, 107, 122, 151, 152
CGI, 99, 152
CIFS, 101, 102, 103, 152
colours, 40, 79, 89
COM-09117 12-step rotary encoder, 30
constructor's gallery, 25
DAC, 14, 26, 27, 28, 47, 48, 49, 66, 67, 68, 72, 77, 78, 107, 110, 114, 118, 119, 120, 121, 125, 142, 152, 159
date format, 79, 148
Debian Wheezy, 22, 52, 57
DHCP, 101, 128, 129, 130, 152
dpkg, 58, 99, 119, 120
DSP, 152
Electromagnetic Interference, 34, 114, 152
EMI, 34, 114, 152
espeak, 14, 18, 76, 85, 86, 107, 141, 142, 143, 151
ferrite core, 34
fsck, 113
GPIO, 14, 17, 19, 20, 21, 23, 27, 28, 29, 30, 33, 36, 37, 38, 39, 40, 42, 43, 45, 46, 48, 68, 74, 78, 105, 117, 121, 122, 152, 158
GPIO header, 20, 21, 33, 40, 42, 68
GPIO pins, 14, 19, 20, 27, 36, 37, 38, 40, 42, 48, 105, 122
Ground Loop Isolator, 35
HD44780, 14, 22, 31, 37, 38, *See* HD44780 controller
HDD44780, 14, 22, 105
HDMI, 23
HiFiBerry, 14, 26, 27, 28, 47, 48, 66, 68, 72, 77, 107, 114, 118, 120, 125, 142, 159
housing the radio, 25
I2C, 14, 24, 27, 28, 37, 40, 41, 42, 43, 46, 60, 61, 74, 105, 121, 151, 152
iPad, 14, 145
iPhone, 14, 145
iPod 4 pole AV, 22
IPv4, 152
IPv6, 115, 152
IQAudio, 14, 26, 27, 28, 29, 31, 46, 49, 70, 74, 78, 107, 142
IR, 14, 17, 19, 27, 28, 37, 43, 44, 45, 46, 73, 74, 76, 107, 122, 151, 152, 157, 159
IR sensor, 45
IR Sensor, 19, 27, 37, 45, 74
Jessie, 14, 22, 23, 37, 52, 53, 57, 101, 117, 118, 119, 126, 128, 153
Jessie Lite, 14, 22, 23, 52, 57, 126
LCD, 14, 15, 16, 17, 20, 21, 22, 24, 27, 28, 31, 32, 33, 34, 37, 38, 39, 40, 41, 42, 45, 46, 60, 62, 74, 79, 80, 81, 83, 84, 87, 88, 105, 106, 107, 111, 112, 114, 115, 119, 121, 122, 133, 137, 147, 150, 151, 152
  HD44780 controller, 14
LED Backlight, 31
LED dimmer, 160
LIRC, 73, 77, 152
M3U, 14, 92, 93, 94, 107, 116, 134, 135, 136, 152
MAC, 129, 130, 152
mains filter, 34
micro SD card, 20
Micro SD slot, 23
micro USB, 22
MP3, 14, 85, 93
MPC, 71, 88, 89, 115, 152
mpd, 51, 57, 59, 71, 72, 85, 88, 90, 91, 98, 103, 104, 107, 115, 116, 118, 119, 123, 131, 136, 137, 139, 140, 145, 147, 151
MPD, 14, 57, 64, 71, 72, 80, 81, 85, 88, 89, 91, 98, 103, 106, 107, 115, 116, 118, 119, 123, 131, 132, 140, 145, 146, 148, 151, 152, 153
MPEG, 93, 152, 153
MPEG3, 93, 152
NAS, 14, 85, 86, 102, 152
NFS, 101, 102, 117, 153
NTP, 153
OLED, 22, 153
Organic Light Emitting Diode, 22
OS, 37, 102, 153, 158
PC, 14, 15, 16, 25, 37, 39, 58, 85, 87, 92, 101, 131, 133, 134, 136, 140, 153

PC speakers., 15, 136
PCF8574, 42, 43, 151
PCM, 48, 153
Pi 3 Model B, 23
Pi Zero, 14, 17, 22
PID, 153
PiFace, 14, 17, 19, 24, 37, 44, 45, 46, 61, 62, 73, 74, 78, 107, 122, 151, 152, 157
PiFace Control and Display, 14, 17, 24, 37, 157
pifacecad, 62, 73, 122
PLS, 90, 91, 92, 93, 107, 115, 153
potentiometer, 43, 114
power adapter, 33
power supply switch, 33
radiod.conf, 42, 43, 48, 72, 77, 78, 79, 80, 86, 106, 107, 124, 141, 147, 148, 154
Raspberry PI, 1, 14, 15, 16, 17, 19, 20, 21, 23, 25, 33, 34, 37, 38, 39, 40, 43, 45, 46, 47, 51, 57, 58, 61, 62, 64, 72, 74, 78, 82, 85, 88, 89, 98, 99, 106, 108, 114, 116, 122, 123, 126, 127, 128, 130, 132, 137, 140, 150, 151, 152
Red Blue Green LED, 108
remote control, 14, 17, 19, 44, 45, 46, 73, 74, 75, 76, 77, 78, 79, 107, 122, 124, 144, 157
Revision 1 board, 30
rotary encoder, 14, 16, 24, 26, 29, 30, 43, 80, 85, 87, 105, 106, 121, 148
rotary encoders, 14, 16, 17, 19, 24, 30, 37, 43, 81, 83, 84, 106, 117, 148
Rotary encoders, 30, 37, 41, 46, 74, 117
RSS, 14, 83, 84, 86, 87, 106, 143, 144, 153
SD, 153
Serial Peripheral Bus interface, 17
service mpd, 71
service radiod, 64, 76, 81, 82, 108, 124, 144
soldering skills, 24
SPI, 14, 17, 24, 42, 44, 46, 61, 153
SSID, 127, 130, 153
TCP/IP, 107, 153
TSOP38238, 37, 45
TSOP382xx, 19
type of radio, 24
UDP, 77, 107, 122, 153
URL, 83, 84, 86, 87, 90, 92, 93, 94, 95, 116, 117, 131, 136, 137, 140, 146, 152, 153
USB, 14, 17, 20, 21, 22, 23, 33, 34, 37, 38, 39, 41, 80, 85, 103, 104, 118, 119, 126, 151, 153
USB adaptor, 17
USB stick, 14, 80, 85, 103, 104
USB to Ethernet adapter, 17, 22
version 1.0 boards, 29
vintage radio, 14, 18, 25, 78, 105, 108
web interface, 14, 99, 100, 101, 128, 146, 147
WEP, 127, 153
wget, 58, 73, 94, 95, 99
WiFi, 23, 126, 130
WIFI, 55, 127, 153
wiring, 26, 27, 30, 31, 37, 42, 43, 79, 106, 111, 114, 117
wiring diagram, 160
WMA, 14, 85
WPA, 127, 128, 153
WPA2, 127, 153
XML, 94, 153