



Raspberry Pi Internet Radio

Constructors Manual



A comprehensive guide to building Internet radios using the Raspberry Pi and MPD (Music Player Daemon)

Bob Rathbone Computer Consultancy

www.bobrathbone.com

7th of July 2023

Version 7.5

Contents

Chapter 1 - Introduction	1
Document Overview	4
Quick links	4
Examples	5
Vintage Radio Conversion.....	11
Chapter 2 - Hardware overview.....	13
Raspberry Pi computer	13
The Raspberry Pi model 3B	13
Raspberry Pi model 4B	13
Audio and video output jack	14
Raspberry Pi Zero and Pi Zero W	15
Pi-Pico is not supported	15
Raspberry Pi 400	16
The HD44780U LCD display.....	17
Midas LCD displays with VEE	17
Midas Character OLED with MC0100 Controller	18
Olimex OLED 128x64 pixel screen.....	18
Sitronix SSD1306 128x64 pixel OLED	18
SSH1106 1.3-inch I2C monochrome OLED.....	19
SH1106 128x32 pixel OLED monochrome OLED.....	19
Grove LCD RGB Backlight	19
Pimoroni Products	20
Pimoroni Pirate Radio	20
Pimoroni Pirate Audio.....	20
Touch-screens	21
Raspberry Pi 7-inch touch screen	21
Adafruit 2.8 and 3.5-inch TFT touch-screens	21
Waveshare 2.8 and 3.5-inch TFT touch-screens	22
Connection via ribbon cable	22
MHS 3.5-inch RPi Display	23
Radio variants	24
Connecting the LCD display	25

Housing the radio.....	25
Building in a IR sensor and remote control.....	26
Chapter 3 – Wiring and Hardware	27
Version 1 boards (early boards)	29
Version 2, 3 or model B+ boards.....	29
Rotary encoder wiring.....	30
Using KY040 Rotary encoders	31
Rotary encoders with RGB LEDs.....	33
RGB I2C rotary encoders.....	34
Rotary encoder wiring.....	35
Optical Rotary encoders.....	36
LCD Module Wiring	37
Power supply considerations	39
GPIO Hardware Notes	40
Parts List.....	41
Chapter 4 - Construction details	42
Construction using an interface board	42
Construction using breakout boards	44
Construction using an Adafruit LCD plate.....	45
Introduction	45
Using other switches.....	46
Using the Adafruit LCD plate with the model B+, 2B and 3B.....	46
Using alternatives to the Adafruit display	46
Construction using an I2C LCD backpack	47
Adafruit I2C Backpack	47
Arduino PCF8574 I2C backpacks	48
Creating the interface board for the I2C back pack.....	48
Fitting a wake-up button.....	49
Installing an IR sensor and remote control.....	50
IR Sensor.....	50
Remote control	50
Remote Control Activity LED.....	51
Construction using an IQaudIO Cosmic Controller	52
Construction using the Pimoroni Pirate radio	53

Construction using the PiFace CAD.....	53
Installing the FLIRC USB remote control.....	54
FLIRC USB Remote Control dongle.....	54
Using DAC Sound Cards.....	55
HiFiBerry DAC.....	55
QaudIO DAC sound products	56
JustBoom DAC products.....	57
PCM5102A DAC Sound Card	58
Pimoroni audio DACs	59
Adafruit speaker bonnet.....	61
Allo DAC products	61
Construction Tips and Tricks	62
Wiring up rotary encoders and switches	62
Selecting an audio amplifier	64
Preventing electrical interference	65
Fit a mains filter	65
Connecting up a USB power adapter.....	66
Cooling the Raspberry Pi.....	66
Miscellaneous	67
Simple tone regulator	67
Using the Adafruit backlit RGB LCD display	68
Chapter 5 – System Software Installation.....	69
Conventions used in this tutorial	69
Useful installation tools	70
Finding the Raspberry Pi on a network using Fing	70
Bitvise and Putty	70
Entering system commands.....	71
Editing configuration files	72
Using the Vi editor	72
Using Nano	72
System Software installation	74
SD card creation using Raspberry Pi Imager	74
Booting the Raspberry Pi for the first time.....	80
Log into the Raspberry Pi for the first time	80

Logging in with the Raspberry Pi desktop.....	80
Logging in with SSH (Bitvise)	82
Enabling SSH using raspi-config	82
HDMI Screen not available.....	83
Disabling the DRM VC4 V3D driver	84
Preparing the Operating System for software installation	84
Update to the latest the packages.....	84
Disable booting to the desktop environment.....	85
Setting the time zone.....	86
Changing the system hostname and password	88
Configuring the Wi-fi Connection via raspi-config	88
Setting up the locale	90
Install other required packages	91
Install Pimoroni ioe-python	93
Extra packages required for Raspberry OS Lite.....	93
Disable PiWiz screen reader (Buster only).....	94
Chapter 6 - Installing the radio Software.....	95
Upgrading from previous versions.....	95
Installing the Music player daemon.....	95
Installing pulseaudio	96
Install display drivers	97
Install the Radio Daemon.....	97
Configuring the radio	99
Configure the SPI Kernel Module.....	103
Select the type of LCD display.....	104
Configuring OLEDs.....	105
Installing the HDMI or touch screen software.....	107
Configuring the audio output	109
Testing the I2C interface.....	110
Installation logs	111
Reboot to enable the software	111
Apply patches to the radio software	112
Installing Pimoroni Pirate Radio (pHat BEAT)	113
Install the Rathbone Internet radio software	114

Installing the Pimoroni Pirate Audio	114
Installing the SSD1306 OLED driver	115
Configuring HDMI or Touchscreen.....	117
Waveshare TFT device installation	117
MHS 3.5-inch RPi display installation.....	118
Installing LUMA monochrome OLEDs.....	119
Installing the Grove LCD RGB	120
Installing the Adafruit 2.5 and 3.5-inch TFT touchscreen.....	120
Using other touch screens	121
Installing PiFace CAD software.....	122
Setting the mixer volume.....	123
Configuring other sound devices	124
Configuring a USB sound device	124
Configuring a Sound Card	125
Configuring Allo Sound Cards.....	127
Connecting a Bluetooth device	129
Install the Bluetooth software	129
Pairing a Bluetooth device	129
Testing Bluetooth.....	131
Configuring the radio and MPD software to use Bluetooth	132
Testing the Music Player Daemon MPD	133
Manually configuring sound cards.....	133
Configuring MPD to use pulseaudio	134
Installing the Infra-Red sensor software.....	135
Install the IR remote control software.....	135
Configuring the IR remote control	140
Method 1 – Using ir-keytable to create the IR configuration	140
Method 2 – Use irrecord to create the IR configuration	146
Configuring roaming Wi-Fi	152
Comitup Wi-Fi roaming	153
Installation from the Comitup Image.....	153
Installing from a Raspbian package	155
Installing the Web interface.....	157
Install the Apache Web Server.....	157

Install the Web Browser server pages	160
Start the radio Web interface	161
Install version 2.x of the Web interface	163
Changing the Web Interface Radio photo	169
Upgrading the Music Player Daemon	170
Installing the MPD upgrade from a Debian package	170
Installing the latest MPC client	171
Compiling and installing the latest Music Player Daemon	172
Installing the speech facility.....	172
The /var/lib/radiod/voice file.....	172
Testing espeak.....	172
Speech Operation	174
Suppressing an individual message	174
Keeping the radio software up-to-date	175
Backing up the SD card	175
Chapter 7 – Configuration.....	176
Configuring the HDMI or Touch Screen	176
Configuring GPIO outputs	177
Switches and rotary encoders GPIO assignments.....	177
Disabling button or rotary encoder GPIOs.....	177
LCD display GPIO assignments	178
Configuring button interface with pull up resistors.....	178
Configuring the remote control activity LED	178
Testing the remote control activity LED.....	178
Specifying the action for exit/shutdown action.....	179
Changing the date format.....	179
Configuring the IQaudIO Cosmic controller and OLED	180
Configuring the Adafruit LCD backlight colours.....	180
Configuring startup mode for Radio or Media player.....	181
Configuring the volume display	181
Configuring the volume range	182
Configuring the MPD client timeout	182
Changing the display language	183
Creating a new language file.....	183

Configuring Music Player Daemon CODECs.....	184
Configuring an RSS feed	184
Configuration of the mute button action	184
Configuring the Alsa Equalizer	185
Disabling the Alsa equalizer	187
Configuration of the FLIRC USB dongle.....	188
Configuring FLIRC from the command line	189
Configuring the display scroll speed	191
Configuring Russian/Cyrillic text	191
Configuring European languages	192
Configuring the Internet Check URL and port.....	192
Configuring MPD interface.....	193
Configuring the remote-control parameters	193
Configuring the radio software UDP server listening port	193
Configure the I2C bus number	194
Configuring character LCD lines and width.....	194
Configuring Rotary Encoders	194
Configuring the Menu rotary switch.....	194
Configuring soft volume (SoftVol).....	195
Chapter 8 – Operation	196
Operation of LCD and OLED versions.....	196
Starting and stopping the program.....	196
Push buttons or Rotary encoders operations	197
Radios with push buttons operation.....	198
Radios with rotary encoders operation	199
Mute function	200
Operation of HDMI and touch screen displays	200
The graphical screen	200
The display window	201
The search window	201
Smaller TFT screens.....	203
Artwork display	203
Volume and Mute controls	204
Source selection.....	204

Other graphic window controls	204
Python pygame colour constants	207
Graphic screen keyboard controls	207
The Vintage Graphic Radio.....	207
Switching between graphics programs.....	209
Configuring a screen saver.....	209
Playing Media.....	210
Playing MP3 and WMA files.....	210
Organising the music files.....	210
MPD Logging	211
Radio program logging.....	211
Other useful logs.....	211
Installation and Configuration Logs	211
Configuration and status files	212
Using the Timer and Alarm functions	212
Setting the Timer (Snooze)	212
Setting the Alarm	213
Using the Alarm and Timer functions together	213
Music Player Clients	214
Using the MPC client.....	214
Adafruit RGB Plate changing colours	214
Shutting down the radio	215
Creating and Maintaining Playlist files.....	215
Creating new playlists	215
Changing the character set	218
Directly creating the Radio.m3u playlist	219
Creating Media playlists.....	220
Maintaining playlists using external MPD clients	223
Accessing Shoutcast	225
Using the get_shoutcast.py program.....	225
Using the Shoutcast Web Interface	226
Using old 5.x Radio playlists	228
Radio stream resources on the Internet.....	228
Getting a radio stream from a Web browser.....	229

Overview of media stream URLs.....	229
M3U and M3U8 Files.....	229
PLS file format.....	230
ASX file	230
Direct stream URLs.....	231
Listening to live Air Traffic Control (ATC).....	232
Mounting a network drive	235
Finding the IP address of the network drive.....	235
The CIFS mount command.....	235
Older NAS drives sec security option.....	236
The NFS mount command	236
Display the share directory	236
Un-mounting the /share directory.....	237
Copy the mount command to the configuration.....	237
Loading the media playlists.....	237
Update the playlists for the new share.....	237
Create a Playlist for the share	237
Disabling the share.....	237
Further information	238
Troubleshooting mount problems.....	238
Controlling the Music Player daemon from Mobile devices	238
Android devices.....	238
Apple devices	239
Chapter 9 -Troubleshooting.....	240
Boot problems.....	240
Newly created desktop version of Raspberry Pi OS will not boot	240
The Raspberry Pi will not boot from a previously good SD card	241
Bad SD card	241
Installation problems	242
Missing package dependency problems.....	242
Confused or unsure of wiring.....	242
Unexpected message during an upgrade	242
Network problems	243
Checking the network quality	243

Checking the Wi-Fi signal strength.....	244
Checking response times with ping	245
Reboot your router	246
Checking the Internet connection	246
Wi-Fi is currently blocked by rfkill.....	246
Wireless network keeps dropping out.....	247
Cannot find my Raspberry Pi.....	247
HDMI/Touchscreen problems.....	247
Screen size error	247
HDMI/Touchscreen radio does not start	248
Test the graphic version of the radio.....	248
HDMI/Touchscreen is displaying upside-down.....	249
The touch screen displays a lightning symbol.....	249
The touch screen displays a thermometer symbol.....	249
Sound is heard but the graphical radio program will not start.....	249
The HDMI/Touchscreen program only displays a blue screen	249
Cannot launch Graphical radio by clicking on the Desktop Icon	249
Clicking on one of the radio desktop icon asks for an execution option	249
Trouble shooting problems with MPD.....	250
MPD fails to install	250
Music Player Daemon won't start.....	251
The MPD program may display a socket error	252
The MPD daemon complains about the avahi daemon	252
LCD Problems	253
LCD screen not showing anything.....	253
The LCD only displays hieroglyphics	253
The LCD displays hieroglyphics or goes blank occasionally	253
LCD backlight not working	253
LCD only displays dark blocks on the first line	254
Constant alternate display of Station Name and Volume	254
Display scrolling too fast or too slow	254
Adafruit LCD backlight problems	254
Grove JHD1313 LCD with RGB backlight	254
Pimoroni Pirate Radio problems.....	255

Volume UP button (Y button) not working.....	255
Playlist problems.....	255
The display shows the message “No playlists”	255
Cannot play newly mounted network share.....	255
I2C and SMBUS problems	255
Import errors.....	255
PiFace CAD and SPI problems	256
PiFace CAD not detected	256
Olimex OLED problems	256
Radio does not start with Olimex screen.....	256
OLED Screen is displaying upside down.....	256
Pimoroni Pirate Audio crashes.....	256
Rotary encoder problems	257
Testing rotary encoders	257
Keep getting erroneous button-down events	257
RGB I2C rotary encoder - Missing module ioexpander.....	257
Stream decode problems.....	257
Cannot mount remote network drive.....	258
Sound problems	259
Re-run the audio configurator program	259
Check to see if any sound cards are configured	259
HiFiBerry or other types of DAC no sound.....	259
Bluetooth device no sound	260
Noisy interference on the radio.....	264
Humming sound on the radio	264
Speaker Tests	264
Cannot change volume when running Airplay.....	265
Volume control errors.....	265
Operational problems	265
When selecting the source, the USB stick isn’t shown.....	265
Radio daemon doesn’t start or hangs.....	265
Volume control not working with DAC or USB speakers	266
The radio keeps skipping radio stations.....	266
Source selection only shows the radio playlist	266

Shoutcast playlist not created	266
A station plays for a few seconds then skips to the next one	266
The volume keeps getting reset to a 100% when the radio is restarted	267
Raspberry Pi shuts down when the Menu button is pressed.....	267
Incorrect clock time displayed	267
IR remote control problems.....	268
The irrecord program complains that lircd.conf already exists	268
The irrecord cannot open /dev/lirc0.....	268
Remote control software does not start up	268
Running irradio gives a ModuleNotFoundError for _client	270
Cannot select source through the Web interface.....	270
Cannot boot from the SD card	271
Using the diagnostic programs	272
Cannot start Bluetooth control program (bluetoothctl).....	272
Running test programs.....	274
Running the radio program in diagnostic mode	274
Using the LCD test code	274
Testing push buttons program.....	274
Testing rotary encoders	274
Testing RGB I2C rotarybencoders	275
The display_current program	276
The wiring program.....	276
The test_gpios program	278
The display_model program	279
The display configuration program.....	279
Running the radio program in nodaemon mode	280
Creating a log file in DEBUG mode.....	280
Display the kernel details.....	281
Displaying information about the Raspberry Pi.....	281
Displaying the GPIO information	281
Displaying information about the Operating system.....	281
Chapter 10 - Streaming to other devices using Icecast2	283
Inbuilt MPD HTTP streamer	283
Introduction to Icecast.....	283

Installing Icecast.....	283
Overclocking older Raspberry PI's	285
Icecast2 Operation.....	285
Switching on streaming on and off	285
Playing the Icecast stream on Windows	286
Running the Icecast Administration Web pages.....	287
Playing the Icecast2 stream on an Apple IPad.....	288
Playing the Icecast2 stream on an Android device	289
Visual streaming indicator	289
Graphical radio streaming operation.....	289
Troubleshooting Icecast2.....	289
Problem - Icecast streaming page says it can't be displayed.....	290
Problem – No Mount Point displayed.....	290
Problem - Cannot play the stream on my Android device.....	290
Problem – Music keeps stopping or is intermittent	290
Chapter 11 - Setting up Spotify.....	291
Spotify hardware requirements.....	291
Spotify installation	291
Spotify operation	294
Exiting Spotify	296
Troubleshooting Raspotify.....	296
Installation problems	296
Raspotify exits with a 101 error code	296
The client connects to Raspotify but no sound heard	296
Cannot change Raspotify volume	297
Raspotify initial sound too loud	297
Chapter 12 - Setting up Airplay.....	298
Installation	298
Configuring the Airplay feature	298
Airplay service check.....	299
Using Airplay on the radio	299
Chapter 13 - Internet Security	301
Some golden Internet Security rules	301
SSH keys installation	302

Raspberry Pi ssh keys	302
Generate a client key	303
Add client public key to Raspberry Pi authorised keys	303
Firewall set-up.....	303
Chapter 14 - Frequently asked questions (FAQs)	305
What is the login name and password?.....	305
How do I change the order of the radio stations?	305
Why are some station names not being displayed in the Web interface?.....	305
Why doesn't the Web interface display URLs until a station is selected?.....	306
Why are music tracks played randomly when loaded?	306
Can the volume be displayed as blocks instead of Volume nn?.....	306
Why do I see a station number on LCD line 3?	306
Is it possible to change the date format?	306
Is there a pause & resume function?.....	306
Is there a reboot or shutdown option?.....	306
Why do I see a different station name from the one in the playlist?	307
What Rotary Encoder can I use for this project?	307
Can this code or documentation be re-used in other projects?.....	307
Can I use an Electronic Ink display?	307
Can you make or sell me a radio?.....	307
Can you recommend the hardware for my project?	307
Can you make a change to the software for my project?.....	308
What if I want to try different hardware?	308
Chapter 15 - Source files and package build	309
The Radio program	309
The Radio Daemon.....	309
The Display Class	309
The Graphical Screen radio programs	310
The Graphics display class.....	310
The Graphics controls class.....	310
The OLED class	310
The button class.....	310
The rotary class	310
The Cosmic controller Class	310

The Event class.....	310
The Menu class	311
The Message class.....	311
The Language class.....	311
The Log class	311
The Volume class	311
The Configuration Class	311
The RSS class	311
The Translate class.....	311
The create_stations program.....	311
The update_stationlist program	311
The display_current program	312
The display_model script	312
The configure_radio.sh script	312
The playlist creation program.....	312
The configure_audio.sh script.....	312
The configure_audio_device.sh script	312
The configure_ir_remote script	313
The set mixer id script.....	313
The UDP network communications class.....	313
The Status LED class.....	313
The Airplay Class	313
The Menu Switch class.....	313
Current_station.....	314
The init file	314
Alsa sound files	314
The remote control daemons	314
The IR remote-control ireventd.service	315
The IR Remote Control irradiod.service.....	315
Downloading the source from GitHub.....	317
Chapter 16 Advanced topics	318
Booting from a USB drive or stick	318
Pi Zero W.....	319
USB Disk Drive power consumption	319

Enabling the Raspberry Pi to boot from a USB drive	319
Creating the bootable USB drive.....	323
Modifying the EEPROM boot order	324
USB 2.x SSD disk drive not booting	325
USB 3.x SSD disk drive slow booting.....	325
Boot up the Raspberry Pi from the USB disk drive	326
Playing music from the USB disk drive.....	327
Repairing a non-bootable SD card	328
Building your own package.....	331
Using PWM GPIOs for sound output on a Pi Zero	332
Fitting a Pimoroni On-Off Shim power switch	333
Adding a Real Time Clock (RTC)	334
Licences, acknowledgements and support.....	335
Licences.....	335
Intellectual Property, Copyright, and Streaming Media.....	335
Technical support.....	336
Acknowledgements.....	337
Disclaimer.....	338
Glossary.....	339
Appendix A - System Files used by the Radio Program	344
A.1 Files added to the system	344
/etc/radiod.conf.....	344
/etc/logrotate.d/radiod	349
/lib/systemd/system/radiod.service.....	349
/usr/lib/systemd/system/irradiod.service.....	349
/etc/asound.conf	350
Cronjob scripts	351
/etc/lirc/lircrc	352
A.2 System files modified by the installation.....	354
/etc/modules	354
/boot/config.txt	354
A.3 X-Windows radio desktop files	354
The lxsession autostart file for the desktop/touchscreen radio.....	354
Desktop radio icon files.....	355

Appendix B – Technical specification and other notes	356
B.1 – Technical specification	356
B.2 -Elecrow 7-inch touch-screen notes	357
B.3 Sound card DT Overlays.....	358
Configuring other audio devices	359
B.4 UDP messages.....	359
B.5 Cyrillic/European character LCDs/OLEDS	360
Romanization of Russian characters	360
Displaying Russian/Cyrillic or European characters.....	360
Character Translation routines	362
Appendix C – Wiring diagrams and lists.....	363
C.1 Push Button and Rotary Encoder 40-pin wiring.....	363
C.2 Push Button and Rotary Encoder 26-pin wiring.....	363
C.3 IQaudio Cosmic Controller wiring	363
C.4 Pimoroni Pirate Radio wiring	364
C.5 Pimoroni Pirate Audio wiring	364
C.6 Vintage Radio Push-button/Rotary Encoder 40-pin wiring	365
C.7 Raspberry Pi Rotary Encoder version with backlight dimmer	366
Appendix D – Using a battery pack	367
Appendix E – Explanation of Vcc, Vdd, Vss and Vee	368
Index.....	369

Figures

Figure 1 Fun radio using an old toaster (Courtesy Robert Knight)	3
Figure 2 Raspberry pi 7-inch touchscreen radio	5
Figure 3 HDMI Television running the radio	5
Figure 4 Vintage tuning touch-screen radio	5
Figure 5 Adafruit 3.5-inch TFT	6
Figure 6 Radio using the Adafruit LCD plate	6
Figure 7 Lego Internet Radio.....	6
Figure 8 Pi radio using rotary encoders	6
Figure 9 Old Zenith radio using rotary encoders	7
Figure 10 Transparent Perspex Radio	7
Figure 11 Perspex radio rear view	7
Figure 12 The Radio running on a Pi Zero	7
Figure 13 Boom Box radio front view	8
Figure 14 Boom Box Radio rear view	8
Figure 15 Raspberry Pi Wine Box radio.....	8
Figure 16 Wine box internet radio internal view.....	8
Figure 17 Very small radio using the Cosmic Controller	8
Figure 18 RPI radio with two-stage valve amplifier	9
Figure 19 The RPi valve radio chassis view	9
Figure 20 Pimoroni Pirate Radio	9
Figure 21 PiFace CAD Radio with IR Remote Control	9
Figure 22 Vintage look-and-feel Internet radio	10
Figure 23 Vintage look-and-feel - rear view.....	10
Figure 24 Philips BX490A (1949) Vintage Internet Radio.....	11
Figure 25 Vintage radio using a touch screen.....	12
Figure 26 Vintage Radio with AM transmitter	12
Figure 27 Lusya DIY 3-channel AM transmitter	12
Figure 28 Raspberry PI Model 3B Computer	13
Figure 29 Raspberry Pi Model 4B Computer.....	13
Figure 30 USB-C plug.....	14
Figure 31 Rasberry Pi MicroHDMI cable	14
Figure 32 Raspberry PI B+ AV cable	14
Figure 33 Raspberry Pi Zero	15
Figure 34 USB Ethernet adapter	15
Figure 35 Unsupported Pi Pico.....	15
Figure 36 Raspberry Pi 400 running the graphical radio.....	16
Figure 37 The HD44780U LCD display.....	17
Figure 38 OLED 4 x20 LCD display	17
Figure 39 Midas LCD display with VEE	17
Figure 40 HD44780 potentiometer wiring.....	17
Figure 41 Midas character OLED.....	18
Figure 42 Sitronix SSD1306 128x64 pixel monochrome OLED	18

Figure 43 1.3-inch SH1106 I2C OLED.....	19
Figure 44 - 0.91-Inch 128x32 OLED.....	19
Figure 45 Grove LCD RGB Backlight	19
Figure 46 Pimoroni Pirate Radio - Rear view	20
Figure 47 Pimoroni Pirate Audio	20
Figure 48 Raspberry Pi 3 with 7-inch touch screen	21
Figure 49 Adafruit 3.5-inch TFT touchscreen.....	21
Figure 50 Waveshare 2.8-inch TFT touch screen.....	22
Figure 51 TFT connected by a 40-pin male/female ribbon cable	23
Figure 52 MHS-3.5-inch RPi Display (Courtesy Brent Fraser)	23
Figure 53 Some examples of radio cases	25
Figure 54 IR Sensor and Remote control	26
Figure 55 Adafruit and IR sensor and activity LED	26
Figure 56 Push-button Wiring version 1 boards	29
Figure 57 Push-button wiring version 2 onwards boards	29
Figure 58 Rotary Encoder Diagram	30
Figure 59 Rotary encoder with push switch	30
Figure 60 Rotary encoder pin-outs	31
Figure 61 KY-040 Rotary Encoder	31
Figure 62 KY040 Missing 10K resistor	31
Figure 63 KY-040 Circuit Diagram	32
Figure 64 Sparkfun RGB rotary encoder	33
Figure 65 RGB Rotary Encoder circuit	33
Figure 66 Transparent knob	33
Figure 67 Pimoroni RGB I2C Rotary Encoders.....	34
Figure 68 Radio project using RGB I2C rotary encoders	34
Figure 69 HD44780U LCD electrical circuit	38
Figure 70 Wire LCD pin 1 (GND) and 5 (RW) together	38
Figure 71 GPIO Numbers.....	40
Figure 72 26-pin header extender	40
Figure 73 40-pin Interface board with ribbon cable	42
Figure 74 Interface board with LCD screen attached	42
Figure 75 Radio controls connections.....	43
Figure 76 Interface board overview	43
Figure 77 GPIO header breakout board	44
Figure 78 Adafruit LCD plate	45
Figure 79 Adafruit LCD plate with ribbon cable adapter	45
Figure 80 Chinese 1602 I2C LCD.....	46
Figure 81 Enabling the backlight	46
Figure 82 Adafruit I2C Backpack	47
Figure 83 LCD connected to an Adafruit I2C backpack	48
Figure 84 Arduino I2C backpack.....	48
Figure 85 Ciseco Humble PI I2C interface board.....	49
Figure 86 The I2C backpack interface board.....	49
Figure 87 Wake-up button.....	49

Figure 88 TSOP38238 IR sensor	50
Figure 89 Soldering precautions	50
Figure 90 LED polarity	51
Figure 91 Adafruit plate with IR sensor and activity LED.....	51
Figure 92 IQaudIO Cosmic controller and OLED display	52
Figure 93 Lego radio with IQaudIO Cosmic controller and OLED	52
Figure 94 PiFace CAD and Raspberry PI	53
Figure 95 PiFace CAD in a case.....	53
Figure 96 HiFiBerry DAC Plus	56
Figure 97 HiFiBerry mounted on the Rasberry Pi	56
Figure 98 IQaudIO DAC plus.....	57
Figure 99 IQaudIO Pi-DigiAMP+	57
Figure 100 JustBoom Amp HAT.....	57
Figure 101 JustBoom Amp Zero pHAT	57
Figure 102 JustBoom Zero stacker requirements	57
Figure 103 Using the 40-pin stacker	58
Figure 104 Pimoroni pHat DAC	59
Figure 105 Pimoroni Audio DAC Shim for Pi Zero/W	60
Figure 106 Adafruit Speaker Bonnet.....	61
Figure 107 Allo Piano 2.1 HiFi DAC with woofer.....	61
Figure 108 Allo Piano 2.1 HiFi DAC (Rear).....	61
Figure 109 Rotary encoder wiring components	62
Figure 110 Using wire strippers	62
Figure 111 Tinning the wires with solder.....	62
Figure 112 Soldering up the switch.....	63
Figure 113 Shrink shrink-wrap with a hair dryer	63
Figure 114 Connecting the rotary encoder an interface board	63
Figure 115 PC speakers	64
Figure 116 Velleman 30W stereo amplifier	64
Figure 117 IQaudIO DAC and 20W Amplifier	64
Figure 118 Vintage radio PA input	64
Figure 119 JVC Bluetooth Speakers	64
Figure 120 Audio output jack amplifier	64
Figure 121 Clip on ferrite core	65
Figure 122 Loop +5V supply around the core	65
Figure 123 Various mains filters	65
Figure 124 Integrated mains socket and filter.....	65
Figure 125 3.5mm Jack Ground Loop Isolator	66
Figure 126 Connecting up a USB power adapter.....	66
Figure 127 Heat sink kit.....	67
Figure 128 Cooling fans.....	67
Figure 129 Simple tone control circuit.....	67
Figure 130 Tone control board	68
Figure 131 IN4148 diode	68
Figure 132 The nano file editor	73

Figure 133 The nano editor help screen	73
Figure 134 Windows Laptop with SD card reader	74
Figure 135 USB SD Card reader.....	75
Figure 136 Raspberry Pi desktop	81
Figure 137 Bitvise connection dialogue	82
Figure 138 Enabling SSH.....	83
Figure 139 raspi-config main screen.....	86
Figure 140 Disabling the graphical desktop.....	86
Figure 141 Desktop enable/disable selection.....	86
Figure 142 Setting the time zone	87
Figure 143 Saving the time zone.....	87
Figure 144 Time zone country selection.....	87
Figure 145 Changing the Raspberry PI password	88
Figure 146 Setting up the Wi-Fi in raspi-config.....	89
Figure 147 Entering Wi-Fi credentials.....	89
Figure 148 Setting up the Wi-fi country.....	89
Figure 149 Selecting change locale	90
Figure 150 Generating the locale.....	90
Figure 151 Selecting the locale	91
Figure 152 Configure radio – Upgrade.....	99
Figure 153 Replace configuration file	99
Figure 154 Configure radio – User interface selection	100
Figure 155 Configure radio - Confirmation screen	100
Figure 156 Push-button voltage selection	100
Figure 157 Rotary encoder type selection	101
Figure 158 Configure radio - wiring selection	101
Figure 159 Configure radio - Display interface selection 1	102
Figure 160 Configure radio - Display selection 2	102
Figure 161 Olimex OLED flip display	103
Figure 162 Enable SPI Kernel Module.....	103
Figure 163 Enable SPI kernel module option	104
Figure 164 Configure radio - Display type selection	105
Figure 165 Configure radio audio output option.....	105
Figure 166 Selectin an OLED display	106
Figure 167 Luma devices.....	106
Figure 168 Selecting the OLED I2C address	106
Figure 169 Touchscreen selection	107
Figure 170 Selecting the type of radio display.....	107
Figure 171 Configuring the HDMI or touch screen display startup	107
Figure 172 Configuring the graphical radio full screen.....	108
Figure 173 Audio configuration selection.....	108
Figure 174 Selecting the audio output device	109
Figure 175 The I2C bus display using the i2cdetect program	110
Figure 176 Basic Alsa sound mixer.....	123
Figure 177 Configure USB DAC	125

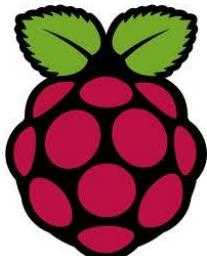
Figure 178 The USB PnP Alsa Mixer	125
Figure 179 Configuring add-on DAC sound cards	126
Figure 180 Set mixer analogue volume	127
Figure 181 Set mixer digital volume	127
Figure 182 Configuring bluetooth devices	132
Figure 183 Alsamixer using Bluetooth devices	132
Figure 184 IR Remote Installation program.....	135
Figure 185 IR configuration IR sensor GPIO selection	136
Figure 186 IR configuration Activity LED GPIO selection	136
Figure 187 IR Remote Control Interface type selection.....	136
Figure 188 LIRC Remotes Database	146
Figure 189 Downloading the comitup image.....	153
Figure 190 Radio Web interface	161
Figure 191 Snoopy Web interface.....	162
Figure 192 Version 2.x Web interface with O!MPD.....	165
Figure 193 OMPD database creation	165
Figure 194 The Alsa	187
Figure 195 FLIRC setup program.....	188
Figure 196 FLIRC keyboard controller.....	188
Figure 197 HDMI and Touch Screen Display.....	200
Figure 198 Graphical scree information display	201
Figure 199 Graphical radio search window	201
Figure 200 Graphical radio search functions	202
Figure 201 Display playlists.....	202
Figure 202 Display of media tracks	202
Figure 203 Displaying artists	203
Figure 204 Track artwork display.....	204
Figure 205 Airplay running on a Graphical screen.....	205
Figure 206 Changing the graphical screen theme	206
Figure 207 The vintage graphic radio on a touch-screen	208
Figure 208 Adding a new station to MPD	223
Figure 209 Adding new station URL in the Web interface.....	224
Figure 210 Moving and deleting entries in the Web Interface.....	224
Figure 211 Shoutcast playlist Web page.....	227
Figure 212 Shoutcast search selection.....	227
Figure 213 Shoutcast playlist summary	228
Figure 214 Live ATC Web page.....	232
Figure 215 WinAmp playing ATC live feed.....	233
Figure 216 WinAmp station information	233
Figure 217 MPDdroid set-up screen	239
Figure 218 MPDdroid play screen.....	239
Figure 219 MPDdroid play queue	239
Figure 220 SD card boot patition	240
Figure 221 The ping command	245
Figure 222 Configuring Icecast2.....	283

Figure 223 Over-clocking the Raspberry PI.....	285
Figure 224 Windows media player	286
Figure 225 Firefox embedded media player.....	287
Figure 226 Icecast2 Status	288
Figure 227 Starting the Spotify Receiver	294
Figure 228 The radio in Spotify mode.....	294
Figure 229 Spotify connecting to the radio	295
Figure 230 Listening to Spotify on the radio.....	295
Figure 231 Spotify playing a music track.....	296
Figure 232 Airplay source selection	299
Figure 233 Running an Airplay device on the radio with Cloudbreak	300
Figure 234 Raspberry Pi booting from an Intenso USB 3.0 2.5-inch disk drive	318
Figure 235 A USB SD card reader	328
Figure 236 PWM Low band pass filters.....	332
Figure 237 Pimoroni On-Off shim power switch	333
Figure 238 Raspberry Pi Radio with Pimoroni on-off shim.....	333
Figure 239 RasClock RTC V4.2 module.....	334
Figure 240 Russian/Cyrillic character LCD.....	360
Figure 241 Wiring Raspberry Pi Radio Rotary Encoder version	366
Figure 242 Raspberry Pi 5 Volt battery pack.....	367
Figure 243 Micro USB breakout board	367
Figure 244 BJT and FET transistors.....	368

Tables

Table 1 Display Type options	24
Table 2 User interface options.....	24
Table 3 Controls and LCD wiring 26 pin version	27
Table 4 Radio and DAC devices 40-pin wiring.....	28
Table 5 Radio A, B, C Rotary Encoder Wiring.....	35
Table 6 RGB LED Rotary Encoder switch wiring.....	35
Table 7 KY-040 Rotary Encoder Wiring	35
Table 8 RGB I2C Rotary Encoders wiring.....	36
Table 9 LCD module wiring for 26 and 40-pin Raspberry Pi's.....	37
Table 10 Parts list (LCD versions)	41
Table 11 I2C Backpack connections	48
Table 12 Remote Control Activity LED	51
Table 13 Adafruit backlit RGB display wiring	68
Table 14 Additional system packages	91
Table 15 PulseAudio installation options.....	96
Table 16 Display driver software installation	97
Table 17 OLED rivers	105
Table 18 IR Sensor Pin outs.....	135
Table 19 Remote Control Key names and functions.....	148
Table 20 Push Button Operation.....	198
Table 21 Rotary Encoder Knob Operation	199
Table 22 Graphic screen keyboard command	207
Table 23 Common MPC commands.....	214
Table 24 Example playlists	215
Table 25 Playlist files and directories.....	216
Table 26 Wi-Fi signal strength.....	244
Table 27 Display classes	309
Table 28 Asound configuration files	314
Table 29 The irradiod service.....	315
Table 30 Sound card Device Tree overlays	358
Table 31 UDP messages	359
Table 32 Character font table selection.....	361
Table 33 Code page translation files.....	362
Table 34 Russian Cyrillic and Romanization display configurations	362
Table 35 40-PinPush-buttons/Rotary encoder Wiring.....	363
Table 36 26-PinPush-buttons/Rotary encoder Wiring.....	363
Table 37 IQaudio Cosmic Controller Wiring	363
Table 38 Pimoroni Pirate radio (pHat BEAT) Wiring	364
Table 39 Pimoroni Pirate radio Audio Wiring	364
Table 40 40-PinPush-buttons/Rotary encoder Wiring.....	365
Table 41 Status LED indications	365
Table 42 Rotary menu switch	365
Table 43 Supply voltage definitions summary	368

Chapter 1 - Introduction



This manual describes how to create one of the most popular Internet Radios using the Raspberry PI educational computer. This manual provides a detailed overview of construction and software installation.

The source and basic construction details are available from the following Web site:

https://bobrathbone.com/raspberrypi/pi_internet_radio.html

The main features of the Raspberry PI internet radio are:



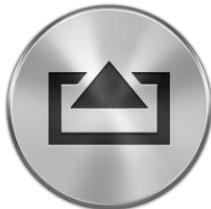
Raspberry Pi Internet Radio

Turn your Raspberry Pi into an Internet Radio using a variety of designs as shown in this manual. It runs on both 32 and 64-bit systems.



Media Player

Play your favourite MP3 tracks from a USB stick, SD card or from a NAS (Network Attached Storage).



Airplay Receiver

This design allows the Raspberry Pi to act as an Airplay receiver. Music tracks can be played from your Apple or Android mobile phone or tablet.



Spotify Receiver

Turn your Raspberry Pi into a Spotify Receiver. This requires a Spotify Premium account.



RSS Feed Reader

This software also allows you to read any configured RSS feed. For example, your favourite news feed.



Bluetooth speaker/headphone support

This manual contains instructions how to run the radio software with Bluetooth speakers or headphones. Bluetooth versions 1.x through 5.x supported depending upon the Raspberry Pi model used.



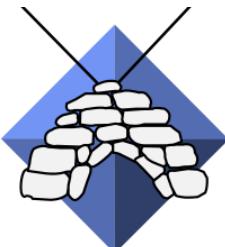
Digital Clock

The Internet Radio software displays as standard a digital clock and date with alarm and snooze functions.



Shoutcast

Shoutcast radio is a streaming audio which is used by some 50,000 Internet Radio stations across the internet. This radio software, using a Web interface, allows multiple playlist creation from Shoutcast radio stations by genre or country which can then be selected through the radio or Web interface menus.



Icecast

Icecast is free streaming software which supports a variety of streams such as MP3 and OGG. Icecast can optionally be installed on the Raspberry Pi and allows the currently playing station or track to be streamed around the local network or out to the Internet (Legal and Copyright issues apply).



User Interface and displays

This design caters for a number of user interfaces such as push-buttons, rotary encoders or touch screens. Also, a number of displays such as 2 and 4-line LCDs, OLED displays, TFT or full graphical displays such as HDMI and touch-screen are supported. Examples are shown later on in this manual.



Web interface

The radio software includes an optional Web interface powered by Apache. This allows stations and playlists to be selected via Web pages on your PC, mobile telephone or tablet.



The **eSpeak** engine is a small, lightweight text-to-speech (TTS) program that supports a large number of languages. It is used with the radio software to assist blind or visually impaired users by “speaking” the menus. It can also be used with radios without a screen to navigate the menus.



Language files enable the messages to be displayed in the user's own language, such as French, Dutch or Spanish. The user can easily create their own language file if required.



Highly configurable allowing a broad range of displays such as 2 or 4-line LCDs, TFTs, graphic screens or HDMI monitors to be selected at installation time. Likewise, the user input devices such as rotary encoders or buttons etc. An audio configuration script selects a range of audio devices such as DACs or Bluetooth speakers.



Support for English, Russian/Cyrillic and Western European character set, dependent upon LCD capabilities. Both native and Romanized (Convert to Latin) characters supported.

A full specification can be found in *Appendix B – Technical specification* on page 356.

REMEMBER TO HAVE FUN DOING THIS PROJECT!



Figure 1 Fun radio using an old toaster (Courtesy Robert Knight)



This is a large document. Use the *Document Overview* on page 4 to navigate to the section you are interested in. The chapters are laid out in the order you will need them.



This manual is continually being updated. Please check to see if there is a more up to date manual before commencing any work.

Is the size of this manual worrying you then try starting with *Raspberry Pi Internet Radio, A Beginners Guide* at:

<https://bobrathbone.com/raspberrypi/documents/RPi%20Radio%20beginners%20guide.pdf>

Document Overview

This is a big document but has been organised logical sections to make it easier to navigate.

Section	Topic	Page
Chapter 1	Introduction and examples	1
Chapter 2	Hardware overview	13
Chapter 3	Wiring information	27
Chapter 4	Construction details	42
Chapter 5	System software installation	69
Chapter 6	Radio software installation	95
Chapter 7	Configuring the radio	176
Chapter 8	Operation	196
Chapter 9	Troubleshooting	240
Chapter 10	Icecast streaming installation	283
Chapter 11	Spotify Installation	291
Chapter 12	Airplay Installation	298
Chapter 13	Internet Security	301
Chapter 14	Frequently Asked Questions FAQs	305
Chapter 15	Source files and package build	309
Chapter 16	Advanced topics, booting from a USB drive	318
Licences	Licence, acknowledgments and support	335
Glossary	List of abbreviations used	339
Appendix A	Radio program files	344
Appendix B	Technical specification	356
Appendix C	Wiring diagrams	363
Index	Document index	369

Quick links

Topic	Page(s)	Topic	Page(s)
Adafruit	45	Pimoroni products	20,53,113,364
Airplay	298	PiFace CAD	53,103,122
Buttons (Switches)	177	Playlist's creation	215
Bluetooth speakers	129	Radio software installation	95, 176
DAC sound cards	55	Radio software operation	196
Glossary	339	RSS feed	184
GPIO configuration	177	Rotary encoders	177
HDMI TV or screens	176	Shoutcast	225
I2C backpacks	47	Spotify	291
IQaudIO Cosmic Controller	52	Speech facility (Espeak)	170
Interface boards	42	Switches (Buttons)	177
Icecast streaming	283	Touch screens	17,176
IR sensors	50,54,188	USB Sound card	124
LCD displays	25,31	Vintage radios	5
Media	210	Web interface	157
Network drives (NAS)	235	Wi-Fi	152
OLED displays	18,256	Wiring	27



Please note that there is also a full document index on page 369 at the end of this document.

Examples

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 the following examples and decide which one is the best for you. Some examples are shown in the following pages. This manual is designed to provide inspiration for your own ideas and unique solution to building an Internet Radio using the Raspberry Pi.

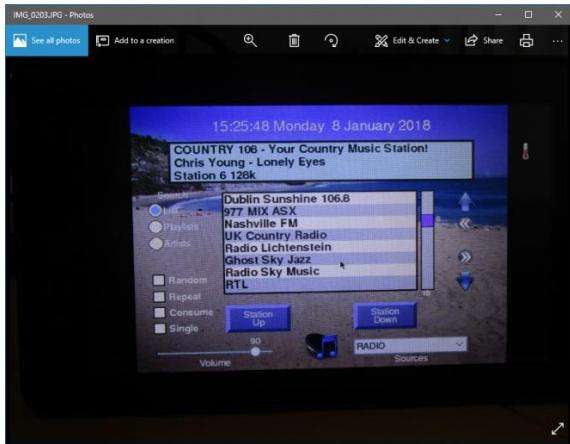


Figure 2 Raspberry pi 7-inch touchscreen radio



Figure 3 HDMI Television running the radio



Figure 4 Vintage tuning touch-screen radio

The Radio software supports the Raspberry Pi 7-inch touch screen. Using the graphic version of this software, the radio can be operated using the touch screen or a mouse and HDMI screen or TV with HDMI. A keyboard can also be attached and used to operate the radio. The touch screen version supports the same functionality as the LCD versions of the radio except for timer and alarm functions. The touch screen can also be configured to use either rotary encoders or buttons.

The HDMI/Touch screen version of the radio can also be configured to run in a window on the Raspberry Pi desktop. Here it is running on the HDMI input of a typical flat-screen television. It can also be configured to use an IR remote control using a FLIRC USB IR detector.

As an alternative to the above design this touch-screen radio is made to look like a vintage radio with a tuning dial. The green slider marks the currently playing station. When a station name is touched on the screen then the slider jumps to that position and plays the selected radio station. The design supports multiple pages of radio stations which can be scrolled left or right. The volume control slider is at the bottom of the screen. This version currently only plays radio stations and not media or Airplay. The touchscreen can also be configured to use either rotary encoders or buttons.



Figure 5 Adafruit 3.5-inch TFT



Figure 6 Radio using the Adafruit LCD plate



Figure 7 Lego Internet Radio



Figure 8 Pi radio using rotary encoders

This example shows an Adafruit 3.5-inch TFT (Thin Film Transistor) touch-screen running the graphical version of the software (Version 6.7 onwards). This is the smallest screen that is currently supported.

Installation of the Adafruit TFT touchscreen is found in the section called *Installing the Adafruit 2.5 and 3.5-inch TFT* on page 120.

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.

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

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 9 Old Zenith radio using rotary encoders

Example of the PI radio from Jon Jenkins built into an old Zenith valve radio case. 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 10 Transparent Perspex Radio



Figure 11 Perspex radio rear view

The above example built by the author has a transparent perspex front and back panel. It uses a Raspberry Pi with a HiFiBerry sound card and a Velleman 30 watt amplifier.



Figure 12 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 built by the author. 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 13 Boom Box radio front view

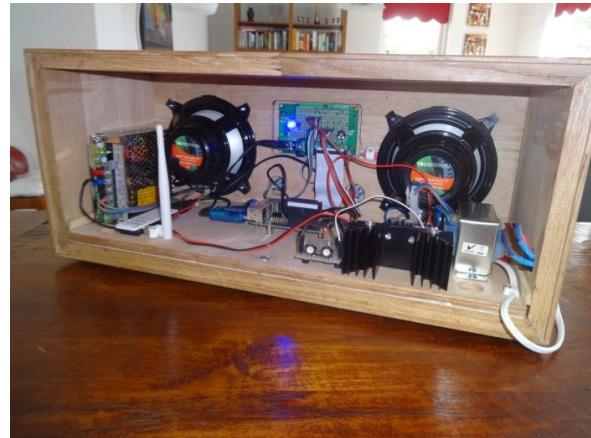


Figure 14 Boom Box Radio rear view

Below is a Raspberry Pi radio built into a old wine box. It uses a 2x8 character LCD and rotary encoders. The amplifier and loud speakers are from a set of old PC speakers.



Figure 15 Raspberry Pi Wine Box radio



Figure 16 Wine box internet radio internal view



Figure 17 Very small radio using the Cosmic Controller

Here is a really cute radio made using the IQaudI0 Cosmic Controller and Olimex 128x64 pixel OLED display. The Cosmic Controller provides an excellent solution where space is limited or you simply want a very small radio.

The audio output is on the rear of the case.

Note: The Cosmic Controller has been discontinued and is no longer available.

Below is a fascinating use of both modern and bygone era technology. The radio shown below was created by Broesel from Austria. In this design a Raspberry Pi has been used with the software described in this manual. However, the audio amplifier has been constructed with an ECL84 vacuum valve. The ECL84 valve provides a two-stage mono audio amplifier driving an elliptical wide frequency response loud speaker. Broesel has very kindly provided the full construction details at the Radio Board Forum. See: <http://theradioboard.com/rb/viewtopic.php?t=6314>



Figure 18 RPi radio with two-stage valve amplifier



Figure 19 The RPi valve radio chassis view



Figure 20 Pimoroni Pirate Radio

The radio software supports the Pimoroni Pirate radio with pHat BEAT.

The pHAT BEAT gives high-quality, digital, amplified, stereo or mono audio and 16 RGB LEDs, in two rows of 8, which are ideal as a VU (Volume Unit) indicator, and 6 buttons to control the radio (Five on the left and one at the top). See:
<https://shop.pimoroni.com/products/pirate-radio-pi-zero-w-project-kit>



Figure 21 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 but is quite slow. See:
http://www.piface.org.uk/products/piface_control_and_display/

The PiFace CAD has 5 push buttons, a reset button (not currently used) and an Infra Red (IR) sensor 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.

Below is an example of a Raspberry Pi Internet radio, made by the author, running the graphic version of the radio which gives a vintage look-and-feel to the final radio. Two vintage Bakelite knobs help complete the vintage appearance.



Figure 22 Vintage look-and-feel Internet radio

The Raspberry Pi model 3B outputs to a Pimoroni 3W stereo D-Class amplifier driving two 3-inch loudspeakers. It is housed in a wooden case painted black with two vintage radio Bakelite knobs.



Figure 23 Vintage look-and-feel - rear view

Vintage Radio Conversion

This version of the software allows for the program to be configured without a screen for use with a vintage radio as shown below:



Figure 24 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 a 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 contained in the following documents:

<http://www.bobrathbone.com/raspberrypi/documents/Raspberry%20PI%20Vintage%20Radio.pdf>

<http://www.bobrathbone.com/raspberrypi/documents/Raspberry%20PI%20Vintage%20Radio%20Operating%20Instructions.pdf>

The illustration below shows a French Radio Schneider Frères Rondo from the 1950's which has been converted to an internet radio by Franz-Josef Haffner, from Germany.



Figure 25 Vintage radio using a touch screen

What makes this project also very interesting is that he has removed all of the RF section of the valve radio leaving only the audio amplifier and power supply. This is an excellent example of combining old and new technology to extend the life of these increasingly rare radios.

See the Vintage Radio supplent for further details using the following link:

<https://bobrathbone.com/raspberrypi/documents/Raspberry%20PI%20Vintage%20Radio.pdf>



Figure 26 Vintage Radio with AM transmitter

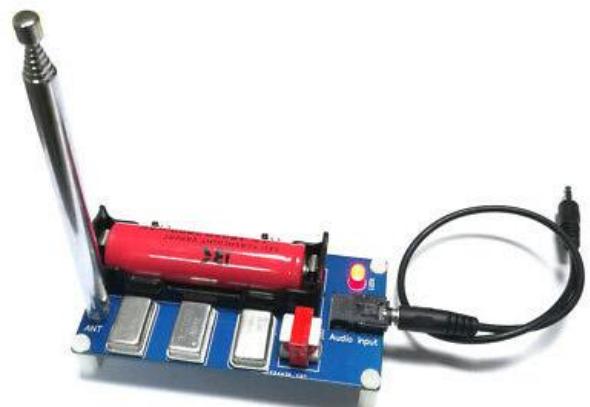


Figure 27 Lusya DIY 3-channel AM transmitter

Another way to connect to a vintage radio (if it is working) is to use a Raspberry Pi and a low power AM transmitter. In the above illustration, the box on the right contains a Pi Zero W running the radio software with a tuning knob. The audio comes out of a USB DAC which connects to the Lusya 3-channel AM transmitter which transmits on the 300 Meter wavelength (AM waveband) to a vintage radio. In this example the radio is the British made **Bush DAC 90A** medium/longwave receiver.

Details on construction can be found in the following document:

<https://bobrathbone.com/raspberrypi/documents/Raspberry%20PI%20Vintage%20Radio.pdf>

Chapter 2 - Hardware overview

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)
- Push buttons or rotary encoders for the user interface
- A display such as a HD44780U LCD, OLED or a Raspberry Pi touch-screen display

Raspberry Pi computer

The **Raspberry Pi** is a credit-card-sized single-board computer developed in the United Kingdom by the [Raspberry Pi Foundation](https://www.raspberrypi.com) originally with the intention of promoting the teaching of basic computer science in schools. It has however become immensely popular with hobbyists and engineers. The official site for the Raspberry Pi Web site is <https://www.raspberrypi.com>

The Raspberry Pi model 3B



Figure 28 Raspberry PI Model 3B Computer

The Raspberry Pi 3B has full size HDMI port.

More information on the Raspberry PI computer may be found here:

[http://en.wikipedia.org/wiki/Raspberry_Pi](https://en.wikipedia.org/wiki/Raspberry_Pi)

If you are new to the Raspberry PI try the following “Getting started” guide:

<https://www.raspberrypi.com/documentation>

Raspberry Pi model 4B

The Raspberry Pi model 4B was released in June 2019.



Figure 29 Raspberry PI Model 4B Computer



Note: The Raspberry Pi 4B requires **Raspberry Pi Buster** or **Bullseye** SD card. It will not work with earlier OS versions. However, the recommended OS for the Radio software is **Bullseye**.



Figure 30 USB-C plug

The power supply on the model 4B uses a USB-C connector. The usual micro-USB power supply will not fit used for other Raspberry Pi models will not fit. The USB-C specification allows the cable to be inserted either way around. When purchasing the model 4B also purchase the official model 4B power supply which is 5 Volt 3 Amps.



Figure 31 Rasberry Pi MicroHDMI cable

The model 4B also requires an official Raspberry Pi Micro-HDMI cable for each of the micro-HDMI ports (This is not the same as the Pi Zero HDMI adapter). Order one or two of these adaptors if using an HDMI or TV screen.

Audio and video output jack

Earlier versions of the Raspberry Pi have a separate audio output jack and composite video output. Later versions (3 and 4) of the Raspberry Pi have a new AV (Audio/Video) port which combines the audio and video signals in a single jack. Instead of using a standard composite cable, this new connector requires a 4 pole 3.5mm 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.

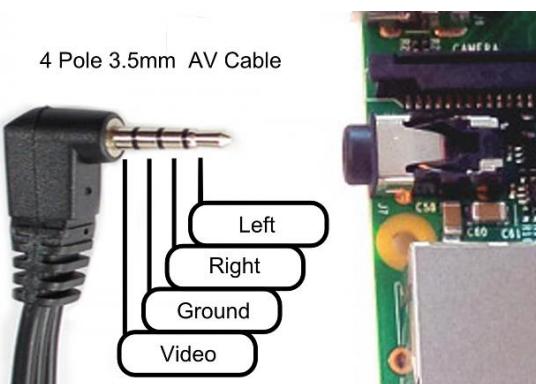


Figure 32 Raspberry PI B+ AV cable

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 and Pi Zero W

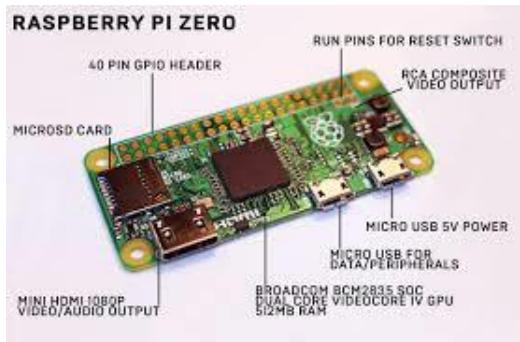


Figure 33 Raspberry Pi Zero

On the original Pi Zero network connection 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 a micro-USB to standard USB adapter. The Pi Zero W has onboard Wi-Fi and Bluetooth and is a better choice for the radio.



Figure 34 USB Ethernet adapter

In October 2021 the 1GHz quad core Pi Zero 2 was added to the list of Raspberry Pi's.

The Raspberry Pi Zero does not have an onboard audio output jack. Sound must be played through either the HDMI port or a USB sound dongle (See Figure 12) or one of the Pi Zero DAC boards available from manufacturers such as **IQaudIO**, **HiFiBerry**, **Pimoroni** or **JustBoom**. Alternatively, use Bluetooth speakers. See *Connecting a Bluetooth device* on page 129.

One useful and simplest audio device is the **Pimoroni Audio DAC Shim** for the Pi Zero/W. Because the Audio DAC SHIM adds no extra bulk to the Raspberry Pi Zero it allows the GPIO header to be accessed as normal, for example other HATs can be fitted on top of it. It pushes over the GPIO header and doesn't require any soldering. It is a very simple way to provide audio output to the Raspberry Pi Zero/W. See *Pimoroni Audio DAC Shim for Pi Zero* on page 60

It is also possible to use a so-called Low-pass filters which allow output from the GPIO PWM (Pulse Width Modulation) pins. This requires soldering and board construction skills and isn't recommended for beginners.

See *Using PWM GPIOs for sound output on a Pi Zero* on page 332 in *Chapter 16 Advanced topics*.

Pi-Pico is not supported



Figure 35 Unsupported Pi Pico

The **Pi Pico** is **NOT** supported nor will it ever be. The Pi Pico uses the RP2040 processor which is a micro-controller. It is completely different from the usual Raspberry Pi's. Unlike the Raspberry Pi, it does not run Linux and therefore cannot run the radio software or the Music Player daemon (MPD). It uses either **Circuit Python** or **C/C++** for development.

Also, there is no HDMI, Ethernet or Wi-Fi port available without additional expansion boards.

More information on the Raspberry Pi Pico can be found at:

<https://www.raspberrypi.org/products/raspberry-pi-pico/>

Raspberry Pi 400

In November 2020 the Raspberry Pi foundation released the model 400. This is housed in a keyboard case with all connectors including a 40-pin GPIO header at the back of the unit. Most notably it does not have a 3.5mm audio Jack. If connected to a HDMI monitor or TV then sound can be played through the HDMI interfaced speakers on the monitor/TV. Alternatively, a USB DAC or Bluetooth speakers can be used. As the RPi 400 has a 40-pin GPIO header it can, in theory, be connected to a suitable DAC via a ribbon cable.



Note: At the time of writing no testing with a DAC other than a USB DAC has been done by the author.



Figure 36 Raspberry Pi 400 running the graphical radio

Specification

- Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
- 4GB LPDDR4-3200
- Dual-band (2.4GHz and 5.0GHz) IEEE 802.11b/g/n/ac wireless LAN
- Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 × USB 3.0 and 1 × USB 2.0 ports
- Horizontal 40-pin GPIO header
- 2 × micro-HDMI ports (supports up to 4Kp60)
- H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES 3.0 graphics
- MicroSD card slot for operating system and data storage
- 78- or 79-key compact keyboard (depending on regional variant)
- 5V DC via USB C connector

The HD44780U LCD display



Figure 37 The HD44780U LCD display

The HD44780U 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 HD44780U LCD controller so there is a wide choice of these displays.

The latest displays use **OLED character** displays (Organic Light Emitting Diode) and give very good results and are becoming more popular when compared to traditional LCD displays.

See <https://en.wikipedia.org/wiki/OLED>



Figure 38 OLED 4 x20 LCD display

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



Also see *Configuring Russian/Cyrillic text* on page 191.

Midas LCD displays with VEE



Figure 39 Midas LCD display with VEE

Some LCDs from Midas are compatible with the HD44780U except for pin 15 (VEE) which outputs a negative voltage. For pin-out details see LCD pin outs on page 30.

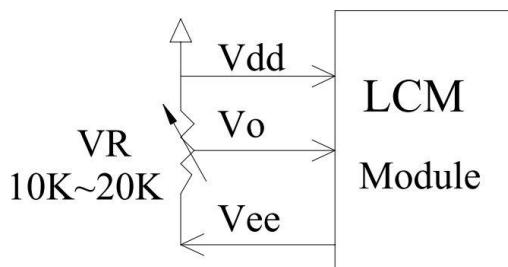


Figure 40 HD44780 potentiometer wiring

The diagram on the left shows the wiring for the 10K contrast potentiometer.

Pin	Name	Description
2	VDD	+5 Volt
3	VO	Contrast adjustment
15	VEE	Negative voltage output



Do not connect pin 15(VEE) to the +5V supply. It will damage the LCD and possibly the Raspberry Pi.



WARNING: DO NOT CONNECT AN I2C BACKPACK TO THIS TYPE OF DEVICE AS BACKPACKS CONNECT +5V ONTO PIN 15 AND WILL DAMAGE BOTH THE BACKPACK AND THE LCD.



The term **LCD** is used throughout this manual to mean both traditional **LCDs** and **OLED character** displays which are gradually replacing **LCDs**.

Midas Character OLED with MC0100 Controller

So-called Character OLEDs are gradually replacing LCDs. Midas market a wide range of such displays.



Figure 41 Midas character OLED

These displays use the **MC0100** controller which is largely compatible with the Hitachi **HD44780U** controller.

This controller can support various built-in font sets such as English, Western European, Japanese and Russian.

Since OLEDs generate their own illumination, they do not need pins 15 and 16 connected for backlighting.

Olimex OLED 128x64 pixel screen



The Olimex 128x64 pixel OLED is a low cost, low power, high contrast LCD display with a UEXT connector. It is controlled via the I2C or SPI interface. The power supply required is only in the range of 1 uA in sleep mode, 200 uA in operating mode and 7mA in display ON mode. View area is 21 x 11 mm. It is particularly useful where space is very limited.

See:

<https://www.olimex.com/Products/Modules/LCD/MOD-OLED-128x64/open-source-hardware>



Note: The Olimex OLED screen is not a particularly fast device when compared with say an LCD or graphics screen. However, its biggest advantage is its size.

Sitronix SSD1306 128x64 pixel OLED



Figure 42 Sitronix SSD1306 128x64 pixel monochrome OLED

The Sitronix SSD1306 128x64 pixel 0.96-inch OLED is display marketed under various names (Such as Makerhawk) has just four connections for the I2C interface.

The wiring from left to right is:

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

The I2C interface hex address for this device **0x3C**. It uses the **ssd1306_class.py** driver.



Note: It is not currently possible to flip the display up-side down due to limitations of the SSD1306 driver software. Use the LUMA.SSD1306 driver instead which can be flipped.

SSH1106 1.3-inch I2C monochrome OLED



Figure 43 1.3-inch SH1106 I2C OLED

This 1.3-inch monochrome display connects via the I2C interface. It uses the SH1106 display chip.

The wiring from left to right is:

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

The I2C interface hex address for this device **0x3C**. This OLED uses the LUMA driver. See *Installing LUMA monochrome OLEDs* on page 119.

SH1106 128x32 pixel OLED monochrome OLED

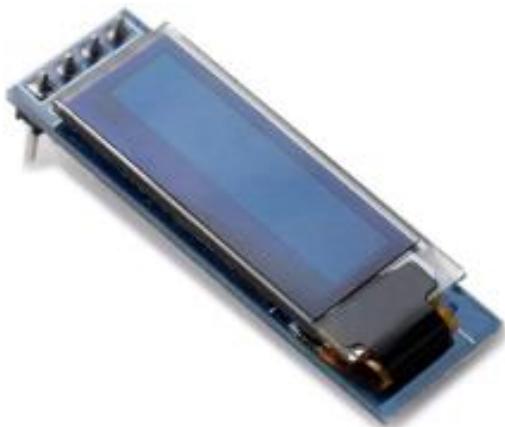


Figure 44 - 0.91-Inch 128x32 OLED

These 0.91 x 0.38-inch 128x32 pixel monochrome OLED display modules connect via the I2C Serial Interface. They are marketed under various names such as WayinTop or AzDelivery. This is the smallest display supported by this project.

Because of its size, the text can be hard to read. Also, one peculiarity is that the PIL driver software recognises the display as being 128x64 pixels and not 128x32. It is most useful where there is little space available for a display.



Note: OLED devices are very slow when compared to character LCDs or Graphics (touch) screens. Their main advantage is their size.

Grove LCD RGB Backlight



Figure 45 Grove LCD RGB Backlight

The **Grove JHD1313 LCD RGB** backlight is a 2x16 LCD with an RGB backlight.

The Grove LCD RGB uses two controllers.
AIP31068L – I2C controller for the LCD
PCA9632DP1 – I2C RGB backlight driver

The interface is the 2-wire I2C interface plus +5V power supply. The backlight can be set to any colour and is configurable in the radio software.

Pimoroni Products

Pimoroni are a UK based company who produce electronic products for both Raspberry Pi and Arduino. They make a range of all-in-one audio boards for Raspberry Pi, with high-quality digital audio.

Their Website is at <https://shop.pimoroni.com/>

Pimoroni Pirate Radio



Figure 46 Pimoroni Pirate Radio - Rear view

The illustration on the left shows the rear of Pimoroni Pirate radio. The amplifier consists of dual I2S DAC/amplifiers for stereo audio (MAX98357A) at 3 Watts per channel.

The Pirate radio comes as a kit (Soldering skills required). The Pimoroni software is disabled and the software from this project used instead. See *Installing Pimoroni Pirate Radio (pHAT BEAT)* on page 113. Note: It does not have a screen.

Pimoroni Pirate Audio



Figure 47 Pimoroni Pirate Audio

No soldering skills are required to construct this project when using a Pimoroni Pirate Audio range of products and a Raspberry Pi Zero with a pre-soldered 40-pin header. This unit requires version 6.14 or later of the radio software.

It comes with a 240x240 pixel colour 1.3-inch IPS (In-plane switching) display which gives a very good viewing angle. The display is driven by an ST7789 controller.

There are four variants of the Pimoroni Audio but they all use the same DAC and display software:

1. Pirate Audio Speaker - MAX98357A DAC with mini 1W / 8Ω speaker
2. Pirate Audio Line-out - PCM5100A DAC chip with 3.5mm output stereo jack
3. Pirate 3W Stereo Amp - MAX98357A DAC with mini 3W amplifier output
4. Pirate 3W Headphone Amp - PCM5100A DAC driving a PAM8908 headphone amplifier



Note: The software for Pimoroni Pirate Audio is very basic at the moment in particular the menu selection function. It is hoped to improve this at a later date.

Touch-screens

Raspberry Pi 7-inch touch screen

As an alternative to building a radio using limited LCD screens it is possible to build a radio using the Raspberry Pi 7-inch or 3.5-inch touch screen or any other HDMI screen (touch-screen or otherwise). If the screen does not have touch capability, then it is possible to use it with a mouse or keyboard or both. Also, the touch screen can be used in conjunction with rotary encoders or push buttons.



Figure 48 Raspberry Pi 3 with 7-inch touch screen

There is a very good setup guide for the Raspberry Pi touch-screen at:

<https://www.modmypi.com/blog/raspberry-pi-7-touch-screen-assembly-guide>

Adafruit 2.8 and 3.5-inch TFT touch-screens

The radio software supports the Adafruit 2.8 and 3.5-inch TFT touch screen (480x320 and 720x480 pixels respectively). The small size can make the controls difficult to use but it will still work. It is best to use a touch-screen stylus.



Figure 49 Adafruit 3.5-inch TFT touchscreen



Note: This software has only been tested with the Raspberry Pi 7-inch touch screen and the Adafruit 3.5-inch TFT touch-screen . Smaller than 7-inch screens may prove difficult to operate. The following resolutions are supported: 800x480, 720x320, 480x320 or 1024x600 pixels.



Please also note that touch screen functionality has nothing to do with the radio software. Touch screens emulate mouse functions such as click, drag and hover using standard mouse routines. Should you use another touch-screen other than the one recommended and this does not work then you need to solve this first (or use a mouse/keyboard). Regrettably the author cannot provide any support on how to configure other touch screens.

If the screen is displaying upside-down then edit the **/boot/config.txt** configuration file and add the following line.

```
lcd_rotate=2
```

Note: you must disable the *DRM VC4 V3D driver* as shown on page 84 for this to work. Reboot the Raspberry Pi for the change to take effect.

Waveshare 2.8 and 3.5-inch TFT touch-screens

Waveshare provide a 2.8 and 3.5-inch TFT touch screen (480x320 and 720x480 pixels respectively). These have 26-pin DIL connector which plugs in directly to the GPIO header.



Figure 50 Waveshare 2.8-inch TFT touch screen

The above screen shows a 2.8-inch Waveshare touch screen (Courtesy - Recep A. Güleç). Note the revised layout due to its size when compared to the 3.5 and 7-inch touch screens.

Connection via ribbon cable

Smaller touch screens usually plug directly into the Raspberry Pi GPIO header, however that may not be convenient especially if you want to build it all into a case. Fortunately, it is possible to purchase a 40-way DIL male to female ribbon cable which allows the display to be panel mounted.

The following illustration shows TFT screen on the connectors side. If it is required the display can be put on the GPIO side but will be displayed upside down. Luckily it is easy to flip the display upside down as shown in the installation instructions.



Figure 51 TFT connected by a 40-pin male/female ribbon cable

MHS 3.5-inch RPi Display

The MHS 3.5-inch RPi Display appears to be a badged version of the same hardware as the Waveshare 3.5" TFT. The installation software is also very similar.



Figure 52 MHS-3.5-inch RPi Display (Courtesy Brent Fraser)

More information the MHS 3.5-inch RPi Display will be found at:
http://www.lcdwiki.com/MHS-3.5inch_RPi_Display

Radio variants

Before starting you need to make a choice which type of radio you are going to build. There are several combinations of user interface and display type which can be constructed as shown in the following tables.

Table 1 Display Type options

Display Type	
1	Two-line 8-character LCD
2	Two-line 16-character LCD
3	Four-line 16-character LCD
4	Four-line 20-character LCD
5	Adafruit 2x16 RGB Plate (I2C)
6	Raspberry Pi 7-inch touch screen
7	Olimex 128 by 64-pixel OLED
8	Adafruit 3.5-inch TFT touch screen
9	No display (Vintage radio design)
10	Pirate Radio (No display)
11	Waveshare 1.8/2.3" touchscreen
12	Grove I2C 2-line 16-char. LCD
13	OLED displays supported by LUMA (SH1106, SSD1306 etc.)

Table 2 User interface options

User interface	
1	Five or six push buttons
2	Two rotary encoders with push buttons
3	Adafruit RGB plate with push buttons
4	Raspberry Pi 7-inch touch-screen
5	Adafruit 2.8" or 3.5" TFT
6	Mouse (HDMI/Touchscreen only)
7	Keyboard (HDMI/Touchscreen only)
8	IQaudIO Cosmic controller
9	Pirate Radio – 6 push-buttons
10	IR remote control – all versions
11	Waveshare 1.8/2.3" touchscreen

Any type of HD44780U LCD display can be used with any user interface. The HD44780U can either be connected directly to the GPIO pins or via a so-called I2C (also known as IIC) backpack.

The Adafruit RGB plate has a two-line 16-character display and comes with five inbuilt pushbuttons. It also has its own I2C interface using the MCP23017 chip so it does not require a separate I2C backpack.

The **PiFace CAD** comes with a two-line 16-character display and comes with six inbuilt pushbuttons. It uses the SPI interface from Motorola.

The touch screens can be used with or without rotary encoders or push buttons. The touch screen variants can also use a mouse and keyboard.

It is a simple choice of which display (two or four lines, 8,16 or 20-characters LCDs or a touch screen or HDMI screen, OLED display or Pirate radio) and whether to use rotary encoders or push-button switches as the user interface. 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 keyboard interface, whilst supported on the touch-screen versions, is a very limited option.

There is a configuration program called **configure_radio.sh** which configures the choice of display and user interface required. It can be safely re-run at any time.

The vintage radio software (Display option 9) 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 than two-line versions.



Note: The touch screen software (**gradio.py** or **vgradio.py**) cannot be run at the same time as the LCD version of the radio software (**radiod.py**). It is a case of using one or the other. It is however it is possible to switch between **gradio.py** or **vgradio.py** programs during operation.

Connecting the LCD display

There are two ways of wiring up the display:

- Directly connect the LCD to the GPIO pins. This uses six GPIO pins.
- Connection via an I2C backpack. This uses the two-pin I2C interface

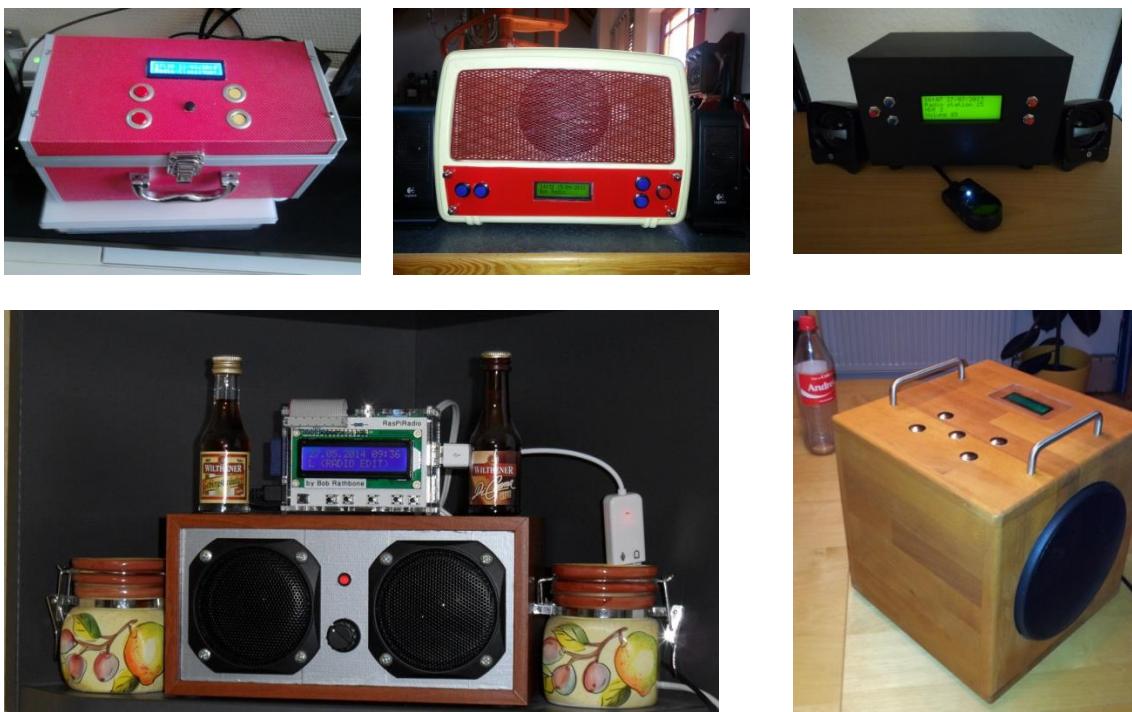
The first choice uses more wiring but is the cheapest option. The second choice uses an I2C backpack which is an extra component to be purchased. However, I2C backpacks are reasonably cheap.

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 53 Some examples of radio cases



Take a look at the constructor's gallery at
https://bobrathbone.com/raspberrypi/pi_internet_radio.html 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 Wi-Fi 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

Building in a IR sensor and remote control



Note: Unfortunately, the IR software will only run on Buster under Python 2 and not Bullseye due to errors with the Python 3 version of LIRC.

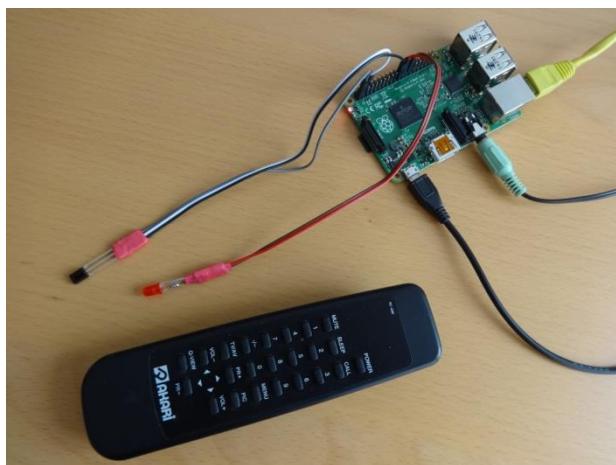


Figure 54 IR Sensor and Remote control

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

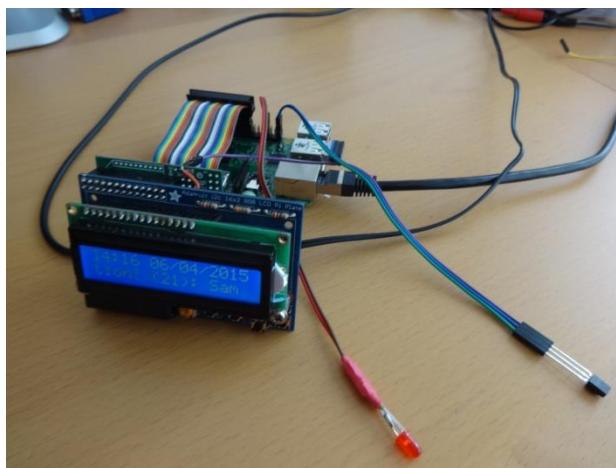


Figure 55 Adafruit and IR sensor and activity LED

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 72 on page 40.



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. If not planning to fit an IR sensor and activity LED then the 26-pin version Raspberry Pi may be used.

Chapter 3 – Wiring and Hardware

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** or **JustBoom** and newer **HiFiBerry** DACs require 40-pin versions of the Raspberry Pi. The 40-pin version of the wiring is recommended for all new projects.



If using DAC products do not use the 26-pin version of the wiring but use the wiring shown in Table 4 on page 28. The table below is now redundant except for very early versions of the radio.

Table 3 Controls and LCD wiring 26 pin version

Pin	Description	Radio Function	Name	LCD pin	Push Buttons	Encoder (Tuner)	Encoder (Volume)
1	3V3	+3.3V 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			MUTE		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 B	
12	GPIO 18**	Channel Down			DOWN	Output A	
13	GPIO 27	LCD Data 4		11			
14	GND	Zero Volts					
15	GPIO 22	LCD Data 5		12			
16	GPIO 23	LCD Data 6		13			
17	3V3	+3.3V supply					
18	GPIO 24	LCD Data 7		14			
19	GPIO 10**	Channel Down	SPI-MOSI		DOWN	Output A	
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-CEO	6			
25	GND	Zero Volts					
26	GPIO 7	LCD RS	SPI-CE1	4			

Colour Legend Radio I2C (shared) SPI Interface

* These pins are used for the I2C LCD backpack if used instead of the directly wired LCD to GPIO pins.

** Pin 18 is used by some DACs so pin 10 should be used for the down switch.

Table 4 Radio and DAC devices 40-pin wiring

Pin	Description	Radio Function	Name	Audio DAC Function	LCD Pin	Push Button	Encoder (Tuner)	Encoder (Volume)
1	3V3	+3.3V supply	+3.3V	+3.3V		+3.3V		
2	5V	5V for LCD	+5V	+5V	2,15			
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	1,3*,5,16		Common	Common
7	GPIO 4	Mute volume				MUTE		Knob Switch
8	GPIO 14	Volume down	UART TX			LEFT		Output A
9	GND	Zero Volts		0V				
10	GPIO 15	Volume up	UART RX			RIGHT		Output B
11	GPIO 17	Menu switch				MENU	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		DOWN	Output A	
17	3V3	+3.3V supply		0V				
18	GPIO 24	Channel up		Rotary Enc B		UP	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-CEO		6			
25	GND	Zero Volts		0V				
26	GPIO 7	LCD RS	SPI-CE1		4			
27	DNC			PiDac+ Eprom				
28	DNC			PiDac+ Eprom				
29	GPIO5	LCD Data 4			11			
30	GND	Zero Volts						
31	GPIO6	LCD Data 5			12			
32	GPIO12	LCD Data 6			13			
33	GPIO 13	LCD Data 7			14			
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 wiring tables. Use column 6 (Push Buttons) for the push button version and the last two columns (Encoder Tuner/Volume) for the rotary encoder version.

Version 1 boards (early boards)

Version 1 boards only had 26 pins and did not have internal pull-up resistors on the GPIO inputs. It has become increasingly difficult to support version 1.0 boards and you are advised to purchase a newer Raspberry Pi board for this project. Tips for using version 1 boards will be retained in this manual; however, if there is a problem, regrettably no support can be provided. As shown in the diagram below, 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 GND(0V) pin via a 10KΩ resistor. See Figure 56 below.

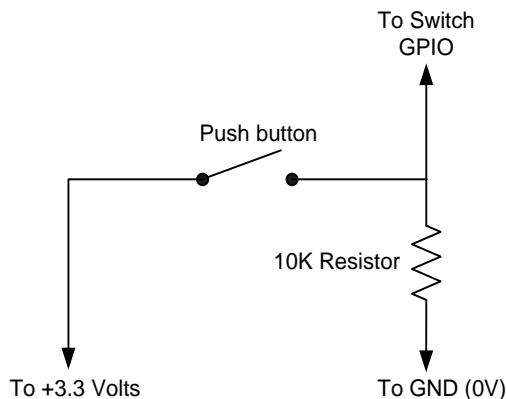
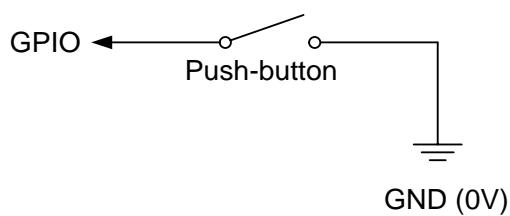


Figure 56 Push-button Wiring version 1 boards

Version 2, 3 or model B+ boards

Wire one side of the push-buttons the GPIO pin as shown in the last column of Table 4 on page 28. Wire the other side of the switches to either +3.3V (Old wiring scheme) or to GND (0V) (Preferred wiring scheme recommended for new projects). Whichever wiring you use; the radio configuration program will ask which wiring scheme is being used. Version 2 onwards boards have internal pull up/down resistors and don't require external resistors. In fact, including these can cause problems.

Preferred wiring for push-buttons



Original wiring scheme for push-buttons

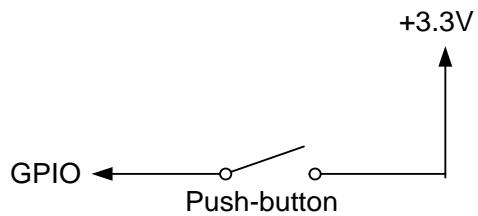


Figure 57 Push-button wiring version 2 onwards boards

The scheme chosen must be configured using the **pull_up_down** parameter in `/etc/radiod.conf` to 'up' or 'down'. See the section called *Configuring button interface with pull up resistors* on page 178.

Rotary encoder wiring

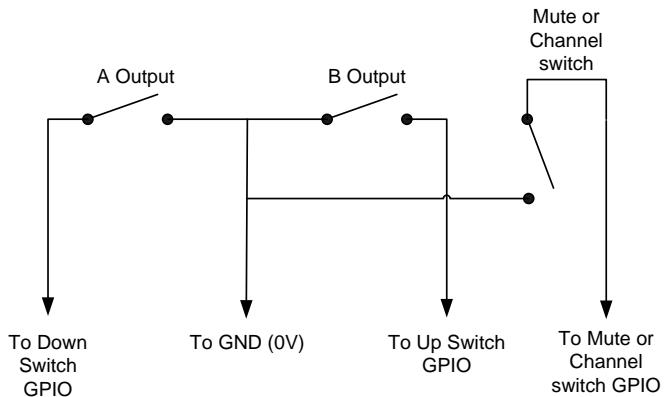


Figure 58 Rotary Encoder Diagram

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 will probably work.



Warning: The push switches (if fitted) on the rotary encoder are wired differently from the push buttons in the earlier 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 59 Rotary encoder with push switch

This project uses a COM-09117 12-step rotary encoder or PEC11R series encoders. 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 can be 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. Note that 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.

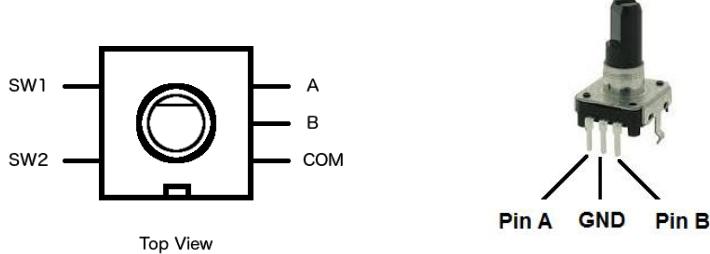


Figure 60 Rotary encoder pin-outs



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.

Using KY040 Rotary encoders

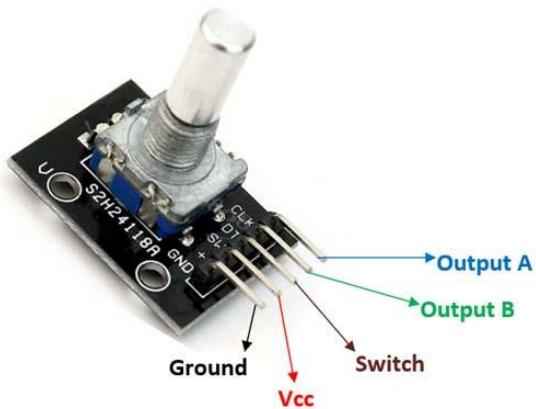


Figure 61 KY-040 Rotary Encoder

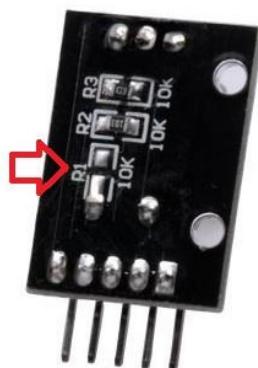


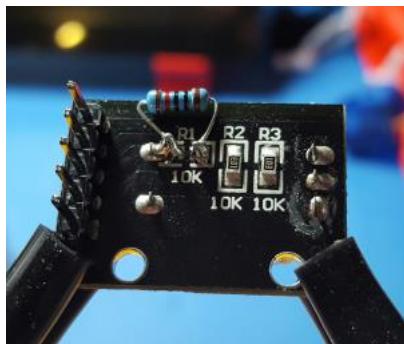
Figure 62 KY040 Missing 10K resistor

These cost-effective Rotary Encoders from Handson Technology originally designed for use with Arduino are now being used more and more by constructors. The KY-040 Rotary Encoder specification shows that these are powered by +5V to the VCC pin.

However, the Raspberry Pi uses a +3.3V supply and cannot tolerate +5V on the GPIO's so the advice is to connect VCC to +3.3V. These encoders work fine with VCC as +3.3V with this project.

More recently, manufacturers have been supplying KY040 encoders with the R1 10K pull-up resistor for the push-button missing. It is uncertain why suppliers aren't including this resistor. For the push button feature to work with these units, the missing 10K ohm resistor needs to be added or configure the pull-up resistors in with the Radio Configuration program described later.

Try to order KY040 encoders with all three resistors fitted.



If you have the soldering skills, you can add a 10K resistor across the R1 pads. If you don't have the ability to use a soldering iron then don't worry as the pull-up resistors can be configured later with software.



Colours: Brown, black orange.

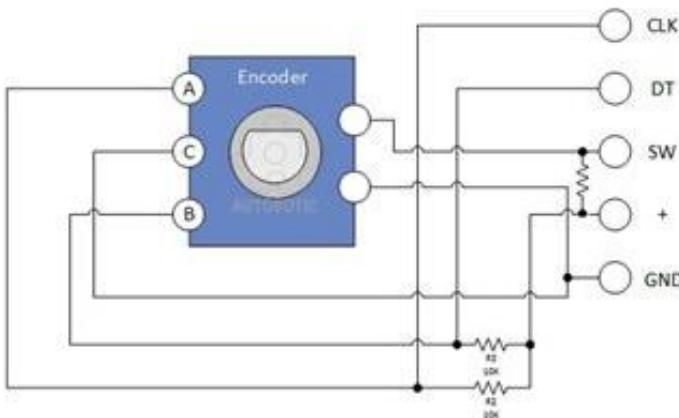


Figure 63 KY-040 Circuit Diagram

From version 7.2 onwards the internal pull-up resistors can be disabled with the radio configuration program as they are not required for the KY040 encoder as it has its own pull-up resistors. See Figure 157 *Rotary encoder* on page 101.



Note: The terms VCC, VEE etc. are explained in *Appendix E – Explanation of Vcc, Vdd, Vss and Vee* on page 368.

The specification shows the rotary encoders are labelled CLK(Clock), DT(Data) and + (VCC) however it is more usual to label these A, B and C.

Connect + to the +3.3V supply.
Do **not** connect to +5V.

Note: Some KY-040 alternatives do not have the resistor between SW and GND.

Rotary encoders with RGB LEDs



Figure 64 Sparkfun RGB rotary encoder

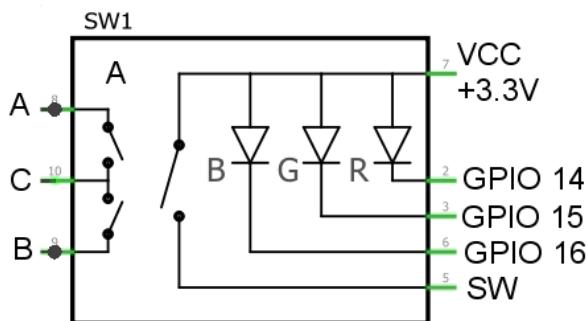


Figure 65 RGB Rotary Encoder circuit



Figure 66 Transparent knob

The LEDs can be driven by configuring the following parameters in `/etc/radiod.conf` as shown below:

```
rgb_red=14  
rgb_green=15  
rgb_blue=16
```

The LEDs are driven by the `status_led_class.py` driver software. The LEDs connect directly to the above GPIO pins 14, 15 and 16 and are driven by 3.3 volts so do not particularly need resistors but 100 Ohm resistors can be used to reduce the brightness.



Warning: Do not, under any circumstances connect these rotary encoders to +5 Volts, even if the specification says so, otherwise the Raspberry Pi will be irreparably damaged!

RGB I2C rotary encoders

These Rotary Encoders use three Red, Green and Blue (RGB) LEDs in the Rotary Encoder Shaft. This type of rotary encoder is entirely driven by the I2C interface. One drawback of these encoders is that they do not have a push button meaning that these must be installed extra to the encoders.

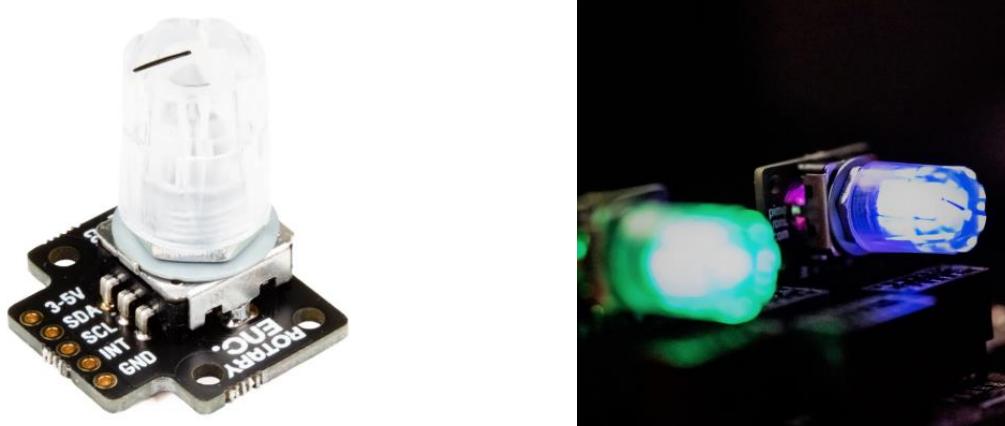


Figure 67 Pimoroni RGB I2C Rotary Encoders

These encoders contain Nuvoton MS51XB9AE microcontroller which let you directly control the RGB LEDs inside the encoder. The default I2C address is 0x0F but it is possible to configure a second device with a separate I2C address (0x1F for the radio project).

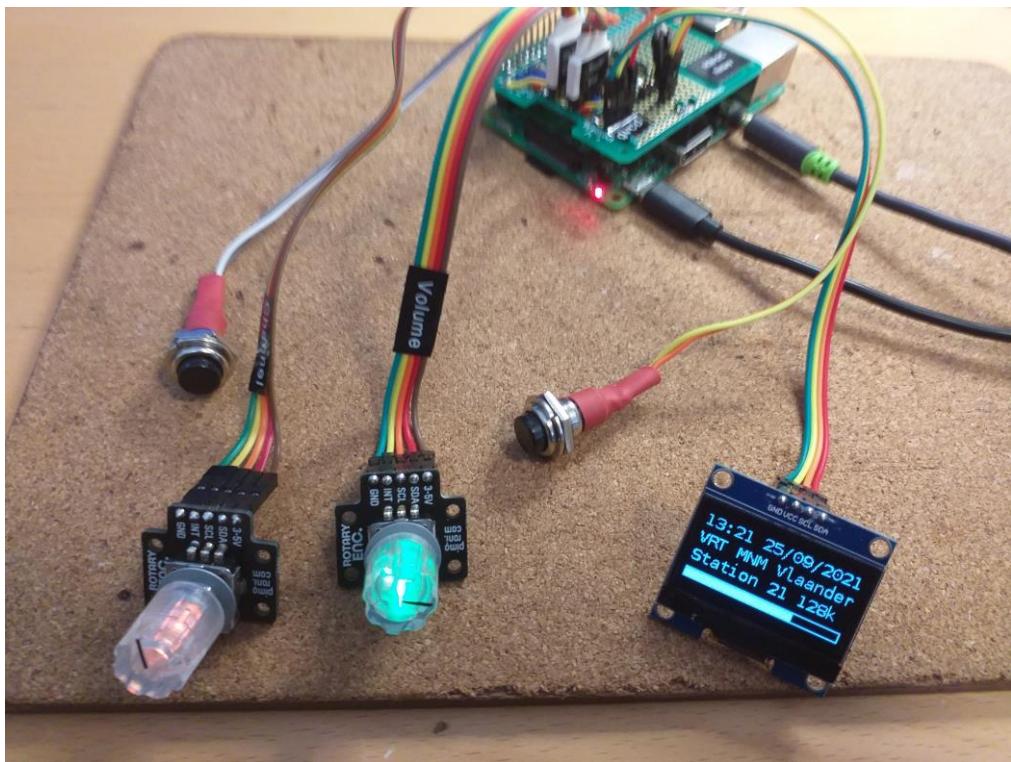


Figure 68 Radio project using RGB I2C rotary encoders

The above photo shows two RGB I2C encoders (Volume 0x0F and Channel 0x1F) running the radio software. The Mute and Menu buttons are wired the same as the standard encoders. Also shown is a 128x64 pixel OLED also connected by I2C (In this case with 0x3C as the address).

Rotary encoder wiring

The following tables show how to wire up the four types of rotary encoders used in this project. The first table shows how to wire up conventional rotary encoders (without pull-up resistors). These have five connections A, B, C and two connections to the switch. See *Figure 60* on page 31.

Table 5 Radio A, B, C Rotary Encoder Wiring

GPIO Pin	Description	Radio Function	Volume Rotary Encoder	Channel Rotary Encoder
6	GND	Zero volts	Common C	
8	GPIO 14	Volume up/down	Output A	
10	GPIO 15	Volume up/down	Output B	
7	GPIO 4	Mute volume	Knob Switch 1 *	
GND	GND 0V	Mute volume	Knob Switch 2 *	
9	GND	Zero volts		Common C
16	GPIO 23	Channel up/down		Output A
18	GPIO 24	Channel up/down		Output B
11	GPIO 17	Menu switch		Knob Switch 1 *
GND	GND 0V	Menu switch GND		Knob Switch 2 *

GND is found on physical pins 6, 9, 14, 20, 25, 30, 34 and 39.

* Note: In the case of RGB LED Rotary Encoders one side of the push switch is wired to the VCC (3.3V) supply internally in the switch. The GPIO goes from low to high when the button is pushed.

Table 6 RGB LED Rotary Encoder switch wiring

7	GPIO 4	Mute volume	Knob Switch 1	
+3.3V	VCC	“ “	Internal connection	
11	GPIO 17	Menu switch		Knob Switch 1
+3.3V	VCC	“ “		Internal connection

The second table shows how to wire up KY-040 rotary encoders (fitted with pull-up resistors). These have five connections namely CLK, DT, SW, VCC and GND. See *Figure 63* on page 32.

Table 7 KY-040 Rotary Encoder Wiring

GPIO Pin	Description	Radio Function	Volume Rotary Encoder	Channel Rotary Encoder
1	3V3	+3.3V supply	+3.3V (VCC)	
6	GND	Zero volts	Common (GND)	
7	GPIO 4	Mute volume	Knob Switch (SW)	
8	GPIO 14 (TX)	Volume up/down	Output B (DT)	
10	GPIO 15 (RX)	Volume up/down	Output A (CLK)	
17	3V3	+3.3V supply		+3.3V (VCC 3.3V)
9	GND	Zero volts		Common C (GND)
11	GPIO 17	Menu switch		Knob Switch (SW)
16	GPIO 23	Channel up/down		Output B (DT)
18	GPIO 24	Channel up/down		Output A (CLK)

Table 8 RGB I2C Rotary Encoders wiring

GPIO Pin	Description	Radio Function	Volume RGB I2C Rotary Encoder	Channel RGB I2C Rotary Encoder	Optional I2C display
1	3V3	+3.3V supply	VCC +3.3V	VCC +3.3V	VCC +3.3V
3	GPIO2	SDA	SDA	SDA	SDA
5	GPIO3	SCL	SCL	SCL	SCL
NC	n/a	n/a	INT (Interrupt)	INT (Interrupt)	
6	GND	Zero volts	GND	GND	GND
		I2C address	Hex 0x0F	Hex 0x1F	Hex 0x3C*
7	GPIO4	Mute	Mute Button		
11	GPIO17	Menu		Menu Button	

* Example OLED I2C address which will differ depending upon the attached device.

The above table shows two RGB I2C encoders. The I2C address for the Channel Rotary encoder is configured with the **gb_set_i2c_address.py** utility.

```
$ ./rgb_set_i2c_address.py
RGB I2C Rotary Encoder set new I2C address 0xf

Usage: sudo ./rgb_set_i2c_address.py --help
       --i2c_address=<i2c_address>
       --new_i2c_address=<new_i2c_address>

Recommended values for current and new addresses are 0x0F and 0x1F
Run "i2cdetect -y 1" to check I2C address before and after
```

First only plug in one RGB I2C Rotary encoder with hex address **0x0F**.

```
$ cd /usr/share/radiod
$ sudo ./rgb_set_i2c_address.py --i2c_address=0x0f -new_i2c_address=0x1F
```

Below is the output of the **i2cdetect -y 1** utility with two RGB I2C rotary encoders (**0x0F** and **0x1F**) and an OLED display (**0x3C**)

```
$ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- -- -- 0f
10:          -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- 1f
20:          -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:          -- -- -- -- -- -- -- -- -- -- -- -- 3c -- -- --
40:          -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:          -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:          -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:          -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

Optical Rotary encoders

Optical rotary encoders are much more expensive than the mechanical (conductive) rotary encoders previously shown. They offer the higher resolution compared to mechanical ones. They are usually used for scientific and industrial applications. They are overkill for this project but may be used. This software has been successfully tested with an HRPG-ASCA #16F optical encoder from Avago. See https://media.digikey.com/pdf/Data%20Sheets/Avago%20PDFs/HRPG_Series.pdf

LCD Module Wiring

The following shows the wiring for a directly wired HD44780U LCD controller. It has 16 or 18 pins. There are two ways of wiring the LCD data lines using either the 26 pin or 40-pin wiring schemes (Table 3 and Table 4 respectively). For all new 40-pin Raspberry Pi's the 40-pin wiring is strongly recommended. The 26-pin version of the wiring can be used on both 26 and 40-pin Raspberry Pi's.

Table 9 LCD module wiring for 26 and 40-pin Raspberry Pi's

LCD Pin	GPIO 26-pin	Pin 26 #	GPIO 40-pin	Pin 40 #	Description
1	n/a	6	n/a	6	Ground (0V) – Wire this directly to LCD pin 5
2	n/a	2	n/a	2	VCC +5V
3	n/a	Note1	n/a	Note1	Contrast adjustment (0V gives maximum contrast)
4	GPIO7	26	GPIO7	26	Register Select (RS). RS=0: Command, RS=1: Data
5	n/a	6 or 9	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	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	GPIO5	29	Data Bit 4 (D4)
12	GPIO22	15	GPIO6	31	Data Bit 5 (D5) Note if using IQaudIO products GPIO22 conflicts !!
13	GPIO23	16	GPIO12	32	Data Bit 6 (D6)
14	GPIO24	18	GPIO13	33	Data Bit 7 (D7)
15	n/a	2	n/a	2	LED Backlight Anode (+5V) or Red LED (Adafruit RGB plate) [2] or VEE negative voltage on Midas HD44780
16	n/a	6 or 9	n/a	6 or 9	LED Backlight Cathode (GND) or Red LED [2]
17					Optional Green LED (Adafruit RGB plate) [2]
18					Optional Blue LED (Adafruit RGB plate) [2]

If using **IQaudIO**, **HiFiBerry** or similar DAC products it is necessary to use the 40-pin version of the wiring (See Table 4).



Note 1: If using the Midas display with VEE on pin 15 do not connect this pin to the +5V supply or you will damage the display. Connect pin 15 as shown in the section called *Midas LCD display* on page 17.



Note 2: The contrast pin 3 (VE) should be connected to the center 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.



Note 3: 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 68.

The following diagram (Courtesy protostack.com) shows the electrical connections for the standard 16 pin LCD. Do not use this diagram for Midas displays. See instead *Midas LCD display* on page 17.

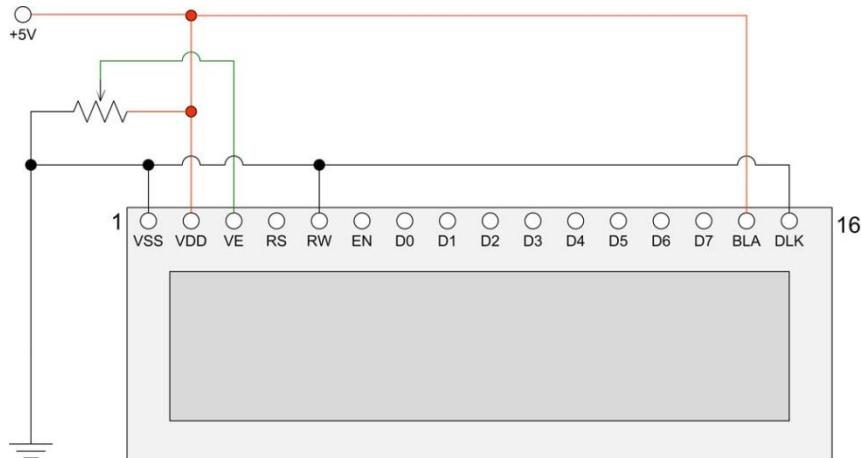


Figure 69 HD44780U LCD electrical circuit



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 (GND 0 Volts) to work. For more information see the section *Using the Adafruit backlit RGB LCD display* on page 68.

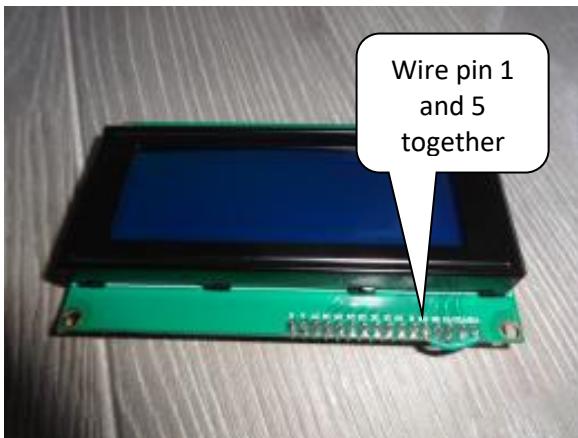


Figure 70 Wire LCD pin 1 (GND) and 5 (RW) together



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 which will probably cause irreparable damage to the Raspberry Pi. If using an I2C backpack this step is not necessary as it is already done in the backpack.



Warning – Some LCD displays such as the Midas with VEE 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. See section called *Midas LCD display* on page 17 for further information.



Note: All settings in the `/etc/radiod.conf` file use GPIO numbers and NOT physical pin numbers. So, in the above example `lcd_width` is GPIO 16 (Physical pin 36).

There is a useful program called `wiring.py` which will display physical the wiring required for the settings found in `/etc/radiod.conf`. See page 276.



Also, there is another program called `test_gpios.py` which will test for button or rotary encoder events. For example, you are not sure how you have wired a button or rotary encoder this program will confirm this for you. See page 278.

Power supply considerations

The Raspberry Pi except for the model 4B uses a standard Micro USB (type B) power connector, which runs at +5V. The model 4B uses a 5 Volt 3 Ampere power supply with a USB-C adaptor. 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. However, due to the extra current that can be drawn by connected USB or other devices, a minimum 2.5A supply will be needed.

Try to find a power adapter that delivers at least 1.5 Ampere. As mentioned above a 2.5A supply may be required if USB are used. See the following article. The newer RPi models can draw up to 1.5 Amps if USB peripherals or sound cards 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 or 4). Some prototyping boards used to provide power in this way.

If using an adaptor or separate 5-volt Power Supply try to use a switched-mode power supply adaptor. This takes less current and generate less heat than 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. 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 65



**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 338.**

Also see *Appendix D – Using a battery pack* on page 367

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.

GPIO Numbers

Raspberry Pi B
Rev 1 P1 GPIO Header

Pin No.		
3.3V	1	2
GPIO0	3	4
GPIO1	5	6
GPIO4	7	8
GND	9	10
GPIO17	11	12
GPIO21	13	14
GPIO22	15	16
3.3V	17	18
GPIO10	19	20
GPIO9	21	22
GPIO11	23	24
GND	25	26

Raspberry Pi A/B
Rev 2 P1 GPIO Header

Pin No.		
3.3V	1	2
GPIO2	3	4
GPIO3	5	6
GPIO4	7	8
GND	9	10
GPIO17	11	12
GPIO27	13	14
GPIO22	15	16
3.3V	17	18
GPIO10	19	20
GPIO9	21	22
GPIO11	23	24
GND	25	26

Raspberry Pi B+
B+ J8 GPIO Header

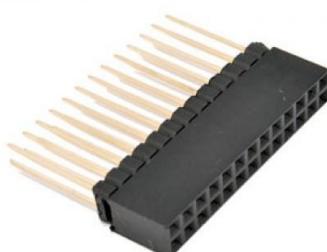
Pin No.		
3.3V	1	2
GPIO2	3	4
GPIO3	5	6
GPIO4	7	8
GND	9	10
GPIO17	11	12
GPIO27	13	14
GPIO22	15	16
3.3V	17	18
GPIO10	19	20
GPIO9	21	22
GPIO11	23	24
GND	25	26
DNC	27	28
GPIO5	29	30
GPIO6	31	32
GPIO13	33	34
GPIO19	35	36
GPIO26	37	38
GND	39	40

Key		
Power +	UART	
GND	SPI	
I ² C	GPIO	

Figure 71 GPIO Numbers



Note: The B+, 2B, 3B, 4B and P400 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 72 26-pin header extender

Parts List

The following table shows the parts list for a basic Raspberry PI Internet Radio. This list is for the version using the HD44780U 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 10 Parts list (LCD versions)

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	Radio Case	See Housing the radio page 25
1	Raspbian Bullseye 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 35 watt)	Any electronics shop
1	Transformer for amplifier	Any electronics shop
1	LCD HD44780U 2 x 16 Display *	Farnell Element 14
1	Prototype board	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 or 40-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	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	TSOP38238 IR Sensor	Adafruit industries and others
1	Red or Green LED and 220 Ohm resistor	Any electronics shop
1	Optional sound card (DAC)	IQaudIO, HifFiBerry or JustBoom
	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 backpack

The above list is not relevant if using a touch screen except when used with rotary encoders or push buttons.

Chapter 4 - Construction details

Construction using an interface board

It isn't necessary to construct the radio connect via an interface board but it does make maintenance easier and is much more reliable and will be needed if you wish to connect to the Raspberry Pi using a ribbon cable. Always bring the ribbon cable into the top of the board as shown in Figure 73 below. If the ribbon cable is connected to the back (underside) of the board the two rows of the 40-pin connector will be swapped over. In the next section how to use breakout boards is covered which may be an easier alternative for many constructors rather than building your own.



Figure 73 40-pin Interface board with ribbon cable

The figure below shows a 4x20 HD44780U LCD plugged into the interface board.



Figure 74 Interface board with LCD screen attached

Below is the other side of the interface board showing the connections to the rotary encoders, an IR sensor the IR activity LED.

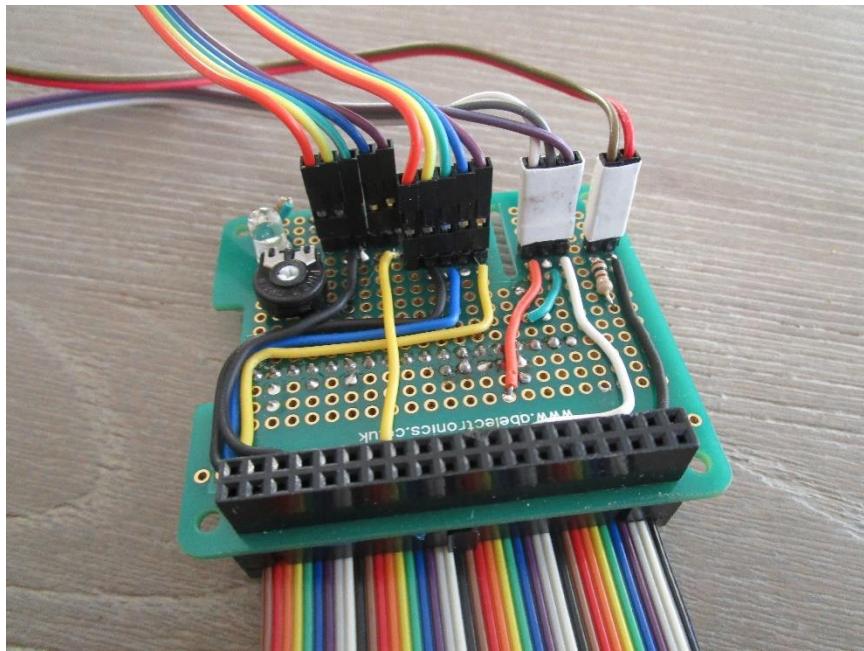


Figure 75 Radio controls connections

Below is the complete overview of the interface card with some test rotary encoders, an IR sensor the IR activity LED.

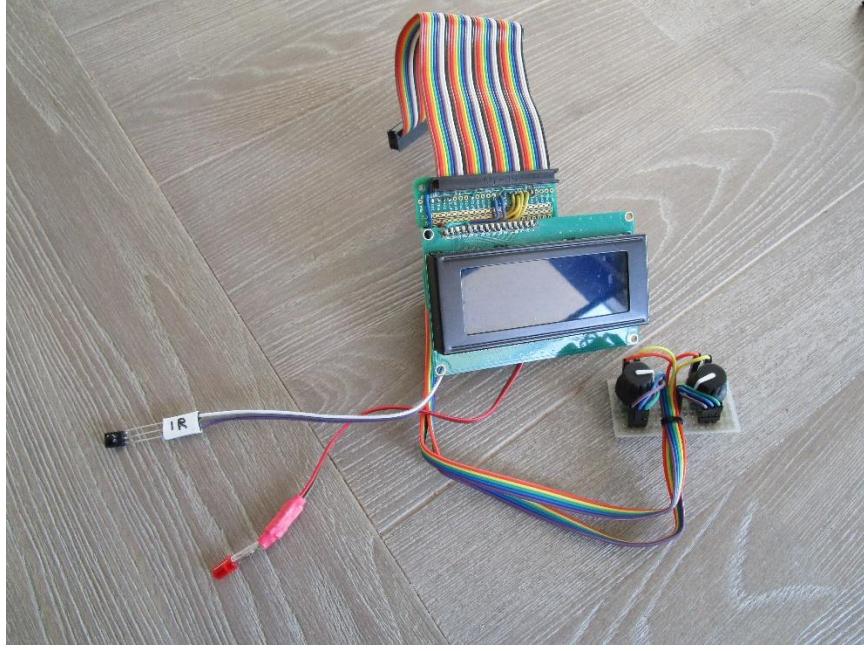


Figure 76 Interface board overview

There are various interface boards available on the market for both 26-pin and 40-pin Raspberry Pis.

Construction using breakout boards

When this project was begun in the early days of Raspberry Pi there was very little in the way of breakout boards. It was necessary to make connections to button, LCDs and Rotary encoders either direct on the GPIO header or via an especially constructed breakout board. Things became more complex when digital sound cards (DAC) were introduced as these occupied the GPIO header and either did not extend the header pins, or if they did so, only extended a few of them.

This has now changed and there is a wide variety of breakout boards available.

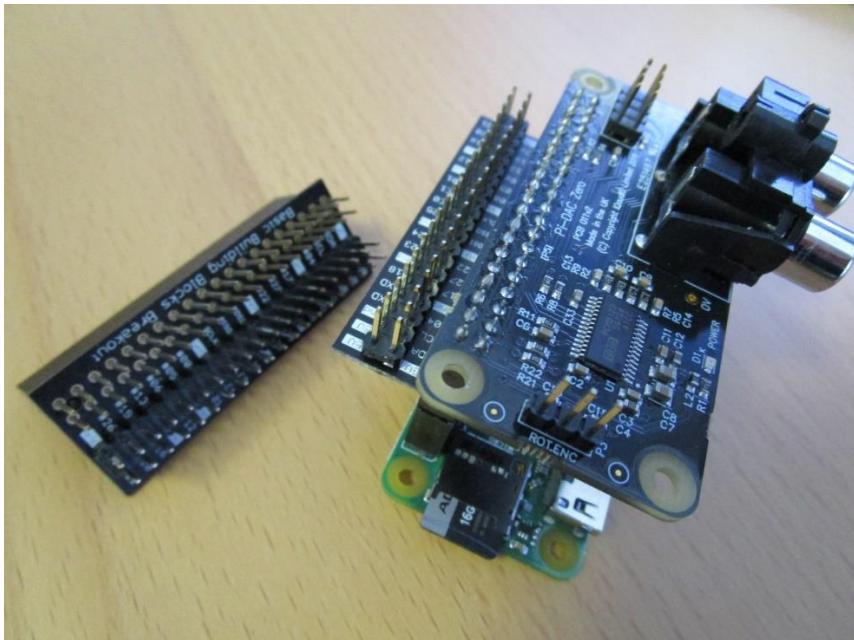


Figure 77 GPIO header breakout board

On the left of the above photo is an example of the 4Tronix GPIO breakout and extender board. On the right of the same photo is the break-out board used with a Raspberry Pi Zero W and an IQaudIO DAC plus. All 40-pins are now made available, except for those used by the DAC, to attach buttons and the like to the GPIO pins.

These boards are available from <http://4tronix.co.uk>

See:

<https://shop.4tronix.co.uk/products/gpio-interceptor-gpio-breakout-for-40-pin-raspberry-pi>



Note: Soldering skills are required to solder the 40-pin header to the breakout board. The above breakout board is only shown as an example and many more are available on the Internet.

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 78 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.

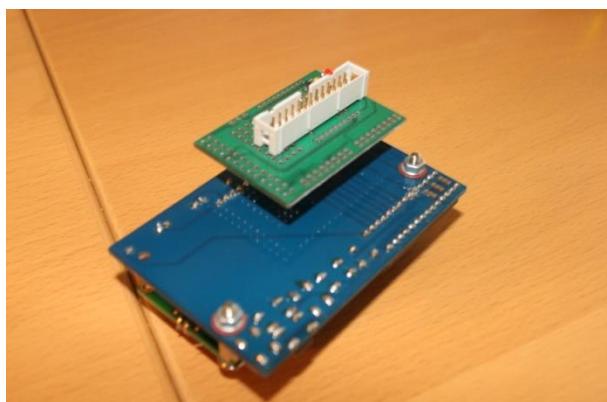


Figure 79 Adafruit LCD plate with ribbon cable adapter

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.

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 three 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 revisions of the Raspberry Pi. 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.

Using alternatives to the Adafruit display

Caution is advised. There are a number RGB boards which are compatible with the software provided by Adafruit Industries. Below is such an example of a Chinese 1602 I2C LCD. When originally tried the backlight wasn't lit and it was necessary to wire Pin 1 (GND 0V) and Pin 16 (Backlight). This is no longer necessary this version of the software as the backlight is software controlled.



Figure 80 Chinese 1602 I2C LCD

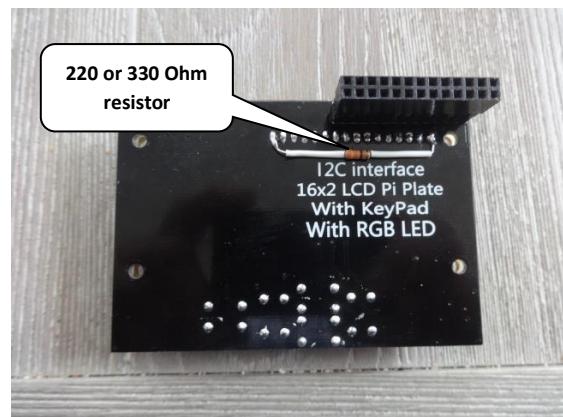


Figure 81 Enabling the backlight.

Note that the RGB part is a RGB LED that you see shining very brightly in the above picture. It is not actually lighting up the LCD backlight unlike the Adafruit RGB plate.

If running the radio version 5.9 or earlier then solder a 220 or 330 Ohm resistor between pin 1 (0V GND) and pin 16 (Backlight). Version 5.10 or above this is no longer necessary.

To switch off the very bright RGB light switch off the colour definitions in `/etc/radiod.conf`.
For example: `bg_color=OFF`

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 – Hex address 0x20
2. Arduino I2C backpack using a PCF8574 port expander – Hex address 0x27 or 0x37

The I2C 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.

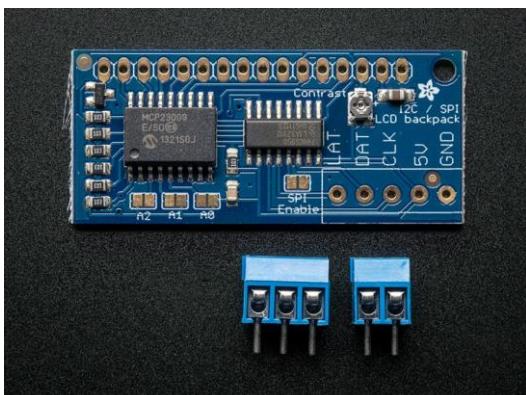
This is configurable in the `/etc/radiod.conf` file.

```
# 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



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.
The Adafruit backpack also supports the SPI interface but is not used in this project

Figure 82 Adafruit I2C Backpack

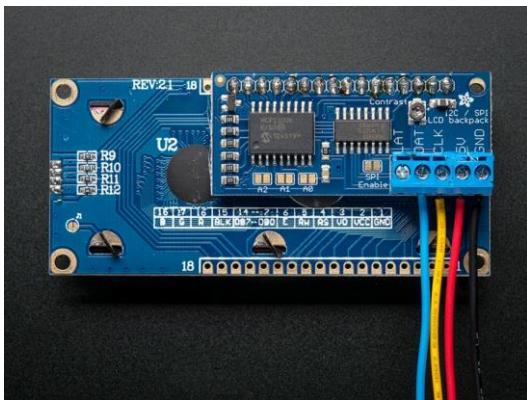


Figure 83 LCD connected to an Adafruit I2C backpack

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 2 (pin 3)
3. Yellow: I2C Clock – GPIO 3 (pin 5)
4. Red: +5 volts – GPIO header pin 2
5. Black: GND (0v) – GND GPIO header 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 hex 0x27. Another manufacture may use hex 0x37. This is configurable in the radio configuration program.



Figure 84 Arduino I2C backpack

The wiring From top to bottom is:

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

The blue potentiometer on the right is the contrast adjustment.

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

To use this device either amend the **i2c_backpack** parameter in **/etc/radiod.conf** (Comment out the ADAFRUIT line) or run the **configure_radio.sh** program.

Table 11 I2C Backpack connections

Backpack	Label	Description	GPIO	Physical pin
1	GND	Zero volts	-	14
2	VCC	+5 Volts	-	4
3	SDA	I2C Data	2	3
4	SCL	I2C Clock	3	5
5	LAT	Latch – Not used	-	-

The LAT pin is for the Adafruit backpack only and is for the SPI interface but isn't used by I2C.

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. Any number of Raspberry Pi prototyping boards are available for all versions of the Raspberry Pi. The Ciseco Humble Pi prototype board shown in Figure 85 has been discontinued.



Figure 85 Ciseco Humble PI I2C interface board

The above figure shows the I2C interface board using the Ciseco Humble PI (Discontinued). 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.



Figure 86 The I2C backpack interface board

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.

Fitting a wake-up button

One of the features of this radio's design is that the menu button (LCD versions) or a special key (Touchscreen version) can do an orderly system shutdown. This is more desirable, and certainly safer and more convenient than pulling the power plug out. The system when properly shutdown goes into a so-called halt state. If the power is left connected the Raspberry Pi, it can be woken up by a button connected between pins 5(GPIO3) and 6(GND).



Figure 87 Wake-up button

On the left is the Raspberry Pi unit which is running the radio on a TV with HDMI inputs as shown in Figure 3 on page 5. A small black wake-up button is fitted to the case and connects to physical pins 5 and 6. When pressed with the Raspberry Pi in a halt state but power still applied it will start its boot-up sequence. It should be noted that pin 5 (GPIO3) is also used as the I2C data line. Although the button could still be fitted it is probably not a good idea as it will disrupt the I2C signal if the wake-up button is pressed.

Installing an IR sensor and remote control



Note: Unfortunately, the IR software will only run on Buster under Python 2 and not Bullseye due to errors with the Python 3 version of LIRC.



Note: This installation procedure is only for LCD versions of the radio. If using a HDMI or Touchscreen display see *Installing the FLIRC USB remote control* on page 54.

IR Sensor

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 pin connectivity depends on the exact hardware being used. See *Table 18 IR Sensor Pin outs* on page 135 for the GPIO pin connection.

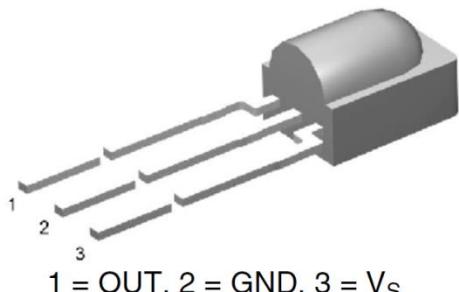


Figure 88 TSOP38238 IR sensor

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 on *Table 18 IR Sensor Pin outs* page 135.

** Caution; Do not accidentally connect 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:

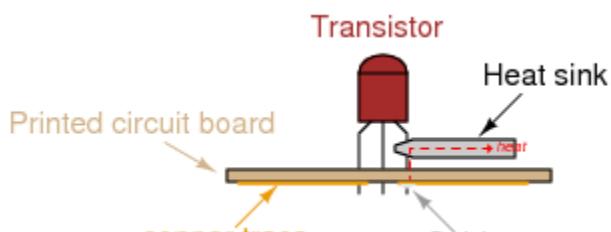


Figure 89 Soldering precautions

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 135.

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.

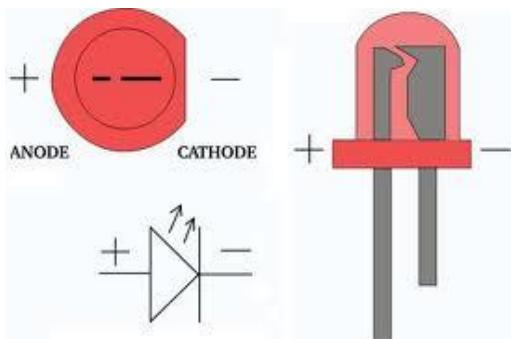


Figure 90 LED polarity

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 the GPIO pin shown in the following table via a 100 Ohm resistor.

The following table shows the GPIO pin used for the LED connections.

Table 12 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
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 etc. sound boards	36	16	40-pin version only
PiFace CAD with IR sensor	16	23	26 or 40-pin version

How to configure the LED is shown on in the section called *Configuring the remote control activity LED* on page 178.

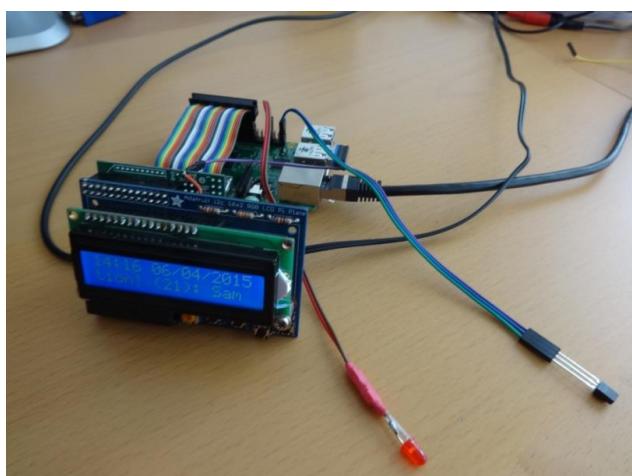


Figure 91 Adafruit plate with IR sensor and activity LED.

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 an IQaudIO Cosmic Controller

IQaudIO manufacture a comprehensive range of sound devices and controller boards. See their Web site at: <http://www.iqaudio.co.uk/>

The radio software provides support for the IQaudIO Cosmic controller.

The IQaudIO Cosmic controller consists of the following:

- A three push-button interface (Channel UP/DOWN and Menu)
- A rotary encoder (Used as volume control)
- Three status LEDs (Normal, Busy and Error)
- A 128 by 64 pixel OLED display (I2C interface)
- Optional IR detector (Ordered separately)

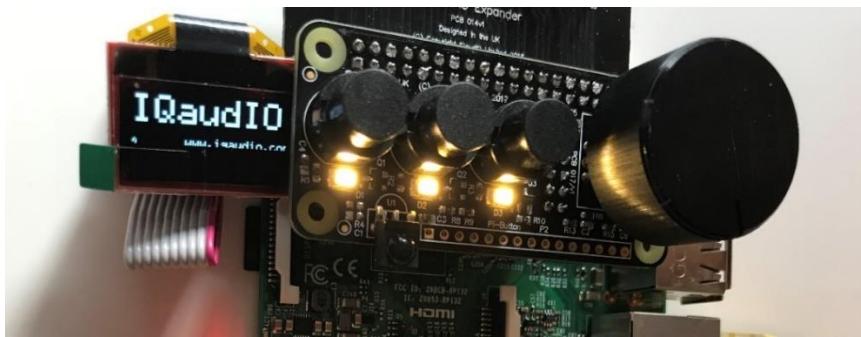


Figure 92 IQaudIO Cosmic controller and OLED display

The main advantage of this hardware, is that it contains everything required by the radio software.

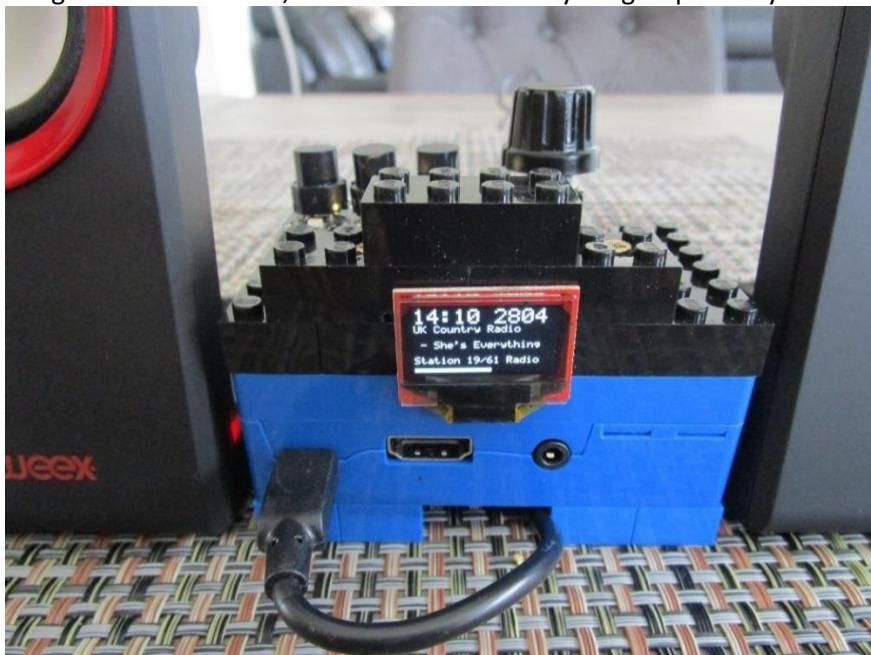


Figure 93 Lego radio with IQaudIO Cosmic controller and OLED

If using the IQaudIO Cosmic controller it is necessary to configure it as shown in the section *Configuring the IQaudIO Cosmic controller and OLED* on 180.

Construction using the Pimoroni Pirate radio

A full set of instruction for building the Pimoroni Pirate radio with pHat BEAT can be found here:
<https://learn.pimoroni.com/tutorial/sandyj/assembling-pirate-radio>

Soldering skills are required.

Construction using the PiFace CAD

Fortunately, no soldering or construction is required with the PiFace CAD. Just plug it in and install and run the software. The PiFace CAD also 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 as the SPI interface on the Raspberry Pi is fairly slow. It also has the disadvantage that the push buttons are on the bottom of the unit.



Figure 94 PiFace CAD and Raspberry PI



Figure 95 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 and the section called *Installing PiFace CAD software* on page 122 for further information on SPI.

Installing the FLIRC USB remote control



Note: This installation procedure is only for the HDMI or Touchscreen display of the radio. If using an LCD display see *Installing an IR sensor and remote control* on page 50.

FLIRC USB Remote Control dongle



The FLIRC USB dongle allows the use of any remote control with your Raspberry Pi. In this design it is intended for use with the graphical version of the radio (Touch-screen or HDMI displays). It allows button presses on a remote control to be mapped to the keyboard input of the Raspberry Pi. For example, pressing the volume up button on the remote control will act just like pressing the + key on keyboard (If so mapped). The graphical version of the radio accepts key presses. The LCD versions of the program don't so FLIRC will not work with the LCD versions. This may however change in a later version.

More can be found at the FLIRC Web site: <https://flirc.tv>

Installation documentation can be found at: <https://flirc.tv/ubuntu-software-installation-guide>



Note: This installation procedure is only for the HDMI or Touchscreen display of the radio. If using an LCD display see *Installing the Infra-Red sensor software* on page 135.

It is first necessary to install FLIRC. First install the necessary libraries.

```
$ sudo apt install libhidapi-hidraw0 libqt5xmlpatterns5
```

Install the FLIRC software by running the following:

```
$ curl apt.flirc.tv/install.sh | sudo bash
```

The following will be displayed:

```
% Total    % Received % Xferd  Average Speed   Time     Time     Time
Current                                         Dload  Upload   Total  Spent  Left
Speed
100  8266  100  8266    0      0  17185      0  --::--  --::--  --::--
17149
:
:
Distribution: debian
Checking for curl...
Detected curl...
Installing flirc deb-repo...
Running apt update... done.
Installing apt-transport-https... done.
Installing /etc/apt/sources.list.d/flirc_fury.list...done.
Running apt update... done.
```

Answer Y to the following question:

Do you want to install the flirc utilities? [Y/n]

Reboot the Raspberry pi in Desktop mode (Important).

On the desktop open **Programs → Accessories → Flirc**



Note: At this stage you may see a pop-up window offering a Firmware Upgrade.

Answer Yes to upgrade the firmware in the FLIRC dongle. Failure to do so may result in the Dongle not connecting.

Now see *Configuration of the FLIRC USB dongle* on page 188.

Using DAC Sound Cards

The sound output of the on-board audio jack on the Raspberry Pi is known to be limited. Using a DAC (Digital Audio Converter) will give much better quality and output. Several types are available.

The one you choose depends upon your requirements. These DAC cards use PCM (Pulse Code Modulation) technology and the Raspberry Pi i2s interface.

If you are going to use an external amplifier then almost any DAC will do. If you want a complete solution, a DAC card with an in-built amplifier (typically 3W up to 25W) is a good choice. The output from these in-built amplifiers is usually so-called class D-type amplification. Another consideration is the required connection to the amplifier copper, S/PDIF or optical (Toslink)?

If price is a big consideration there are a number of very reasonably priced DACs which emulate some of the better-known ones and use the same device driver software as the one, they are emulating. See *Table 30 Sound card Device Tree overlays* on page 358.

For more information on DACs see https://en.wikipedia.org/wiki/Digital-to-analog_converter

HiFiBerry DAC

This version supports the HiFiBerry DAC from HiFiBerry. See <https://www.hifiberry.com>. There is a comprehensive range of DACs available from this manufacturer. A few are shown below:

1. HiFiBerry DAC+ Light/Light – Entry level and standard solution
2. HiFiBerry Digi+ Light – Optical (TOSLink) and RCA connectors
3. HiFiBerry AMP+ –Standard DAC with a 25W D-Class amplifier
4. HiFiBerry DAC+ Zero Form Factor
5. HiFiBerry 3W Miniamp Zero Form Factor

The HifBerry DAC Plus

The HiFiBerry 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 96 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 97 below. The Radio controls and LCD screen for example are then connected on this header.



Figure 97 HiFiBerry mounted on the Raspberry 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** or by running the **configure_radio.py** program.



Note: All settings in the **/etc/radiod.conf** file use GPIO numbers and NOT physical pin numbers. So in the above example **down_switch** is GPIO 10 (Physical pin 19).



The Pimoroni pHat is compatible with HiFiBerry DAC (Not DAC+) and uses the same Device Tree (DT) overlay.

Qaudio DAC sound products

IQaudio DAC also have a comprehensive range of products. Again, these provide excellent results. These cards fit within the Pi's form factor and provide additional full access to the Pi's 40way I/O signals allowing easy addition of IR sensors, Rotary Encoder or i2c devices (such as OLED screens) etc. See <http://iqaudio.co.uk>

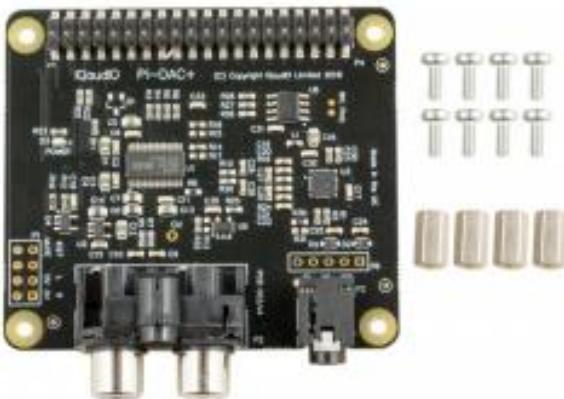


Figure 98 IQaudIO DAC plus

DAC for the Raspberry Pi. This latest revision of the IQaudIO Pi-DAC PRO and is pre-programmed for auto detection. Line out: 2x Phono/RCA Headphone: 3.5mm socket.

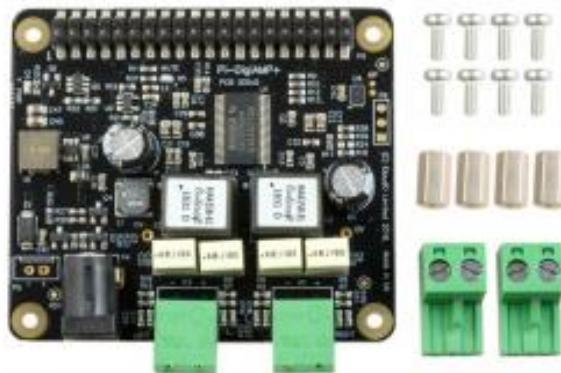


Figure 99 IQaudIO Pi-DigiAMP+

This provides the DAC+ along with a 35W amplifier which fits the Raspberry Pi A+/B+/RPi2/3/3B+. This card requires supports up to 24v power supply and delivers the full 2.5amp to the Pi.

JustBoom DAC products

The construction using **JustBoom** products is similar to other sound cards. 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 by running the audio configuration program.

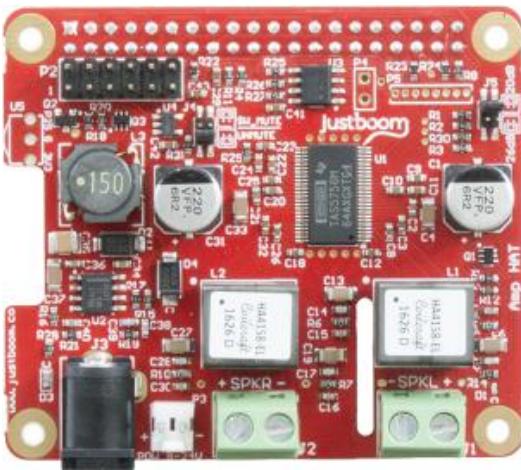


Figure 100 JustBoom Amp HAT

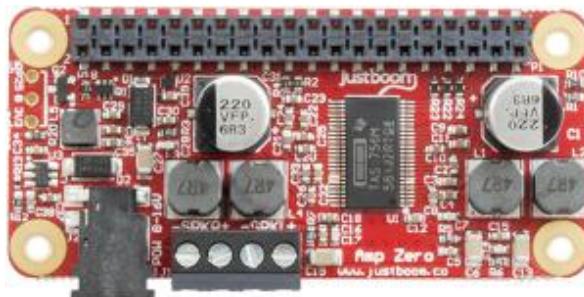


Figure 101 JustBoom Amp Zero pHAT

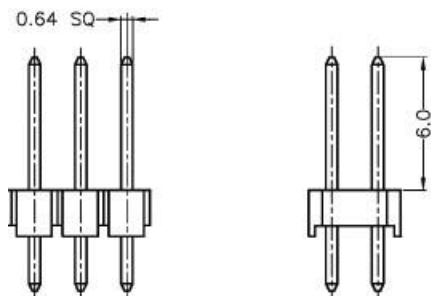


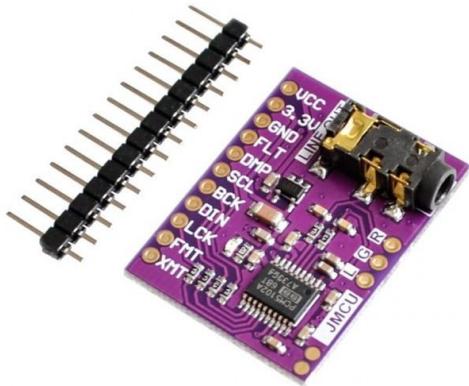
Figure 102 JustBoom Zero stacker requirements

The **JustBoom** Zero boards are used with stackers or installed directly on the Raspberry Pi Zero. Some stackers and some 2x20 male headers on the market though are too thin or too short to provide good contact with the board. Use stackers that the pins are squared and are at least 0.6mm in width. If you are soldering the 2x20 male header on the Raspberry Pi Zero make sure that the pins are 0.6mm in width and 6mm in usable height.



Figure 103 Using the 40-pin stacker

PCM5102A DAC Sound Card



DAC	Function
FLT	Filter Select. Normal latency (Low) / Low latency (High)
DMP	De-emphasis control for 44.1kHz sampling rate: Off (Low) / On (High)
FMT	Audio format selection: I2S (Low) / Left justified (High)

Also edit `/boot/config.txt` and enable the I2S dtoverlay.

```
dtparam=i2s=on
```

Plug a suitable stacker onto the Raspberry Pi Zero. Plug the JustBoom Zero board on top of the stacker so that the pins protrude through the JustBoom Zero board.

Plug the radio interface card or ribbon cable (not shown) on top of these protruding pins.

There are a number of inexpensive DACs which use the PCM5102A chip and have a 3.5mm Stereo Jack 24. These give a very reasonable 24-bit sound. Despite the fact that they claim to be a pHat they do not plug directly into the Raspberry Pi. Use either jumper wires or build your own interface board. These boards use the HiFiBerry DAC device driver and the I2S dtoverlay. Wiring as below:

DAC	RPi	Pin
VCC	5V	2
3.3V	N.C.	-
GND	GND	6
FLT	N.C.	-
DMP	N.C.	-
SCL	N.C.	-
BCK	GPIO18	12
DIN	GPIO21	40
LCK	GPIO19	35
FMT	N.C.	-
XMT (1)	3.3V	1

N.C. Not Connected. (1) Connect via 10K resistor.

Pimoroni audio DACs

Pimoroni pHat DAC

The Pimoroni pHAT DAC provides an affordable high-quality DAC for the Raspberry Pi. The 3.5mm stereo jack comes soldered onto the board already. Though designed to match the format of the Raspberry Pi Zero it is compatible with all 40-pin GPIO Raspberry Pi variants.

Features:

- 24-bit audio at 192KHz
- Line out stereo jack
- pHAT format board
- Uses the PCM5102A DAC to work with the Raspberry Pi I2S interface

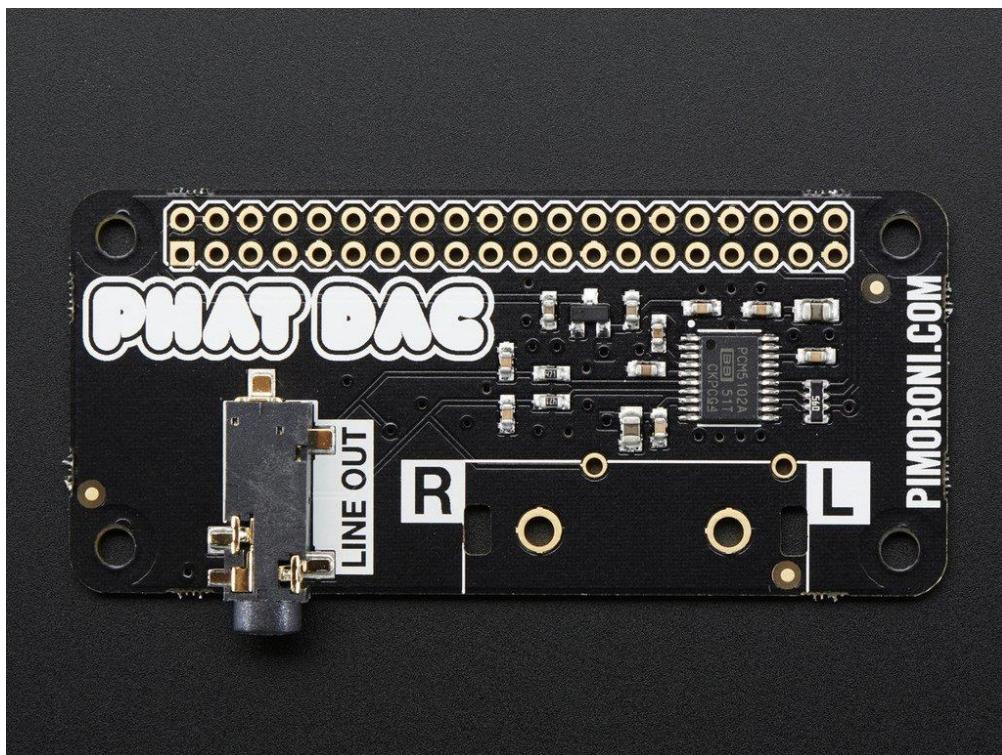


Figure 104 Pimoroni pHat DAC



Note 1: Do not use the 40 female header that comes with the board but use a 40-pin extender so that other cards can be used on top of it.



Note 2: The Pimoroni pHat is not completely compatible with the HiBerry DAC although it uses the same software driver. In particular to use the Alsa sound mixer a package called **pulseaudio** is required. However, **pulseaudio** is not compatible with several of the features of this package such as the **espeak** speech package. Normally the **pulseaudio** package must be removed as shown in the section called *Installing pulseaudio* on page 96. The Pimoroni pHat DAC will run fine without any mixer controls.

Pimoroni Audio DAC Shim for Pi Zero

Pimoroni now supply a DAC which uses PCM5100A DAC chip. It has a friction fit header which slips pushes down over the Pi Zero 40-pin GPIO header and doesn't require any soldering and is removable.

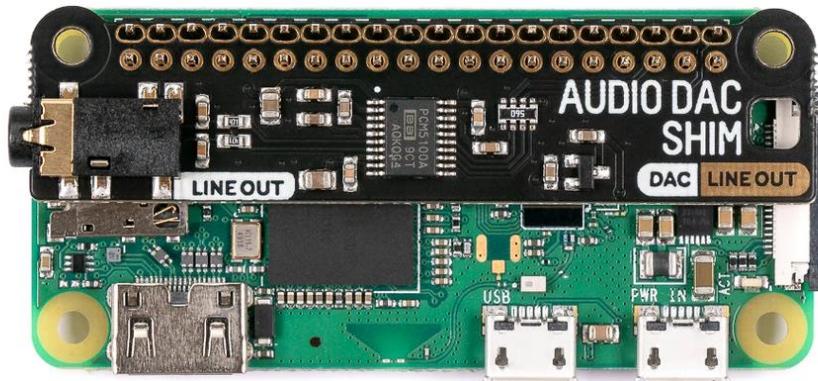


Figure 105 Pimoroni Audio DAC Shim for Pi Zero/W

The above figure shows a Pimoroni Audio DAC shim fitted to a Pi Zero W. The PCM5100A DAC chip takes high quality digital audio from the Pi Zero and pipes out crisp, line-level 24-bit / 192KHz stereo audio through the 3.5mm jack.

Because the Audio DAC SHIM adds no extra bulk to the Raspberry Pi Zero it allows the GPIO header to be accessed as normal, for example other HATs can be fitted on top of it. It should also fit easily inside any standard case. It's a simple way to add an audio output to the Pi Zero. Pimoroni even claim that it can also be fitted to a Raspberry Pi 400.

It uses the same driver as HiFiBerry DAC (Not DAC Plus or Digi). This can be configured during installation of the radio software.

Alternatively, it can be manually configured using by adding the following lines to the [All] section of **/boot/config.txt** file:

```
dtoverlay=hifiberry-dac  
gpio=25=op,dh
```

If you're using a Pi that has an audio jack you might also need to disable onboard audio by either adding a # to the beginning of the following line or setting **on** to **off**.

```
#dtparam=audio=on
```

Adafruit speaker bonnet



Figure 106 Adafruit Speaker Bonnet

The Adafruit speaker bonnet is primarily designed for the Raspberry Pi Zero format but can be used on any Raspberry Pi.

It consists of a stereo 3W amplifier connected to the I2S interface of the Raspberry Pi via a 40-pin DIL female header.

It can be purchased with two miniature speakers or it may be used with any small 4 or 8-Ohm speakers.

Allo DAC products

Allo is a manufacturer of very high-quality audiophile sound products. A good example is the **Allo Piano 2.1 HiFi DAC (EU)** with **woofer output** as shown below.

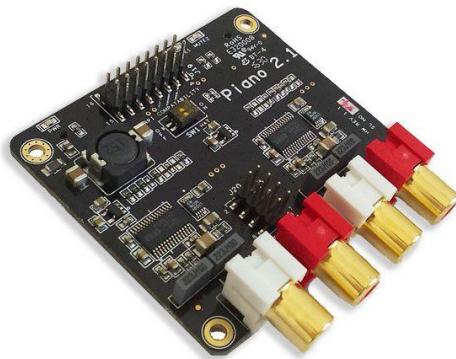


Figure 107 Allo Piano 2.1 HiFi DAC with woofer

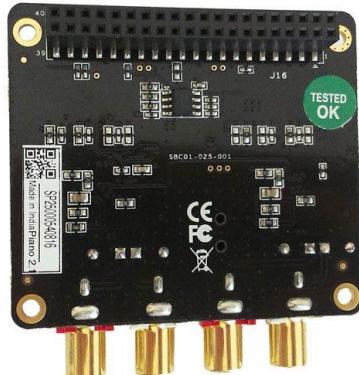


Figure 108 Allo Piano 2.1 HiFi DAC (Rear)

The **Allo Piano** has two outputs, one for normal sound ranges and one for a sound woofer. The board comes pre-programmed as 2.1 for the Raspberry Pi.

The subwoofer has 2 outputs (Stereo left and Right) but is mono only. However, it uses a second I2S channel on GPIO5 and mute signal on GPIO6 both used to control the woofer amplifier. This conflicts with the LCD signals `lcd_data4` and `lcd_data5` pins.



Allo products currently need to be specially configured to correct this conflict. See *Configuring Allo Sound Cards* on page 127.

Further information about Allo products can be found at <https://www.allo.com/sparky/index-dac.html>

Construction Tips and Tricks

This section contains some construction tips which may be useful. It goes without saying that having the correct tools such as a good fine tipped soldering iron, wire strippers and the like will greatly help constructing the radio.

Wiring up rotary encoders and switches



Figure 109 Rotary encoder wiring components

Purchase prototype board jumper wires with at least one end fitted with female connectors. The best type is the ribbon type particularly in the case of rotary encoders. Strip off five wires for the rotary encoder or two for push buttons. Also, it is better to use shrink wrap to cover the wires after they have been soldered onto the switch or rotary encoder. This improves both insulation and strength.



Figure 110 Using wire strippers

Cut the plugs off the end that is to be soldered to the rotary encoder or switch. Leave the other end with *female* connectors. Using good wire strippers, strip a few millimetres off the wires. Separate the wires for about 30 millimetres.

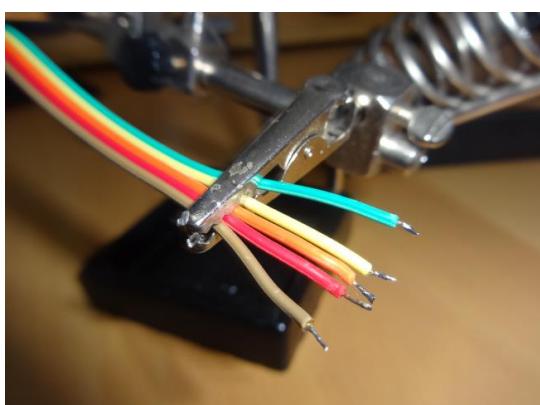


Figure 111 Tinning the wires with solder

Twist the copper strands together as tightly as possible and tin the wires with a little solder. A so called “Extra pair of hands” is very useful for gripping the wires using crocodile clips.

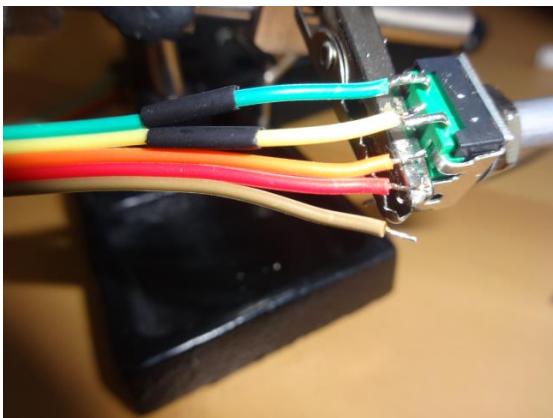


Figure 112 Soldering up the switch



Figure 113 Shrink shrink-wrap with a hair dryer

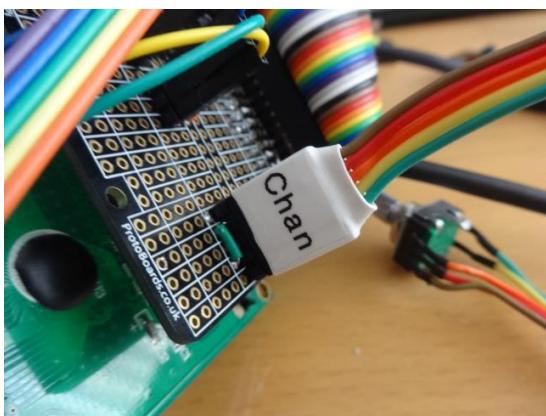


Figure 114 Connecting the rotary encoder an interface board

Tin the switch connections with solder. Cut a few millimetres of shrink wrap and slide onto the wires to be soldered. Make sure that the shrink wrap sleeves are well away from the heat of soldering iron as these will shrink easily with the slightest bit of heat. Tack the wire onto the top of the switch connector. Don't attempt to twist the wire around the connector. Just tack it on top with a little bit of heat from the soldering iron.

After soldering on all wires push the shrink wrap over the newly soldered wires. Using a hair dryer heat the shrink wrap until it has completely shrunk tightly over the wires.

The finished switch can either be connected directly to the Raspberry Pi or to an interface board.

If using an interface board, the female connectors can be bound together to form a plug using a larger piece of shrink-wrap. Slip the shrink wrap over the female pins and heat with a hair dryer. This can then be labelled up using Dymo tape machine or a CD marker pen. Push the newly created plug onto the male pins on the interface board.

Selecting an audio amplifier

There is a wide range of amplifiers that can be used with the radio which fall into five main categories:

1. A set of PC speakers with amplifier (Logitech or similar) – the simplest option
2. A dedicated AB or Class D stereo amplifier (Velleman or similar)
3. A combined DAC (I2S) and amplifier from manufacturers such as **IQaudIO** or **HiFiBerry**
4. The audio stage of an existing (vintage) radio. Use the PA or record player input (Usually mono only)
5. Bluetooth speakers or headset



Figure 115 PC speakers

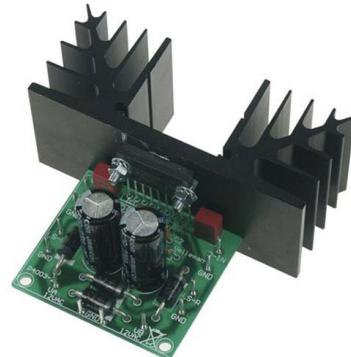


Figure 116 Velleman 30W stereo amplifier



Figure 117 IQaudIO DAC and 20W Amplifier



Figure 118 Vintage radio PA input



Figure 119 JVC Bluetooth Speakers



Figure 120 Audio output jack amplifier



The above manufacturer's products are examples only and do not imply any specific recommendations. There are an enormous number of solutions available. The choice is determined by, amongst other things, price, mono/stereo, power output and quality.

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 121 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 122 Loop +5V supply around the core

Fit a mains filter



Figure 123 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 124 Integrated mains socket and filter

Use an I2C LCD backpack

If all else fails replace the directly wired LCD wiring with an I2C backpack. See the section called *Construction using an I2C LCD backpack* on page 47 for further information.

Preventing ground loops



Figure 125 3.5mm Jack Ground Loop Isolator

Avoid creating ground loops in the first place during construction. Ground loop issues usually cause a humming or electronic 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 Raspberry Pi or sound card and the amplifier input.

For further information on ground loops:

[https://en.wikipedia.org/wiki/Ground_loop_\(electricity\)](https://en.wikipedia.org/wiki/Ground_loop_(electricity))

Connecting up a USB power adapter



Figure 126 Connecting up a USB power adapter

It is convenient to connect all power supply components for the Raspberry Pi, amplifier and USB hub etc via a single mains switch. USB 240V AC to +5V power adapters are designed to connect directly to the mains and not via a mains switch. One idea is to purchase a European round pin adapter and use standard electrical connector blocks to connect the incoming AC power cable to the two pins of the power adapter. AC cables to other components such as the amplifier can also be connected to the connector blocks. Use electrical tape or shrink-wrap to isolate the connector blocks.

Cooling the Raspberry Pi

The Raspberry Pi is built from commercial chips which are qualified to different temperature ranges; the LAN9512 is specified by the manufacturers being qualified from 0°C to 70°C, while the Application Processor (AP) is qualified from -40°C to 85°C. Operation outside these temperatures is not guaranteed. The temperature of the CPU should not really go above 80°C. If it does the CPU will throttle back its processor clock speed to reduce the temperature. The official line from the Raspberry Pi is that does not require cooling even though you can get temperature warnings when using the Raspberry Pi 7-inch touch screen.

The **vcgencmd** command can be used to check the CPU temperature.

```
$ /opt/vc/bin/vcgencmd measure_temp  
temp=67.1'C
```

For most of the radio designs in this document no specific cooling is required. If you need to cool the Raspberry Pi (Those with touchscreens in particular) then a variety of heat sinks and cooling fans are available as shown in Figure 127 below. Even with a heat sink good ventilation is necessary.



Figure 127 Heat sink kit

A variety of heat sink and cooling fan kits are available for the Raspberry Pi. Fans are a less good idea as they will produce a background hum. Also, as the CPU gets hot and the RPi is mounted vertically the adhesive softens and the fan and its heat sink tend to fall off unless secured in some way. Try to use a heat sink first.



Figure 128 Cooling fans

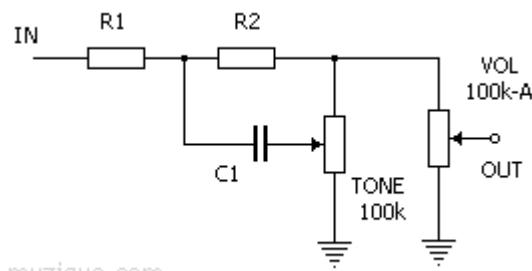
Miscellaneous

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>



muzique.com

Figure 129 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 $R1=10k$, $R2=47k$, $C1=0.022\mu F$ and $100k$ for the tone and volume pots.



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.

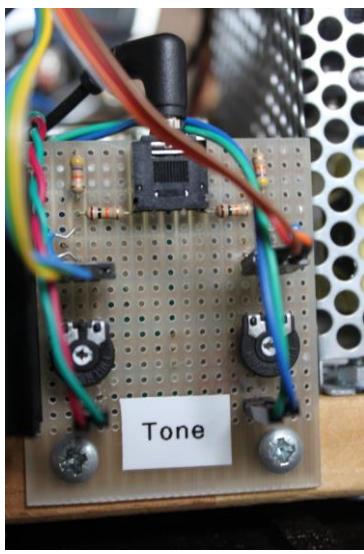


Figure 130 Tone control board

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 the 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 13 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



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.

Figure 131 IN4148 diode

- 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)

Chapter 5 – System Software Installation

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
```

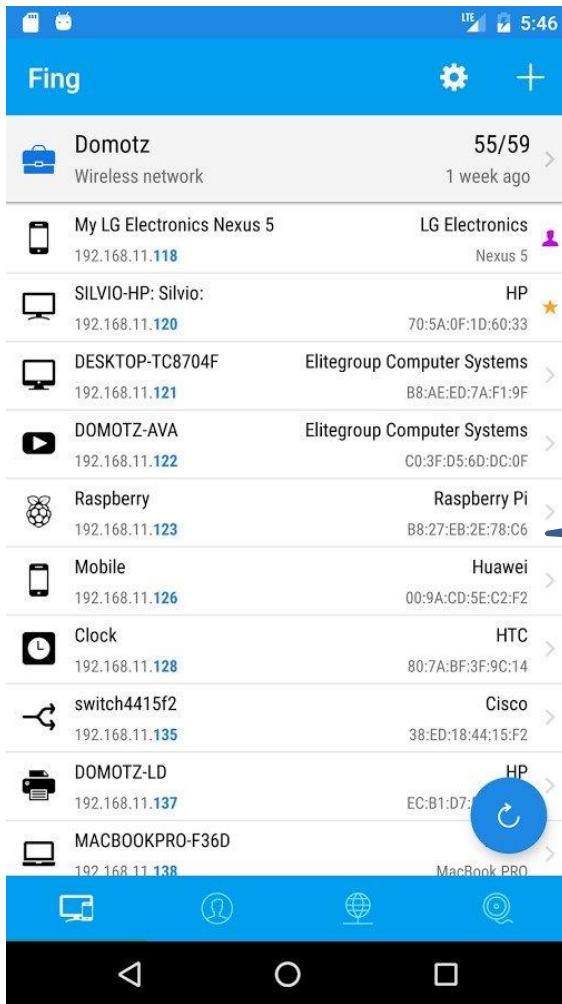
Some commands produce output which does not need to be shown. In such a case a ‘:’ is used to indicate that some output has been omitted.

```
$ 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
```

END OF EXAMPLE COMMANDS.

Useful installation tools

Finding the Raspberry Pi on a network using Fing



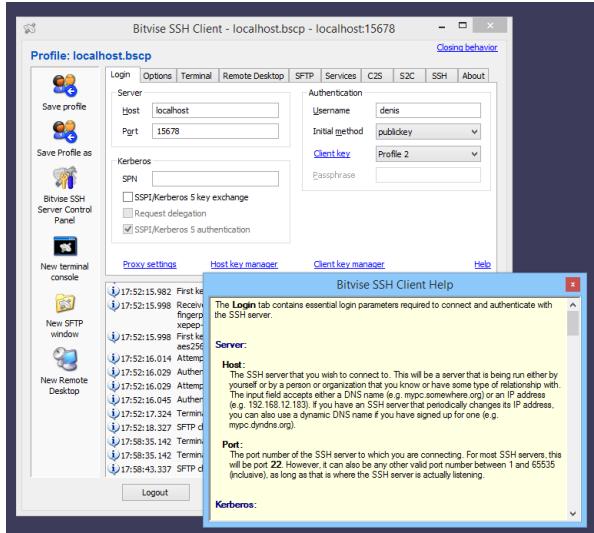
The **Fing** App is a free network toolkit and scanner for iOS (Apple) and Android. It will discover all the devices on your network, identify intruders and can also run Internet speed tests.

In this context it is very useful for finding out the IP address of any Raspberry Pi connected to the network. It will only see it of course once the Raspberry Pi has been set up first on the Wireless Network or hard-wired in. It scans both Wireless and hard-wired Ethernet devices.

Once detected it is the possible to the SSH to the Raspberry Pi using **Bitvise** or Putty.

See <https://www.fing.com/products/fing-app> for further information.

Bitvise and Putty



Bitvise is a free SSH client available for Windows or Mac to connect SSH server enabled Unix or Linux operating systems. It is a graphical based SSH client just like Putty with more features. It supports the File Transfer using SFTP Secure File Transfer Protocol). Once installed you can easily make a terminal connection to the Raspberry Pi using the IP address discovered by Fing above as well as easily transferring files.

See <https://www.bitvise.com/ssh-client> for more information.

Entering system commands

If you are new to Linux there are a couple of things that may cause confusion.

1. Entering program names on the command line
2. File path names
3. File permissions

Take the following examples:

```
$ raspi-config
```

The following command fails

```
$ configure_radio.sh  
$ -bash: configure_radio.sh: command not found
```

The following two commands both work.

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

```
$ /usr/share/radio/configure_radio.sh
```

The third command has a **./** in front of it, the first one doesn't. Why?

The reason is that the **raspi-config** program is in the **/usr/bin** directory which is in the **PATH** environment directive. This can be seen with the following command.

```
$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin://usr/bin:/sbin:/bin:/usr/local/games  
:/usr/games
```

The second program is located in the **/usr/share/radio** directory which is not in the PATH directive. The **./** in front of the command means that the program or script will be found the current directory.

So, this means for programs not in directories specified in the **PATH** environment directive, you must either specify the full path name to the command or change to its directory and then enter the command with a **./** in front of it.

In the system prompt for user pi you will see a **~** character. The **~** character means the home directory for the current user, this case pi. So **~** is the same as **/home/pi**.

```
pi@raspberry3:~ $
```

For information on file permissions see the following link:

https://wiki.archlinux.org/index.php/File_permissions_and_attributes

Editing configuration files

At various points during the installation procedures in this manual you will be asked to edit certain configuration files such as **/etc/radiod.conf** (The radio configuration file) or **/boot/config.txt** (The boot configuration file). There are various text editors that can be used but the main ones in the case of the Raspberry Pi are:

1. Nano - **nano** is a small, free and friendly editor particularly suited for use by beginners.
2. Vi – **vi** is usually the professional user's choice of editor. It is very powerful but a lot harder to use for someone unfamiliar with it.

Usage:

```
nano <filename>
```

or

```
vi <filename>
```

Where *<filename>* is the name of the file to be edited.

See <https://www.raspberrypi.org/documentation/linux/usage/text-editors.md> for an overview of available text editors.

It is important to know that most configuration files are owned by root so you may be able to read them but not write them. For example:

```
$ nano /etc/radiod.conf
```

This will allow you to read the file but not change it. To give user **pi** temporary root user permissions so that you can save changes to the file use the **sudo** command in front of the editor command:

```
$ sudo nano /etc/radiod.conf
```

Or

```
$ sudo vi /etc/radiod.conf
```



Note: Make sure that you edit the file **/etc/radiod.conf** and not **/usr/share/radio/radiod.conf** which is the distribution file. The latter is copied to the **/etc** directory during installation and configuration.

Using the Vi editor

This is too big a subject to cover here. Type “Using vi” into a search engine such as Google or Bing to display various tutorials on how to use **vi**.

Using Nano

When **nano** is started it will display the contents of the file being edited. For example, **/etc/radiod.conf**. In this example the following screen will be displayed.

```

64.tlp - pi@192.168.2.21:22 - Bitvise xterm - pi@pitest: ~/develop/pi/radio6
GNU nano 2.7.4                               File: /etc/radiod.conf

# Raspberry Pi Internet Radio Configuration File (40 Pin version)
# $Id: radiod.conf,v 1.15 2017/11/13 12:27:27 bob Exp $

# Configuration file for version 6.0 onwards
# 40 pin version to support IQ Audio and other sound cards

[RADIOOD]

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

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

# Set date format, US format = %H:%M %m/%d/%Y
#dateformat=%H:%M:%S %d/%m/%Y
dateformat=%H:%M:%S %A %e %B %Y

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

# Volume display text or blocks
volume_display=blocks

# 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

^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos   ^Y Prev Page
^X Exit      ^R Read File   ^\ Replace    ^U Uncut Text ^T To Spell   ^L Go To Line ^V Next Page

```

Figure 132 The nano file editor

Hold down the Ctrl key and press the letter G on the keyboard to display the help text. The following screen will be displayed:

```

64.tlp - pi@192.168.2.21:22 - Bitvise xterm - pi@pitest: ~/develop/pi/radio6
GNU nano 2.7.4                               File: /etc/radiod.conf

Main nano help text

The nano editor is designed to emulate the functionality and ease-of-use
of the UW Pico text editor. There are four main sections of the editor.
The top line shows the program version, the current filename being
edited, and whether or not the file has been modified. Next is the main
editor window showing the file being edited. The status line is the
third line from the bottom and shows important messages. The bottom two
lines show the most commonly used shortcuts in the editor.

Shortcuts are written as follows: Control-key sequences are notated with
a '^' and can be entered either by using the Ctrl key or pressing the Esc
key twice. Meta-key sequences are notated with 'M-' and can be entered
using either the Alt, Cmd, or Esc key, depending on your keyboard setup.
Also, pressing Esc twice and then typing a three-digit decimal number
from 000 to 255 will enter the character with the corresponding value.
The following keystrokes are available in the main editor window.
Alternative keys are shown in parentheses:

^G (F1) Display this help text
^X (F2) Close the current file buffer / Exit from nano
^O (F3) Write the current file to disk
^R (F5) Insert another file into the current one

^W (F6) Search for a string or a regular expression
^R (M-R) Replace a string or a regular expression
^K (F9) Cut the current line and store it in the cutbuffer
^U (F10) Uncut from the cutbuffer into the current line

^J (F4) Justify the current paragraph

^X Exit          ^P Prev Line        ^Y Prev Page        M-\ First Line
^L Refresh       ^N Next Line        ^V Next Page        M-/ Last Line

```

Figure 133 The nano editor help screen

The ^ character means the Control-key (Ctrl). So, for example ^O above is Ctrl + O. For more information on **nano** see <https://www.nano-editor.org/dist/v2.0/nano.html>

System Software installation

Raspberry Pi OS (previously called **Raspbian**) is the Foundation's official supported operating system. The latest version of **Raspberry Pi OS** is called **Bullseye**. Create a new SD card with **Bullseye** or **Bullseye Lite**. There is also a "Full" version of **Bullseye** however this is unnecessary for this project.

A lot of very useful Raspberry Pi documentation will be found at:

<https://www.raspberrypi.org/documentation>



Note: The touch-screen or HDMI TV version of the software requires a desktop version of the operating system so use **Raspberry Pi Bullseye** and not the **Lite** version. Only use **Lite** for LCD versions of the radio.



Warning: Version 7.4 onwards is designed to work on primarily on **Raspberry Pi Bullseye** but can run with restrictions on **Buster**. **Bullseye** uses Music Player Daemon (MPD) version 0.22.6.

SD card creation using Raspberry Pi Imager

Use at least a 16 Gigabyte Card for **Bullseye Lite** or 32 Gigabyte for **Bullseye Desktop/Full**. Create an SD card running the latest **32-bit** version of **Raspberry Pi Bullseye** or **Bullseye Lite**.



Note: This version now works on either **32-bit** or **64-bit** architecture of the Raspberry Pi Operating System. Please note that the **64-bit** version is in the Beta testing stage. The software has exactly the same functionality for both architectures.

There are a couple of ways of doing this but in this tutorial, we are using the **Raspberry Pi Imager** software to create the SD card. You will need a Windows PC or Laptop with a SD Card Reader as shown in Figure 134 below.



Figure 134 Windows Laptop with SD card reader



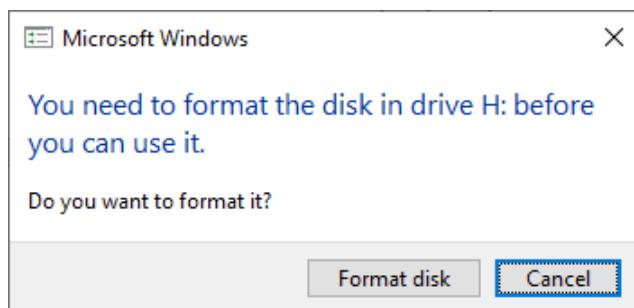
If your PC does not have an SD Card Reader then you can use a USB SD Card reader as shown on the left. The Raspberry Pi Imager software can then be used to write the Raspberry Pi OS to SD Card.

Figure 135 USB SD Card reader

An Apple Mac PC can also be used but will require that you download the Mac OS version of the software.

First insert the SD card into the SD card reader on your PC.

When you insert an SD Card that already has an OS other than Windows you may see the following:



Ignore this message and close the above dialogue box. The above may occur a number of times during this process.



Note: It is possible to boot from a USB 3.0 disk drive or stick instead of from an SD card. This is described in the section called *Booting from a USB drive* on page 318 (Chapter Advanced topics). However, always start with an SD card first.



Note: If installing on a very old Raspberry Pi 1B with a 26-pin header this only runs properly on **Buster**. If using a model 1B install **Buster** and not **Bullseye**.

Using a Web browser, go to <https://www.raspberrypi.org/software/>

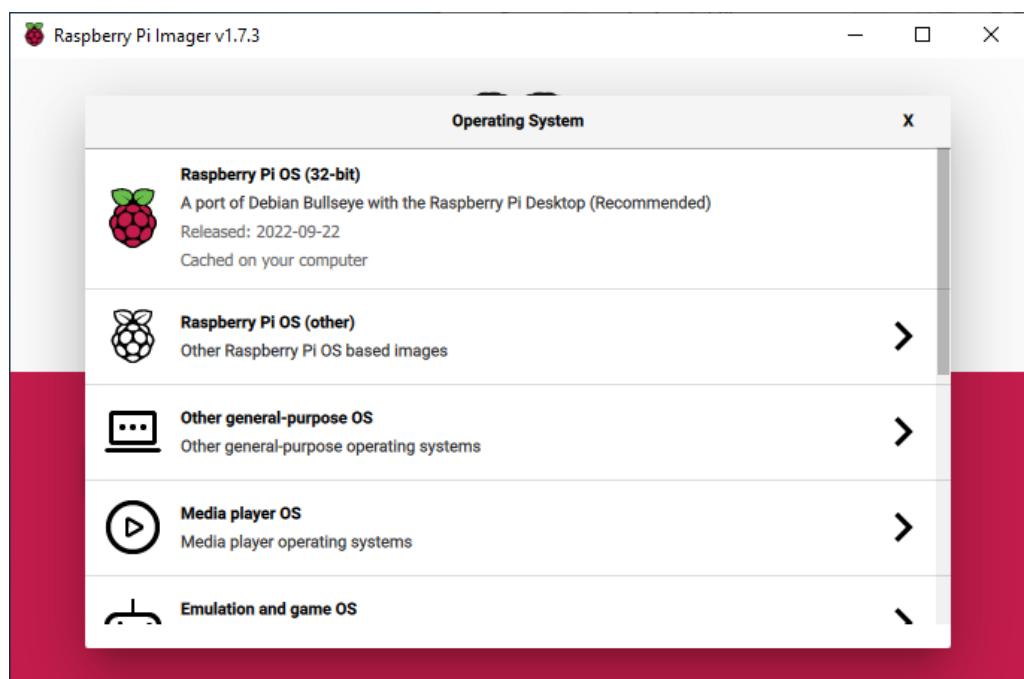
Download and install the **Raspberry Pi Imager** software for your PC Operating System (Normally Windows or Mac OS).

Once installed you will see the Raspberry Pi Imager Icon on your desk-top.
Click on the desktop Imager icon.

The following screen will be displayed:



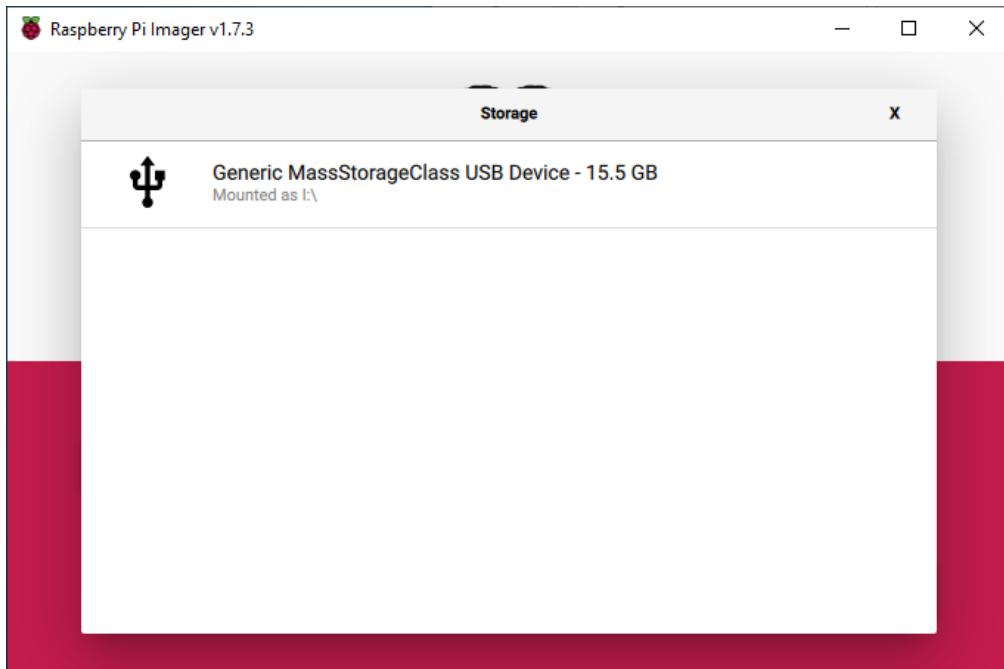
Insert the SD card into your card reader. Ignore any Windows messages to format the card as it isn't necessary. Click on "Choose OS"



The download options will be displayed. Select the first option **Raspberry Pi OS (32-bit)**. If you want the **64-bit** version then select that from **Raspberry Pi OS (other)**.

The program will return to the first screen. Select CHOOSE STORAGE. This will display the available USB devices. The actual drive letter shown will have been assigned by your PC. You should see something like the following. In this example it is drive **H:**

Select your SD card. The imager software should only display USB devices.



Note: If there are multiple choices of USB drives make sure that you select the correct one. Do not proceed until you are absolutely sure you are selecting the correct drive and not for example a USB backup drive.



Note: Do **not** press write at this stage as you need to set up SSH (Secure Shell login) and your Wi-Fi identifier (SSID) and password and other optional changes.



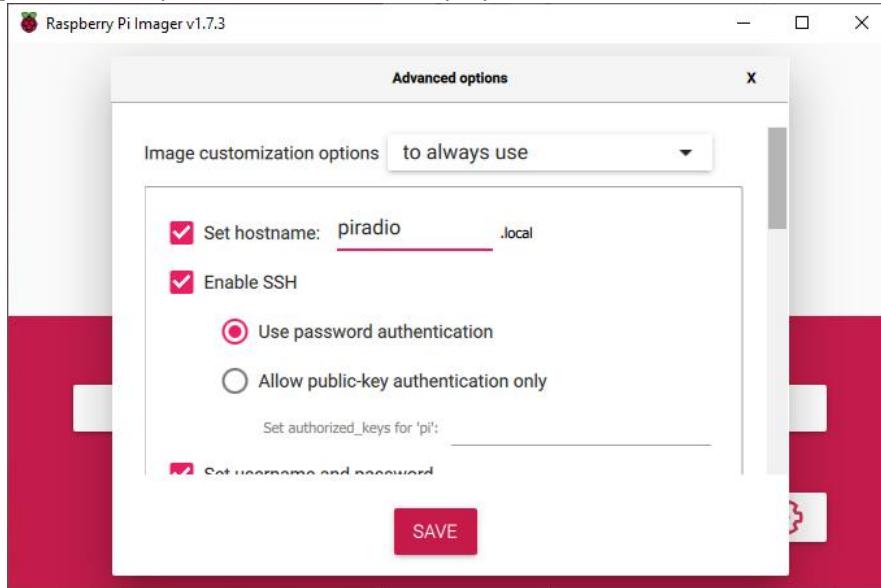
Note: The recommended user name is **pi**. It is possible to use another username other than **pi** when creating the SD card. However only limited testing has been carried out.

Once both the OS and USB drive have been selected the following screen is displayed:



To select the customisation menu press either the configuration icon (Gear wheel) in the bottom right or **CTRL**, **Shift** and **X** keys together. Clicking on WRITE will ask if you want to edit the parameters.

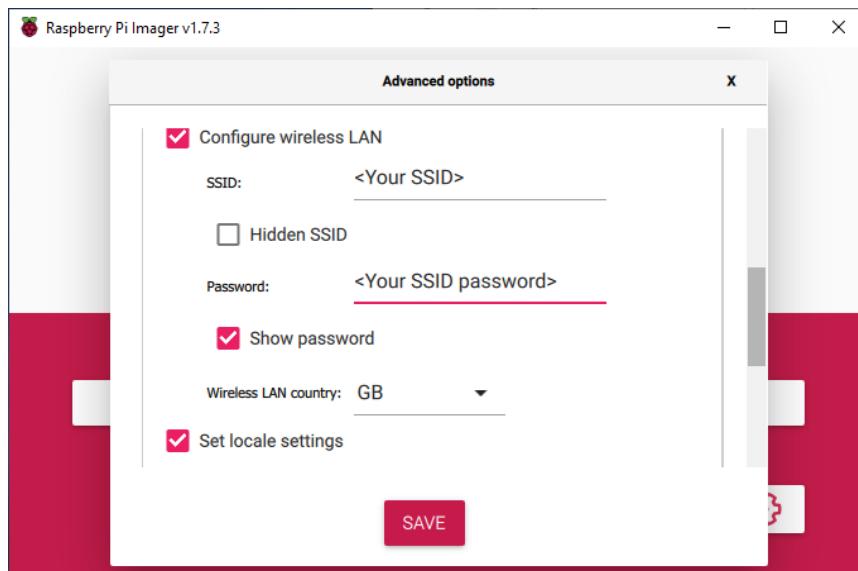
The following Advanced Options screen will be displayed:



Click the “Set hostname” and “Enable SSH” tick boxes. Enter a hostname (such as **piradio**) and a password for user **pi**. You can change the username from **pi** but it is recommended to use user **pi**. Make a note of the username and password.

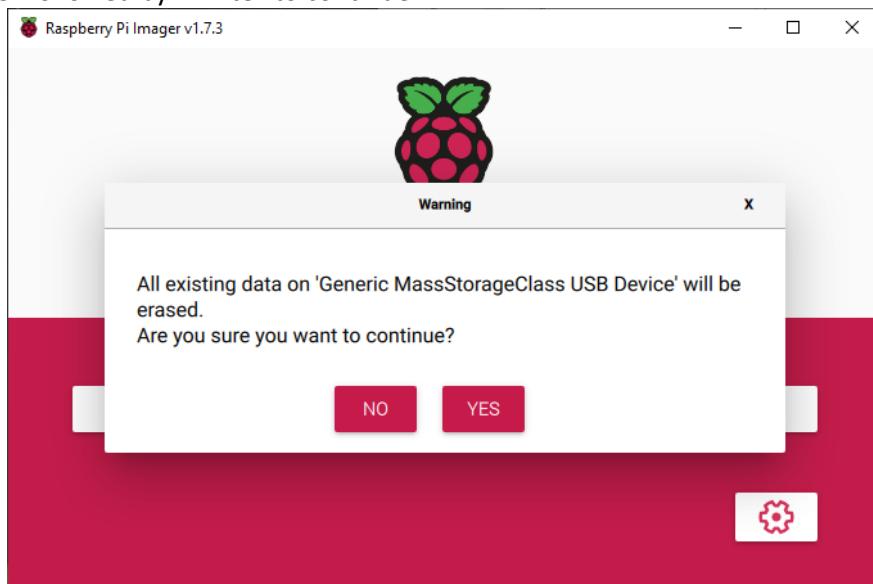
Important: Enable the SSH option “Use password authentication”. If you select “Allow public-key authentication only” you will have to set up SSH keys between the PC and the RPi which is more complicated and not covered yet in this manual.

Do not press SAVE just yet. Ignore “Disable overscan” for now, as this is used to prevent a black border around a HDMI screen. Now scroll down to the “Configure wireless LAN” option using the right-hand scroll bar. Enter the SSID and password for your router. This information is usually given on the router itself or with router documentation.



You must also set the locale using the two-letter country code for your location, for example GB (Great Britain), DE (Germany) or NL(Netherlands). It is required for correct set-up of Wi-Fi. There are other options in this menu but these are not required for this project.

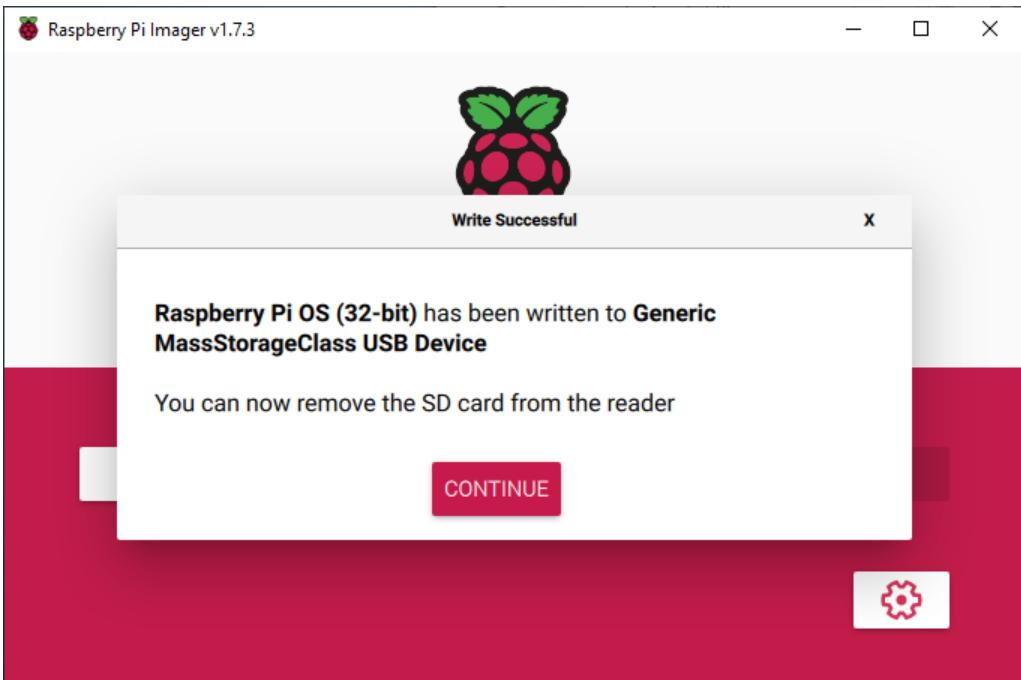
Click on “Save” followed by “Write” to continue.



Click “YES” to continue with writing to the new SD card.



This will take about 10 minutes to complete including the verification stage.



Remove the SD card from your computer and insert it into the Raspberry Pi (Remove power from the Raspberry Pi first).

Booting the Raspberry Pi for the first time

Power up the Raspberry Pi. The first boot will take a little longer as there are a few jobs that the OS must carry out, such as re-sizing the file system on the SD card.



BOOT PROBLEMS: If the Raspberry Pi will not boot up the first time see the section called *Boot problems* on page 240 in *Chapter 9 - Troubleshooting*.

Log into the Raspberry Pi for the first time

There are two ways of logging into the Raspberry Pi:

1. Using the graphical Linux Windows desktop
2. Using SSH to log into the RPi over the network

Logging in with the Raspberry Pi desktop

If you installed the OS with the Linux desktop you will need an HDMI monitor, a USB keyboard and mouse. Boot up the Raspberry Pi with the new SD card with the **Raspberry Pi Buster** Operating System (OS). If you have used **Buster** or **Bullseye** desktop 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 to the left of the "Welcome" message. With **Raspberry Pi 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**. Alternatively log in using SSH (See following section).

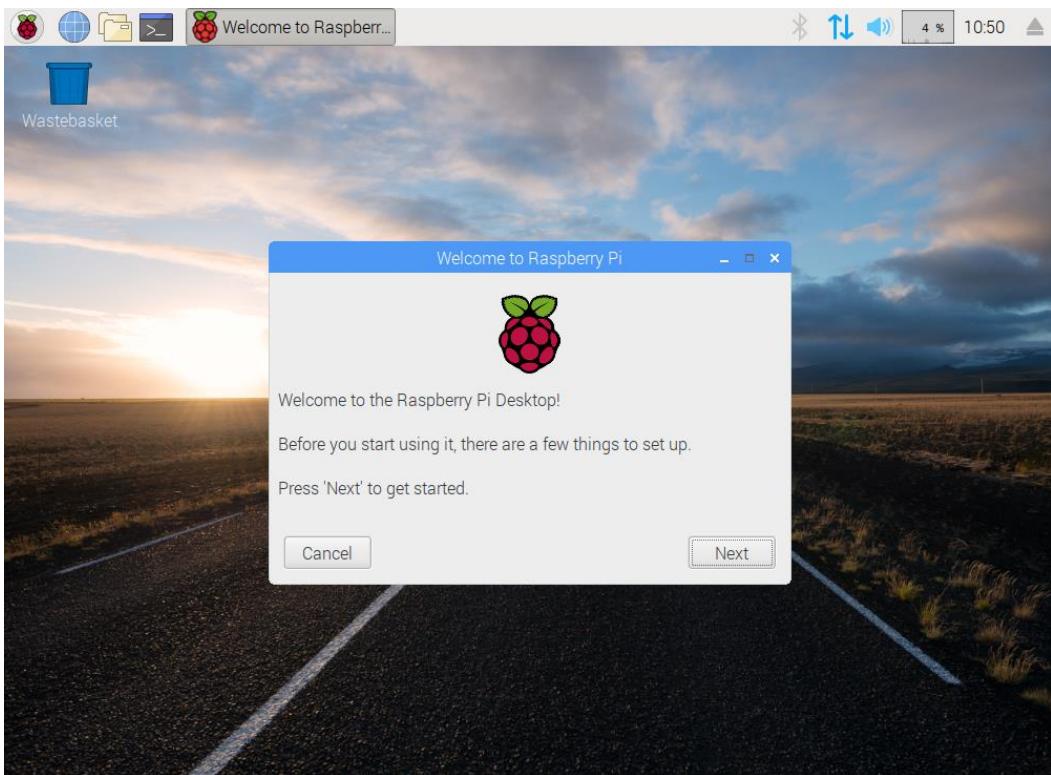


Figure 136 Raspberry Pi desktop

If you created the SD card with the Lite version of the OS then you will not see a desktop. Instead, you will see a terminal session which will allow you to carry out the next steps of the installation.



IMPORTANT: A new screen driver for the Desktop namely the Broadcom VideoCore 4 driver (**vc4-kms-vc3**) has been recently introduced. However, the **vc4-kms-vc3** driver (**dtoverlay**) is not compatible with all HDMI displays. If the system fails to display a Desktop, then see *Disabling the DRM VC4 V3D driver* on page 84.

Logging in with SSH (Bitvise)

If you don't have a USB keyboard, mouse and screen then you can log into the RPi using the Bitvise or Putty utility running on a Personal Computer. Run the **Bitvise** interface and using **Fing** to find the IP address of the Raspberry Pi as shown in the section *Finding the Raspberry Pi on a network using Fing* on page 70. Enter the IP address from Fing into the Host field (192.168.1.36 in this example). Also enter the user name **pi** and the password you set up.

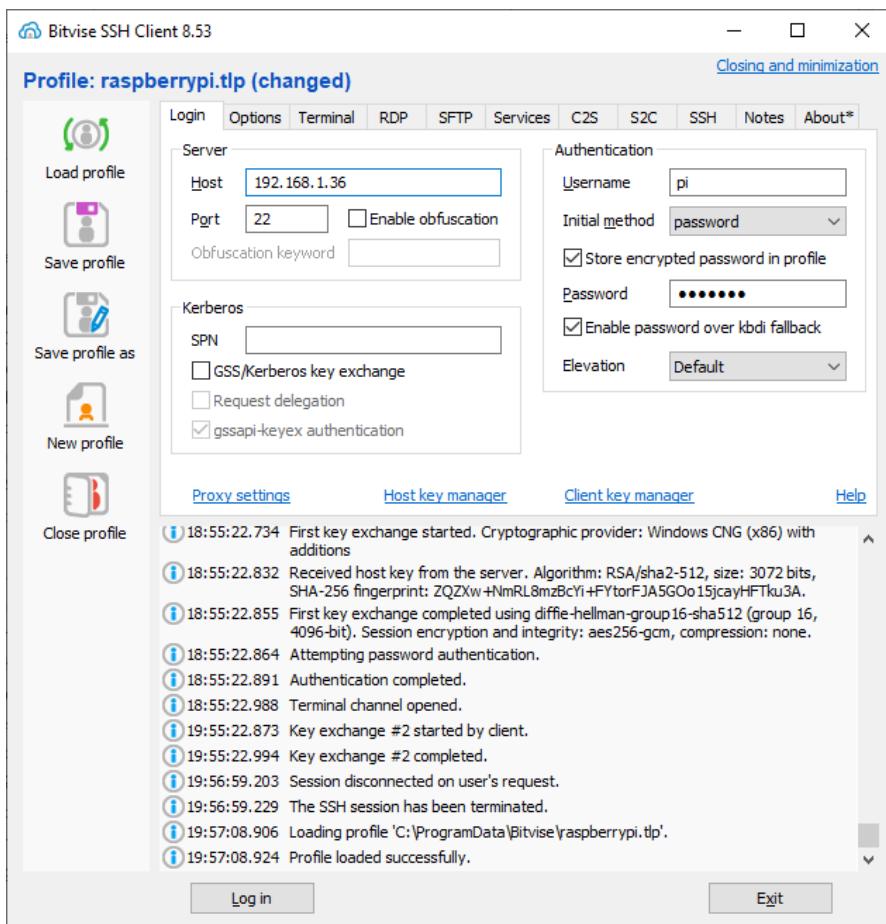


Figure 137 Bitvise connection dialogue



Note: You can also use the host name of the Raspberry Pi instead of its IP address. If you set the SD card up according to the instructions shown earlier. For example if the hostname was "raspberrypi" then try the name **raspberrypi** first. If that doesn't work then try **raspberrypi.lan**, **raspberrypi.local** or **raspberrypi.home** in the host field. If the hostname was "piradio" then try the names **piradio**, **piradio.lan**, **piradio.local** or **piradio.home**. If none of these work just use the IP address.

Enabling SSH using raspi-config

If SSH was enabled when creating the SD card this step should not be necessary. You will need a USB keyboard, mouse and HDMI display screen. Once logged in run **raspi-config**.

```
$ sudo raspi-config
```

Select option 3 – Interface options. The following screen will be displayed

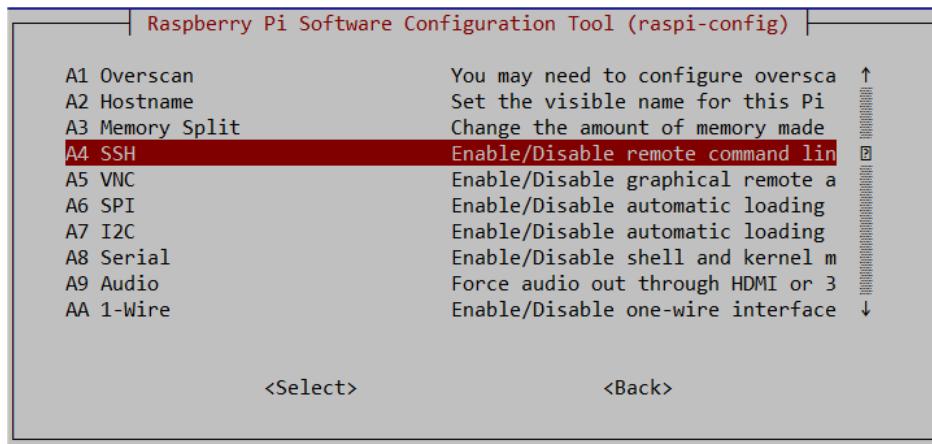


Figure 138 Enabling SSH

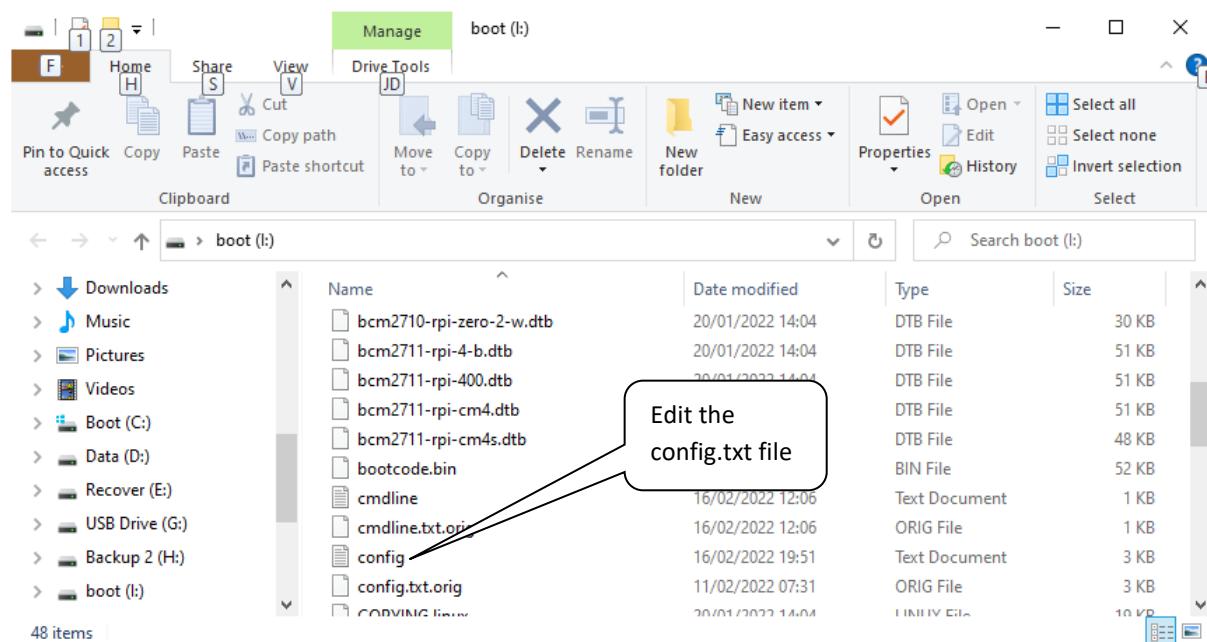
Reboot the Raspberry Pi after which it will be possible to log into the Raspberry Pi using SSH.

```
$ sudo reboot
```

Now log into the Raspberry Pi using SSH.

HDMI Screen not available

If the HDMI display is not available or not working due to the **vc4-kms-vc3** overlay it is necessary to disable the **vc4-kms-vc3** overlay or use the previous **vc4-fkms-vc3** overlay. Put the SD card into a PC. A new drive will be seen called boot. In the following example this is drive (i:). This will vary with each system. Look for the text file config.txt.



Open the config.txt file using notepad. Carry out the edit shown in *Disabling the DRM VC4 V3D driver* on page 84. Once the config.txt file has been edited and saved, put the SD card back into the Raspberry Pi and reboot the system.

```
$ sudo reboot
```

Disabling the DRM VC4 V3D driver

To disable the **vc4-kms-v3d** driver (`dtoverlay`) first log into the Raspberry Pi using SSH or by using the method described in *HDMI Screen not* on page 83.

If logged in via SSH, edit the **/boot/config.txt** file either using **sudo nano** or **vi**.

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

If editing the config.txt file as shown in *HDMI Screen not available* on page 83, use the PC text editor to disable the VC4 driver as shown below. Comment out the **dtoverlay** statement for the VC4 driver.

```
# Enable DRM VC4 V3D driver  
#dtoverlay=vc4-kms-v3d
```

Or try using the older **dtoverlay**:

```
dtoverlay=vc4-fkms-v3d
```

Reboot the system.

```
$ sudo reboot
```

After logging back in the following message may be displayed if a new password wasn't set.

SSH is enabled and the default password for the 'pi' user has not been changed. This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

 **Note:** Security is becoming more and more of an issue for devices connected to the internet. If SSH has been enabled then please change the user password at the first opportunity. See the section called *Chapter 13 - Internet Security* page 301 for further information on security issues.

Preparing the Operating System for software installation

Update to the latest the packages

Run the following command to update the library list.

```
$ sudo apt update
```

The above command will take some time! If you see a message similar to the following.

```
E: Repository 'http://raspbian.raspberrypi.org/raspbian Bullseye InRelease'  
changed its 'Suite' value from 'testing' to 'stable'
```

Run the following command

```
$ sudo apt update --allow-releaseinfo-change
```

Now re-run the update.

```
$ sudo apt update
```

Run the following command to upgrade to the latest packages for this release.

```
$ sudo apt full-upgrade
```



Note: The **update/full-upgrade** commands will also automatically update the Raspberry Pi with the latest stable firmware. Older Raspbian versions such as Stretch required a separate command called **rpi-update** to update the firmware. Do not use **rpi-update** with **Bullseye** or **Buster** unless specifically advised to do so by an official source.

Reboot the Raspberry Pi.

```
$ sudo reboot
```



Important: After upgrading the system the repository locations may no longer be valid. Re-run **apt update** to refresh the package list. Failing to do this may result in packages failing to install.

Re-run the update command to update the library list.

```
$ sudo apt update
```

Once you have updated the operating system login to the system and run **raspi-config**.

```
$ sudo raspi-config
```



Warning: If you are intending to run the touch-screen/HDMI version of the radio, do not be tempted to start removing components of the **pygame** software such as games as this may unfortunately remove graphic libraries used by the radio software.

Disable booting to the desktop environment

If you are planning to use a touch-screen or HDMI display skip this section.

The desktop environment is not required for the LCD, TFT or OLED versions of the Radio and takes a lot of processing power. It is enabled by default in **Bullseye** but is not installed with **Bullseye Lite**. If you are not planning to use the touch-screen or HDMI version of the radio or you don't otherwise plan to use it then disable it.

Run **raspi-config**:

```
$ sudo raspi-config
```

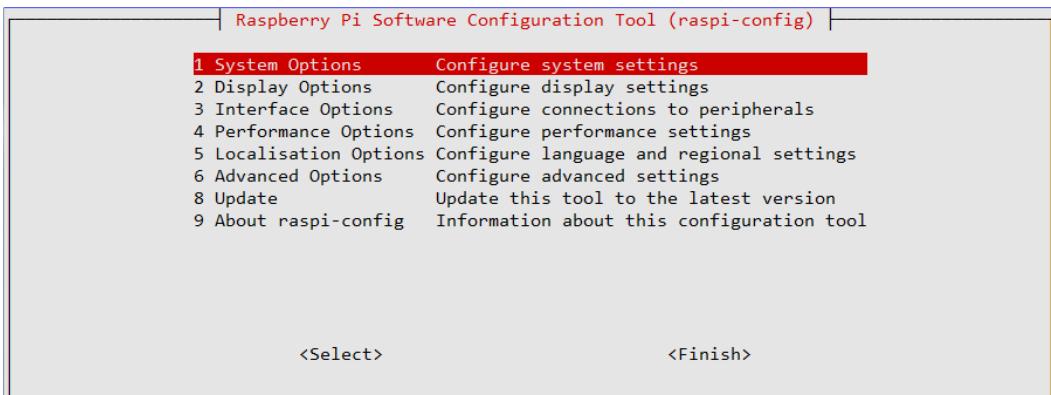


Figure 139 raspi-config main screen

Select option 1 System options

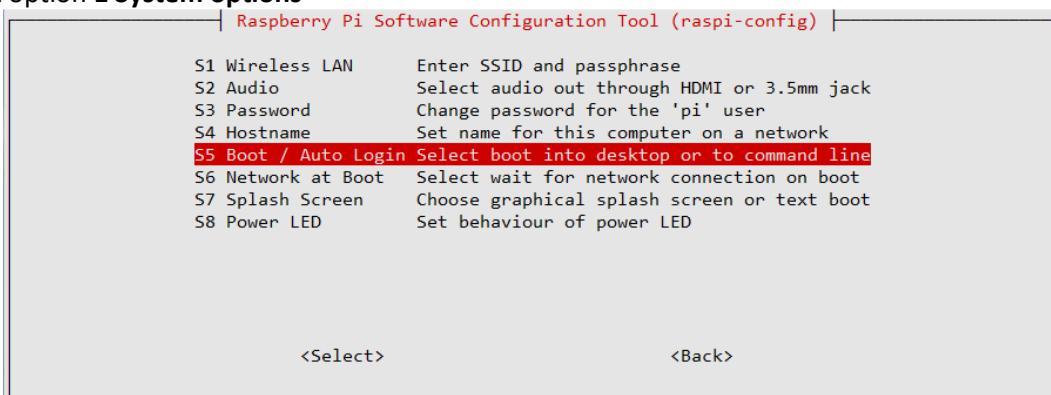


Figure 140 Disabling the graphical desktop

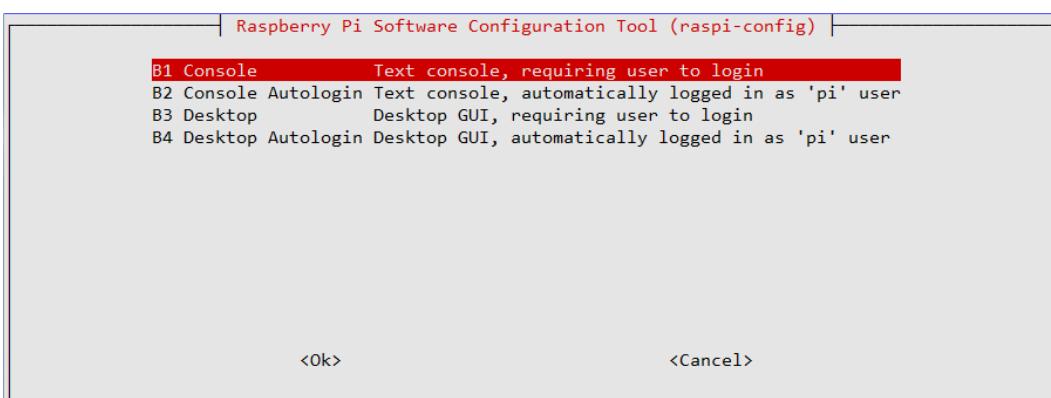


Figure 141 Desktop enable/disable selection

Select option **B1**(secure) or **B2**(insecure) to disable the desktop and select OK

Setting the time zone

The **Raspberry Pi Buster** operating system is usually set to UK time or the time zone that you set with the **Raspberry Pi OS imager** settings. If you still need to set the time zone then the easiest way to set the time zone for your country if you are in a different time zone is to use the **raspi-config** program and select option 5 “Localisation Options”:

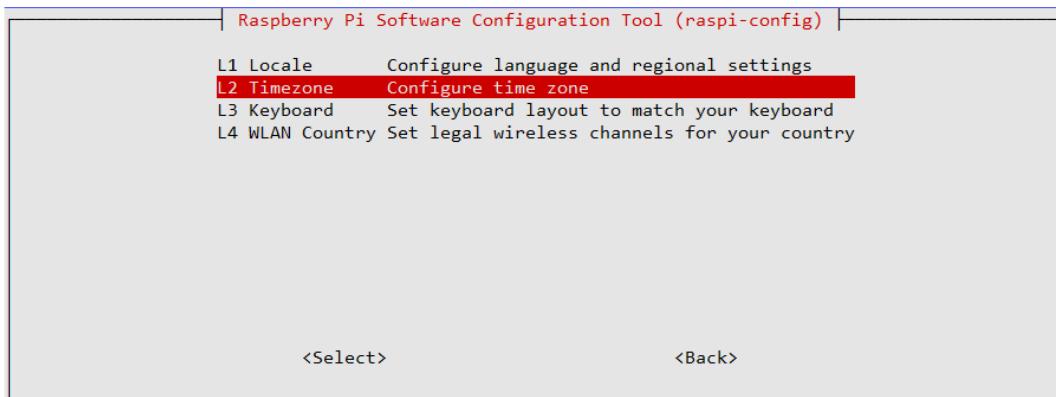


Figure 142 Setting the time zone

Select option L2 “Timezone”:

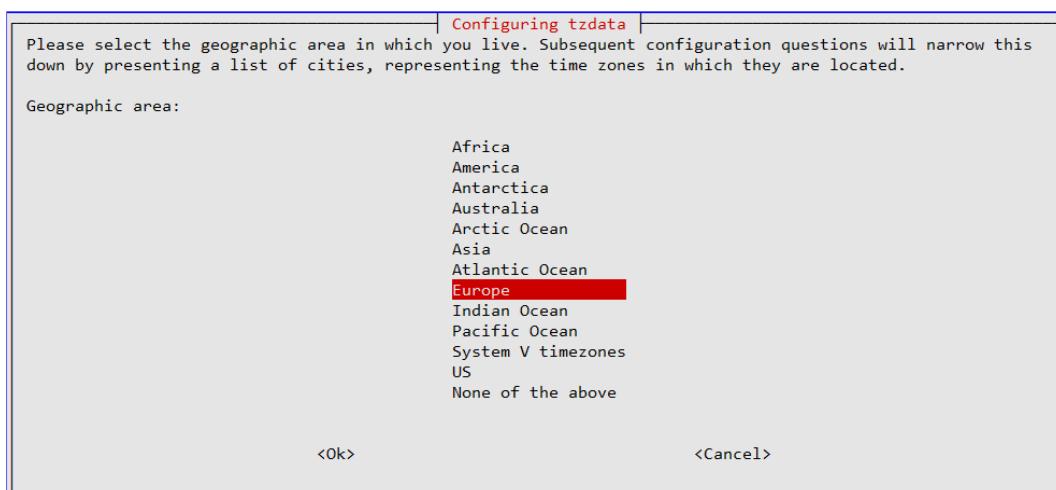


Figure 143 Saving the time zone

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 time zones for the selected region.



Figure 144 Time zone country selection

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

Changing the system hostname and password



If the **RaspberryPi** Imager was used to create the SD card it may be that the hostname and password may have already been set, in which case this option can be skipped.

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 (WI-FI) 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. Select option **1 System options**

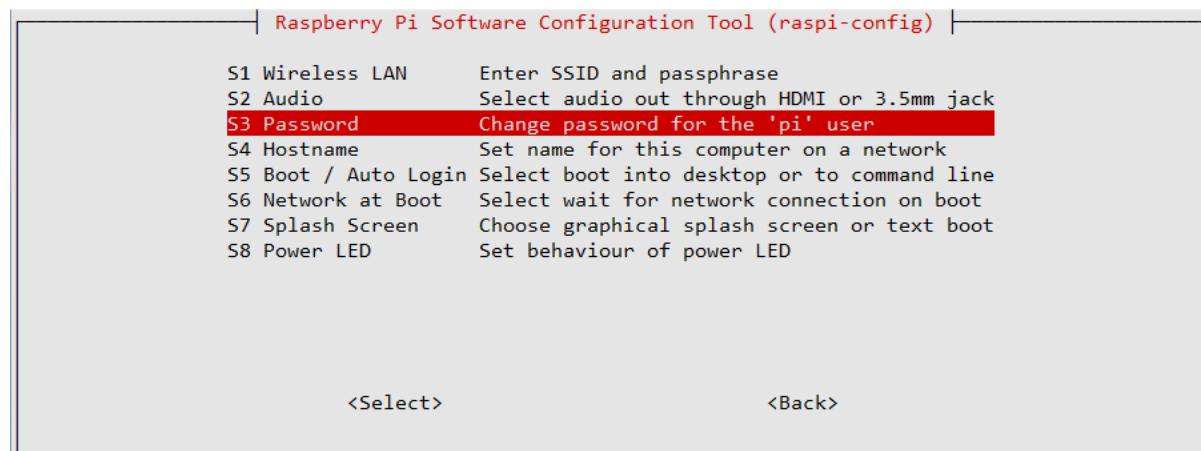


Figure 145 Changing the Raspberry PI password

Option **S3** 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 **S4 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**. Once the hostname has been changed the program will ask if you wish to reboot. Answer “yes” to reboot.

Configuring the Wi-fi Connection via raspi-config



If the **RaspberryPi** Imager was used to create the SD card it may be that the Wi-Fi interface has already been configured, in which case this option can be skipped.

Select the country you are in for legal Wi-Fi channel selection. From the first **raspi-config** screen select **1 System Options** followed by option **S1 Wireless LAN**:

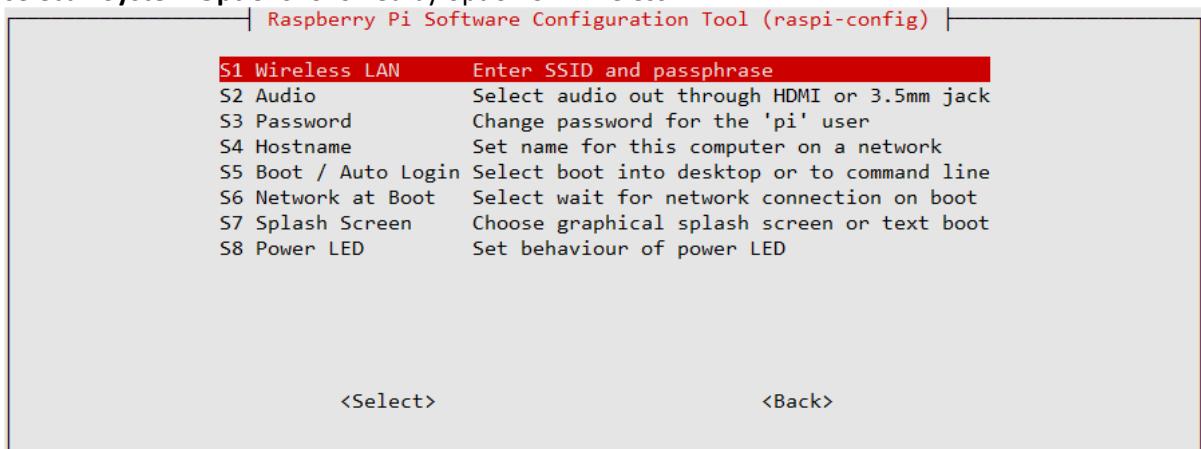


Figure 146 Setting up the Wi-Fi in raspi-config

Now enter the SSID and passphrase for your network. Save the settings and reboot the Raspberry Pi. If this option is not available then use the procedure

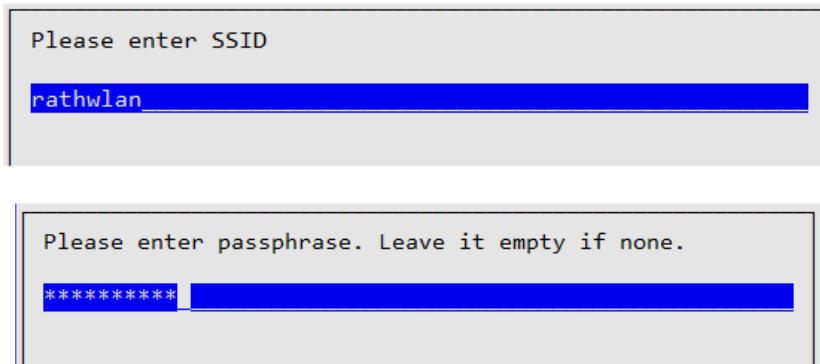


Figure 147 Entering Wi-Fi credentials

Reboot the system. After reboot it is possible that you may see the following message:

```
Wi-Fi is disabled because the country is not set.
Use raspi-config to set the country before use.
```

In such a case re-run **raspi-config**. Select localisation options and select **L4** to set Wi-Fi country:

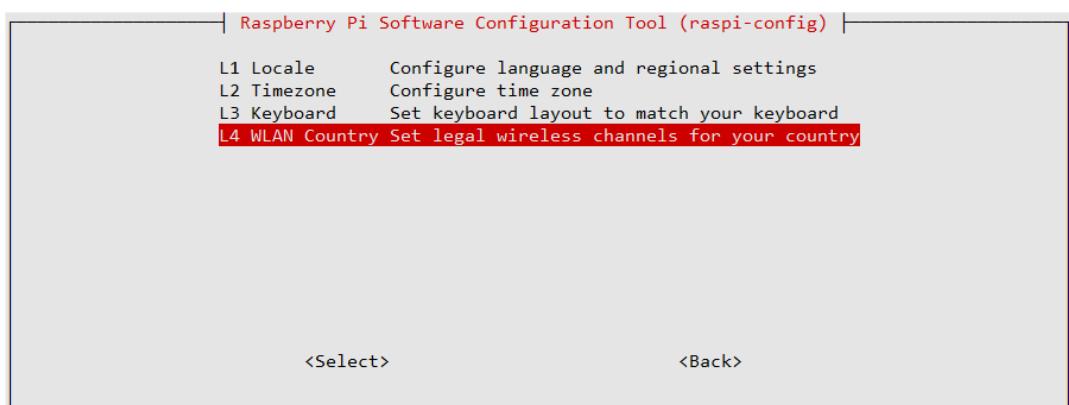


Figure 148 Setting up the Wi-fi country

Select your country from the dropdown menu and exit the **raspi-config** to save the setting.

Setting up the locale

As default the language used by Raspbian is English. To change this in Advanced options select option I1 Change Locale.

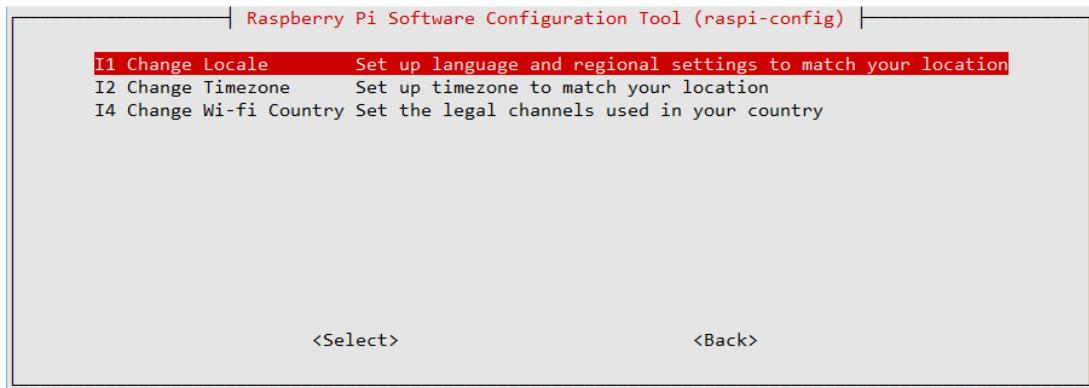


Figure 149 Selecting change locale

Select a locale beginning with the two-letter international code for your country and language.

Usually this will be a locale containing the string ISO or UTF-8.

fi_FI.ISO-8859-1 (Try this one first)
fi_FI.ISO-8859-15@euro
fi_FI.UTF-8

In the following example for the Netherlands the locale is this **nl_AW UTF-8** for the Netherlands (nl).

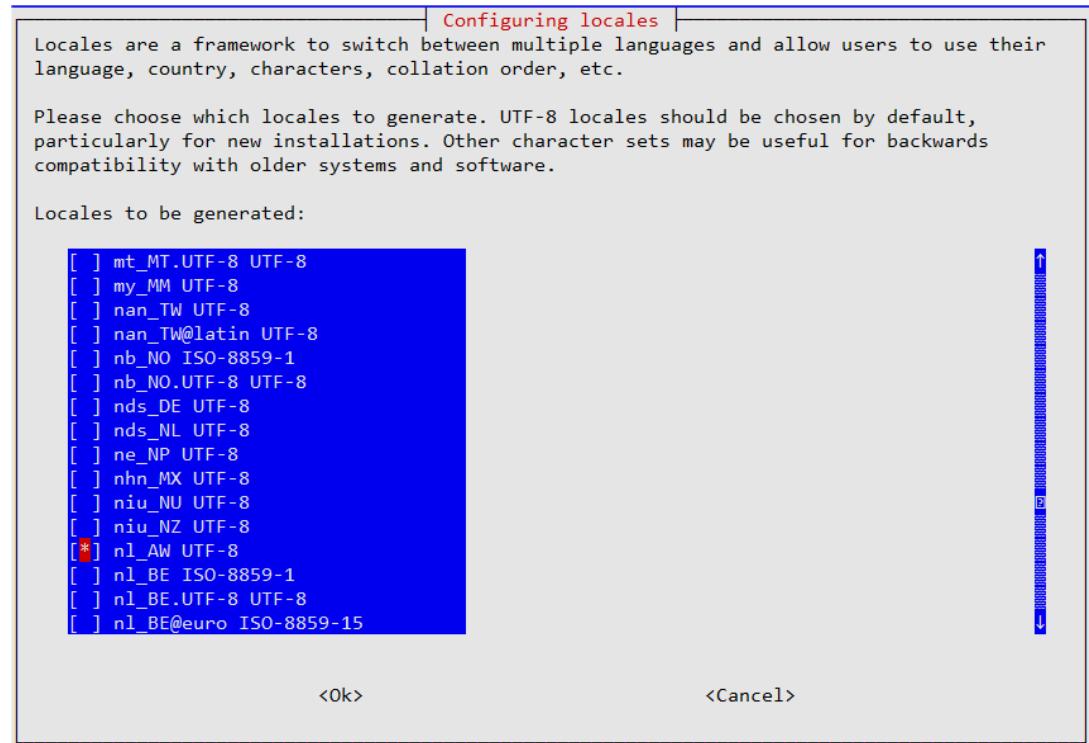


Figure 150 Generating the locale.



Warning: Do not be tempted to generate “All locales” as this is not only unnecessary and futile and will take a very long time.

Press OK to continue. The program will now generate the selected locales.

```
Generating locales (this might take a while)...
en_GB.UTF-8... done
nl_AW.UTF-8... done
Generation complete.
```

The following screen is displayed:

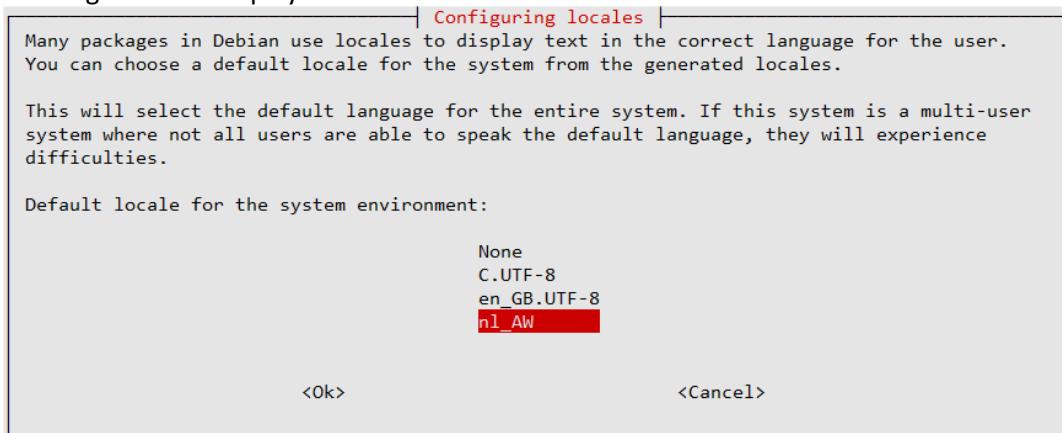


Figure 151 Selecting the locale

Select the required locale **nl_AW** in this example and press OK to save. The new locale will become active after the next reboot.

Install other required packages

The table below shows additional packages which are required depending upon the design.

Table 14 Additional system packages

Package	Grove	HDMI/ Touchscreen	Olimex	Shoutcast	RGB I2C Rotary Encoder	Page
ffmpeg	No	Yes	n/a	n/a	No	91
python-requests	No	n/a	n/a	Yes	No	92
libffi-dev build-essential libi2c-dev i2c-tools python3-dev	No	No	Yes	n/a	No	
python3-pil	Yes	No	Yes	n/a	No	92
scratch	No	Optional	No	No	No	92
ioe-python	No	No	No	No	Yes	93

Install the **ffmpeg** video converter

This section is only applicable if the HDMI/Touchscreen version of the radio is to be used. If using an LCD version of the radio then skip this section. The **ffmpeg** video converter is required to extract artwork (if included) from music files mpeg files. Install **ffmpeg** video converter with the following command (Note that it may already be installed):

```
$ sudo apt install ffmpeg
```

```
Reading package lists... Done
:
The following packages have unmet dependencies:
ffmpeg : Depends: libavdevice57 (>= 7:3.2.10) but it is not going to be
          installed
          Depends: libSDL2-2.0-0 (>= 2.0.4) but it is not installable
E: Unable to correct problems, you have held broken packages.
```

If the installation fails with the above message re-run the **apt update** command to update the library list and retry installing **ffmpeg**.

```
$ sudo apt update
```

Run the following for further information about **ffmpeg**.

```
$ man ffmpeg
```

Install libraries for the Olimex OLED or SSD1306 OLED

If using the IQaudio Cosmic controller or SSD1306 OLED is also necessary to install **libffi-dev** plus other necessary libraries. If you are not using the IQaudio controller then skip this section. Carry out the following instructions.

```
$ sudo apt -y install libffi-dev
$ sudo apt -y install build-essential libi2c-dev i2c-tools python3-dev
```

If using **Raspbian Lite** also install the **python3-pil** package:

```
$ sudo apt -y install python3-pil
```

Install the anacron package

The **anacron** package is required to run the **/etc/cron.weekly/radiod** and **/etc/cron.daily/radiod** scripts. These scripts are required to run the **create_stations.py** and **/etc/cron.daily/radiod** playlist maintenance programs.

Install it with the following command.

```
$ sudo apt -y install anacron
```

Install the Scratch package

This section is only applicable if the HDMI/Touchscreen version of the radio is to be used. There is an option to provide extra backgrounds. If using an LCD or OLED versions of the radio then skip this section. Scratch used to be installed by default, however, in the latest versions of **Raspberry Pi OS Desktop** it isn't. Extra backgrounds can be used by installing **Scratch**. Scratch is a visual programming tool for children.

Scratch is not actually used by the radio software but the graphic files in the **/usr/share/scratch/Media/Backgrounds** directory provide additional background wallpaper for the full feature graphical radio (gradiod.py).

The background wallpaper is set by the following parameter in **/etc/radiod.conf** and can be amended to use a different background. This can only be done after installing the radio software in the next section.

The default wallpapers are available in the **/usr/share/rpd-wallpaper** directory. Amend the wallpaper command once you have installed the radio software. The default

```
wallpaper=/usr/share/rpd-wallpaper/canyon.jpg
```

If **scratch** is installed other additional wallpapers can be used:

```
wallpaper=/usr/share/scratch/Media/Backgrounds/Nature/beach-malibu.jpg
```

To install scratch run the following:

```
$ sudo apt install scratch
```

Install Pimoroni ioe-python

The **ioe-python** python package is required for RGB I2C Rotary Encoders.

```
$ git clone https://github.com/pimoroni/ioe-python
$ cd ioe-python/
$ sudo ./install.sh --unstable
```

The following message will be displayed.

```
Would you like to copy examples to /home/pi/Pimoroni/pimoroni-ioexpander?
[y/N]
```

Unless you particularly want the example code answer N.

Reboot

```
$ sudo reboot
```

Extra packages required for Raspberry OS Lite

If you are running **Raspberry OS Lite** then not all required packages are installed. Run the following commands.

```
$ sudo apt install python-configparser
$ sudo apt install python3-pip
```

The following are required for any displays that are using **I2C**.

```
$ sudo apt install -y i2c-tools
```

Disable PiWiz screen reader (Buster only)

Since December 2020 release of Buster, the developers have installed the PiWiz screen reader.

As a result, every 15 seconds a computerized voice says
"To install the screen reader press control alt space".

This feature is normally disabled during installation using the desktop interface. However, if you are using a headless installation procedure it may not be disabled.

The message is not only very annoying, but hogs the sound system meaning that the radio (MPD) cannot play audio content. To disable PiWiz run the following:

```
$ sudo rm /etc/xdg/autostart/piwiz.desktop
```

On some later versions of Bullseye this file has already been removed so if it isn't found this step can be ignored.

Chapter 6 - Installing the radio Software

Upgrading from previous versions

If upgrading from version 6.x or earlier it is necessary to replace your existing configuration file with the new one supplied in the upgrade. First take a note of any specific configuration items that are not standard or make a copy of your existing **/etc/radiod.conf** configuration file.



Important: The name format of Radio playlists has changed from version 7.3 onwards. See Creating and Maintaining Playlist files on page 215. Also, in this release, it is now possible to add, move and delete entries in the play list using any MPD client that is capable of doing so. See *Maintaining playlists using external MPD clients* on page 223.



Note: The location of the log file has changed from **/var/log/radio.log** to **/var/log/radiod/radio.log**. Also, its ownership has been changed from **root** to user **pi**. This has been done to reduce the number of programs that need to run with **sudo**.

Installing the Music player daemon

If you haven't already done so upgrade the system packages as shown in *Update to the latest the packages* on page 84.

After reboot install the [Music Player Daemon](#) (mpd) and its client (mpc) along with the Python3, MPD and **python3-rpi.gpio** libraries. Note that you must install **python3-mpd** and not **python.mpd**.

```
$ sudo apt install mpd mpc python3-mpd python3-rpi.gpio
```

Answer yes 'y' when asked to continue.

Install the client library needed by the radio software (It may be already installed).

```
$ sudo apt install libmpdclient2
```

If the installation says it cannot find the above packages, then re-run the update command to update the library list and retry installing the above packages.

```
$ sudo apt update
```



IMPORTANT: Unfortunately, the configuration file (**/etc/mpd.conf**) supplied with MPD version 0.22.6 on **Bullseye** is corrupt. It is necessary to correct this as show below.

Find the following entries in **/etc/mpd.conf** somewhere around line 232.

Corrupt /etc/mpd.conf file

```
#audio_output {
#    type          "alsa"
#    name          "My ALSA device"
#    device        "hw:0,0"
#    mixer_type    "software"
##    mixer_device  "default"      # optional
##    mixer_control "PCM"         # optional
```

```

##      mixer_index      "0"          # optional
#

```

Change the above entries to the following:

Corrected /etc/mpd.conf file

```

audio_output {
    type      "alsa"
    name      "My ALSA Device"
    device    "hw:0,0"
    mixer_type "software"
#    mixer_device   "default"    # optional
#    mixer_control   "PCM"        # optional
#    mixer_index     "0"          # optional
}

```



If installing on **Bullseye Lite** this will take quite a long time as there are a lot of software libraries to be installed.



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



The version of **MPD** released with **Bullseye** is **0.22.6** which is reasonably up-to-date. You should install MPD version **0.22.6** first. When it is all working if you are having problems playing certain stations, it is well worth installing the latest version of MPD as shown in the section called *Compiling and installing the latest Music Player Daemon* on page 172.

Installing pulseaudio

The **pulseaudio** package may or may not need to be installed. This has been different between various releases of the operating system. Currently the **pulseaudio** package causes a lot of problems with the radio software, in particular **espeak** and the Alsa sound system. If you do not intend to use espeak and need pulseaudio, for example, for blue-tooth devices or pHat BEAT then install it. The following table shows the options:

Table 15 PulseAudio installation options

Type of radio installation	Install pulseaudio
Using espeak	No
Pimoroni Pirate Radio with pHat BEAT	Yes (Installed by phatbeat)
Pimoroni Pirate Audio/mini-speaker	No
Adafruit speaker bonnet	No
LCD display radio	No unless using above DACs
HDMI or touch-screen displays	No unless using above DACs
IQaudIO, HiFiBerry or JustBoom DACs	No
Bluetooth sound devices	Yes



The Pimoroni installation software for Pirate Radio installs **pulseaudio**.

To install **pulseaudio**:

```
$ sudo apt install pulseaudio
```

To remove **pulseaudio**:

```
$ sudo apt remove pulseaudio
```

If installing on Bullseye disable the `pulseaudio-enable-autospawn.service`.

```
$ sudo systemctl stop pulseaudio-enable-autospawn.service
```

Also remove unwanted libraries.

```
$ sudo apt autoremove
```

Install display drivers



Note: If you are installing the software for any of the products listed below then first carry out the instructions in the appropriate section. Do this before installing the radio software.

Table 16 Display driver software installation

Device	Section/Devices	Page
Pirate Radio	Installing Pimoroni Pirate Radio (pHat BEAT)	113
Pirate Audio	Installing the Pimoroni Pirate Audio	114
SSD1306 0.9-inch OLED	Installing the SSD1306 OLED driver	115
Waveshare TFTs	Waveshare TFT device installation	117
MHS RPi displays	MHS 3.5-inch RPi display installation	118
SH1106 1.3-inch OLED	Installing LUMA monochrome OLEDs	119
LUMA OLED devices	SSD1306, SSD1309, SSD1325, SSD1331, SH1106, WS0010	119
Grove LCD RGB	Installing the Grove LCD RGB	120
Adafruit TFT	Installing the Adafruit 2.5 and 3.5-inch TFT touchscreen	120
PiFace CAD	Installing PiFace CAD software	122

Install the Radio Daemon

First install the **python-configparser** package:

```
$ sudo apt install python-configparser
```

The Raspberry PI Internet Radio software is distributed as a Debian package. This can be downloaded from http://www.bobrathbone.com/raspberrypi/pi_internet_radio.html

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.

For 32-bit systems:

```
$ wget http://www.bobrathbone.com/raspberrypi/packages/radiod_7.5_armhf.deb
```

Run **dpkg** to install the package.

```
$ sudo dpkg -i radiod_7.5_armhf.deb
```

For 64-bit systems:

```
$ wget http://www.bobrathbone.com/raspberrypi/packages/radiod_7.5_arm64.deb
```

Run **dpkg** to install the package.

```
$ sudo dpkg -i radiod_7.5_armhf.deb
```

If upgrading or re-installing the radio software use the following to overwrite the existing package download file (Substitute correct file name):

```
$ curl -L -O http://bobrathbone.com/raspberrypi/packages/radiod_7.5_arm64.deb
```

The **dpkg** program will install the files.

```
(Reading database ... 131542 files and directories currently  
installed. Preparing to unpack radiod_6.14_armhf.deb ...  
Raspberry PI internet radio installation  
Stopping radiod service  
Unpacking radiod (7.5)
```

If updating from an earlier version of the radio software you may see the following message.
Enter 'I' or 'Y' to install the new version.

```
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] ? I
```

Configuring the radio

Once that is done the installation will run the **configure_radio.sh** script. This updates the configuration settings in **/etc/radiod.conf**.



Note: This configuration program does the basic configuration to get the radio going with the hardware you are using. More complex configuration options are explained in *Chapter 7 – Configuration* on page 176.

The configuration program is automatically run when installing the radio package but can safely be run at any time using the following commands:

```
$ cd /usr/share/radio  
$ sudo ./configure_radio.sh
```

The installation script detects if this is a software upgrade and if that is the case displays the following screen. Normally select option 2 if upgrading the software. This means that the configuration will not be changed.

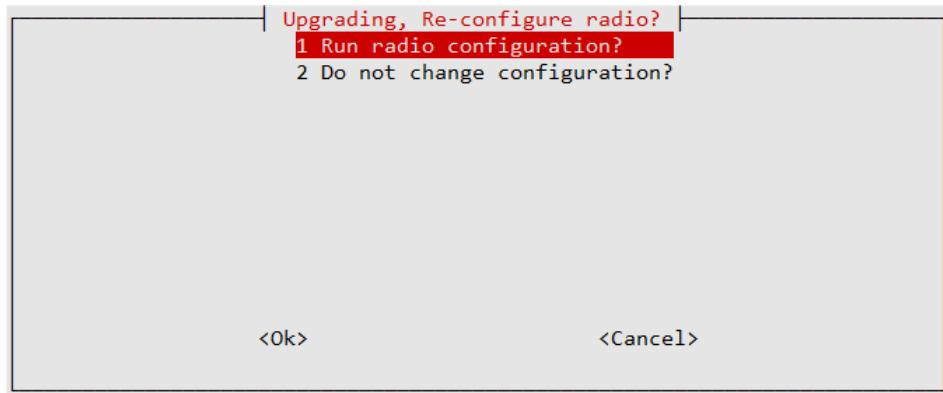


Figure 152 Configure radio – Upgrade

If you selected option 1 above and you are upgrading the software from a previous version, the program will ask if you wish to overwrite the existing configuration. Unless you have a heavily modified configuration you may safely overwrite the configuration file:

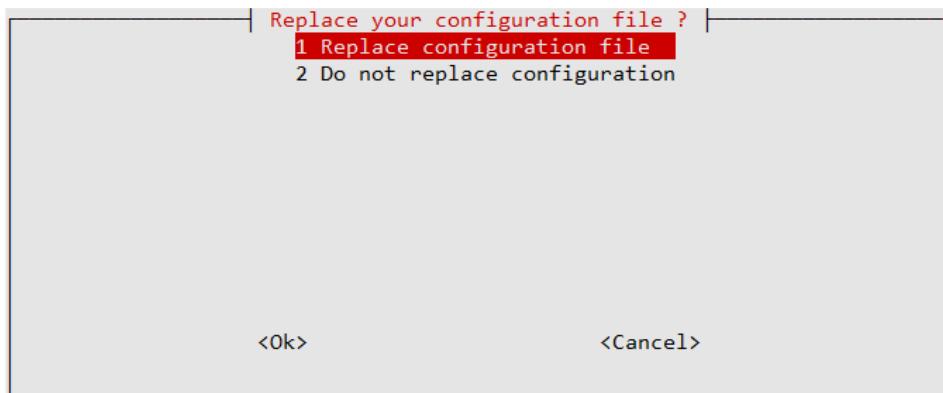


Figure 153 Replace configuration file

If option 1 is selected the existing configuration will be replaced. A backup copy of the original configuration is written to **/etc/radiod.conf.save**. The following screen will then be displayed:

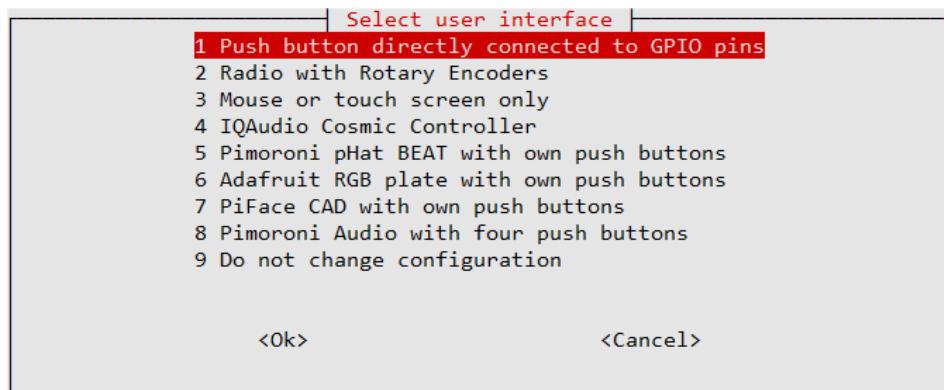


Figure 154 Configure radio – User interface selection



Note: This configuration program can be re-run at any time in the future. Change directory to **/usr/share/radio** and run **configure_radio.sh**. To do this run the following:

```
$ cd /usr/share/radio  
$ sudo ./configure_radio.sh
```

Select option 1 if push buttons are being used or option 2 if using rotary encoders. Otherwise select the type of user interface being used in option 3 onwards.

This screen and all following screens have the option to not change the configuration. The program will always ask you if your choice is correct and give you the opportunity to change the selection.

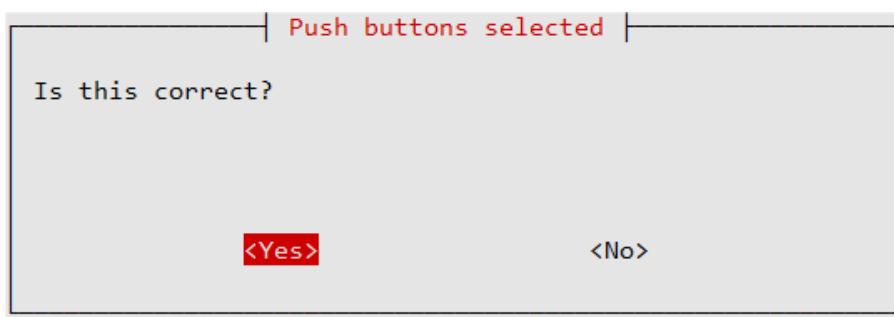


Figure 155 Configure radio - Confirmation screen

If the push-button interface is selected, the program will ask how they have been wired. The push-buttons can be wired to either +3.3V (The original scheme) or GND(0V).

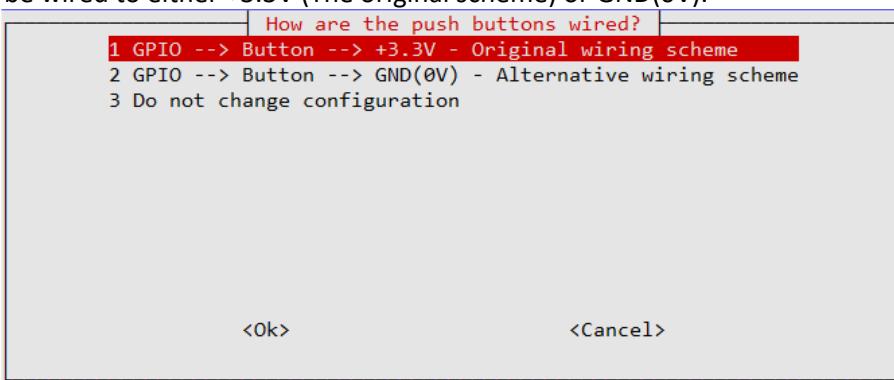


Figure 156 Push-button voltage selection

If option 2 Rotary Encoders was selected then the following screen will be seen:

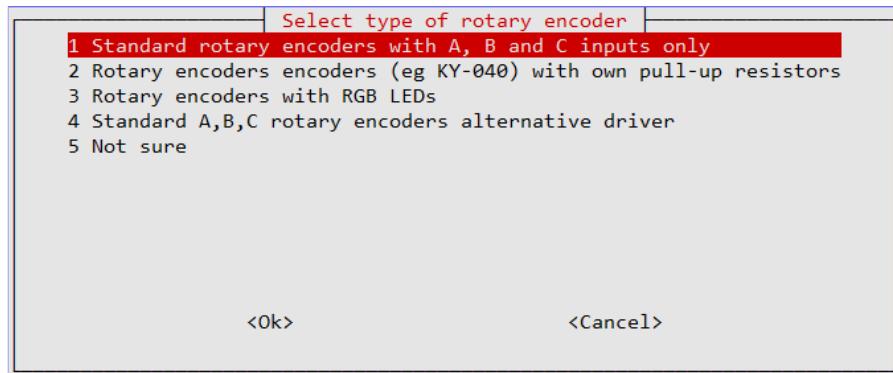


Figure 157 Rotary encoder type selection

Traditional rotary encoders have three connections called A, B and C. They usually also have two connections to a switch (See [Rotary encoder wiring on page 30](#)). If these rotary encoders are being used the select option 1 as shown above.

If rotary encoders such as the KY040 (See [Using KY040 Rotary encoders on page 31](#)) are being used, these have their own 10K pull-up resistors and so do not need the internal GPIO pull-up resistors as well. If the internal GPIO pull-up resistors are also enabled then they are paralleled across the rotary encoders pull-up resistor giving a value of 5K for the pull-up which is half the recommended 10K value. Select option 2 KY040 encoders to disable the internal GPIO pull-up resistors. Select option 2. This can also be done by setting `rotary_gpio_pullup=None` in `/etc/radiod.conf`.

 If you are missing the push-button 10K resistor as shown in [Figure 62 KY040 Missing 10K resistor on page 31](#) then select option 1 (Standard A, B and C inputs).

If using rotary encoders with RGB LEDs select option 3.

If there are problems with the standard A, B, C rotary encoders try option 4 - Alternative driver.

 **Note:** Option 2 will not work with traditional A, B and C encoders. If unsure select option 3 and the internal pull-up resistors will be enabled and will work with both types of encoders.

Next select the GPIO header wiring. This is either 26-pin (Older RPi models) or 40-pin wiring.

Note that the 26-pin wiring scheme can also be used on 40-pin headers if so wished.

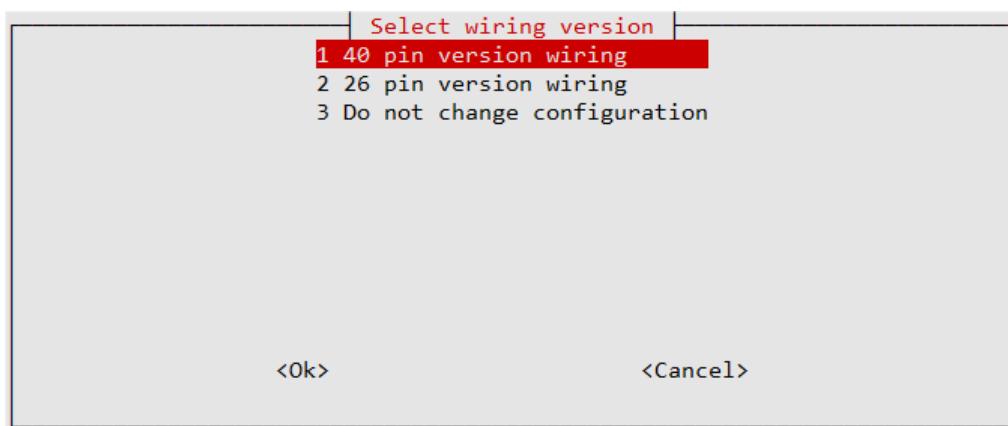


Figure 158 Configure radio - wiring selection

Normally select the 40-pin option unless you have used the 26-pin wiring scheme. See [Table 3 Controls and LCD wiring 26 pin version](#) and [Table 4 Radio and DAC devices 40-pin wiring](#).

Confirm selection and continue to the next screen:

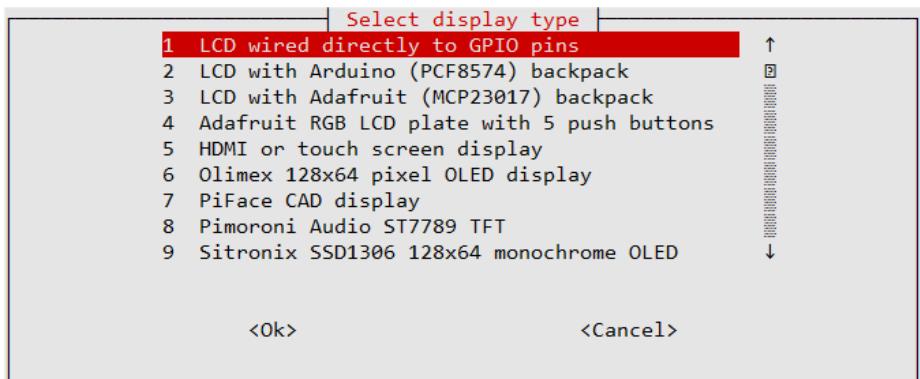


Figure 159 Configure radio - Display interface selection 1

Notice that there is a scroll bar to the right of the options. Use the Up/Down keys to scroll down to the remaining options. Select the option which matches the display you are using.

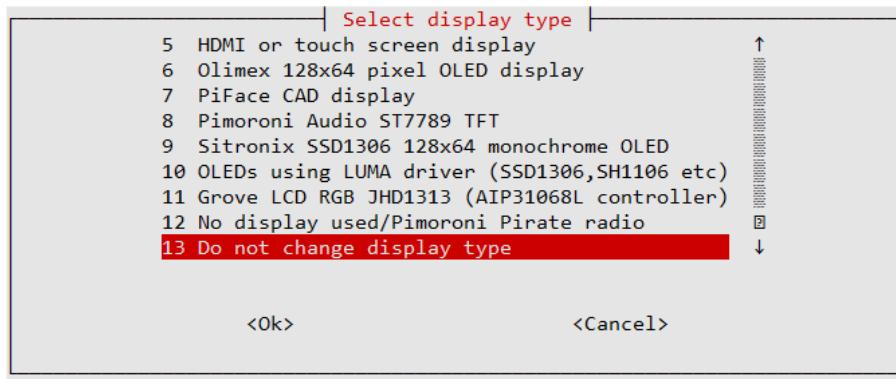


Figure 160 Configure radio - Display selection 2

Select the correct option for the display interface and confirm selection. Again, it is possible leave the configuration unchanged. There is also an option 10 (Scroll down) called "Do not change display type".

If option 5 – ‘HDMI or touch screen display’ was selected go to the section called *Configuring HDMI or Touchscreen* on page 117.

If options 6, 8, 9 or 10 were selected go to *Configuring OLEDs* on page 105.

If option 2, 3, 4 or 6 was selected then this will require the hex address to be configured. Otherwise, the program will skip the screens in the next section and go to the section called *Select the type of LCD display* on page 104.

If option 6 (Olimex OLED) was selected then the following will be displayed:

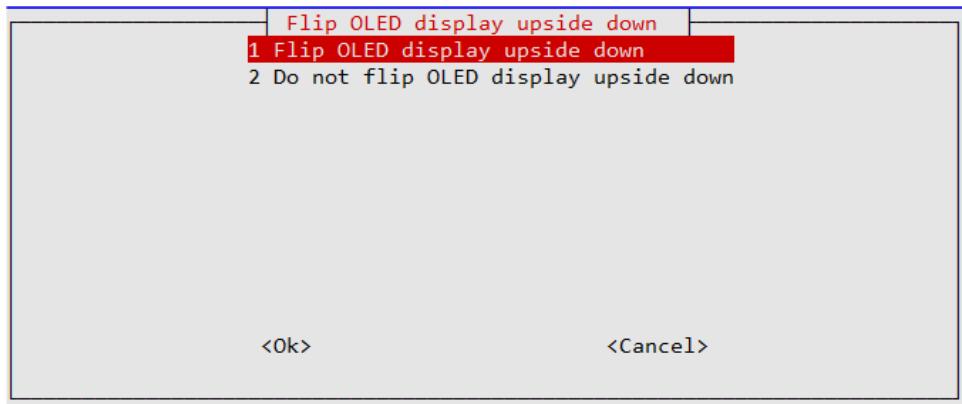


Figure 161 Olimex OLED flip display

This option sets the `flip_display_vertically` parameter in `/etc/radiod.conf` to **yes** or **no** and allows the Olimex OLED display to be flipped vertically. This option doesn't work with the ST7789TFT.

Configure the SPI Kernel Module

Skip this section unless installing PiFace CAD or devices using the ST7789 OLED display.

If installing the radio on PiFace CAD or the ST7789 OLED display it is necessary to enable the SPI interface. For example, if PiFace CAD was selected the following message will be seen:

```
The chosen display PiFace CAD requires the
SPI kernel module to be loaded at boot time.
The program will call the raspi-config program
Select the following options on the next screens:
  5 Interfacing options
  P4 Enable/Disable automatic loading of SPI kernel module

Press enter to continue:
```

Press enter and select option 1:

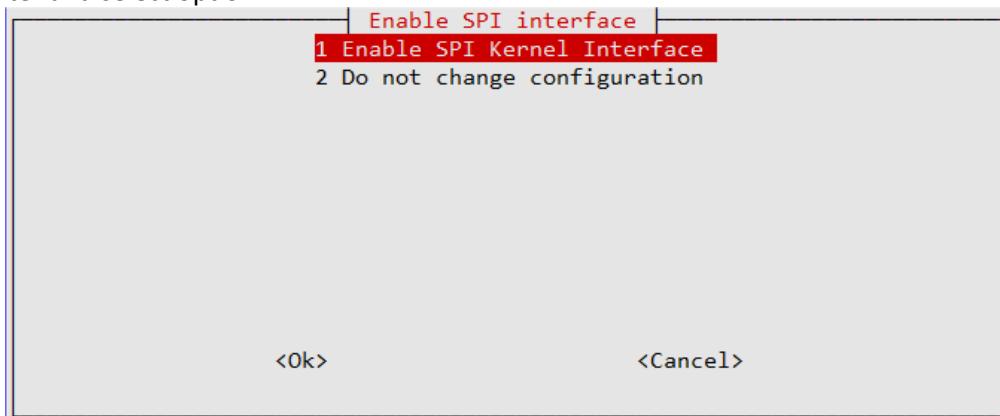


Figure 162 Enable SPI Kernel Module

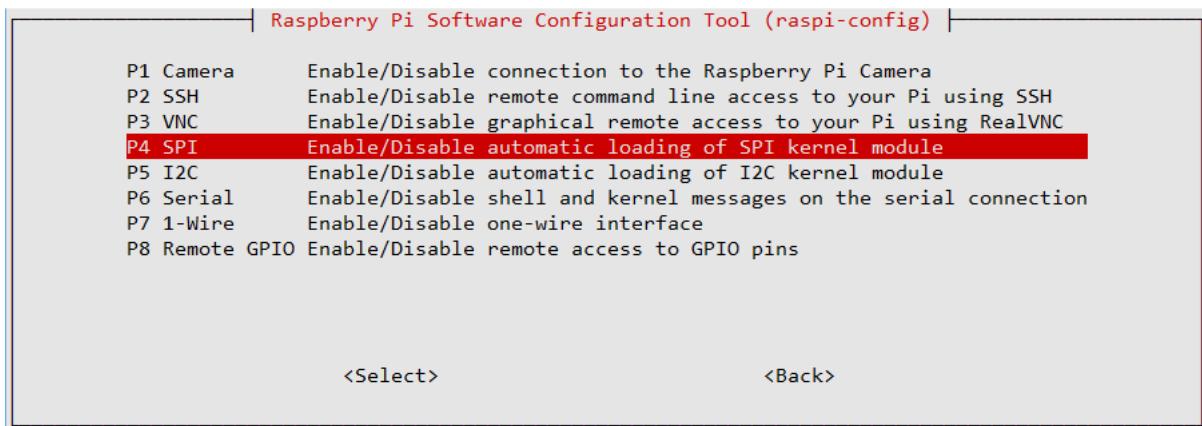
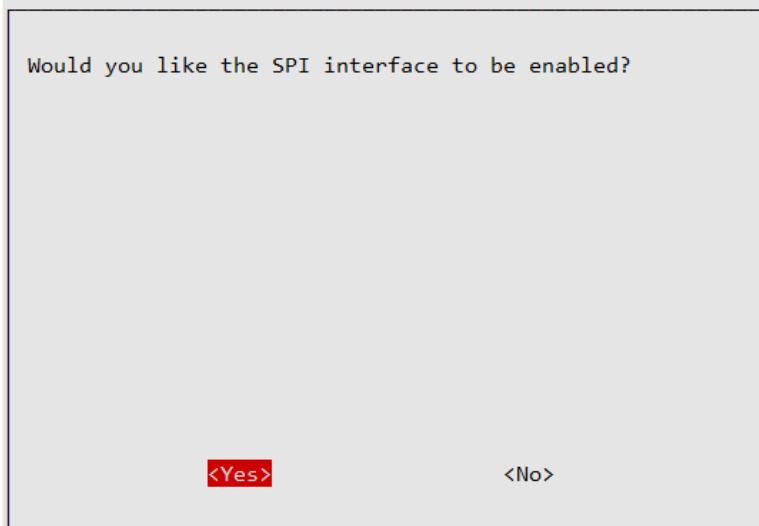
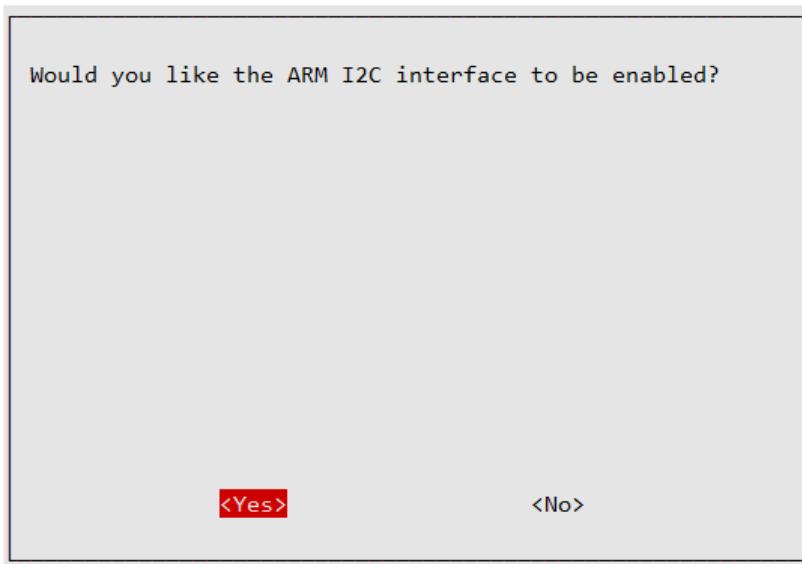


Figure 163 Enable SPI kernel module option

Select the option P4 SPI. The following screen is displayed:



Select Yes to enable the SPI kernel module. Select “Finish” to exit.



Select the type of LCD display

Skip this section if you are not using an LCD display directly connected to the GPIO pins. Confirm the selection and continue to the next screen to select the type of LCD display. This section is not relevant for a HDMI or touch screen.

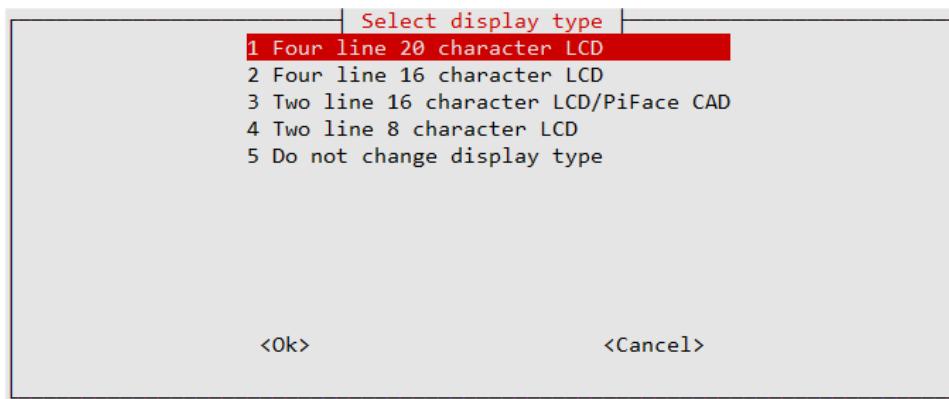


Figure 164 Configure radio - Display type selection

Select the type of display to be used and confirm the selection. The installation script asks if you wish to configure the audio device:

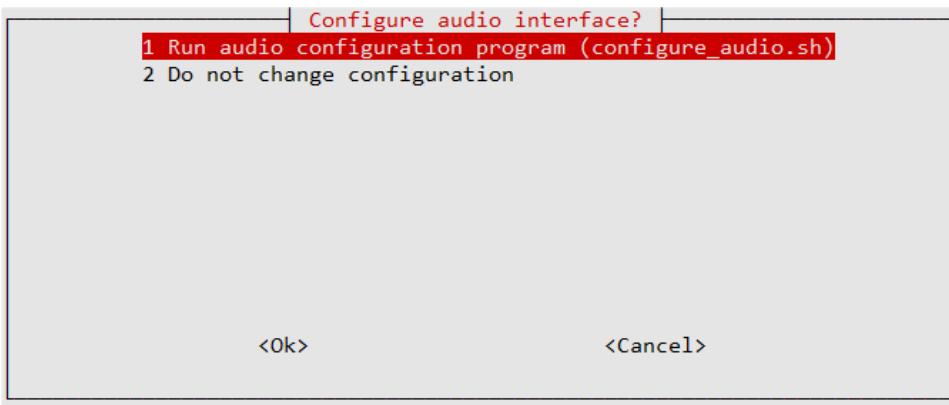


Figure 165 Configure radio audio output option

You should select option 1 to run the audio device configuration program.

If you selected option 1 then go to the section *Configuring the audio output* on page 109.

Configuring OLEDs

If not using an OLED display then go to the section *Configuring the audio output* on page 109.

There are four drivers available for OLED displays. These can have either the 4-wire **I2C** interface (+5v, GND, Data and Clock) or the **SPI** interface. They are selected by the **display_type** parameter in **/etc/radiod.conf**. This configuration

Table 17 OLED drivers

display_type	Interface	Lines	Driver	Option	Notes
ST7789TFT	SPI	5	st7789tft_class.py	8	Used by the Pimoroni Pirate Audio hat
OLED_128x64	I2C/SPI	5	oled_class.py	6	The driver automatically detects I2C or SPI
SSD1306	I2C	4	ssd1306_class.py	9	From version 7.2 onwards
LUMA	I2C	4	luma_class.py	10	From version 7.3 onwards

The Luma device supports multiple devices namely devices using SSD1306, SSD1309, SSD1325, SSD1331, SH1106 or WS0010 hardware driver chips. Only SSD1306 and SH1106 have been tested.

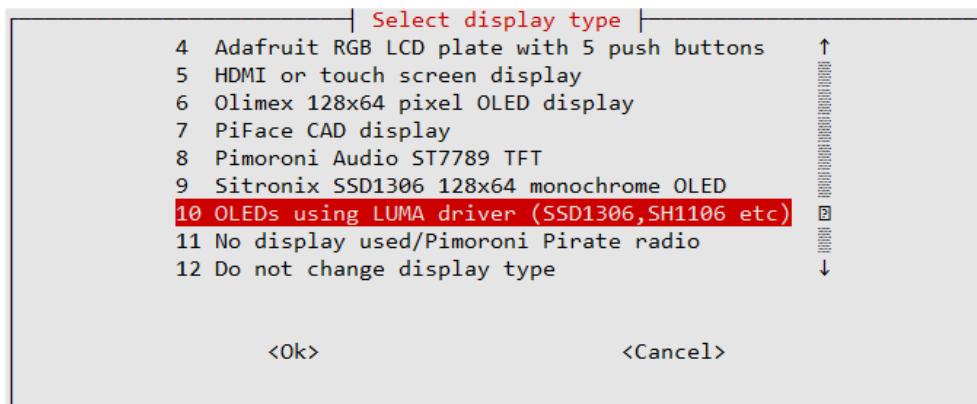


Figure 166 Selectin an OLED display

Select option 6, 8, 9 or 10 depending upon the type of OLED being configured.

If option 10 (Luma device driver) was selected the following screen will be displayed:

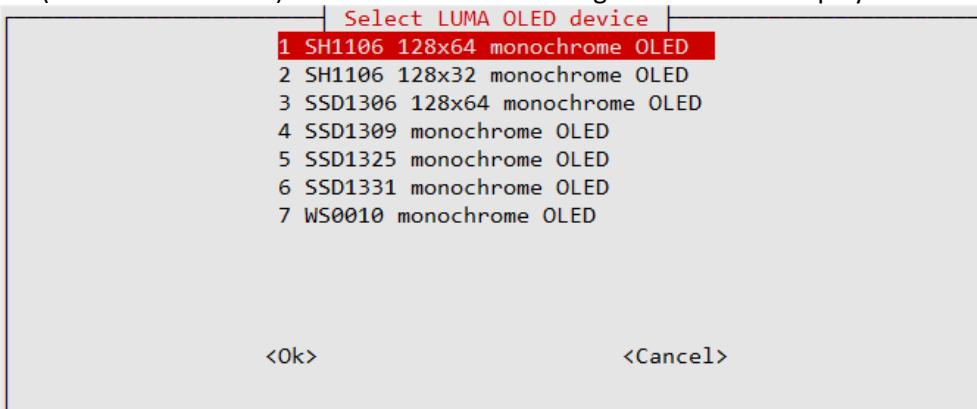


Figure 167 Luma devices

Select the device that your OLED is using.

If using devices with an I2C interface then select the appropriate I2C device address. OLED devices usually use address **0x3C** but check your device specification or run **i2c_detect.py -y 1** to display it.

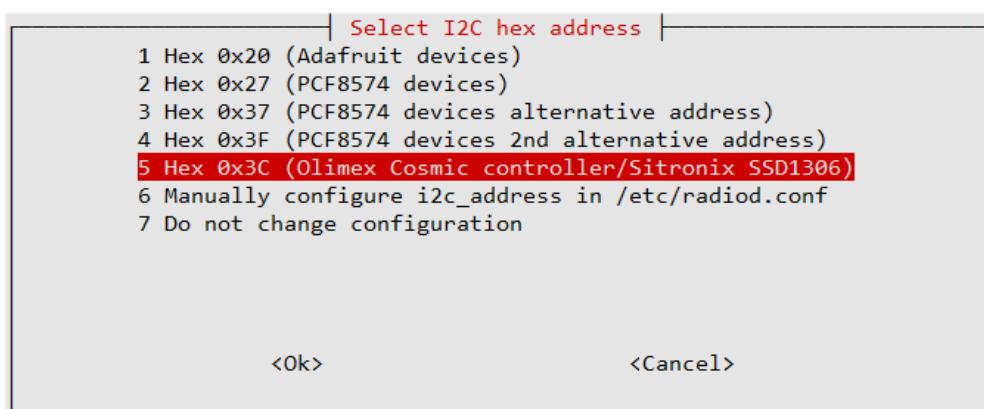


Figure 168 Selecting the OLED I2C address

After configuring I2C address go to the section *Configuring the audio output* on page 109.

Installing the HDMI or touch screen software

This section is only relevant if configuring an HDMI or touchscreen interface. If using an LCD display then skip this section. The following screen is displayed:

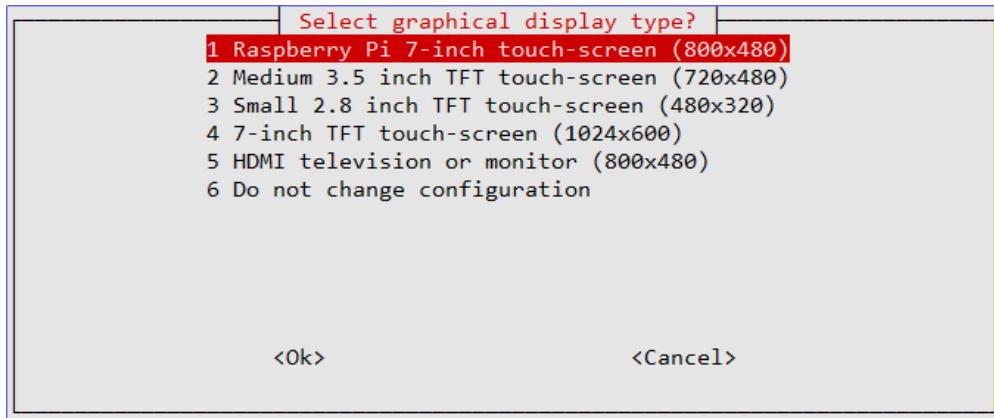


Figure 169 Touchscreen selection

Select the type of screen that is connected to the Raspberry Pi.

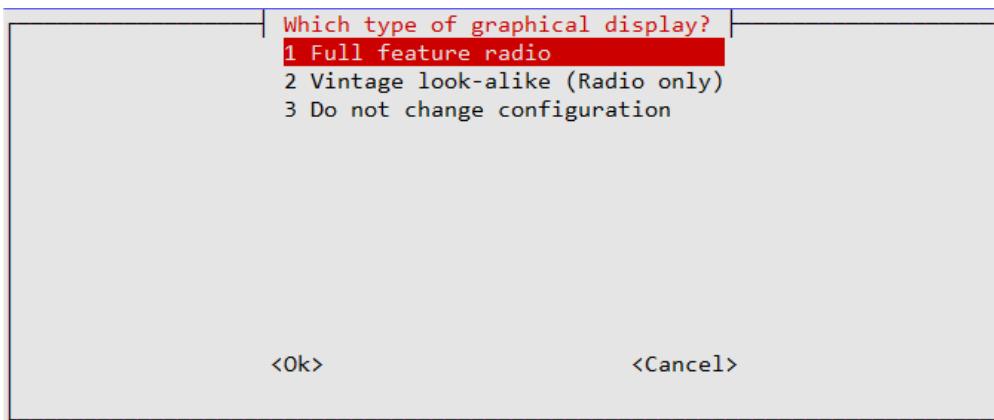


Figure 170 Selecting the type of radio display

Select which radio program to start-up. . See *Figure 2 Raspberry pi 7-inch touchscreen radio* and *Figure 4 Vintage tuning touch-screen radio* on page 5.

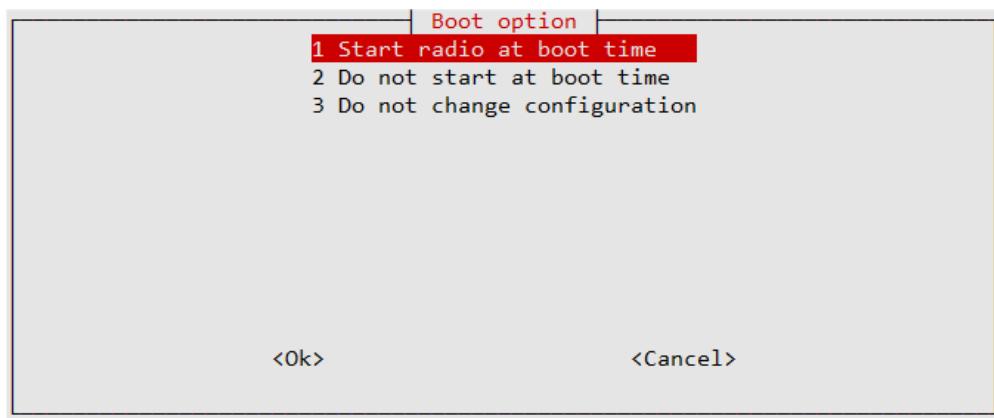


Figure 171 Configuring the HDMI or touch screen display startup

Normally select option 1 to automatically start the gradio.py program when the Graphical Desktop is loaded. This copies a desktop configuration to the file **/home/pi/Desktop/gradio.desktop** file.

There is also a similar file created called **vgradio.desktop** for the vintage graphical radio. For more information see the *Appendix A.3 X-Windows radio desktop files* on page 354354.

The installation script also copies the graphic screen configuration to **/etc/radiod.conf**. It also disables start-up of the **radiod** service which is only used for the LCD versions of the radio.

Now select the option to display the radio full screen (7-inch touchscreen) or in a desktop window (Large HDMI monitor or TV).

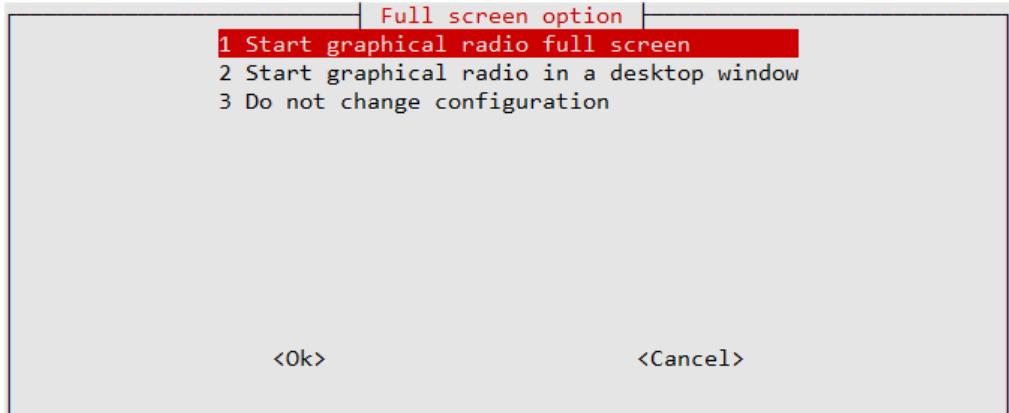


Figure 172 Configuring the graphical radio full screen

Configuration of the HDMI/Touch screen is shown in the section *Configuring the HDMI or Touch Screen* on page 176. Operation of the HDMI/Touch screen is shown in the section called *Operation of HDMI and touch screen displays* on page 200.

At the end of the radio configuration process the radio installation calls the **configure_audio.sh** script which then displays the screen shown below in Figure 173.



Currently it is not possible to configure anything but the onboard audio jack or HDMI when the **configure_audio.sh** script is called directly from the initial installation script.

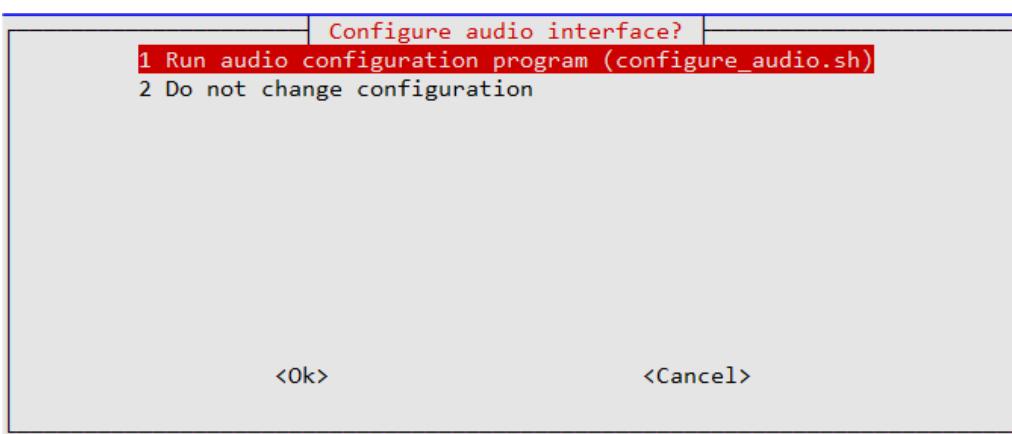


Figure 173 Audio configuration selection

If you wish to configure the on-board audio select option 1. If you wish to configure DAC, USB or Bluetooth devices then select option 2 to exit the program, then run the following command:

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

Configuring the audio output

If building a new radio start by using the Raspberry Pi on-board audio output jack. Leave configuring a digital audio card such as HiFiBerry or IQaudio until later unless you are using a Raspberry Pi Zero which doesn't have an on-board audio jack. In that case there is no choice but to configure the sound card. The package installation **configure_radio.sh** script will automatically run the **configure_audio.sh** script.

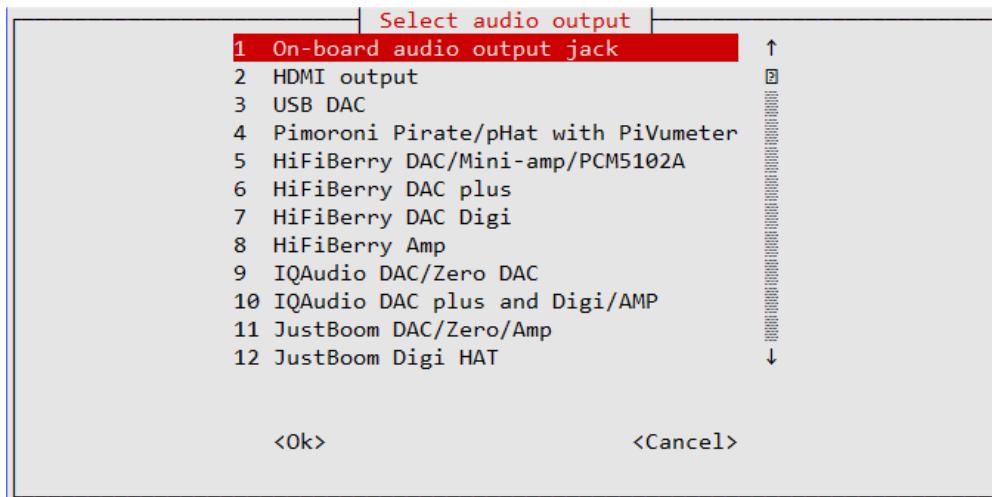
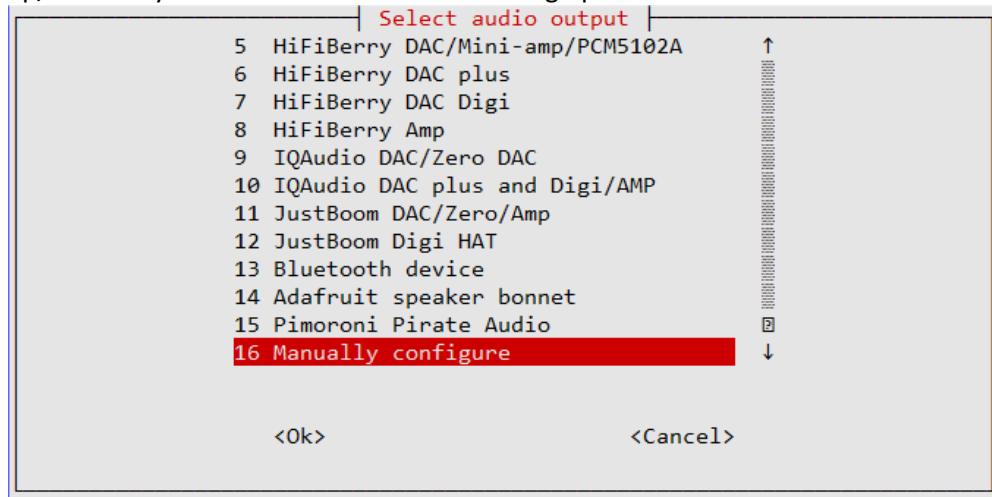


Figure 174 Selecting the audio output device

Use the Up/Down keys to scroll down to the remaining options:



Please note the scroll bar on the right side of the above screen. There are more options after option 12 (Bluetooth device). Use the Up/Down keys to scroll up and down.



This configuration program can be safely re-run at any time in the future. Change directory to **/usr/share/radio** and run **configure_audio.sh**. To do this run the following:

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

Testing the I2C interface.



Version 6.13 onwards comes with its own **SMBus** library (`smbus2`) in the `smbus2` sub-directory of the `radio` package.

The I2C interface should have already been enabled by the installation program. To test the I2C interface, carry out the following:

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

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

If you are using a revision 1 Raspberry Pi (Very old V1 boards) carry out the following:

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

This will search **/dev/i2c-0** (Very old v1 RPis) or **/dev/i2c-1** (Later RPi versions) 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**. The OLED 128x64 pixel display uses address **0x3C**. See Figure 175 *The I2C bus display using the i2cdetect program*.

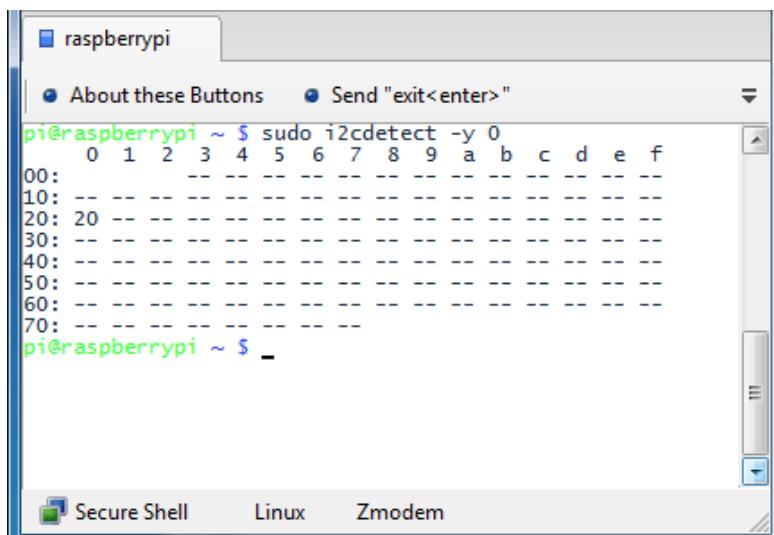


Figure 175 The I2C bus display using the `i2cdetect` program

If the following is seen instead then it is necessary to run enable the I2C module at boot time using `raspi-config`.

```
Error: Could not open file `/dev/i2c-1' or `/dev/i2c/1': No such file or directory
```

If problems with `i2cdetect` are still encountered, then edit the **/boot/config.txt** file using `sudo nano` and change the following line:

```
#dtparam=i2c_arm=on
```

Change to:

```
dtparam=i2c_arm=on
```

Also, the **i2c-dev** module must be added to the **/etc/modules** file.

```
i2c-dev
```

Reboot and retry the i2cdetect program.



Note: If the Arduino **PCF8574** backpack is using another address other than **0x27**, **0x37** or **0x3F** then you must modify the **i2c_address** parameter in **/etc/radiod.conf**. For example, if the backpack is using the address **0x2F** then modify the **i2c_address** parameter to match this as shown in the example below:

```
# The i2c_address parameter overrides the default i2c address. 0x00 = use default
# Some backpacks use other addresses such as 0x3F, then set i2c_address=0x3F
i2c_address=0x2F
```

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 *Chapter 9 -Troubleshooting* on page 240.

Installation logs

A log of the changes made by the radio configuration program will be written to the **/usr/share/radio/logs/install.log** file. For the audio configuration program changes will be written to the **/usr/share/radio/logs/audio.log** file.

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 *Chapter 9 -Troubleshooting* on page 240.

The radio daemon (LCD versions only) can be started and stopped with the **systemctl** command:

```
$ sudo systemctl start radiod
$ sudo systemctl stop radiod
```

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 systemctl disable radiod
```

To re-enable it:

```
$ sudo systemctl enable radiod
```

Apply patches to the radio software

DO NOT SKIP THIS SECTION.

Patches will be announced on Twitter at: https://twitter.com/bob_rathbone and should always be applied to the current software release.

Follow this Twitter feed for announcements about new patches. Patches can be viewed at http://www.bobrathbone.com/raspberrypi/pi_internet_radio.html

Patches take the form:

radiod-patch-<version>-<patch-number>.tar.gz

Where; <version> is the package version number, 7.5 in this case.

<patch-number> is the patch number from 1 onwards.

For example:

radiod-patch-7.5-1.tar.gz



Always check for the latest patches on the Web site. They will not be listed in this document.

To apply this patch (if it exists) run the following commands:

```
$ cd /usr/share/radio  
$ wget http://www.bobrathbone.com/raspberrypi/packages/radiod-patch-7.5-  
1.tar.gz  
$ tar -xvf radiod-patch-7.5-1.tar.gz
```

Modify the above command as necessary.

To see the details of the patch run the following command:

```
$ cat README.patch
```

Restart the radio software to activate the patches.



Any patch greater than 1 will also include all previous patches where relevant so it is not necessary to install previous patches. So for example patch 3 will include patches 1 and 2.



Do not apply any patches from a previous version of the software to the current version.
This will most likely cause the current software to malfunction.

All relevant patches in a particular version will normally be included in the next version of the software.

Installing Pimoroni Pirate Radio (pHat BEAT)

If not using the Pimoroni Pirate Radio then skip this section

To use the Pimoroni Pirate radio with **pHat BEAT** it is necessary to install the Pimoroni software pHat BEAT first. Do this before installing the Rathbone radio software.

Once the Pimoroni software is installed and tested it is then necessary to install the Rathbone radio software as shown in *Chapter 6 - Installing the radio Software* on page 95. Only the VU meter and pHat audio software is used by the Rathbone software. pHat uses the VLC radio not MPD.

The following instructions are based on the following link:

<https://github.com/pimoroni/phat-beat#full-install-recommended>

Run the following commands from the pi user home directory:

```
$ cd  
$ curl https://get.pimoroni.com/phatbeat | bash
```

This displays the following:

```
% Total    % Received % Xferd  Average Speed   Time     Time     Time  
Current                                         Dload  Upload   Total  Spent  Left  
Speed  
100 35982  100 35982      0       0    113k      0 --::-- --::-- --::--  
114k
```

This script will install everything needed to use
pHAT Beat

Always be careful when running scripts and commands copied
from the internet. Ensure they are from a trusted source.

This script should -- only be run on a Raspberry Pi with RPi OS --
other systems and SBCs are not supported and may explode!

If you want to see what this script does before running it,
you should run: 'curl https://get.pimoroni.com/phatbeat'

Note: pHAT Beat uses the I2S interface
bash: line 494: [: !=: unary operator expected

Do you wish to continue? [y/N] y

Answer yes (y).

The following is displayed:

```
pHAT Beat comes with examples and documentation that you may wish to  
install.  
Performing a full install will ensure those resources are installed,
```

```
along with all required dependencies. It may however take a while!  
Do you wish to perform a full install? [y/N] y
```

Again, answer yes(y).

The installation will take quite some time as it does a system upgrade and then builds the software as well as installing any other required packages so be patient.

Eventually the installation program displays:

```
All done!
```

Install the Rathbone Internet radio software

Do the following:

1. Now carry out the instructions shown in section *Install the Radio Daemon* on page 97.
 - a. Select option 5 Pimoroni pHat BEAT with own push buttons
 - b. Select option 1 40-pin wiring
 - c. No display used/Pimoroni Pirate radio
 - d. Audio configuration – Select 4 Pimoroni Pirate/pHat with PiVumeter
2. Since the Pirate Radio does not have a screen, you can optionally install **espeak** as shown in the section called *Installing the speech facility* on page 170 to hear choices when using the menu button.

Finally reboot the Raspberry Pi to start the radio.

```
$ sudo reboot
```



If no sound is heard from the Pirate radio, then use the volume up (+) button to increase the volume until sound is heard.

Installing the Pimoroni Pirate Audio

If you are not using either the Pimoroni Pirate Audio; skip this section.

Install the packages required by the Pimoroni Pirate Audio .

```
$ sudo apt -y install python3-rpi.gpio python3-spidev python3-pip python3-pil python3-numpy python3-pip git
```

Install setup tools

```
$ sudo pip3 install setuptools
```

Install the st7789 library if using the Pimoroni Pirate Audio. Not needed for the SSD1306 OLED.

```
$ sudo pip3 install st7789
```

Now carry out the instructions shown in *Chapter 6 - Installing the radio Software* on page 95. Select the following.

1. User interface - Select option 8 Pimoroni Audio with four push buttons
2. Select option 1 40-pin wiring
3. Display type - Select Pimoroni Audio ST7789 TFT

When configuring the audio output, select option 14 *Pimoroni Pirate Audio (HiFiBerry DAC)*. This will configure **/boot/config.txt** with the following lines.

```
dtoverlay=hifiberry-dac
```

and

```
dtparam=audio=off
```

More details about the configuration for the Pirate Audio are found in section *C.5 Pimoroni Pirate Audio wiring* on page 364.

Note 1: Button Y (Volume up) can be GPIO 24 instead of 20 on versions of the Pirate Audio card produced before January 2020.

The new settings in **/etc/radiod.conf** are normally:

```
up_switch=16  
down_switch=5  
left_switch=6  
right_switch=20
```

or on some variants.

```
right_switch=24
```

Manually edit **/etc/radiod.conf** if your Pirate Audio is using GPIO24.

Now carry out the instructions shown in section *Install the Radio Daemon* on page 97.

Installing the SSD1306 OLED driver

SSD1306 devices can either use the **ssd1306_class** driver or the **luma_class** driver configured for SSD1306 (The Luma driver supports multiple devices). The 0.9-inch SSD1306 OLED display works with either the **ssd1306_class** or **luma_class** driver.

To use the *Luma driver* see *Installing LUMA monochrome OLEDs* on page 119.

Otherwise follow the instructions below to use the SSD1306 driver. These install the **Adafruit_Python_SSD1306** driver software.

```
$ sudo apt install git  
$ git clone https://github.com/adafruit/Adafruit_Python_SSD1306.git  
$ cd Adafruit_Python_SSD1306  
$ sudo python3 setup.py install
```

Now carry out the instructions shown in section *Install the Radio Daemon* on page 97.

Configuring HDMI or Touchscreen

If using a touch-screen or HDM TV/Monitor add the following lines to **/boot/config.txt**.

```
hdmi_group=2  
hdmi_mode=4  
hdmi_cvt 800 480 60 6 0 0 0  
max_usb_current=1
```

If the screen upside-down then add the following line. Note: you must disable the *DRM VC4 V3D driver* as shown on page 84 for this to work.

```
# Rotate screen 180  
lcd_rotate=2
```

Now carry out the instructions shown in section *Install the Radio Daemon* on page 97.

Waveshare TFT device installation

If not using Waveshare devices skip this section.

Full instructions for installing and configuring the Waveshare TFTs will be found at:

https://www.waveshare.com/wiki/RPi_LCD_User_Guides

Below are the basic installation instructions extracted from the above site. First clone the Waveshare installation software.

```
$ cd  
$ git clone https://github.com/waveshare/LCD-show.git
```

It is now necessary to change to the LCD-show directory and run the correct installer for the type of Waveshare screen being installed. For example:

LCD28-show Waveshare TFT 2.8"
LCD35-show Waveshare TFT 3.5"
LCD35B-show Waveshare TFT 3.5" type B
LCD35C-show Waveshare TFT 3.5" type C

There are other installation scripts for different Waveshare LCD models in the

Below is an example of the installation for the Waveshare TFT 3.5" type C.

```
$ cd LCD-show/  
$ ./LCD35C-show
```

This command to rotate the screen 180 degrees and can be run at any time (reboot required)

```
$ ./LCD35C-show 180
```

However, do refer to the Waveshare documents as the above are only very basic instructions.

Edit **/boot/config.txt** and force the console to 720 x 480.

```
framebuffer_width=720  
framebuffer_height=480
```



Warning: The Waveshare installation script currently copies its own **boot.config** to **/boot/config.txt** and will overwrite any previous settings. For example, Bluetooth settings. The file it copies to config.txt (config-32c.txt-retropie) is also in **DOS** format. Use **dos2unix** to convert it back to Unix format. See below

```
$ sudo apt install dos2unix  
$ sudo dos2unix /boot/config.txt
```

If you are using Bluetooth then after running the Waveshare configurator and correcting the DOS format add the following two lines to the **/boot/config.txt** configuration file.

```
dtoverlay=pi3-miniuart-bt  
core_freq=250
```

Below is an example of Waveshare driver code added to the **/boot/config.txt** configuration file.

```
dtoverlay=waveshare35b:rotate=270  
hdmi_force_hotplug=1  
hdmi_group=2  
hdmi_mode=1  
hdmi_mode=87  
hdmi_cvt 480 320 60 6 0 0 0  
hdmi_drive=2  
display_rotate=0
```

Now carry out the instructions shown in section *Install the Radio Daemon* on page 97.

MHS 3.5-inch RPi display installation

If not using MHS devices skip this section.

Full instructions for installing and configuring the MHS devices will be found at:

http://www.lcdwiki.com/MHS-3.5inch_RPi_Display

Below are the basic installation instructions extracted from the above site. First clone the Waveshare installation software.

```
$ cd  
$ git clone https://github.com/goodfet/LCD-show.git
```

It is now necessary to change to the **LCD-show** directory and run the correct installer for the type of MHS screen being installed. For example:

MHS35-show MHS 3.5-inch RPi display
MHS55-show MHS 5.5-inch RPi display
MHS35B-show MHS 3.5-inch RPi display
LCD7B-show MHS 7-inch RPi display

There are other installation scripts for different MHS LCD models in the

Below is an example of the installation for the MHS TFT 3.5" RPi display

```
$ cd LCD-show/  
$ ./MHS35-show
```

This command to rotate the screen 180 degrees and can be run at any time (reboot required)

```
$ ./MHS35-show 180
```

However, do refer to the MHS documents as the above are only very basic instructions.

Edit **/boot/config.txt** and force the console to 720 x 480.

```
framebuffer_width=720  
framebuffer_height=480
```

Now carry out the instructions shown in section *Install the Radio Daemon* on page 97.

Installing LUMA monochrome OLEDs

The LUMA device driver (**luma_class.py**) interfaces monochrome OLEDs which are using SSD1306, SSD1309, SSD1325, SSD1331, SH1106 or WS0010 interface chips. In this release only SSD1206 and SH1106 OLEDs have been tested.

The devices test used 64x64 or 128x64 pixels resolution. Other resolutions may also work.

If using any of the above devices then first install the necessary dependencies.

Install dependencies:

```
$ sudo apt install python3-pip  
$ sudo -H pip3 install --upgrade luma.oled  
$ sudo apt install python3 python3-pil libjpeg-dev zlib1g-dev libfreetype6-  
dev liblcms2-dev libopenjp2-7 libtiff5 -y  
$ sudo pip3 install pathlib  
$ sudo pip3 install luma.core
```

For more information on the Luma driver see:

<https://luma-oled.readthedocs.io/en/latest/install.html>

<https://luma-oled.readthedocs.io/en/latest/api-documentation.html>

When the **configure_radio.py** program is run and the LUMA OLED driver is selected the **display_type** parameter in **/etc/radiod.conf** is set to **LUMA.<DEVICE>** where **<DEVICE>** is the chip being used.

For example, LUMA.SH1106 for OLEDs using the sh1106 chip

```
display_type=LUMA.SH1106
```

Now carry out the instructions shown in section *Install the Radio Daemon* on page 97.

Installing the Grove LCD RGB

The **Grove LCD** with RGB backlight is capable of displaying any colour wanted. For instance, the colour AQUA would be RGB (5,195,221). To turn the word “AQUA” into its RGB value the PIL (Python Image Library) package is required. Install it with the following command.

```
$ sudo apt install python3-pil
```

Now carry out the instructions shown in section *Install the Radio Daemon* on page 97.

Installing the Adafruit 2.5 and 3.5-inch TFT touchscreen

If not using the Adafruit TFT then skip this section.



If using the Adafruit TFT display this procedure needs to be carried out before attempting to install the Radio software. This procedure requires the desktop version of the Raspberry Pi OS



Note: There are two types of TFT touch-screen available screen available from Adafruit namely Capacitive or Resistive. The one used in this project is the Resistive type.

Plug the TFT screen into the Raspberry Pi 40 pin header. Depending upon the type of display the TFT screen will have either a 26-pin or 40-pin female header which plugs into the GPIO header.

The basic installation instructions can be found on the Adafruit Web site:

<https://cdn-learn.adafruit.com/downloads/pdf/adafruit-pitft-3-dot-5-touch-screen-for-raspberry-pi.pdf>

Or from the Bob Rathbone Web site:

<http://bobrathbone.com/raspberrypi/documents/Adafruit/adafruit-pitft-3-dot-5-touch-screen-for-raspberry-pi.pdf>

Software installation is also covered in the above manual.

Look at the section called *FBCP Install Commands* (HDMI Mirroring).

Below is the easy installation for the Adafruit TFT displays:

Install pre-requisite python3-pip (May already be installed)

```
$ cd ~  
$ sudo apt install -y git python3-pip
```

Download the Adafruit installer scripts:

```
$ sudo pip3 install --upgrade adafruit-python-shell click  
$ git clone https://github.com/adafruit/Raspberry-Pi-Installer-Scripts.git
```

Now run the setup scripts.

```
$ cd Raspberry-Pi-Installer-Scripts/  
$ sudo python3 adafruit-pitft.py --display=35r --rotation=90 --install-type=fbcp
```

Replace 35r with 28r for the 2.8-inch TFT. To flip the display, change the rotation from 90 to 270 and re-run the python3 script.

The above command adds the following to **/boot/config.txt** (may vary).

```
# --- added by adafruit-pitft-helper Sun Jul 11 11:15:00 2021 ---
hdmi_force_hotplug=1
dtparam=spi=on
dtparam=i2c1=on
dtparam=i2c_arm=on
dtoverlay=pitft35-resistive,rotate=90,speed=2000000,fps=20
# --- end adafruit-pitft-helper Sun Jul 11 11:15:00 2021 ---
hdmi_cvt=720 480 60 1 0 0 0
```

Make sure that the resolution specified matches your device. This is 720x480 in the above example. The supported resolutions are 480x320 or 720x480 pixels for the 2.8-inch and 3.5-inch TFT respectively. Smaller resolutions are not supported.

Edit **/boot/config.txt** and force the console to the same size with the same values.

```
framebuffer_width=720
framebuffer_height=480
```



It is very important that the resolution set in the above instructions matches your device and that they have been set to the same value. If not, the results will be unpredictable.

The TFT screen should already be calibrated but if calibration required refer to the section called *Resistive Touchscreen Manual Install & Calibrate* in the Adafruit in the Adafruit manual.

There is also an interactive installation script:

```
sudo python3 adafruit-pitft.py
```

See the Adafruit manual for the full procedure.

Reboot the Raspberry Pi.

```
$ sudo reboot
```

If everything has been correctly configured the Raspberry Pi desktop should be seen on the TFT display. If a HDMI monitor is also plugged in then the RPi desktop should also been seen mirrored on it.

Now carry out the instructions shown in section *Install the Radio Daemon* on page 97.

Using other touch screens

There are various other touch screens on the market but this version of the software does not support screen sizes of less the 480 x 320 pixels. There is a **screen_size** parameter in **/etc/radiod.conf** configuration file.

```
# Size is in pixels. Supported is 800x480 (7" screen) or 720x480(3.5"
screen)
# or 480x320 (2.8" or 3.5" screen)
screen_size=800x480
```

Another important aspect of screen size are the following parameters in **/boot/config.txt**.

```
framebuffer_width=1280  
framebuffer_height=720
```

Changing the above can force a console size. By default, it will be display's size minus overscan settings in **/boot/config.txt**.

If using the Elecrow 7-inch TFT Capacitive touch screen display then see *B.2 -Elecrow 7-inch touch-screen notes* on page 357.

Now carry out the instructions shown in section *Install the Radio Daemon* on page 97.

Installing PiFace CAD software

Version 7.4 onwards re-introduces **PiFace CAD**. The PiFace CAD software is no longer supplied as a package but as source which has to be compiled into a library (egg). It also requires the **SPI** interface to be enabled and also needs the **lirc** package to be installed.

First install git which is required to download the software from <https://github.com/piface>

```
$ sudo apt install git
```

Install pifacecommon

```
$ cd  
$ git clone https://github.com/piface/pifacecommon.git  
$ cd pifacecommon/  
$ sudo python setup.py install  
$ sudo python3 setup.py install
```

Install pifacecad:

```
$ cd  
$ git clone https://github.com/piface/pifacecad.git  
$ cd pifacecad/  
$ sudo python setup.py install  
$ sudo python3 setup.py install
```

Now install the IR remote control software as shown in the section called *Install the IR remote control software* on page 135. Put the settings for the IR Sensor to use GPIO25 and 0 for the IR remote Led (No LED available).

If you configured the IR sensor to use LIRC for **Buster** also carry out the instructions below.
Amend the LIRC import in the pifacecad **ir.py** Python script

```
$ cd /usr/local/lib/python3.9/dist-packages/pifacecad/  
$ cp ir.py ir.py.lirc
```

Edit **ir.py** using sudo nano or sudo vi and change:

```
import lirc
```

to

```
import pylirc
```

Run the Hello World demo using Python3:

```
$ python3
>>> import pifacecad
>>> cad = pifacecad.PiFaceCAD()      # create PiFace Control and Display
object
>>> cad.lcd.backlight_on()          # turns the backlight on
>>> cad.lcd.write("Hello, world!") # writes hello world on to the LCD
```

A full test can be run with the following:

```
$ cd /usr/share/radio
$ ./lcd_piface_class.py
```

Setting the mixer volume

All sound output goes through a mixer. After rebooting the Raspberry Pi, for the on-board output jack, run the **alsamixer** program:

```
$ alsamixer
```

The following screen is displayed:

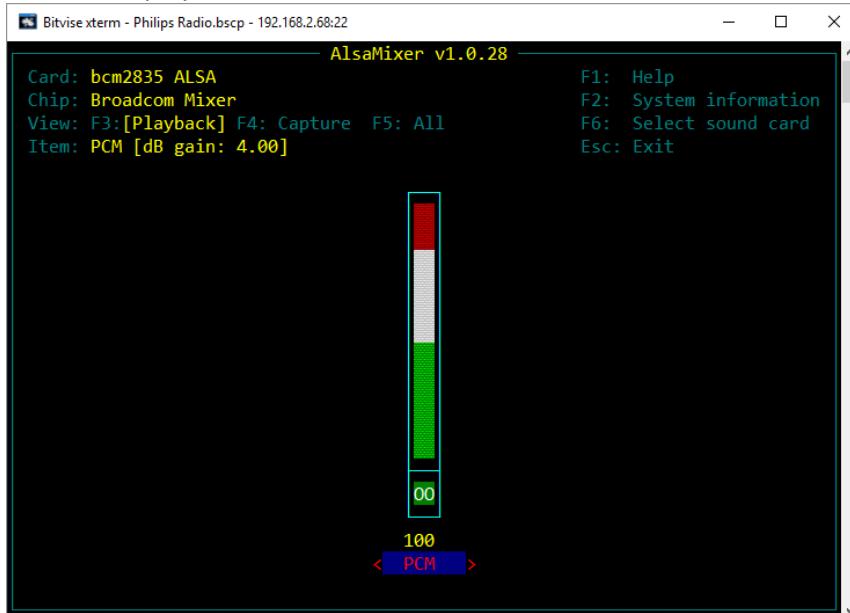


Figure 176 Basic Alsa sound mixer

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 124)
- Sound cards such as **HiFiBerry**, **IQaudio**, **JustBoom** and **Pimoroni pHat** DAC and DAC+ products (See page 125)
- Bluetooth speakers or headphones (See page 129).

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 devices namely the audio output jack and HDMI. **Card 1** is a USB PnP sound device.

To configure other sound devices run the **configure_audio.sh** utility.

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

Configuring a USB sound device

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

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

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

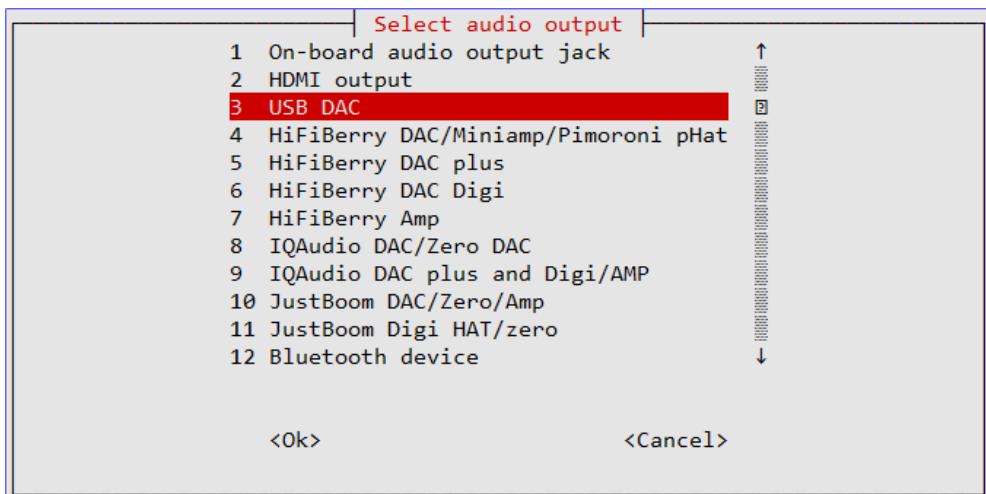


Figure 177 Configure USB DAC

Reboot when prompted. After rebooting the Raspberry Pi run the **alsamixer** program.

```
$ alsamixer
```

The following screen is displayed:

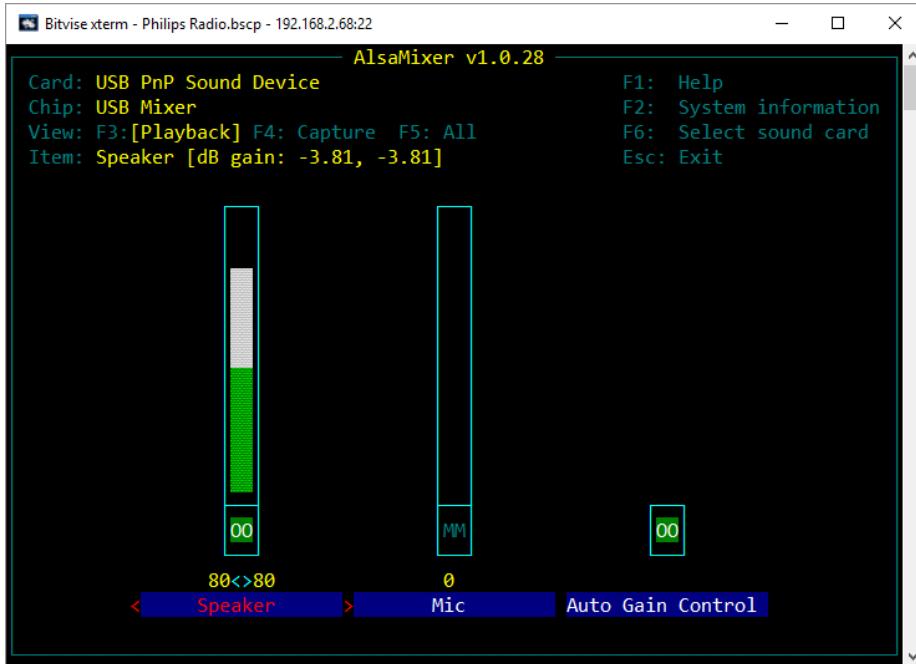


Figure 178 The USB PnP Alsa Mixer

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). Press **Esc key** or **Ctl Z** key to exit.

Configuring a Sound Card

This section covers configuration of add on DAC boards such as **HiFiBerry**, **IQaudIO** and **JustBoom DAC**, **DAC+** and Amplifier products. Older versions of the **HiFiBerry DAC** that used the 26-pin GPIO header are not supported.

To configure add on audio cards run the Run the **configure_audio.sh** utility:

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

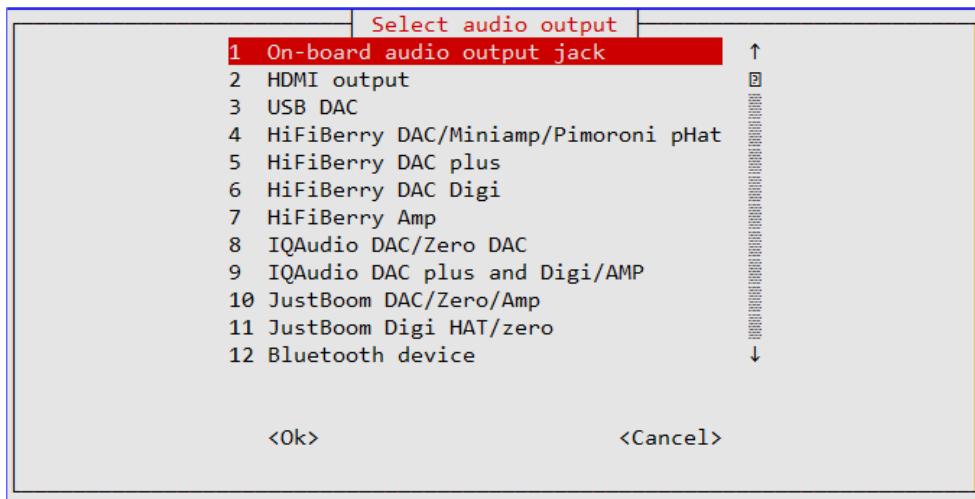
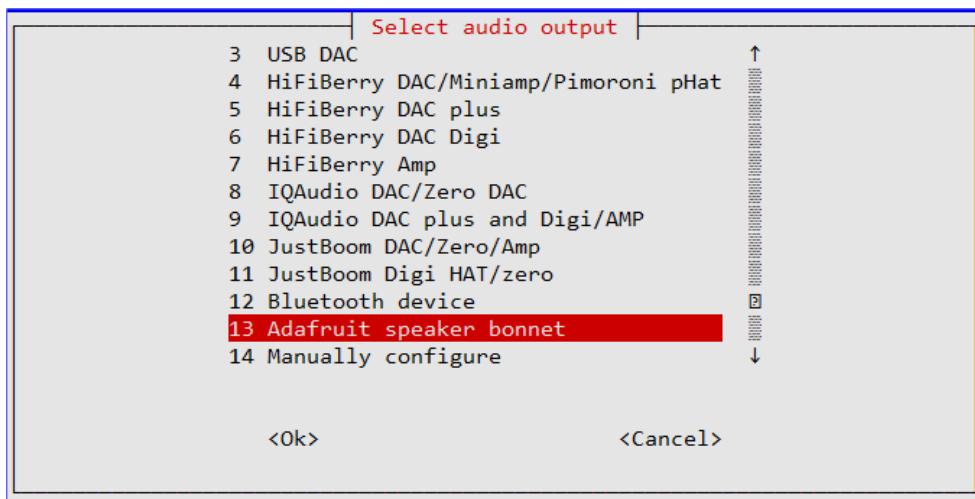


Figure 179 Configuring add-on DAC sound cards

More options are available by scrolling down with the down arrow key:



Select option for the DAC being used and press OK. Reboot when prompted by the next screen.
If using Bluetooth devices such as speakers or headphones then select option 12 Bluetooth device.

The Pimoroni pHat is compatible with HiFiBerry DAC (Not DAC+) and uses the same Device Tree (DT) overlay so select HiFiBerry DAC if using the pHat.

After rebooting run the **alsamixer** program.

```
$ alsamixer
```

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

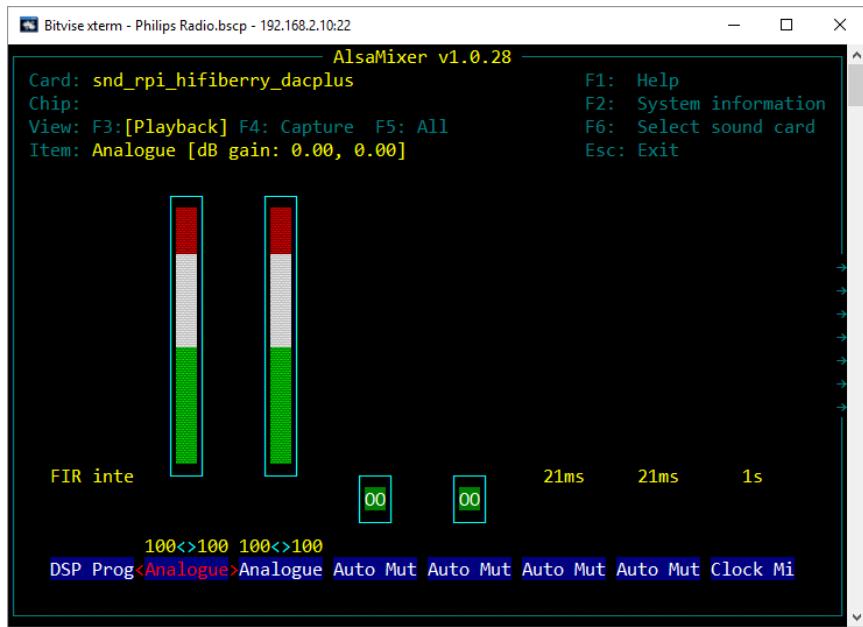


Figure 180 Set mixer analogue volume

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

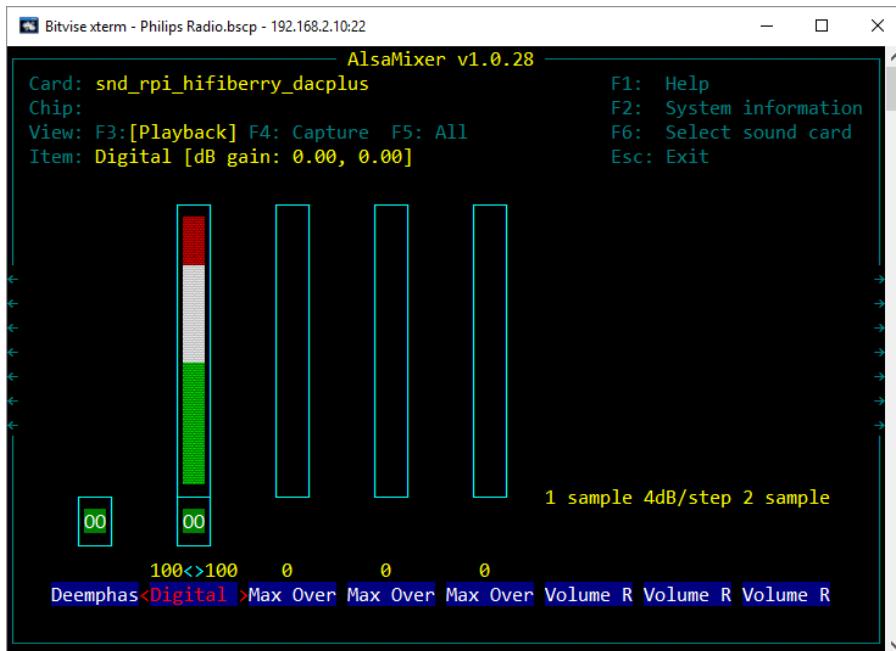


Figure 181 Set mixer digital volume

Configuring Allo Sound Cards

The Allo Piano, in particular, has two outputs, one for normal sound ranges and one for a sound woofer. The board comes pre-programmed as 2.1 for the Raspberry Pi. The subwoofer has a right and left output but is mono only.

However, it uses a second I2S channel on GPIO5 and mute signal on GPIO6 both used to control the woofer amplifier. This conflicts with the LCD signals **lcd_data4** and **lcd_data5** pins. Other Allo products are not affected as they only use one I2S channel. There are two ways to correct this.

Method 1

Edit **/etc/radiod.conf** and locate the LCD GPIO connection definitions.

```
# LCD GPIO connections for 40 pin version of the radio
lcd_select=7
lcd_enable=8
lcd_data4=5
lcd_data5=6
lcd_data6=12
lcd_data7=13
```

Change the **lcd_data4/5** assignments in **/etc/radiod.conf** to:

```
lcd_data4=26
lcd_data5=27
```

Wire LCD **data4/5** to GPIO 26 (physical pin 37) and GPIO 27 (physical pin 13).

Method 2

Configure the LCD to use an I2C backpack. This frees up 6 GPIOs including GPIO5 and 6. See *Construction using an I2C LCD backpack* on page 47.

Configuring Allo driver dtoverlay

Allo manufacture several audio products as well as the Piano 2.1 with woofer. Currently these cannot be configured via the radio installation program. It is necessary to load the correct **device tree overlay (dtoverlay)** by adding the following line for the corresponding Allo audio card the end of the **/boot/config.txt** file.

Allo Piano HIFI DAC

```
dtoverlay=allo-piano-dac-pcm512x-audio
```

Allo Piano 2.1 HIFI DAC with woofer

```
dtoverlay=allo-piano-dac-plus-pcm512x-audio
```

Allo Boss HIFI DAC / MINI BOSS HIFI DAC

```
dtoverlay=allo-boss-dac-pcm512x-audio
```

Allo DIGIONE

```
dtoverlay=allo-digione
```

Also disable the on-board output jack by amending it in the **/boot/config.txt** file.

```
dtparam=audio=off
```

Reboot the Raspberry Pi to load the Allo dtoverlay.

```
$ sudo reboot
```

Verification

Check, if the sound card is enabled with “aplay”. The output below will vary depending upon which Allo DAC has been selected:

```
$ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: PianoDAC [PianoDAC], device 0: PianoDAC multicodec-0 []
Subdevices: 0/1
Subdevice #0: subdevice #0
```

Connecting a Bluetooth device

Install the Bluetooth software

Usually, all necessary Bluetooth is installed for the full versions of Raspbian but may be missing for the Lite version. To make sure all required software is installed run the following:

```
$ sudo apt-get install pulseaudio pulseaudio-module-bluetooth
```

Add user pi to the bluetooth group:

```
$ sudo usermod -G bluetooth -a pi
```

Reboot the Raspberry Pi.

```
$ sudo reboot
```

Pairing a Bluetooth device

Switch on the Bluetooth speakers or headphones. Reboot the Raspberry Pi.

To pair your Bluetooth device run **bluetoothctl**. This will enter its own shell.

```
$ bluetoothctl
Agent registered
[bluetooth]#
```

Do not mistake the # prompt for the root (super-user) prompt. Put scanning on.

If you see the following message:

```
$ bluetoothctl
[bluetooth]# scan on
No default controller available
```

See *Cannot start Bluetooth control program (bluetoothctl)* on page 272.

Switch on the Bluetooth agent and set as default

```
[bluetooth]# agent on
Agent registered
[bluetooth]# default-agent
Default agent request successful
```

Switch on scanning.

```
[bluetooth]# scan on
Discovery started
[CHG] Controller DC:A6:32:05:36:9D Discovering: yes
[NEW] Device C0:48:E6:73:3D:FA [TV] Samsung Q7 Series (65)
[NEW] Device 00:75:58:41:B1:25 SP-AD70-B
```

When you see your Bluetooth speaker or headphones switch scan back off.

```
[bluetooth]# scan off
:
[CHG] Controller DC:A6:32:05:36:9D Discovering: no
Discovery stopped
```

In this example the device name is **SP-AD70-B** and has a Bluetooth ID of **00:75:58:41:B1:25**.

Now pair the device using its ID:

```
[bluetooth]# pair 00:75:58:41:B1:25
Attempting to pair with 00:75:58:41:B1:25
[CHG] Device 00:75:58:41:B1:25 Connected: yes
[CHG] Device 00:75:58:41:B1:25 UUIDs: 0000110b-0000-1000-8000-00805f9b34fb
[CHG] Device 00:75:58:41:B1:25 UUIDs: 0000110e-0000-1000-8000-00805f9b34fb
[CHG] Device 00:75:58:41:B1:25 UUIDs: 0000111e-0000-1000-8000-00805f9b34fb
[CHG] Device 00:75:58:41:B1:25 ServicesResolved: yes
[CHG] Device 00:75:58:41:B1:25 Paired: yes
Pairing successful
[CHG] Device 00:75:58:41:B1:25 ServicesResolved: no
[CHG] Device 00:75:58:41:B1:25 Connected: no
```

Now connect and trust the device:

```
[bluetooth]# connect 00:75:58:41:B1:25
Attempting to connect to 00:75:58:41:B1:25
[CHG] Device 00:75:58:41:B1:25 Connected: yes
Connection successful
[CHG] Device 00:75:58:41:B1:25 ServicesResolved: yes
```

Trust the new Bluetooth device

```
[SP-AD70-B]#trust 00:75:58:41:B1:25
[CHG] Device 00:75:58:41:B1:25 Trusted: yes
Changing 00:75:58:41:B1:25 trust succeeded
```

Connect the device

```
[SP-AD70-B] # connect 00:75:58:41:B1:25
Attempting to connect to 00:75:58:41:B1:25
[CHG] Device 00:75:58:41:B1:25 Connected: yes
Connection successful
```

Note that the Bluetooth prompt displays the name of the connected device. Now exit **bluetoothctl**.

```
[SP-AD70-B]# exit
$
```

You can also use **bluetoothctl** with commands following it from the normal pi user prompt.

```
$ bluetoothctl paired-devices
Device 00:75:58:41:B1:25 SP-AD70-B
```

The info command displays the state of the paired Bluetooth device.

```
[SP-AD70-B]# info 00:75:58:41:B1:25
Device 00:75:58:41:B1:25 (public)
  Name: SP-AD70-B
  Alias: SP-AD70-B
  Class: 0x00240404
  Icon: audio-card
  Paired: yes
  Trusted: yes
  Blocked: no
  Connected: yes
  LegacyPairing: no
  UUID: Audio Sink          (0000110b-0000-1000-8000-00805f9b34fb)
  UUID: A/V Remote Control   (0000110e-0000-1000-8000-00805f9b34fb)
  UUID: Handsfree            (0000111e-0000-1000-8000-00805f9b34fb)
```

The following displays all available commands:

```
$ bluetoothctl help
```

Testing Bluetooth

Once Bluetooth is running switch on the Bluetooth Speaker or device. The Bluetooth device should indicate that it is connected to the Raspberry Pi by some means (See your device manual). Now run the following:

```
$ aplay /usr/share/sounds/alsa/Rear_Right.wav
```

If things are working correctly, a spoken “Rear right” should be heard from the Bluetooth device. If not see *Bluetooth device no sound* on page 240 in Chapter 9 -Troubleshooting.

Configuring the radio and MPD software to use Bluetooth

Now re-run the **configure_audio.sh** configuration script and select Bluetooth.

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

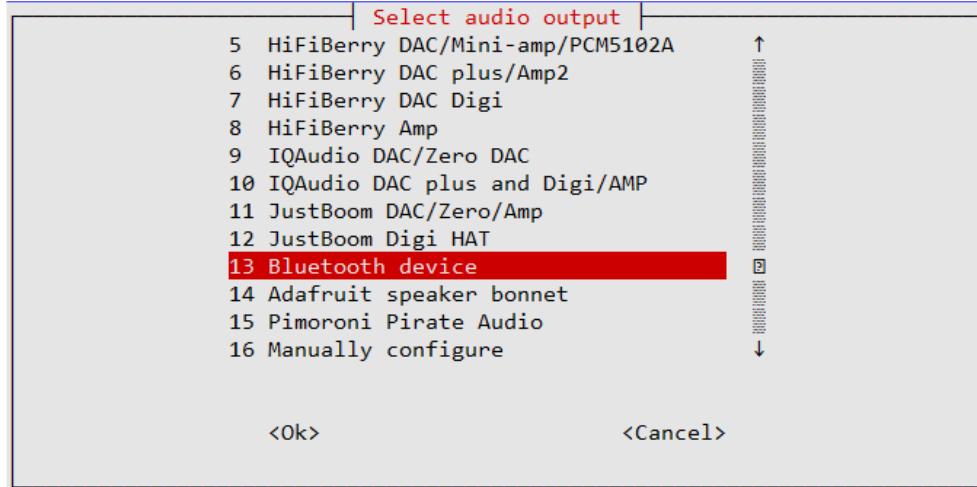


Figure 182 Configuring bluetooth devices

Reboot the Raspberry Pi.

```
$ sudo reboot
```

Using the alsamixer with Bluetooth devices

If using the Bluetooth speakers or headphones using **bluealsa**, use the following command to invoke the **alsamixer**.

```
$ alsamixer -D bluealsa
```

The following screen will be displayed:

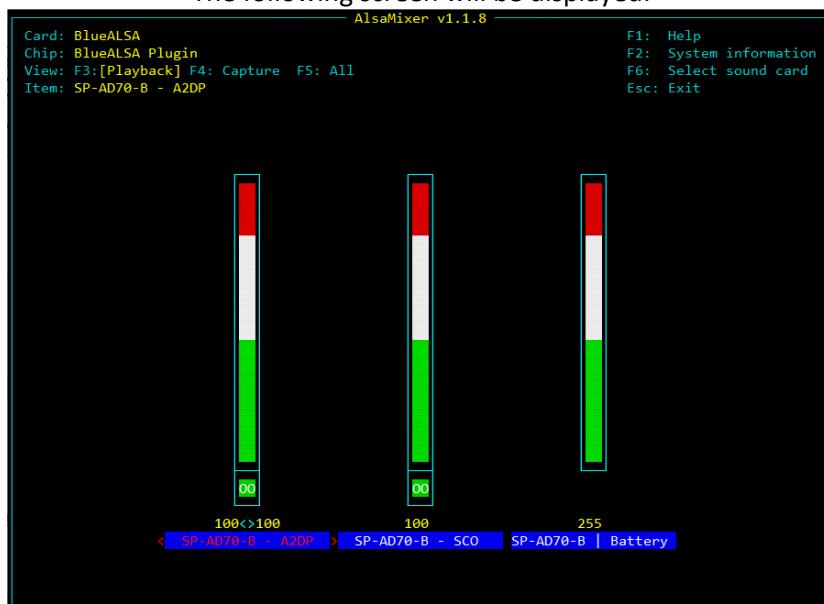


Figure 183 Alsamixer using Bluetooth devices

Controls from left to right, A2DP=Master volume, SCO=Messages volume (“Waiting for connection”, “connected”), Battery=Battery state messages (“low Battery”)

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 systemctl status mpd
● mpd.service - Music Player Daemon
  Loaded: loaded (/lib/systemd/system/mpd.service; disabled; vendor preset: enabled)
  Active: active (running) since Mon 2019-11-04 11:22:03 GMT; 6min ago
    Docs: man:mpd(1)
          man:mpd.conf(5)
          file:///usr/share/doc/mpd/user-manual.html
   Main PID: 1056 (mpd)
     Tasks: 7 (limit: 2061)
    Memory: 10.6M
      CGroup: /system.slice/mpd.service
              └─1056 /usr/bin/mpd --no-daemon
```

If the following is seen:

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

Start the MPD daemon.

```
$ sudo systemctl start mpd
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_stations.py** program as shown in the section called Creating new playlists on page 215.

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:0,0"
#  mixer_type  "hardware"
#  mixer_type  "software"
}
```

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.

Configuring MPD to use pulseaudio

In this version **pulseaudio** is removed due to the fact that for some unknown reason MPD has problems if **pulseaudio** is installed and MPD is configured to use the default ALSA system. Some DACs such as the Adafruit Bonnet require **pulseaudio**. MPD can be configured to use either the default **Alsa** sound system or the Pulse audio server. If you want to use **pulseaudio**, stop the radio and install **pulseaudio**:

```
$ sudo systemctl stop radiod
$ sudo apt install pulseaudio
```

Either re-run the **configure_radio.sh** program or manually change the **audio_output** type statement in **/etc/mpd.conf** to **pulse**. The ‘.’ character means output not shown.

```
audio_output {
    type      "pulse"
    name     "IQAudio DAC+"
    device   "hw:0,0"
    mixer_type  "software"
    :
}
```

Reboot the Raspberry Pi to restart the radio.

```
$ sudo reboot
```

See section *B.3 Sound card DT Overlays* on page 358 for supported/tested overlays.

Installing the Infra-Red sensor software

Before starting, the IR sensor needs to be wired to the correct GPIO pin. The following table shows the correct GPIO pin assignment for the IR receiver depending upon the hardware being used. Configuration commands shown later use the GPIO number shown in bold in the table below.

Table 18 IR Sensor Pin outs

Radio Type	Pin	GPIO	Type of Raspberry PI
Two- or Four-line LCD with Push Buttons	21	9	Any (No DAC)
Two- or Four-line LCD with Rotary encoders	21	9	Any (No DAC)
Two- or Four-line LCD with I2C backpack	21	9	Any (No DAC)
Adafruit RGB plate with push buttons	36	16	40-pin version only
All versions using DAC sound cards	22	25	40-pin version only
IQaudio Cosmic Controller and OLED display	22	25	40-pin version only



Note: If you have wired your IR sensor to a different IR sensor other than shown in the above table, simply use any of the above GPIO numbers and then later amend the `dtoverlay= gpio-ir,gpio_pin=x` in `/boot/config.txt` where `x` is the GPIO number you have used.

Install the IR remote control software

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

```
$ sudo apt update  
$ sudo apt upgrade
```

Run the IR remote installation program:

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

The following screen will be displayed. Select option 1 – Run IR Remote Control Configuration.

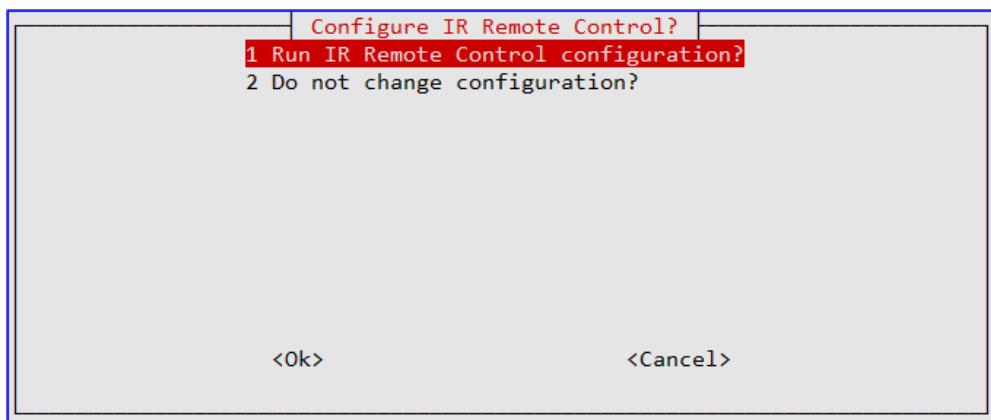


Figure 184 IR Remote Installation program

It is now necessary to select which GPIO is to be used for the IR sensor. This is either 9, 16 or 25.

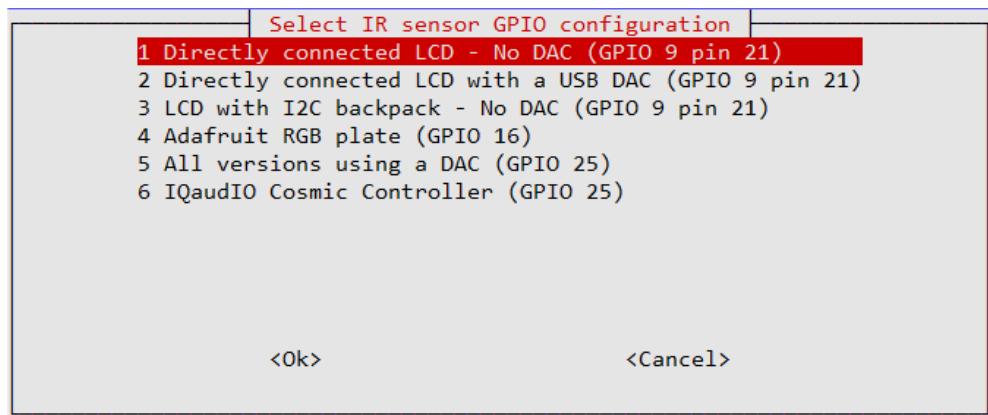


Figure 185 IR configuration IR sensor GPIO selection

Now select the Remote Activity LED GPIO. This is either GPIO 11, 13, 14 or 16.

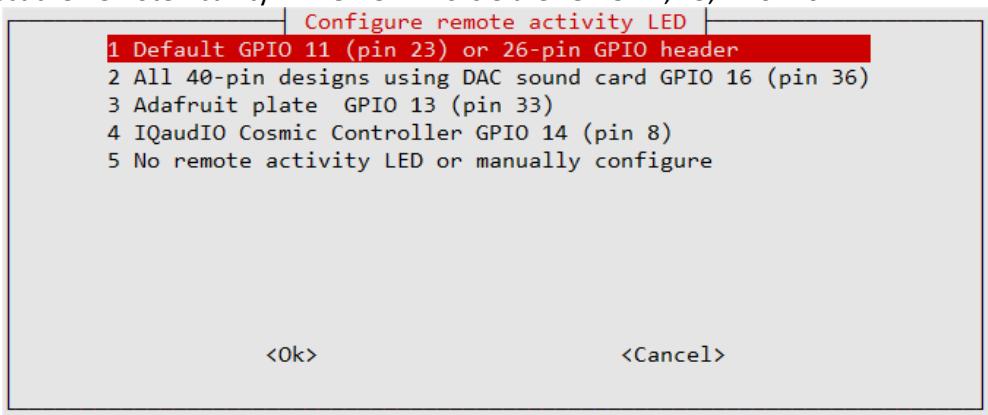


Figure 186 IR configuration Activity LED GPIO selection

The configurator program will now ask what IR software interface (service) you wish to use. There are two types a) Kernel events (new method) or b) LIRC (old method). These are explained in detail in the section *The remote control daemons* on page 314.

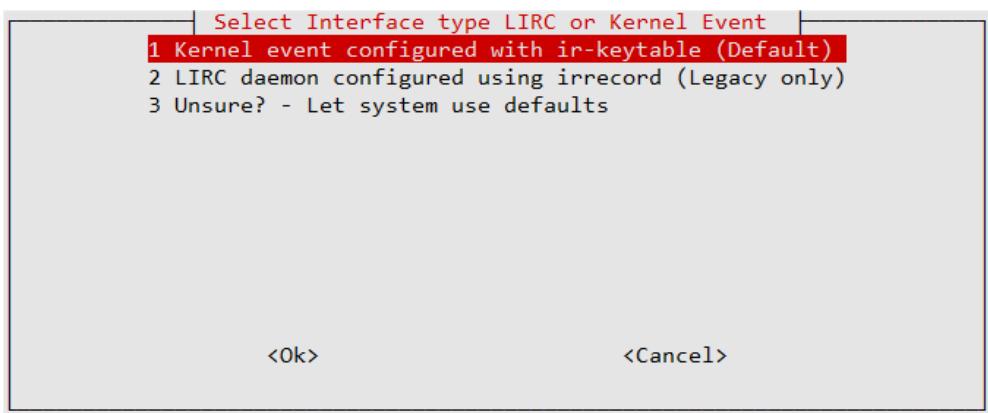


Figure 187 IR Remote Control Interface type selection

You should select option 1 unless you have a very good reason to use LIRC such as you already have configured the IR remote control to use LIRC and are upgrading from Version 7.4 or earlier. The LIRC method is very unreliable and difficult to configure and will be dropped when the next version of the Raspberry Pi OS is released (bookworm). This action will configure either the **ireventd** (Kernel events) or **irradiod** (LIRC daemon) service.

Once the Activity LED selection has been made the program will install LIRC and Kernel Event components and configure the **/boot/config.txt** file. It installs the **lirc**, **ir-keytable**, **lirc-compat-remotes** and **python3-evdev** packages.

The program will display the following instructions to complete the set-up process. In the following it gives the example for the Kernel events service.

```
Configuration of Kernel Event completed OK
Reboot the system and then run the following
to configure your IR remote control
    sudo ir-keytable -v -t -p rc-5,rc-5-sz,jvc,sony,nec,sanyo,mce_kbd,rc-
6,sharp,xmpir-keytable

Create myremote.toml using the scan codes from the ir-keytable program
output
See the example in /usr/share/radio/myremote.mytoml
Then copy your configuration file (myremote.toml) to /etc/rc_keymaps
    sudo cp myremote.toml /etc/rc_keymaps/.

Reboot the Raspberry Pi

A log of this run will be found in /usr/share/radio/install_ir.log
```

If you selected the LIRC option the following instructions will be displayed.

```
Configuration of LIRC completed OK
Reboot the system and then run the following
to configure your IR remote control
    sudo irrecord -f -d /dev/lirc0 ~/lircd.conf

Then copy your configuration file (myremote.conf) to /etc/lirc/lircd.conf.d
    sudo cp myremote.conf /etc/lirc/lircd.conf.d/.

Reboot the Raspberry Pi
```

The program adds the **gpio-ir** dtoverlay to the **/boot/config.txt** file for Buster. The **gpio_pin** varies with the selection you made.

```
dtoverlay= gpio-ir, gpio_pin=25
```

In the case of LIRC there is an error in the **paths.py** executable which deletes **_client.so** shared library if **lirc** is loaded as root user. It is necessary to edit the **paths.py** file and disable the offending lines. You can skip these instructions if you selected option 1 – Kernel events.

Edit **/usr/lib/arm-linux-gnueabihf/python3.9/site-packages/lirc/paths.py** and comment out the section which unlinks **_client.so** as shown below:

```
#if os.path.exists(os.path.join(HERE, '_client.so')):
#    try:
#        os.unlink(os.path.join(HERE, '_client.so'))
#    except PermissionError:
#        pass
```

Reboot the radio:

```
$ sudo reboot
```

After reboot check that the correct IR daemon is running. Use Ctrl-C to exit

If the Kernel Events interface was configured check **ireventd**.

```
$ systemctl status ireventd.service
● ireventd.service - Radio remote control daemon
  Loaded: loaded (/lib/systemd/system/ireventd.service; enabled; vendor
  preset: enabled)
    Active: active (running) since Sat 2023-06-24 12:22:28 BST; 1h 51min
      ago
      Main PID: 3055 (python3)
        Tasks: 1 (limit: 1599)
          CPU: 336ms
        CGroup: /system.slice/ireventd.service
                  └─3055 python3 /usr/share/radio/ireventd.py nodaemon
:
:
```

If the LIRC interface was configured check **irradiod**.

```
$ systemctl status lircd.service
● lircd.service - Flexible IR remote input/output application support
  Loaded: loaded (/lib/systemd/system/lircd.service; enabled; vendor
  preset: enabled)
    Active: active (running) since Fri 2022-02-11 19:53:31 GMT; 12h ago
  TriggeredBy: ● lircd.socket
    Docs: man:lircd(8)
           http://lirc.org/html/configure.html
  Main PID: 488 (lircd)
    Tasks: 2 (limit: 1592)
      CPU: 72ms
    CGroup: /system.slice/lircd.service
              └─488 /usr/sbin/lircd --nodaemon
:
:
```

You can ignore any warnings about the configuration at this stage.

Now test the remote control. Run the test program

```
$ sudo mode2 --raw -d /dev/lirc0
Using driver default on device /dev/lirc0
Trying device: /dev/lirc0
Using device: /dev/lirc0
Running as regular user pi
```

Press buttons on the remote control. Output similar to the following should be seen every time a button is pressed:

```
space 16777215
pulse 60
pulse 127838
space 1727845
space 1702207
pulse 4552
space 4431
```

```
pulse 631  
:
```



Note that some remote such as the Samsung remote have a select button or buttons (for example VCR and TV) which change the protocol or coding or both of the IR signal transmitted. Make sure that you use the same mode when setting up the remote control and using it avoid confusion.

Installing pylirc2 (Bullseye only)

Skip this section if installing LIRC on **Buster** or you selected option 1 – Kernel events. There is currently incompatibility between Python3 and **Pylirc**. To fix the problem you download the file **pylircmodule.c** and placed it in the folder **/home/pi/pylirc2-0.1**. Then **Pylirc** should be recompiled and installed.

Install required libraries

```
$ sudo apt-get install python3-dev liblircclient-dev
```

Go to the home directory

```
$ cd
```

Download the **pylirc2** source.

```
$ wget  
https://files.pythonhosted.org/packages/a9/e1/a19ed9cac5353ec07294be7blaefc8  
f89985987b356e916e2c39b5b03d9a/pylirc2-0.1.tar.gz
```

Please note the above instruction is all one line.

Extract the source using tar.

```
$ tar -xvf pylirc2-0.1.tar.gz
```

Change to the **pylirc2-0.1** directory

```
$ cd pylirc2-0.1
```

Replace the **pylircmodule.c** file:

```
$ rm pylircmodule.c  
$ wget https://raw.githubusercontent.com/project-  
owner/Peppy.doc/master/files/pylircmodule.c
```

Compile **pylirc2**

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

Install the **pylirc** share **Python3** distribution packages

For Bullseye

```
$ sudo mv /usr/local/lib/python3.9/dist-packages/pylircmodule.cpython-39-arm-linux-gnueabihf.so /usr/local/lib/python3.9/dist-packages/pylirc.cpython-39-arm-linux-gnueabihf.so
```

Please note the above instruction is all one line.

For Buster

```
$ sudo mv /usr/local/lib/python3.7/dist-packages/pylircmodule.cpython-37m-arm-linux-gnueabihf.so /usr/local/lib/python3.7/dist-packages/pylirc.cpython-37-arm-linux-gnueabihf.so
```

Configuring the IR remote control

There are two ways to do this.

1. Use new **ir-keytable** method using Kernel events (Recommended and most reliable method)
2. Create one using the **irrecord** utility program (Old LIRC method – largely redundant)

All new users should use the **IR-KEYTABLE** method to configure the IR remote control. You must use the method that you selected in *Figure 187 IR Remote Control Interface type selection* on page 136.

Method 1 – Using ir-keytable to create the IR configuration

This method uses the **ir-keytable** program. It supersedes the **LIRC** method. It has significant advantages over the older **LIRC** architecture.

You should also have already installed the general IR software as shown in *Installing the Infra-Red sensor software* on page 135.

The configuration files for **ir-keytable** have the **.toml** extension. These can be listed with the following command:

```
$ ls /lib/udev/rc_keymaps/
adstech_dvb_t_pci.toml    encore_enltv.toml      pixelview_002t.toml
af9005.toml                evga_indtube.toml    pixelview_mk12.toml
alink_dtu_m.toml          eztv.toml            pixelview_new.toml
:
```

This will list over 430 different remote controls and if you are lucky, it may find one that matches the one you are using. If not, create one using the following method.

```
$ sudo ir-keytable
```

This will list the available input devices of which there are several. The one that is of interest is **gpio_ir_recv**.

The **gpio_ir_recv** driver

```
Found /sys/class/rc/rc0/ with:
  Name: gpio_ir_recv
  Driver: gpio_ir_recv
```

```
Default keymap: rc-rc6-mce
Input device: /dev/input/event0
LIRC device: /dev/lirc0
Attached BPF protocols:
Supported kernel protocols: lirc rc-5 rc-5-sz jvc sony nec sanyo
mce_kbd rc-6 sharp xmp imon
Enabled kernel protocols: lirc rc-6
bus: 25, vendor/product: 0001:0001, version: 0x0100
Repeat delay = 500 ms, repeat period = 125 ms
```

Now run:

```
$ sudo ir-keytable -v -t -p rc-5,rc-5-sz,jvc,sony,nec,sanyo,mce_kbd,rc-6,sharp,xmp
```

Note the above is all one line. This will display the protocols available.

```
Found device /sys/class/rc/rc0/
:
Opening /dev/input/event0
Input Protocol version: 0x00010001
Protocols changed to rc-5 rc-5-sz jvc sony nec sanyo mce_kbd rc-6 sharp xmp
```

Now press a button on the remote control such as Volume Up. This will display the **scancode** for that button.

```
Testing events. Please, press CTRL-C to abort.
7800.012059: lirc protocol(necx): scancode = 0x833356
7800.012097: event type EV_MSC(0x04): scancode = 0x833356
7800.012097: event type EV_KEY(0x01) key_down: KEY_VOLUMEUP(0x0073)
7800.012097: event type EV_SYN(0x00).
7800.188028: event type EV_KEY(0x01) key_up: KEY_VOLUMEUP(0x0073)
7800.188028: event type EV_SYN(0x00).
```

Now create file called **myremote.toml** as shown in the example below. The name can be anything you wish, but it must end in the **toml** extension. The name field should be **myremote**. The protocol should be that shown above (**necx** in this example). Assign the scan code for each key as show in the above event. The variant field is normally the same as the protocol field but this will be indicated in the event output if it is different. There is an example **myremote.toml** file in the **/usr/share/radio/remotes** directory.



Note: There wasn't a **necx** protocol. The nearest one was **nec** so this was used.

```
[protocols]
name = "myremote"
protocol = "nec"
variant = "nec"
[protocols.scancodes]
0x833356 = "KEY_VOLUMEUP"
0x833357 = "KEY_VOLUMEDOWN"
0x833395 = "KEY_MUTE"
0x833358 = "KEY_CHANNELUP"
0x833359 = "KEY CHANNELDOWN"
0x833396 = "KEY_MENU"
0x833342 = "KEY_UP"
```

```

0x833343 = "KEY_DOWN"
0x833344 = "KEY_LEFT"
0x833345 = "KEY_RIGHT"
0x833341 = "KEY_OK"
0x833360 = "KEY_NUMERIC_0"
0x833361 = "KEY_NUMERIC_1"
0x833362 = "KEY_NUMERIC_2"
0x833363 = "KEY_NUMERIC_3"
0x833364 = "KEY_NUMERIC_4"
0x833365 = "KEY_NUMERIC_5"
0x833366 = "KEY_NUMERIC_6"
0x833367 = "KEY_NUMERIC_7"
0x833368 = "KEY_NUMERIC_8"
0x833369 = "KEY_NUMERIC_9"
0x833373 = "KEY_EXIT"

```



Note: The remote-control power on/off button must be called **KEY_EXIT** and not **KEY_POWER**. If **KEY_POWER** is used the event system issues its own system shutdown command instead of letting the radio decide the exit action, either system shutdown or stop radio. See the **exit_action** parameter in **/etc/radiod.conf** file.

Now write the new remote-control definition (myremote.toml) to the key table:

```

$ sudo ir-keytable -c -w myremote.toml
Read myremote table
Old keytable cleared
Wrote 11 keycode(s) to driver
Protocols changed to rc-5

```

Display the new table:

```

$ sudo ir-keytable -r
scancode 0x833341 = KEY_OK (0x160)
scancode 0x833342 = KEY_UP (0x67)
scancode 0x833343 = KEY_DOWN (0x6c)
scancode 0x833344 = KEY_LEFT (0x69)
scancode 0x833345 = KEY_RIGHT (0x6a)
scancode 0x833356 = KEY_VOLUMEUP (0x73)
scancode 0x833357 = KEY_VOLUMEDOWN (0x72)
scancode 0x833358 = KEY_CHANNELUP (0x192)
scancode 0x833359 = KEY CHANNELDOWN (0x193)
scancode 0x833360 = KEY_NUMERIC_0 (0x200)
scancode 0x833361 = KEY_NUMERIC_1 (0x201)
scancode 0x833362 = KEY_NUMERIC_2 (0x202)
scancode 0x833363 = KEY_NUMERIC_3 (0x203)
scancode 0x833364 = KEY_NUMERIC_4 (0x204)
scancode 0x833365 = KEY_NUMERIC_5 (0x205)
scancode 0x833366 = KEY_NUMERIC_6 (0x206)
scancode 0x833367 = KEY_NUMERIC_7 (0x207)
scancode 0x833368 = KEY_NUMERIC_8 (0x208)
scancode 0x833369 = KEY_NUMERIC_9 (0x209)
scancode 0x833373 = KEY_EXIT (0x74)
scancode 0x833395 = KEY_MUTE (0x71)
scancode 0x833396 = KEY_MENU (0x8b)
Enabled kernel protocols: lirc nec

```

Now test it:

```

$ ir-keytable -t
7427.808043: lirc protocol(necx): scancode = 0x833356

```

```
7427.808079: event type EV_MSC(0x04): scancode = 0x833356
7427.808079: event type EV_KEY(0x01) key_down: KEY_VOLUMEUP(0x0073)
7427.808079: event type EV_SYN(0x00).
7427.864040: lirc protocol(necx): scancode = 0x833356 repeat
7427.864075: event type EV_MSC(0x04): scancode = 0x833356
7427.864075: event type EV_SYN(0x00).
```



Note: If you press a key that you did not previously configure, it will display the scan code but it will not have a key name such as KEY_VOLUMEUP.

If all is OK it is now necessary to make the changes permanent when the RPi is rebooted.

Copy the newly created **myremote.toml** to **/etc/rc_keymaps/**

```
$ sudo cp myremote.toml /etc/rc_keymaps/myremote.toml
```

Reboot the Raspberry pi

```
$ sudo reboot
```

After reboot check that the IR key table is OK

```
$ ir-keytable -r
scancode 0x833341 = KEY_OK (0x160)
scancode 0x833342 = KEY_UP (0x67)
scancode 0x833343 = KEY_DOWN (0x6c)
scancode 0x833344 = KEY_LEFT (0x69)
scancode 0x833345 = KEY_RIGHT (0x6a)
scancode 0x833356 = KEY_VOLUMEUP (0x73)
scancode 0x833357 = KEY_VOLUMEDOWN (0x72)
scancode 0x833358 = KEY_CHANNELUP (0x192)
scancode 0x833359 = KEY_CHANNELDOWN (0x193)
scancode 0x833395 = KEY_MUTE (0x71)
scancode 0x833396 = KEY_MENU (0x8b)
: {Rest of output not shown}
```



Note: If you called your configuration anything else other than **myremote.toml** you not only need to change the previous instructions to use that name but also edit it in **/etc/radiod.conf** and change the **keytable** parameter to match the name you chose.

Edit **/etc/radiod.conf** and find the keytable entry and change the name to match the name you used.

```
# ireventd daemon keytable name
keytable=myremote.toml
```

This is necessary as the **ireventd.service** (**ireventd.py**) reads this entry to load they key table at the start of its run.

Now reboot the system and test the remote control:

```
$ reboot
```

Operation

The keys should work as expected, for example, Volume Up/Down, Mute and Channel Up/down. Pressing any of the numeric keys will cause the radio to play the station or track number entered. For example, press 5 and the 5th station in the playlist will play. Press 128 within two seconds and the radio will jump to the 128th station or track (if it exists). If it doesn't exist it will jump to station 1.

You can check the status of the **ireventd.service** using **systemctl**.

```
$ systemctl status ireventd.service
● ireventd.service - Radio remote control daemon
  Loaded: loaded (/lib/systemd/system/ireventd.service; enabled; vendor
  preset: enabled)
    Active: active (running) since Wed 2023-06-14 10:44:28 BST; 4h 43min
      ago
      Main PID: 541 (python3)
        Tasks: 1 (limit: 1599)
          CPU: 686ms
        CGroup: /system.slice/ireventd.service
                  └─541 python3 /usr/share/radio/ireventd.py nodaemon

Jun 14 10:44:28 bobrath systemd[1]: Started Radio remote control daemon.
:
```

Testing the ireventd daemon

If the **ireventd.service** isn't running then it is possible to test it as follows:

First stop the service.

```
$ sudo systemctl stop ireventd.service
```

Run the **ireventd.py** software from the command line. For example:

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

This will display the usage instructions:

```
This program must be run with sudo or root permissions!
Usage: sudo ./irradiod.py start|stop|status|nodaemon|flash|config|send <KEY>
```

The following command flashes the activity LED six times if fitted.

```
$ sudo ./irradiod.py flash
```

If the service is crashing you can test it using the following command to run it as a foreground process:

```
$ sudo ./ireventd.py nodaemon
```

If you see the following:

```
pidfile /var/run/ireventd.pid already exist. Daemon already running?
```

Run

```
$ sudo ./ireventd.py stop
```

If the program is running normally, it displays the following (pid and GPIO number will vary):

```
IR Remote control listener running pid 1847
Using IR events architecture
Flashing LED on GPIO 16
Read myremote table
Old keytable cleared
Wrote 22 keycode(s) to driver
Protocols changed to nec
:
/dev/input/event0 gpio_ir_recv gpio_ir_recv/input0
Listening for IR events:
```

Pressing a key on the remote-control will display the key that was pressed. The radio program will respond with 'OK'. For example, the following will be displayed if the Channel UP key is pressed.

```
KEY_CHANNELUP
OK
```

It can also be used to send a valid test key or command to the radio program.

```
$ sudo ./irradiod.py send
Usage: ./irradiod.py send <KEY>
Where <KEY> is a valid IR_KEY
    KEY_VOLUMEUP,KEY_VOLUMEDOWN,KEY_CHANNELUP,KEY CHANNELDOWN,KEY_MENU
    KEY_UP,KEY_DOWN,KEY_LEFT,KEY_RIGHT,KEY_OK,KEY_INFO,KEY_MUTE,KEY_EXIT
    PLAY_<n> Where <n> is 1 to 999, play station/track n
```

For example, to play station 12.

```
$ sudo ./irradiod.py send PLAY_12
```

The status command:

```
$ sudo ./ireventd.py status
Remote control running pid 1975
```

Finally, the IR configuration can be displayed.

```
$ sudo ./ireventd.py config
Remote Control daemon configuration
-----
LED = GPIO 16
HOST = localhost
PORT = 5100
LISTEN = 5100
dtoverlay= gpio-ir, gpio_pin=25
Key map: /etc/rc_keymaps/myremote.toml
```

Method 2 – Use irrecord to create the IR configuration

This was the original method to configure IR remote controls but hasn't worked reliably since the IR driver was moved into the kernel. You can either try to download a configuration for your remote control if it exists, or create one using the **irrecord** method.



Note: This method is highly unreliable. Unless you already have a configuration file for your remote control or you manage to download one you are strongly advised to use *Method 1 – Using ir-keytable to create the IR configuration* on page 140.

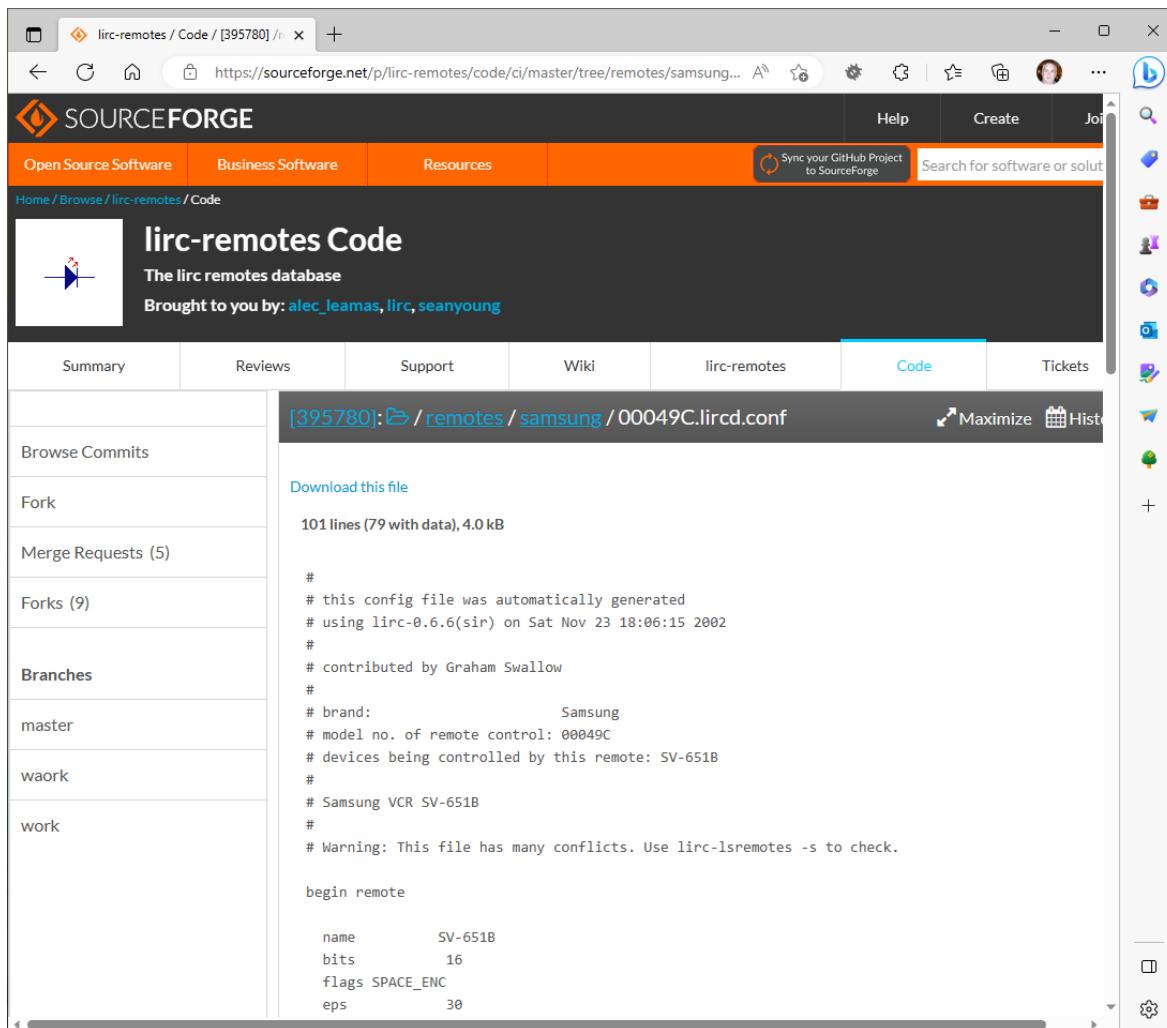
Download an LIRC configuration from Sourceforge

It may be possible to download a ready-made configuration from **sourceforge**. Go to <http://lirc-remotes.sourceforge.net/remotes-table.html>

lircd.conf file	Supported remotes	Timing Raw lircmd.conf
2wire/2wire_lircd.conf	2wire	✓ No
3m/MP8640_lircd.conf	MP8640	✓ No
abit/AU10_lircd.conf	ABIT_WINDVD	✓ No
abs/8776_lircd.conf	ABS_8776	✓ No
accesshd/DTA1080U_lircd.conf	AccessHD_DTA1080U	✓ No
accessmedia/ThinBox_lircd.conf	AccessMedia_ThinBox	✓ No
acer/AT3201W_lircd.conf	Acer_AT3201W	✓ No
acer/Aspire_6530G_lircd.conf	Acer_Aspire_6530G_MCE	✓ No
acer/RC-17DE0_irman_lircd.conf	ACER_RC-17DE0	✓ No
acer/RC-802_lircd.conf	Acer_RC-802	✓ No
aconatic/AN-2121_DRS_lircd.conf	Aconatic_AN-2121_DRS	✓ No
acorp/acorp_878y_lircd.conf	Acorp_878	✓ No
adaptec/AVC-2410_lircd.conf	AVC-2410	✗ No
adb/I-CAN_3000_lircd.conf	ADB_ICAN3000	✓ No
admiral/G0014AJ_lircd.conf	admiral-G0014AJ	✓ No
ads/Instant_TV_DVB-T_lircd.conf	ADS_Tech_Instant_TV_DVB-T	✓ No
ads/PTV-334_irman_lircd.conf	adstech	✗ No
ads/PTV-350_lircd.conf	ADS_Instant_TV_PCI	✓ No
adstech/usbx-707_lircd.conf	ADSTech_USBX-707	✗ No
advanced_acoustic/MAP-305II_DAI.lircd.conf	Advance_Acoustic_MAP-305II/DAI	✓ No
aim/RC126_lircd.conf	AIM-RC126	✓ No

Figure 188 LIRC Remotes Database

Find your remote control in the list and click on it. The configuration will be displayed as shown in the example below:



Click on the caption **Download this file** and save it to your PC and copy it to the Raspberry Pi. Alternatively copy the “Download this link” caption link from the caption and use **wget** to download it directly to the Raspberry Pi.

For example:

```
$ wget https://sourceforge.net/p/lirc-remotes/code/ci/master/tree/remotes/samsung/3F14-00048-180.lircd.conf?format=raw
```

This creates a file called **3F14-00048-180.lircd.conf?format=raw**. Rename it so that the file ends in the name **.conf**

```
mv 3F14-00048-180.lircd.conf?format=raw 3F14-00048-180.lircd.conf
```

or use any name you wish as long as it ends in **.conf**

```
$ mv 3F14-00048-180.lircd.conf?format=raw myremote.lircd.conf
```

Now copy it to the **/etc/lirc/lircd.conf.d** directory

```
$ sudo cp myremote.lircd.conf /etc/lirc/lircd.conf.d/.
```

Change the permissions of all configuration files in **/etc/lirc/lircd.conf.d/** directory to read.

```
$ sudo chmod og+r /etc/lirc/lircd.conf.d/*.conf
```

Create configuration using irrecord

To create your own configuration file run the configuration **irrecord** program.

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

If you see the following:

```
irrecord: could not open /dev/lirc0
:
```

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 program asks for a name for the remote control. Enter **myremote** or any other name you wish (no spaces or special characters).

```
Enter name of remote (only ascii, no spaces) :myremote
```

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 19 Remote Control Key names and functions*.

Table 19 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
KEY_EXIT	Exit/shutdown	Exit/shutdown	Exit/shutdown	Exit/shutdown

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



Note: The **KEY_OK** and **KEY_MENU** do the same thing. The **KEY_EXIT** key performs either exit program or shutdown depending upon the **exit_action** parameter in **/etc/radid.conf**. This also applies to the Shutdown Button in the Graphics versions of the radio.

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.

During recording of the keys, you may almost certainly see the following:

```
Something went wrong: Signal length is 0
That's weird because the signal length must be odd!
Please try again. (28 retries left)
```

Wait about five seconds and hold the selected key down until it is detected. Unfortunately, the program will try up to 23 times before it asks for the next key.

On completion of the key assignments the following is displayed:

```
Successfully written config file myremote.lircd.conf
```

Terminate the program by pressing **Enter** key when prompted for the next key.



It is better to create separate **myremote** files with just a few keys each and copy them into **/etc/lirc/lircd.conf.d**. For example, **myremote1.conf**, **myremote2.conf**, **myremote3.conf** etc. The **lircd** daemon will read in all of the *conf files that it finds in the **/etc/lirc/lircd.conf.d** directory.

It is now necessary to check the new **myremote.lircd.conf** file for bad key entries. In the output below the **KEY_CHANNELUP** entry is OK. The **KEY_CHANNELDOWN** and **KEY_MENU** keys did not produce good key definitions. You must delete these bad entries before copying them to the **/etc/lirc/lircd.conf.d** directory.

```
:
```

```
    name KEY_CHANNELUP
      8986    4443    604    1652    605    1652
      609     523    605    523    605    523
      605     523    605    523    610    1652
      605    1652    605    1656    605    523
      605     523    605    1652    610    1653
      605     523    606    523    610    523
      605     523    605    523    605    1652
      610    1652    605    523    610    1652
      605     523    605    1652    610    1652
      605    1652    613    521    606    520
      605    1652    610    523    605    1647
      610
```

```
    name KEY_CHANNELDOWN
      8993    2196    605
```

```
    name KEY_MENU
      8989    2198    611
```

```
end raw_codes
```

Now copy the new **myremote.lircd.conf** (or the name you used) to **/etc/lirc/lircd.conf**:

```
$ sudo cp myremote.lircd.conf /etc/lirc/lircd.conf.d/.
```

Restart the **lircd** daemon to reflect the changes. This must be done every time that a new key configuration file is copied to or changed in the **/etc/lirc/lircd.conf.d** directory.

```
$ sudo systemctl restart lircd
```

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

Testing the remote control

Run **irw** and press each key on the remote control in turn:

```
$ irw
0000000000000001 00 KEY_VOLUMEUP myremote
0000000000000001 01 KEY_VOLUMEUP myremote
0000000000000002 00 KEY_VOLUMEDOWN myremote
0000000000000002 01 KEY_VOLUMEDOWN myremote
0000000000000003 00 KEY_CHANNELUP myremote
0000000000000003 01 KEY_CHANNELUP myremote
0000000000000003 02 KEY_CHANNELUP myremote
0000000000000004 00 KEY_CHANNELDOWN myremote
0000000000000004 01 KEY_CHANNELDOWN myremote
0000000000000004 02 KEY_CHANNELDOWN myremote
0000000000000006 00 KEY_MENU myremote
0000000000000006 01 KEY_MENU myremote
0000000000000006 02 KEY_MENU myremote
0000000000000005 00 KEY_MUTE myremote
0000000000000005 01 KEY_MUTE myremote
```

Use Ctrl-C to exit. If keys are not responding repeat the previous Remote-Control installation procedure. Do not proceed if **irw** does not produce any output. Correct the problem and retry.



Note that the **ir-ctl** program mentioned in some guides as a replacement for **irw** does not really provide any useful information such as the key names. Use the above **irw** program.

Enable and start and check the irradiiod daemon

If you haven't already done so set the **remote_led** parameter **/etc/radiod.conf** as shown in the section called *Remote Control Activity LED* on page 51. If you ran the **configure_ir_remote.sh** program this should have already been done.

Disable the **ireventd** service:

```
$ sudo systemctl disable ireventd.service
```

Configure the **irradiod** daemon to start at boot time and start it.

```
$ sudo systemctl enable irradiod  
$ sudo systemctl start irradiod
```

The activity LED should flash a few times. Check the status of irradiod.

```
$ sudo systemctl status irradiod
```

If not running check the **remote_control.py** program to see if this provides any clues as to why it is not running.

```
$ cd /usr/share/radiod  
$ sudo ./irradiod.py nodaemon  
IR Remote control listener running pid 1924  
Using pylirc module  
Flashing LED on GPIO 16  
Listening for input on IR sensor  
KEY_UP  
KEY_DOWN  
KEY_RIGHT  
KEY_LEFT  
KEY_OK  
KEY_MENU  
^C  
Stopping remote control pid 1924  
Killed
```

If you see the following error then **/etc/lirc/lircrc** is missing:

```
IR Remote control listener running pid 1523  
Using pylirc module  
Flashing LED on GPIO 16  
piradio: could not open config file /etc/lirc/lircrc  
piradio: No such file or directory  
Unable to read configuration!  
Possible configuration error, check /etc/lirc/lircd.conf  
Activation IR Remote Control failed - Exiting  
Reboot the system to check the new IR remote configuration is working  
properly.
```

Copy the **lircrc.dist** file to **/etc/lirc/lircrc** and restart lircd and re-test with the **remote_control.py** program. This file tells LIRC what keys will be used with the radio program.

```
$ sudo cp /usr/share/radio/lircrc.dist /etc/lirc/lircrc  
$ sudo systemctl restart lircd
```

Now reboot the system.

```
$ sudo reboot
```

After rebooting make sure the **radiod** and **irradiod** daemons are running and check that the radio is listening on UDP port 5100 (or as configured in **/etc/radiod.conf**):

```
$ sudo systemctl status irradiod radiod
```

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

If the above UDP socket on port 5100 is not seen then troubleshoot the reason why the radio daemon isn't running.

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 roaming Wi-Fi

The wireless adaptor is normally configured either using **raspi-config**. See section *Configuring the Wi-fi Connection* on page 88 or using the Raspberry Pi imager software. This section describes two methods of enabling Wi-Fi roaming.

It is possible to add multiple Wi-Fi access points to the **/etc/wpa_supplicant/wpa_supplicant.conf** file. This file will already contain an entry for your Wi-Fi access point if you followed the procedure in section *Configuring the Wi-fi Connection* on page 88.

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=GB

network={
    ssid="YOUR_SSID"
    psk="YOUR_SSID_KEY"
}
```

The above configuration is for a router using WPA encryption.

Add additional Wi-Fi entries after the above lines. In this example for OFFICE and HOME:

```

network={
    ssid="OFFICE_SSID"
    psk="OFFICE_SSID_KEY"
}

network={
    ssid="HOME_SSID"
    psk="HOME_SSID_KEY"
}

```

Substitute your SSID and KEY with the actual SSID and KEY for your Wi-Fi access points.
With this method all of the Wi-Fi access points to be roamed must be pre-configured. The Raspberry Pi will automatically switch between all configured Wi-Fi access points.

Comitup Wi-Fi roaming

Comitup works by configuring a Wi-Fi Hot-Spot on the Raspberry Pi which can be connected to using either a Mobile Phone, tablet or PC via a Web Interface. The web interface then allows selection of any available Wi-Fi network. More information on **comitup** can be found at:
<https://davesteele.github.io/comitup>

There are two ways of installing comitup:

- Download the ready-made image from <https://davesteele.github.io/comitup>
- Install the comitup Rasbian package using **apt install**

Installation from the Comitup Image

To burn onto an SD card for the Raspberry Pi, first download the image from the above web site. There are currently two versions namely **Bullseye Desktop** and **Bullseye Lite**. Download the zip file to your PC (Not the torrent or Magnet versions). You should find the file in your **Downloads** folder:

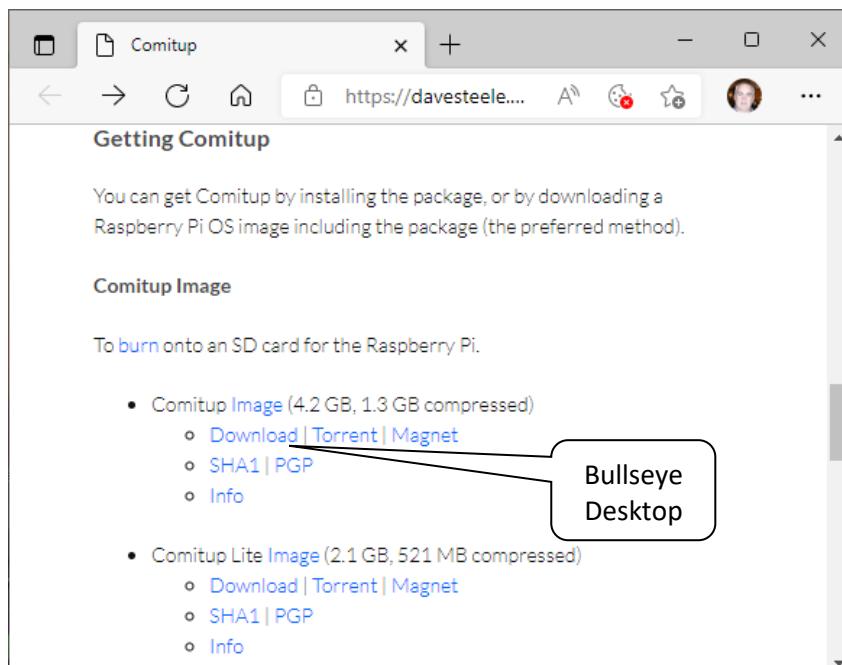
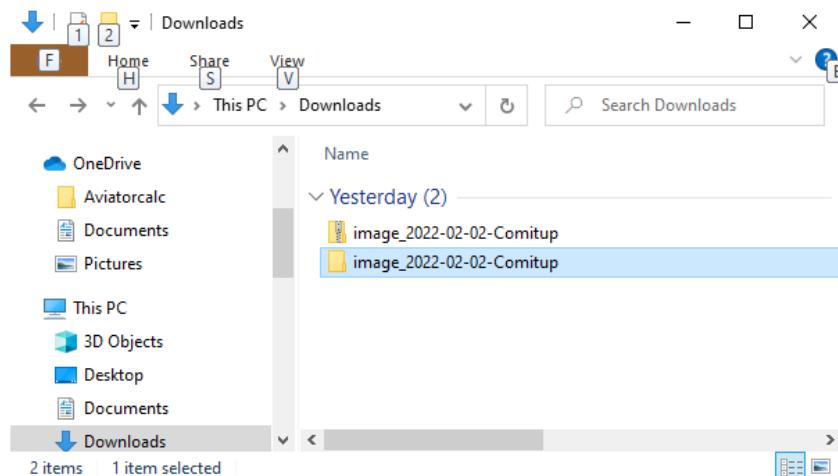
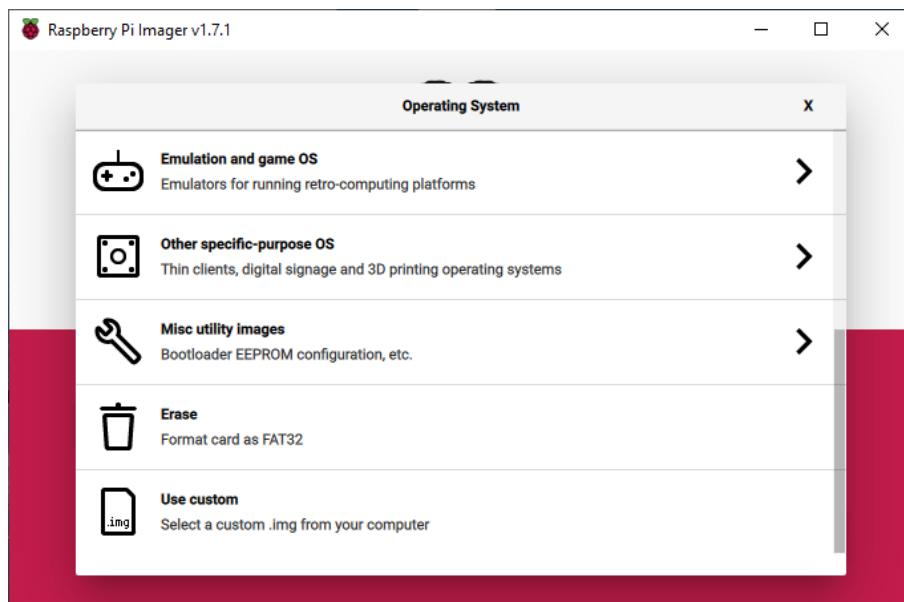


Figure 189 Downloading the comitup image

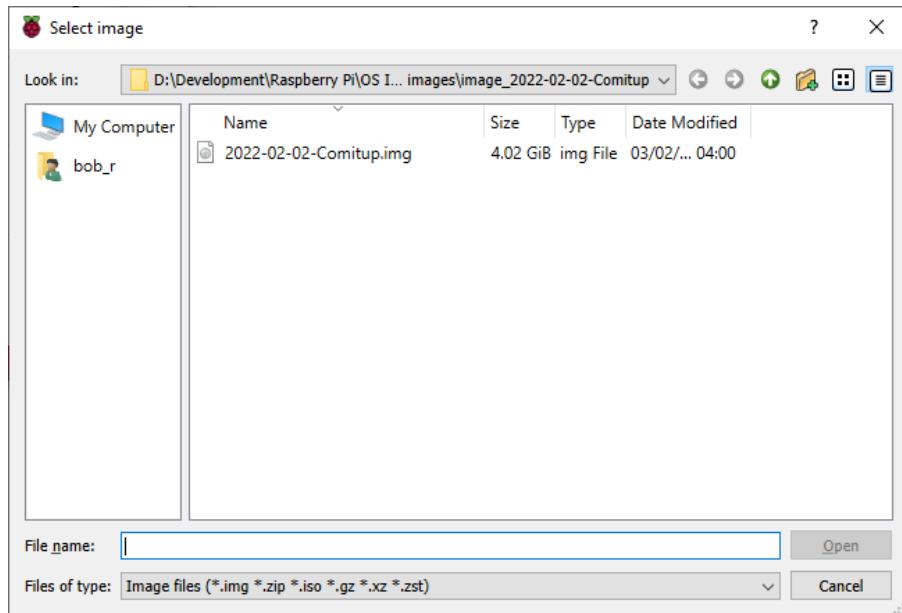
Using **Winzip** or **7Zip** unzip the image file; Typically this will unzip the compressed image to a directory with the same name as the zip file:



Now open the Raspberry Pi Imager software (See SD card creation using Raspberry Pi Imager on page 74 for more detail). Select “CHOOSE OS” and then scroll down to “Use custom”.



This will open a browser window. Go to your downloads browser and select the img file that was just unzipped.



Click and open the image file. Next select “CHOOSE STORAGE” and select your USB device. Finally select “WRITE”. Don’t change the hostname at this stage. It will have already been set to comitup-xxx where xxx is a three-digit number and will vary for example, comitup-123.

Installing from a Raspbian package



At the moment this procedure does not seem to produce a working comitup service. It used to work and is being investigated. Please use the procedure described in *Installation from the Comitup Image* on page 153.

Install required libraries

```
$ sudo apt install python3-tabulate firewalld libteam-utils
```

Install the Comitup package:

```
$ sudo apt install comitup
```

The current **/etc/wpa_supplicant/wpa_supplicant.conf** must be moved out of the way and control handed over to comitup. Rename

```
$ cd /etc/wpa_supplicant  
$ sudo mv wpa_supplicant.conf wpa_supplicant.conf.save
```

Comitup requires access to port TCP 53 however the system-resolved service uses that port so must be disabled.

```
$ sudo systemctl disable systemd-resolved
```

The current Web interface (If installed) must be placed under comitup control. In this case it is the Apache web interface that must be disabled and configure to be controlled by comitup.

```
$ sudo systemctl disable apache2.service
```

Add the following line to the **/etc/comitup.conf** file

```
web_service: apache2.service
```

Enable comitup and associated web service.

```
$ sudo systemctl enable comitup comitup-web
```

Reboot the Raspberry Pi

```
$ sudo reboot
```

After reboot there will no longer be a connection via the normal Wi-Fi Interface. Use a mobile phone or PC and use the Wi-Fi configuration screen to select the comitup

Once logged back in it is possible to check the status of the comitup and comitup-web services.

```
$ sudo systemctl status comitup
● comitup.service - Comitup Wi-Fi Management
  Loaded: loaded (/lib/systemd/system/comitup.service; enabled; vendor
  preset: enabled)
    Active: active (running) since Wed 2020-06-10 19:43:15 BST; 11min ago
      Docs: man:comitup(8)
   Main PID: 493 (comitup)
     Tasks: 2 (limit: 2061)
    Memory: 26.3M
   CGroup: /system.slice/comitup.service
           └─493 /usr/bin/python3 /usr/sbin/comitup
                 ├─731 dnsmasq --conf-file=/usr/share/comitup/dns/dns-hotspot.conf
                 └─731 dnsmasq --conf-file=/usr/share/comitup/dns/dns-hotspot.conf
--interface=wlan0
```

```
$ sudo systemctl status comitup-web
● comitup-web.service - Comitup Web Service
  Loaded: loaded (/lib/systemd/system/comitup-web.service; static; vendor
  preset: enabled)
    Active: active (running) since Wed 2020-06-10 19:54:22 BST; 3s ago
      Docs: man:comitup-web(8)
   Main PID: 2389 (comitup-web)
     Tasks: 1 (limit: 2061)
    Memory: 12.8M
   CGroup: /system.slice/comitup-web.service
           └─2389 /usr/bin/python3 /usr/sbin/comitup-web
```

To use comitup enter <http://10.41.0.1> into a web browser running on a PC, tablet or mobile telephone.

Notes:

The IP address is defined in `/usr/share/comitup/comitup/nm.py`

Disabling comitup

If for any reason comitup is a problem then it can be disabled and the standard Wi-Fi configuration restored with the following instructions.

Disable comitup and associated web service.

```
$ sudo systemctl disable comitup comitup-web
```

Restore `/etc/wpa_supplicant/wpa_supplicant.conf` (This assumes that this was previously a working configuration)

```
$ cd /etc/wpa_supplicant  
$ sudo mv wpa_supplicant.conf.save wpa_supplicant.conf
```

Re-enable the **system-resolved** service.

```
$ sudo systemctl enable systemd-resolved
```

Re-enable the **apache2** service.

```
$ sudo systemctl enable apache2.service
```

Reboot the Raspberry Pi

```
$ sudo reboot
```

Installing the Web interface

MPD has several Web clients. See the following link: <https://www.musicpd.org/clients/>. There are three versions of the Web interface used by the radio software:

1. Version 1.10 **Snoopy** Web interface (Optional)
2. Version 2.0 **Snoopy** and **O!MPD** Web interface for Buster
3. Version 2.2 **Snoopy** and **O!MPD** Web interface (The ! character is not a spelling mistake).

All three options use the Apache Web server.

It is recommended to install **O!MPD** as it provides a much more sophisticated user interface but it does require setting up a **MySQL** database. The Snoopy interface is much simpler but also more limited. You can also install the **Snoopy** interface and install and **O!MPD** at a later date.

Install the Apache Web Server

Install Apache the Web server. Make sure that the system is up to date with the following commands otherwise installation of Apache may fail.

Re-run the update to refresh package lists.

```
$ sudo apt update
```

Now upgrade the packages

```
$ sudo apt upgrade
```

Re-run the update to refresh package lists.

```
$ sudo apt update
```

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

Run the following command:

```
$ sudo apt install apache2 php libapache2-mod-php php7.4-gd php7.4-mbstring
```

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

```
$ sudo apt -f install
```

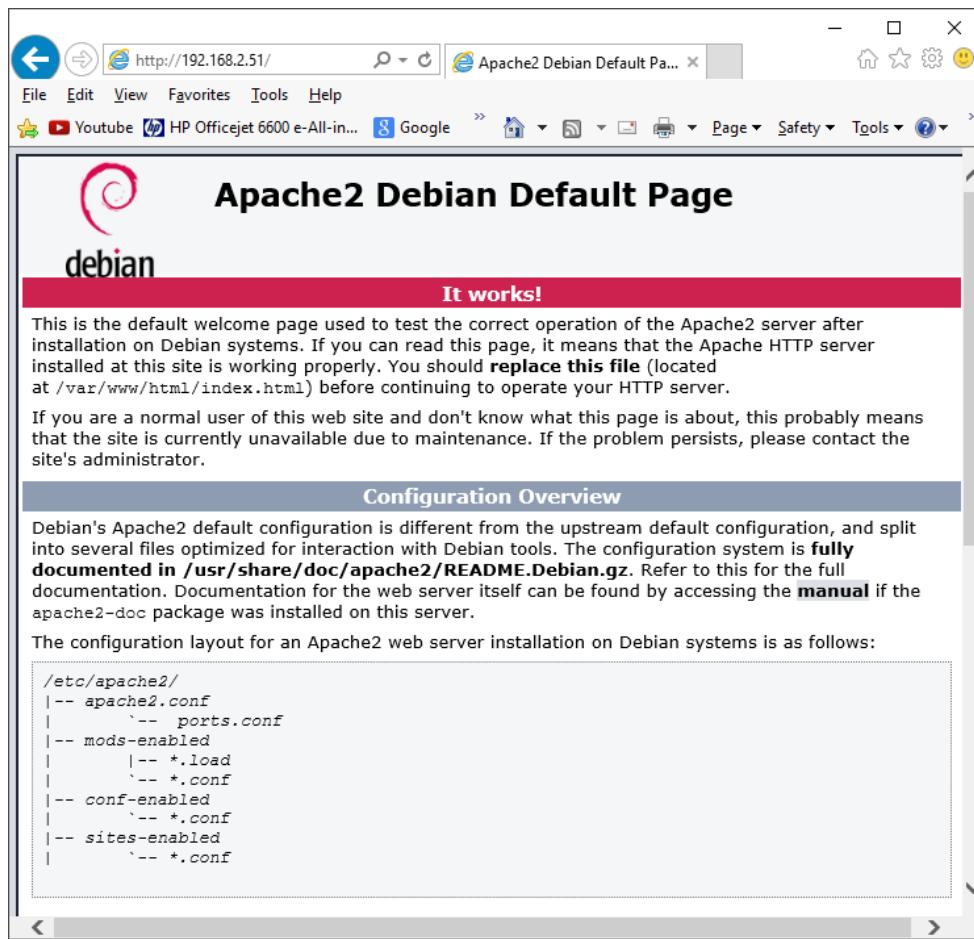
Test the Apache Web browser

Point your Web browser at the IP address or network name of the Raspberry PI. For example:
<http://192.168.2.51>.



Note: Make sure that you specify **http** and not **https** as the protocol. So <https://192.168.2.51> will not work as the Apache Web server is listening on port 80 (http) and not port 443 (https)

You should see the following display.



If it is not running check the apache2.service. The following display should be displayed. If not, there should be the appropriate error message.

```

$ systemctl status apache2
● apache2.service - The Apache HTTP Server
  Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor
  preset: enabled)
    Active: active (running) since Fri 2022-02-18 07:50:11 GMT; 9s ago
      Docs: https://httpd.apache.org/docs/2.4/
     Process: 11481 ExecStart=/usr/sbin/apachectl start (code=exited,
    status=0/SUCCESS)
    Main PID: 11485 (apache2)
       Tasks: 6 (limit: 2054)
         CPU: 139ms
      CGroup: /system.slice/apache2.service
              └─11485 /usr/sbin/apache2 -k start
                  ├─11486 /usr/sbin/apache2 -k start
                  ├─11487 /usr/sbin/apache2 -k start
                  ├─11488 /usr/sbin/apache2 -k start
                  ├─11489 /usr/sbin/apache2 -k start
                  └─11490 /usr/sbin/apache2 -k start

Feb 18 07:50:11 bullseye2 systemd[1]: Starting The Apache HTTP Server...
Feb 18 07:50:11 bullseye2 systemd[1]: Started The Apache HTTP Server.

```

Install the Web Browser server pages



Note: This procedure describes the simpler more limited **Snoopy** interface. If you wish to directly install version 2.x with **O!MPD** there is no need to install version 1.10 first. If you wish to install the **O!MPD** Web interface, go to the section called *Install version 2.x of the Web interface* on page 163.

Download the **Snoopy** radio Web pages Debian package from the Bob Rathbone Web site.

Run the following to install the version 1.10 Snoopy Web interface:

```
$ wget http://www.bobrathbone.com/raspberrypi/packages/radiodweb_1.10_armhf.deb
```

Now run:

```
$ sudo dpkg -i radiodweb_1.10_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.

The following error message may appear:

```
apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
```

The message may be ignored or it can be suppressed by editing the **/etc/apache2/apache2.conf** file and adding a **ServerName** directive.

Edit **/etc/apache2/apache2.conf** file.

Add the following line anywhere in the **apache2.conf** file.

```
ServerName localhost
```

Start the radio Web interface

Point your web browser at the IP address of the Raspberry Pi. For example: <http://192.168.1.168> you should see the following display:



Figure 190 Radio Web interface

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

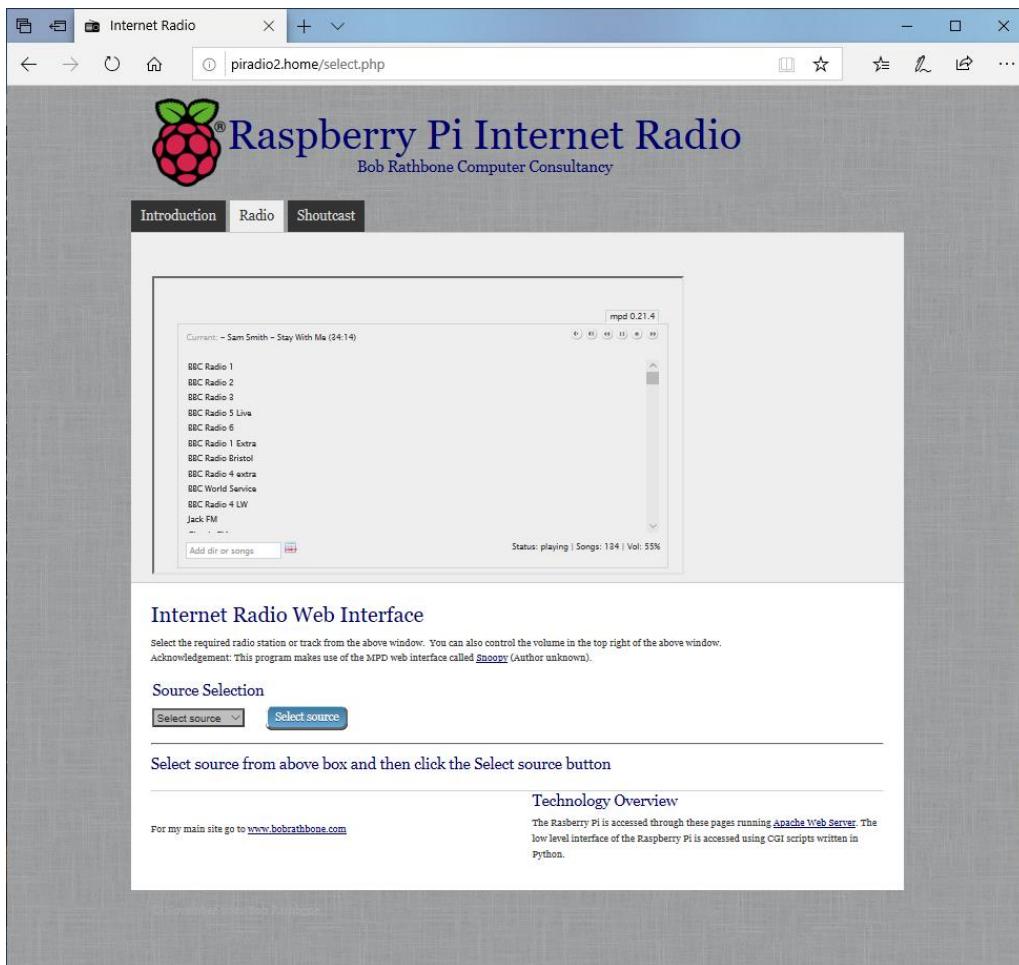


Figure 191 Snoopy Web interface

Click on any station in the list to select a station. After a short pause the station should start playing if it is online.

Source Selection

Select source	Submit
Select source Radio Media Airplay Spotify	

www.bobrathbone.com

The example on the left shows four music sources that can be selected namely Radio, Media, Airplay and Spotify. Version 1.8 onwards also displays Shoutcast

The desired source can be selected from the source drop-down selection box. Click on the required source then click on 'Submit' button to load the selected source in the radio. If you have more than one Media or Radio playlist, then repeatedly clicking on the appropriate source and Submit button will cycle through the playlists for that source. The name of the new playlist, however, is not displayed.

The Shoutcast tab is explained in *Using the Shoutcast Web Interface* on page 226.



Note: The radio tab only displays radio stations or media tracks from MPD. It is not currently capable of displaying Spotify or Airplay details which can only be seen on the radio itself.

Install version 2.x of the Web interface

Install Apache as described in the section called *Install the Apache* on page 157. Once done install required packages.

Install version 2.x of the Web interface

For Bullseye

Install PHP4 and the MariaDB database.

```
$ sudo apt install php7.4-gd php7.4-mbstring mariadb-server php-mysql
```

Install the Radio Web pages.

```
$ wget http://www.bobrathbone.com/raspberrypi/packages/radiodweb_2.2_armhf.deb
```

Install the **radiodweb** package:

```
$ sudo dpkg -i radiodweb_2.2_armhf.deb
```

For Buster

Install PHP3 and the MariaDB database.

```
$ sudo apt install php7.3-gd php7.-mbstring mariadb-server php-mysql
```

Download the **radiodweb** package:

```
$ wget http://www.bobrathbone.com/raspberrypi/packages/radiodweb_2.0_armhf.deb
```

Install it

```
$ sudo dpkg -i radiodweb_2.0_armhf.deb
```

For both Buster and Bullseye

If any of the previous commands fail for any reason run the following command:

```
sudo apt --fix-broken install
```

Enable the new PHP modules:

```
$ sudo phpenmod gd mbstring
```

Restart Apache:

```
$ sudo apachectl restart
```

Create MySQL user and password

```
$ sudo mysql  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
:  
MariaDB [(none)]>
```

Enter the following commands at the **MariaDB [(none)]>** prompt to create a password hash. All commands end with a semicolon (;).

```
SELECT PASSWORD('raspberry');  
+-----+  
| PASSWORD('raspberry') |  
+-----+  
| *1844F2B11CCAEF3B31F573A1384F608BB6DE3DF9 |  
+-----+  
1 row in set (0.008 sec)
```

Now grant permissions to user 'pi'. Note that the command in bold below is all one line.

```
GRANT USAGE ON *.* TO 'pi'@'localhost' IDENTIFIED BY PASSWORD  
'*1844F2B11CCAEF3B31F573A1384F608BB6DE3DF9';  
Query OK, 0 rows affected (0.001 sec)
```

Grant permissions to user pi to create the OMPD database with the following two commands.

```
GRANT ALL PRIVILEGES ON ompd.* TO 'pi'@'localhost';  
FLUSH PRIVILEGES;
```

Exit mysql.

```
exit
```

Restart Apache2

```
$ sudo apachectl restart
```

Enter the address of your Raspberry Pi into the browser. In this example that would be **http://192.168.1.249**. The following should be seen:



Figure 192 Version 2.x Web interface with O!MPD

Note that there is an extra tab called **O!MPD** when compared to the **Snoopy** interface. No, the ! character is not a spelling mistake. Now click on the **O!MPD** tab. The following should be seen:

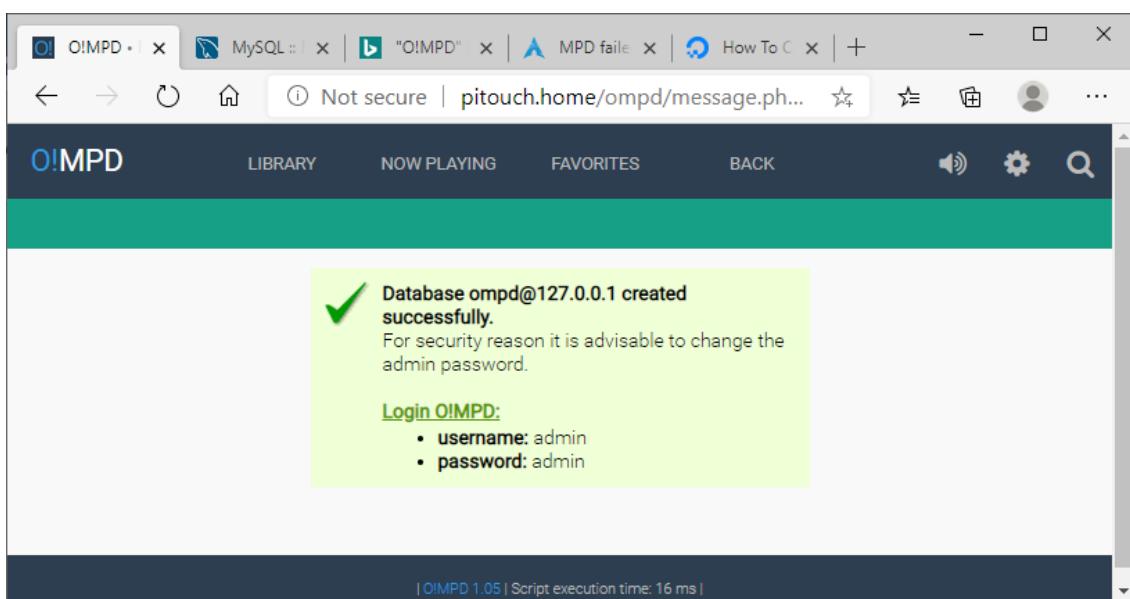
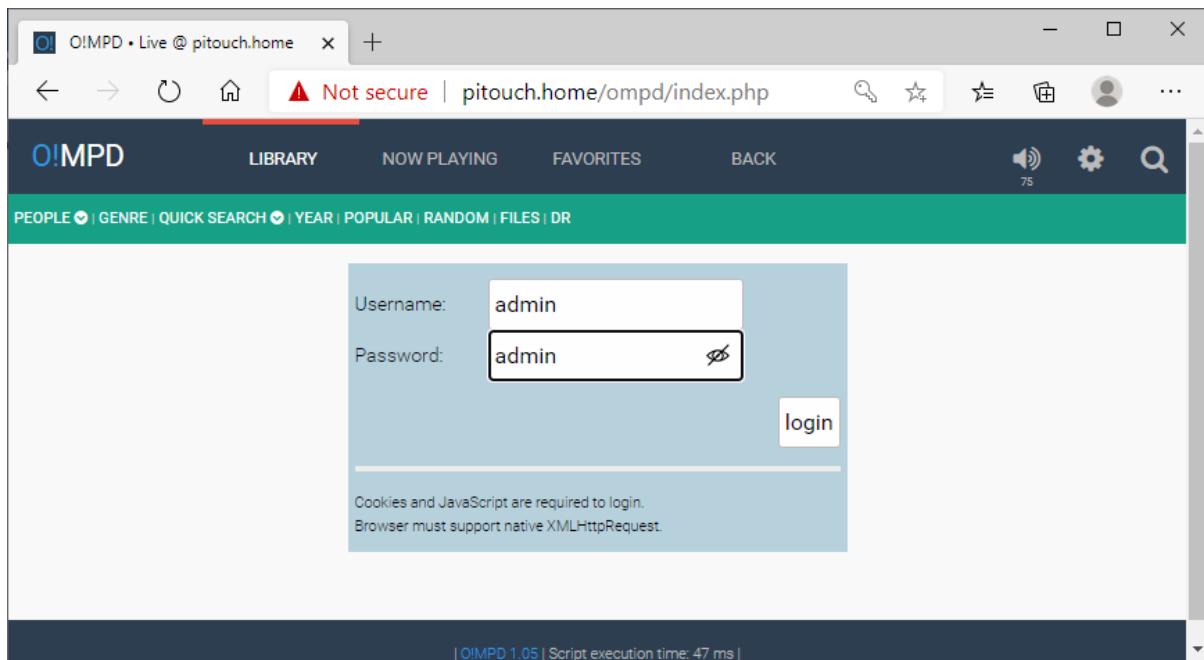
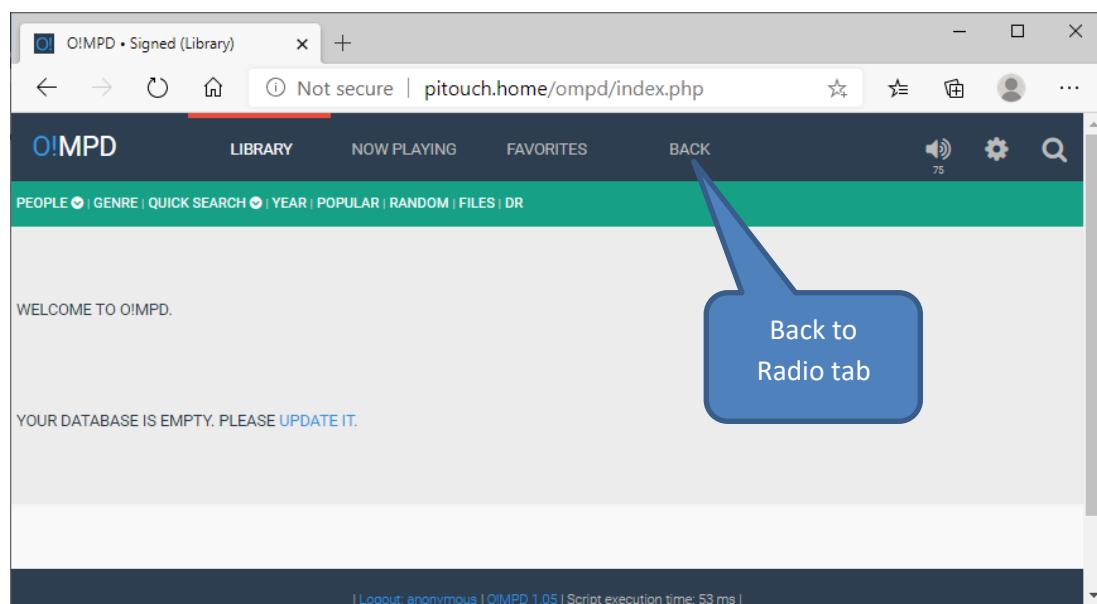


Figure 193 OMPD database creation



If you have already created a music library by running the `create_playlist.sh` program then it is necessary to update the **O!MPD MySql** database. Otherwise skip these instructions.



Note: The “(i) Not secure” message in the above screens can be ignored. It is because the web pages do not have a security certificate to verify that the pages come from a legitimate source and also are not using the HTTPS protocol. Since these pages are not coming from a source on the Internet and are locally installed, they do not pose a security risk and come from a reliable source namely the **Bob Rathbone radioweb** package.

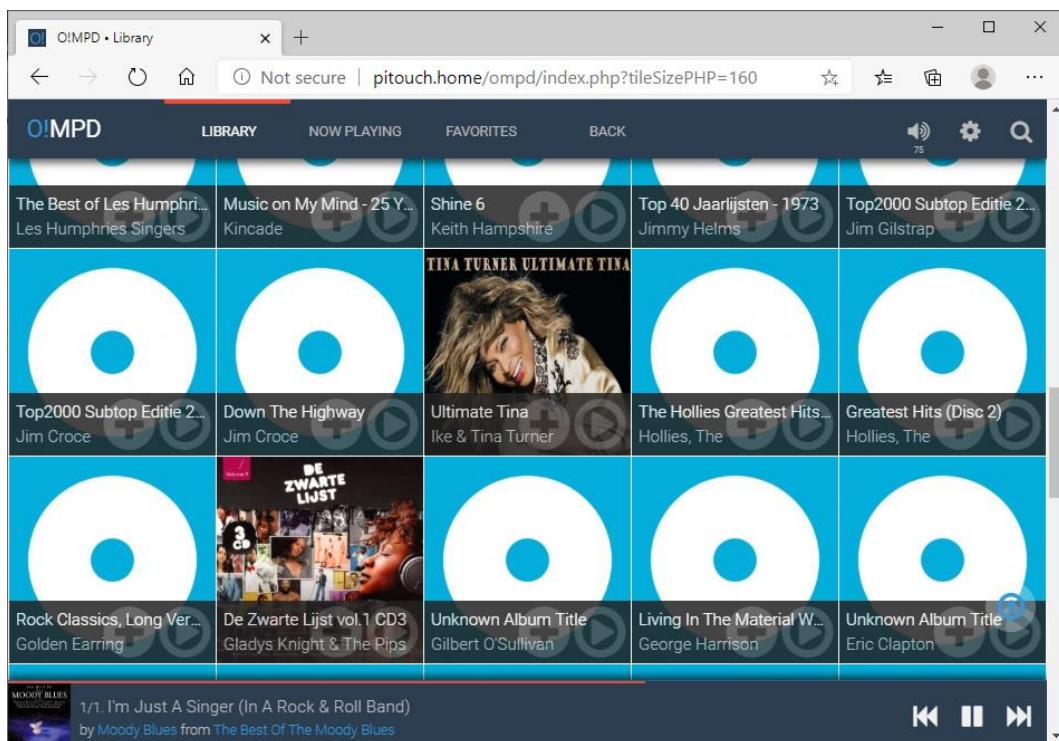
The screenshot shows the O!MPD configuration interface. At the top, there are tabs for LIBRARY, NOW PLAYING, FAVORITES, and BACK. Below the tabs is a green header bar with links for PLAYER PROFILE, STREAM PROFILE, DOWNLOAD PROFILE, SKIN PROFILE, USERS, and UPDATE ALL. The main content area is titled 'SESSION PROFILE' and contains four entries: Player profile (Music Player Daemon (default)), Stream profile (Source), Download profile (Source), and Skin profile (ompd_default). A blue callout bubble points to the 'Update' link under the 'CONFIGURATION' section, which includes links for Settings, Users, Online, User statistics, Media statistics, Update, and PHP information. At the bottom, a track is playing: '35/148. Camila Cabello - Liar' by 'from CROOZE.fm - a unique mix of Smooth & Acid Jazz, Funk, Soul, Latin, R&B, Lounge and Chill music.'

Click here to
update
database

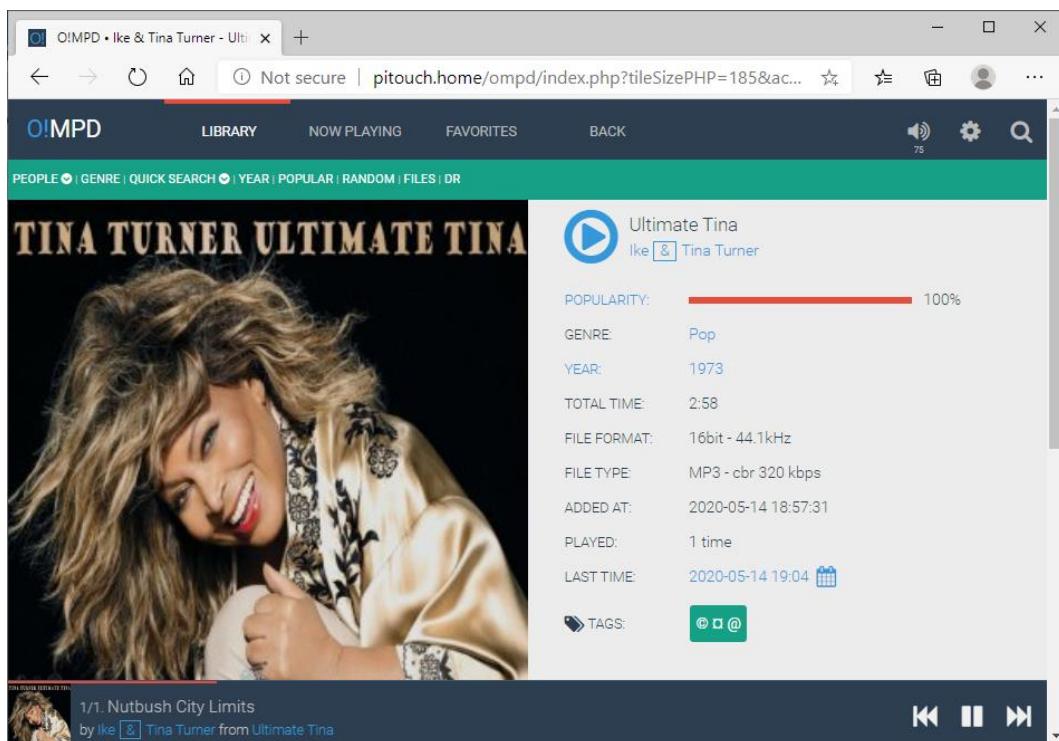
The media library will now be updated:

The screenshot shows the O!MPD configuration interface with the title 'Signed (Configuration)'. The main content area is titled 'CONFIGURATION > UPDATE' and shows a progress bar for 'Structure & image' at 100%. Below the progress bar, there is a section for 'File info:' showing a track: '1/148. bbcmedia_radio1_mf_p?s#BBC Radio 1' by 'from BBC Radio 1'.

The Web interface will now display the contents of the media library.

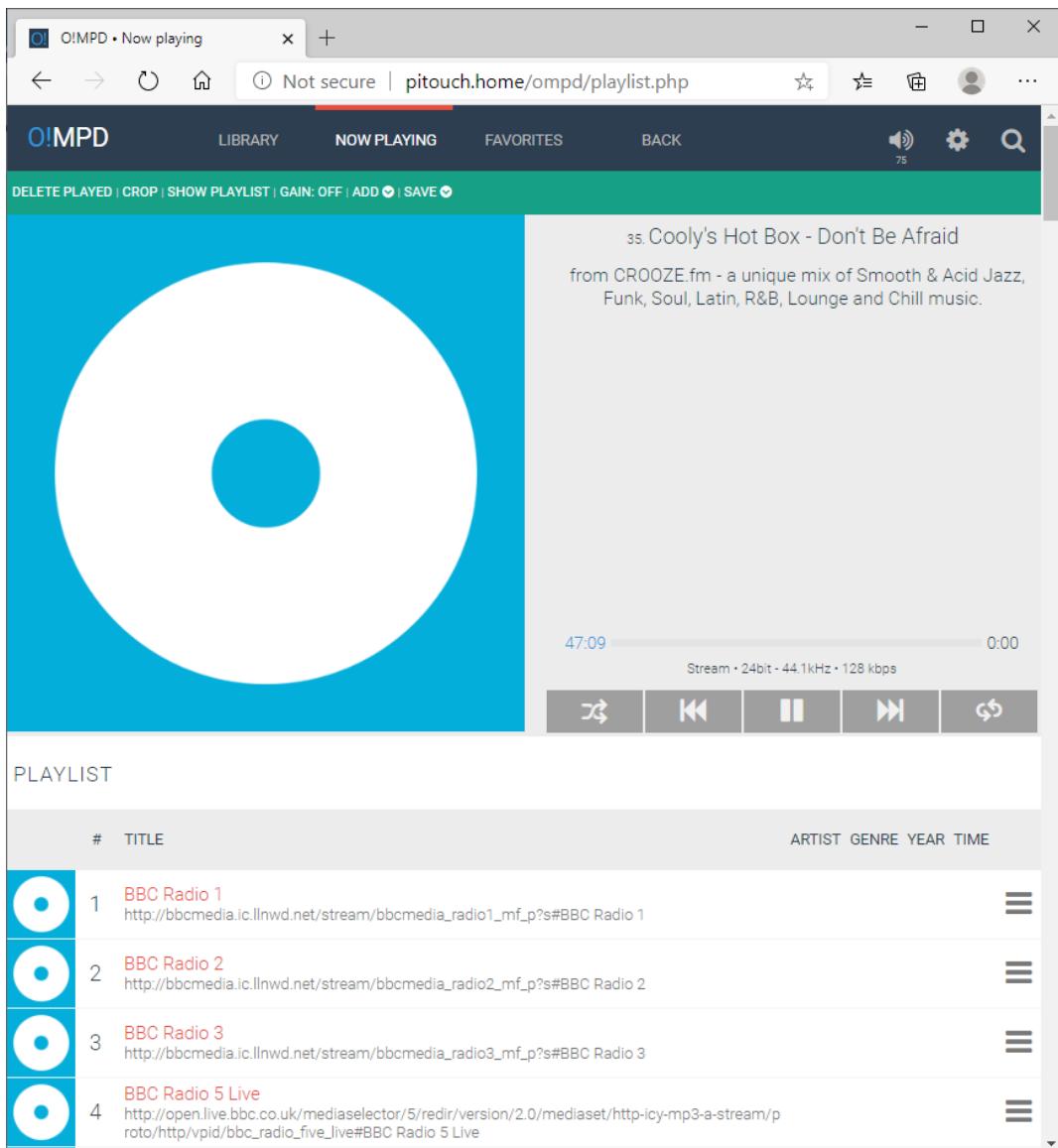


Clicking on one of the tracks will display its details along with the album artwork.



Clicking on the blue play icon will play the track.

To play a radio station click on the “NOW PLAYING” top menu item. Select the required radio station to start playing the stream.



Changing the Web Interface Radio photo

If you want to change the photo displayed by the Web interface on the first page, 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. Copy the new jpeg photo to the pi home directory with any ftp program.

Now copy it to the Web pages image directory using **sudo**.

```
$ sudo cp radio.jpg /var/www/html/images/.
```

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.

```
</td>
```

Upgrading the Music Player Daemon

The Raspberry Pi usually is installed with the **Raspberry Pi OS** (Formerly **Raspbian**) operating system. At the time of writing the latest version of **Raspbian** is called **Bullseye** which uses version **5.10.17** of the Linux kernel or later.



Note: The MPD version described in this procedure is **0.23.8** whilst the version released with Bullseye is **0.22.6** and is already fairly out-of-date. The version released with **Buster** is **0.21.5** which is even worse as it is years out of date. The latest version of MPD is much more stable. Due to their age, it is impossible to get support on the older versions of MPD.

There are two methods of updating MPD.

1. Download a ready-made Debian package **mpd_0.23.8_armhf.deb** (or latest version)
2. Download the latest MPD source, then compile and install it.

Installing the MPD upgrade from a Debian package

Two versions of the Raspberry Pi OS are supported in this tutorial.

- **Bullseye** – Install version **0.23.13** (**mpd_0.23.13_armhf.deb**)
- **Buster** – Install version **0.23.5** (**mpd_0.23.5_armhf.deb**)

Version **0.23.8** will not run on Buster as it requires the **libicu67** library which is not available in **Buster**. If using **Buster** substitute all commands using **0.23.13** with **0.23.5**. For example:

```
$ wget https://bobrathbone.com/raspberrypi/packages/mpd_0.23.13_armhf.deb
```

For **Buster**:

```
$ wget https://bobrathbone.com/raspberrypi/packages/mpd_0.23.5_armhf.deb
```

The **mpd_0.23.13_armhf.deb** (or **mpd_0.23.5_armhf.deb**) upgrade package is currently available from the **package's** directory on the Bob Rathbone Web site. This is an upgrade; **mpd** and **mpc** as supplied with the OS must be installed first.

```
$ sudo apt install mpd mpc
```

Answer yes 'y' when asked to continue.

Install the required libraries. Some of these may already be installed if using the desktop version. Execute all three lines below. Do not copy the \$ sign in the commands below.

```
$ sudo apt install libsidutils0 libresid-builder0c2a libcurl3-gnutls  
$ sudo apt install libaudiofile1 libyajl2  
$ sudo apt install libiso9660-11 libzzip-0-13 libao4
```

Now install the **libadplug** library. There are two versions depending upon the OS version being used.
For Bullseye (It may already be installed)

```
$ sudo apt install libadplug-2.3.3-0
```

For Buster

```
$ sudo apt install libadplug-2.2.1-0v5
```

Download the MPD update software.

```
$ wget https://bobrathbone.com/raspberrypi/packages/mpd_0.23.13_armhf.deb
```

Now install with **dpkg**.

```
$ sudo dpkg -i mpd_0.23.13_armhf.deb
```

Ignore the warnings for existing **/var/log/mpd** and **/var/lib/mpd** directories.

Reboot the Raspberry Pi to enable the new version.

```
$ sudo reboot
```



Note: For some unknown reason when **mpd.socket** is run from this package it changes the ownership of the **/var/run/mpd** directory from **mpd:audio** to **root:root** which causes **mpd.service** to fail with a permissions error. To correct this problem the following line has been added to **mpd.service** just before the **ExecStart** statement.

```
ExecStartPre=-/bin/chown mpd:audio /var/run/mpd
```

There is no need to add this line yourself as it is already added by the installation script. If the Radio/MPD service fails to start see the section called *Music Player Daemon won't start* on page 251.

Installing the latest MPC client

The **mpc_0.34_armhf.deb** upgrade package is currently available from the **packages** directory on the Bob Rathbone Web site. The latest version of **mpd** must be installed first as shown in the previous section.

Download the MPC software upgrade package.

```
$ wget https://bobrathbone.com/raspberrypi/packages/mpc_0.34_armhf.deb
```

Now install with **dpkg**.

```
$ sudo dpkg -i mpc_0.34_armhf.deb
```

Ignore the warnings for existing **/var/log/mpd** and **/var/lib/mpd** directories.

Reboot the Raspberry Pi.

Compiling and installing the latest Music Player Daemon

How to download and compile the latest version of Music Player daemon is described in the following document:

<https://bobrathbone.com/raspberrypi/documents/Compiling%20and%20installing%20MPD.pdf>

Installing 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 337). This facility requires installation of the **espeak** package.

See [http://elinux.org/RPi_Text_to_Speech_\(Speech_Synthesis\)](http://elinux.org/RPi_Text_to_Speech_(Speech_Synthesis))

The speech facility makes use of the **/var/lib/radio/language** file as already described in the section called *Creating a new language file* on page 183

Install the **espeak** package:

```
$ sudo apt 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
```

Speak hostname and IP address when in the Information Menu.

```
speak_info=no
```

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 (Stop the radio first).

```
$ espeak -ven+f2 -k5 -s130 -a20 "Hello Bob" --stdout | aplay
```

To see the capabilities of **espeak** see the Web site <http://espeak.sourceforge.net/> or run:

```
$ espeak -h
```

If no sound is heard 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.

```
$ sudo mpc pause  
$ $ aplay /usr/share/sounds/alsa/Noise.wav
```

Or if Scratch is installed

```
$ 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:

```
$ sudo nano /etc/asound.conf
```

Add the following lines:

```
ctl.!default {  
    type hw  
    card 1  
}  
  
pcm.!default {  
    type plug  
    slave {  
        pcm "plughw:0,0"  
        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 2 device 1) then change the device definition in the above file.

```
pcm "plughw:1,0"
```

Retest with **aplay** (No need to reboot).



Note: This author does not provide support for **espeak**.

See: <https://sourceforge.net/p/espeak/discussion/> for general support issues.

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 135.

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.

Suppressing an individual message

It is possible to suppress speech of 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 ! character to the beginning of the text to be spoken as shown in the example below:

```
the_time: !The time is
```

The exclamation message is removed if the message is displayed on a display. Only speech is affected.

Keeping the radio software up-to-date



The Radio software may be updated from time to time especially if a new version of the operating system is released or a new feature is added to the software. To keep up to date follow the author on Twitter at: https://twitter.com/bob_rathbone
(https://twitter.com/bob_rathbone)

Backing up the SD card

Having spent a lot of time and effort installing and configuring the Radio software it is a very good idea to create a backup of the SD card should it ever become corrupted. There are various ways of doing this. For backing up under Linux see:

<https://www.raspberrypi.org/documentation/linux/filesystem/backup.md>

One of the easiest ways of backing up the SD card on a Windows machine using Windows Disk Imager (Win32DiskImager) described in the following link.

<https://www.raspberrypi.org/forums/viewtopic.php?t=26463>

This allows you to create a copy of the SD card in an image (.img) file. This can then be compressed using **winzip/Zzip** or any other zip utility to reduce the space on disk.

Chapter 7 – Configuration

This section covers manual configuration of the radio. A number of programs are provided to help with configuration. These are:

1. **configure_radio.sh** – Configure the basic radio interfaces.
2. **configure_audio.sh** – Configure the audio output.
3. **set_mixer_id.sh** – Configure the Alsa mixer volume control ID (normally called by the radio program)

The above programs are found in the **/usr/share/radio** directory and are normally used to configure the radio and audio output. The following descriptions cover manual configuration and those configuration parameters not set by the above programs.

Configuring the HDMI or Touch Screen

In the **/etc/radiod.conf** file there is a section called [SCREEN] as shown below. This is the HDMI/Touch Screen default configuration.

```
# Graphics (touch screen) screen settings
[SCREEN]
# Size is in pixels. Supported is 800x480 (7" screen) or 720x480(3.5" screen)
# or 480x320 (2.8" or 3.5" screen) or 1024x600 (Maximum)
# Also see framebuffer_width and framebuffer_height parameters in /boot/config.txt
screen_size=800x480
fullscreen=yes

# Screen save time in minutes, 0 is no screen saver
screen_saver=0

# Title %V = version %H = hostname
window_title=Bob Rathbone Internet Radio Version %V - %H
window_color=turquoise
banner_color=white
labels_color=white
display_window_color=lightblue
display_window_labels_color=black
slider_color=darkgreen
display_mouse=yes

# Wallpaper backgrounds. See /usr/share/scratch/Media/Backgrounds
wallpaper=/usr/share/scratch/Media/Backgrounds/Nature/beach-malibu.jpg

# Set date format for graphic screen
dateformat=%H:%M:%S %A %e %B %Y

# Allow switching between vgradio and gradio
switch_programs=yes

# The following is specific to the vintage graphical radio
scale_labels_color=white
stations_per_page=40
display_date=yes
display_title=yes
```

Parameter	Explanation
fullscreen	Set to yes or no . If using a large HDMI monitor or TV set to no .
window_title	Title to display in the desktop window if fullscreen=no
window_color	Window background colour if wallpaper (See below) not specified.
banner_color	This is the colour of the time and date banner.
labels_color	This is the colour of the radio and MPD option labels.
display_window_color	This is the background colour of the station/track display window.
display_window_labels_color	Colour of the display window text.

slider_color	The color of the slider in the slider window next to the search window.
display_mouse	Future use – hide mouse yes/no .
wallpaper	Background wall paper. Any jpeg or gif file can be specified. See directory <code>=/usr/share/scratch/Media/Backgrounds</code> .
dateformat	The format for displaying the time and date banner.

The following settings are specific to the vintage graphical radio:

scale_labels_color	This sets the color of the station names on the tuning scale
stations_per_page	This is the maximum number of stations that will be displayed on each page.
display_date	Display the date at the top of the screen yes/no
display_title	Display the station title at the bottom of the screen yes/no



All parameters use the American spelling for color and not the British spelling.

The **wallpaper** parameter overrides the **window_color** parameter.

The parameters allow any theme to be configured for the HDMI or Touch Screen window.

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.

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. Normally there is no need to change these as they are set by the **configure_radio.sh** program.

```
# Switch settings for 40 pin version (support for IQaudIO)
menu_switch=17
mute_switch=4
up_switch=24
down_switch=23
left_switch=14
right_switch=15
```

Disabling button or rotary encoder GPIOs

It is possible to disable a button or rotary encoder GPIO configuration. For example, if you are building a radio with an amplifier with its own volume control you may not need the music player daemon volume control as well. In this case set these to 0 to disable them.

```
menu_switch=17
up_switch=24
down_switch=23
mute_switch=0
left_switch=0
right_switch=0
```

This will free the GPIOs originally configured for the volume control for other uses (4, 14 and 15 in this example).

LCD display GPIO assignments

The default LCD settings for a 40-pin Raspberry Pi are shown below. Again, there is no need to change these unless your wiring is different. Again, setting these to 0 disables the outputs.

```
# LCD GPIO connections for 40 pin version of the radio
lcd_select=7
lcd_enable=8
lcd_data4=5
lcd_data5=6
lcd_data6=12
lcd_data7=13
```

Configuring button interface with pull up resistors

This applies to the radio with push buttons only. The original design of the radio wires the push buttons from low to high (GND 0V to 3.3V). It is now more usual to configure the buttons to operate from high to low (3.3V to GND 0V). This is the case for rotary encoders. It may be easier to wire up the buttons to GND 0V. In such a case it is necessary to configure the **pull_up_down** parameter in **/etc/radiod.conf** to 'up' as shown below.

```
# Pull GPIO up/down internal resistors (Applies button interface only).
# Default:down
pull_up_down=up
```

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 51. Which pins you connect to will depend on the type of radio you are building. Table 12 on page 51 shows the required LED connections. Boards such as 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 AdaFruit plate or Vintage radio. For 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 51.

```
# Output LED for remote control, default GPIO 11 (pin 23) or
# GPIO 13 (pin 33) for AdaFruit plate
# 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 **irradiod.py** program.

```
$ cd /usr/share/radio/
$ ./irradiod.py flash
```

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 51).

Specifying the action for exit/shutdown action

The radio can be exited by holding the menu button in for three seconds. The action depends upon the **exit_action** parameter. This is either set to **stop_radio** or **shutdown**.

```
# Action on exiting radio. Stop radio only or shutdown the system  
# exit_action=stop_radio  
exit_action=shutdown
```



From version 7.3 onwards the graphic versions of the radio have an exit/shutdown button (as shown on the left). The exit/shutdown button is displayed by default. This can be changed by changing it to no.

```
display_shutdown_button=no
```

If the exit button is enabled and clicked, the radio program will either exit to the operating system (handy during testing) or it will perform a system shutdown.

It is also possible to specify the shutdown command with the **shutdown_command** parameter in **/etc/radiod.conf**. The default command is shown below.

```
shutdown_command="sudo shutdown -h now"
```

This can be replaced with required command for example "x735off" for a X735 V2.5 shield. Put the command between apostrophes for example "x735off".

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
%H:%M %m%d	Short date display for Olimex OLED

Seconds can also be displayed:

%H:%M:%S %d/%m/%Y Display seconds (%S) on 20 character displays only

Configuring the IQaudIO Cosmic controller and OLED

The **configure_radio.sh** program can be used to set the configuration to use the 128 by 64 pixel OLED supplied with the Cosmic controller. This sets the **display_type** parameter to **OLED_128x64**.

```
display_type=OLED_128x64
```

When running with the Cosmic controller OLED screen there are two relevant settings in the **/etc/radiod.conf** file which are not set by the **configure_radio.sh** program.

The OLED display can be flipped vertically by setting the **flip_display_vertically** parameter to yes.

```
flip_display_vertically=yes
```

Note: If upgrading you will need to add this parameter to the **[RADIO]** section of the **/etc/radiod.conf** file.

The three LEDs on the Cosmic Controller board are driven by the *status_led_class.py* program. This class was originally written for the Vintage radio but is now also used with this board. Configure the following parameters in **/etc/radiod.conf** as shown below:

```
rgb_red=14  
rgb_green=15  
rgb_blue=16
```

The left LED means an ERROR, the middle LED means NORMAL operation and the right-most is BUSY.

If you want to switch off the status LEDs then set them to 0. However, GPIO 15 is switched on automatically at boot time. To switch it off add the following two lines to **/etc/rc.local**.

```
gpio -g mode 15 out  
gpio -g write 15 0
```

Set the date format so that it displays fits the display.

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

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 startup mode for Radio or Media player

The radio can be configured to start in **RADIO**, **MEDIA**, **LAST** modes or a **playlist** name. The default is **_Radio**. To change this, edit **/etc/radiod.conf** and change the **startup=_Radio** parameter to **RADIO**, **MEDIA** or **LAST** to start the radio with the last playlist used in the previous run. For example:

```
# Startup option either RADIO,MEDIA or LAST a playlist name  
startup=MEDIA  
#startup=Radio
```

Alternatively, the radio can be configured to load a specific playlist. To display the available playlists run the following command:

```
$ mpc lsplaylists  
USB_Stick  
UK_stations  
Beatles  
Radio  
:
```

To configure the radio to start with a specific playlist change the **startup=** statement.

```
#startup=RADIO  
startup=USB_Stick
```

If you configure **startup=RADIO** the program will load the first available Radio playlist. Likewise if you configure **startup=MEDIA** the program will load the first available Media playlist.

Configuring the volume display

The volume can be displayed as either text or as a series of blocks. This is configured in **/etc/radiod.conf** using the **volume_display** parameter. The default is text.

```
# Volume display text or blocks  
volume_display=text
```

```
12:01 30/08/2017  
WPJR Country  
Lonestar - Mr. Mom  
Volume 75
```

To display the volume as a series of blocks change this to ‘blocks’:

```
volume_display=blocks
```

12:01 30/08/2017
WPJR Country
Lonestar - Mr. Mom



If the timer or alarm functions are being used then the volume display reverts back to text display so as to allow display of the alarm or timer values.

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 **volume_range** parameter 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, $100/20 = 5$

So, if the volume displayed on the LCD is 10 and the range is 20, then the MPD volume is $10 \times 5 = 50\%$.

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.

Configuring the MPD client timeout

When the radio program tries to connect to radio stream it will time out after so many seconds. In all previous versions this timeout was hard set to ten seconds. This is configurable from three to fifteen seconds using the **client_timeout** parameter in **/etc/radiod.conf**. The default is ten seconds.

```
# MPD client timeout from 2 to 15 seconds default 10
client_timeout=10
```

Changing the display language

The language file is stored in **/home/pi/radio/language** directory. This contains the text that will be either displayed or spoken. The default language is English. The **language.en** file is copied to **/var/lib/radiod/language**. The language file (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
airplay: Airplay
alarm: Alarm
alarmhours: Alarm hours
alarmminutes: Alarm minutes
colour: Colour
consume: Consume
current_station: Current station
information: Information display
loading: Loading
loading_media: Loading media library
loading_radio: Loading radio stations
main_display: Main
media_library: Media library
menu_find: Find
menu_option: Menu option:
menu_search: Search
:
```

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 message. Lines beginning with # are for any comments.

```
# 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 systemctl restart radiod
```

Configuring Music Player Daemon CODECs

It is possible to configure the Music Player Daemon CODECs list in **/etc/radiod.conf**. There is a parameter called CODECS with the CODECs list between quotes.

```
# Codecs list for media playlist creation (Run 'mpd -V' to display others)  
CODECS="mp3 ogg flac wav"
```

To see what CODECs are available in MPD run the following command.

```
$ mpd -V
```

A CODEC defines the method for encoding and decoding a digital stream. CODEC is a portmanteau of Coder-Decoder. See Wikipedia article <https://en.wikipedia.org/wiki/Codec> for more information on CODECS.

Configuring 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 feed 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 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.

If configured, the RSS feed will be automatically displayed by stepping through the menus.

Configuration of the mute button action

When the mute button is pressed the volume is reduced to zero and the stream is either paused or stopped depending upon the setting of the **mute_action** parameter in **/etc/radiod.conf**.

```
# Action when muting MPD. Options: pause(Stream continues but not processed)  
# or stop(stream is stopped)  
# mute_action=stop  
mute_action=pause
```

pause – The radio stream continues to be downloaded but is not processed (default)

stop – The radio stream is stopped altogether.

Both have their own characteristics. When the radio is un-muted using the **stop** option it will play the old remaining stream in its buffer for about 30 seconds before jumping to the new live stream. It does have the advantage that no Internet bandwidth is being consumed.

The **pause** option continues to download the radio stream, but not processing it and consuming Internet bandwidth. When the radio starts playing again MPD simply starts processing the live stream again. There is no buffer to empty so no “jumping” to the new stream. The behaviour of pause and stop is controlled by the Music Player daemon over which the author has no control.

Configuring the Alsa Equalizer



Note: The author of the radio software does not currently have a configuration that is compatible with either **Pulseaudio** or **Bluetooth** audio. Devices. If you wish to use any of these then you cannot currently use the Alsa equalizer. This may change in a future release.

Install the Alsa plugin with **apt**:

```
$ sudo apt install -y libasound2-plugin-equal
```

Amend the “device” parameter in the **audio_output** block in **/etc/mpd.conf** configuration file.

```
audio_output {
    type          "alsa"
    name          "IQAudio DAC+"
    device        "plug:plugequal"
    mixer_type    "software"
}
```

In the above example we are using an IQaudIO card but may be any sound card.

Save the existing **asound.conf** file just in case you need to restore the original file

```
$ sudo cp /etc/asound.conf /etc/asound.conf.save
```

Copy the **asound.conf.dist.equalizer** to **/etc/asound.conf**

```
$ cd /usr/share/radio/asound/
$ sudo cp asound.conf.dist.equalizer /etc/asound.conf
```

The new **/etc/asound.conf** file should look as shown below:

```
pcm.!default {
    type plug
    slave.pcm plugequal;
}
ctl.!default {
    type hw card 0
}
ctl.equal {
    type equal;
}
pcm.plugequal {
```

```
    type equal;
    slave.pcm "plughw:0,0";
}
pcm.equal {
    type plug;
    slave.pcm plugequal;
}
```

If your sound system is using card 1 (for example a USB audio device) then change the hardware settings in the above configuration to use card 1.

```
:
type hw card 1
:
slave.pcm "plughw:1,0";
```

It is also necessary to amend the card number in the **equalizer.cmd** file in the **/var/lib/radiod** directory

```
#!/bin/bash
# Change -c 0 to your alsound device number
lxterminal --command="/bin/bash -c 'sudo -H -u mpd alsamixer -c 1 -D equal'"
```

Reboot the Raspberry Pi.

```
$ sudo reboot
```

After reboot run the Alsa Equalizer as user **mpd**.

```
$ sudo -H -u mpd alsamixer -c 0 -D equal
```

If using card 1 change the **-c 0** parameter above to **-c 1**. The following screen will be displayed:

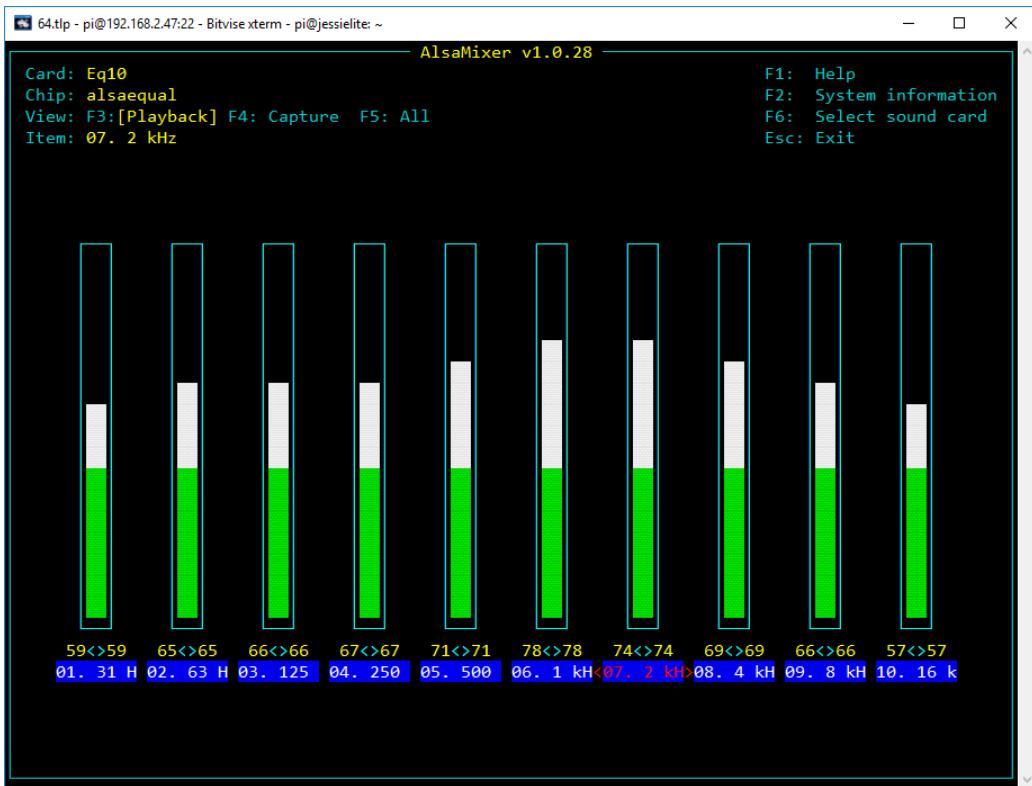


Figure 194 The Alsa

Use the Tab key to move along to the desired frequency to be changed. In this example, it is the <2KHz> block. Use the up and down arrows to adjust the level. The settings are saved in the `/var/lib/mpd/.alsaequal.bin` file. Changes to the sound should be heard.



Note: If you set a particular frequency value too high you will cause unpleasant distortion to the sound output.

Disabling the Alsa equalizer

Restore the original `asound.conf` file:

```
$ cd /usr/share/radio/asound/
$ sudo asound.conf /etc/asound.conf
```

Restore the original “device” parameter in the `audio_output` block in `/etc/mpd.conf` configuration file.

```
audio_output {
    type          "alsa"
    name          "IQAudio DAC+"
    device        "hw:0,0"
    #device       "plug:plugequal"
    mixer_type    "software"
}
```

Reboot the Raspberry Pi to restore the original sound configuration.

Configuration of the FLIRC USB dongle



Note: This configuration procedure is only for the HDMI or Touchscreen display of the radio. If using an LCD display see *Installing the Infra-Red sensor software* on page 135.

First of all, install the **FLIRC** software as shown in the section called *Installing the FLIRC USB remote control* on page 54.

Click on the left-hand program icon (A Raspberry) and select Accessories. In Accessories select **Flirc**. The following screen will be displayed. However, on a 7-inch touchscreen you may not be able to see the whole FLIRC window. In this case use the procedure called Configuring FLIRC from the command line on page 189. The first time you run this program it may ask you if you want to upgrade the firmware. Always upgrade the firmware:

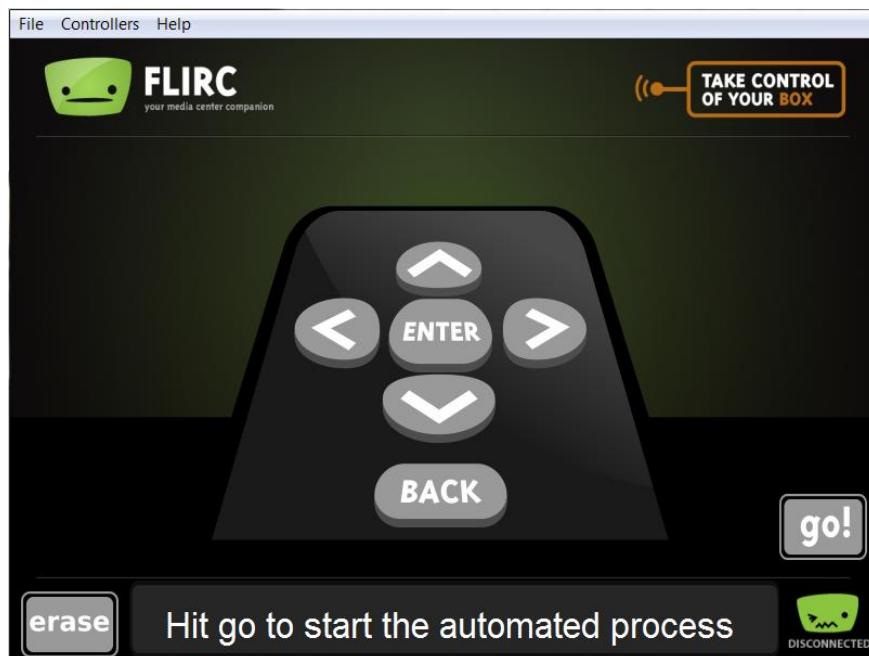


Figure 195 FLIRC setup program

On the **Controllers** drop down menu select the **Full Keyboard** controller.



Figure 196 FLIRC keyboard controller

Now map the buttons on the remote control to the keys shown in Table 22 Graphic screen keyboard command on page 207. For example, press the letter **m** on the above keyboard and then press the Mute button on the Remote Control. For volume control **up** press Shift key followed by the + key on the keyboard, then press the volume up button on the remote control. Do the same with the – key for volume down. Full instructions for configuring FLIRC are to be found at:

<https://flirc.gitbooks.io/flirc-instructions/>

Configuring FLIRC from the command line

If using a small touchscreen there may not be enough room to see the Flirc screen. If so, do the following:

- 1) Amend the **fullscreen=yes** parameter to **fullscreen=no** in **/etc/radiod.conf**
- 2) Reboot the Raspberry PI
- 3) When rebooted open a terminal session on the desktop (Don't use remote SSH).
- 4) In the terminal window on the command line run the following:

```
$ flirc_util format
```

Now record the buttons:

```
$ flirc_util record up  
Press any button on the remote to link it with '+'
```

'up' is the name of the key. Now press the Channel Up key. The following will be displayed:

```
Successfully recorded button.
```

Repeat the command for each key name.

They are:

```
pageup, pagedown,  
+, -,  
left, right,  
up, down,  
return,  
l (small letter L), p, a,  
r, t, c, s, m and d.
```

In the case of the + and – keys press shift first, followed by the + or – key.

Test and if necessary, repeat key-mapping. If configuring on an HDMI Television do not configure volume (+-) or mute (m) keys as the TV will provide these functions.

The configured keys can be displayed with the **flirc_util keys** command, however this command may be missing from the latest version of **flirc_util**.

```
$ flirc_util keys

Recorded Keys:
Index hash      key
----  -----
 0  7D14E297    down
 1  ED385097    up
 2  58C86297    right
 3  41787497    left
 4  BF8F6297    return
 5  AB616762    r
 6  2676D097    t
 7  B6536297    c
 8  9206E297    s
 9  590C3E97    l
10  E8E8D097    p
11  C49C5097    a
12  F1EFD097    e
13  B9F03963    escape
14  D1F15097    pageup
15  53DA6297    pagedown
16  9F5BE297    escape
17  A8DDF497    left_ctrl Q
18  66FFBE97    d
```

Saving the configuration:

```
flirc_util saveconfig my_flirc_config

Saving Configuration File 'my_flirc_config.fcfg' to Disk
[=====] 100%

Configuration File saved
```

There is also a **loadconfig** command.

What if a key does not work after configuring it.

First delete the key by its index. In this example the key d (Display Window) command isn't working.

```
$ flirc_util delete_index 18
```

Re-record the key

```
$ flirc_util record d
Press any button on the remote to link it with '+'
Successfully recorded button.
```

Re-test and repeat until a reliable ‘hash’ is received from the remote control.
If a key is multiply defined delete the first one you see by its index.

```
13 B9F03963 escape
14 D1F15097 pageup
15 53DA6297 pagedown
16 9F5BE297 escape
```

```
$ flirc_util delete_index 13
```

Re-test and if necessary, re-record.

There is a help facility for the flirc_utility.

```
$ flirc_util help
```

Configuring the display scroll speed

This option applies to LCDs only. It does not apply to PiFace CAD, touch screens or OLED devices.

The parameter **scroll_speed** in **/etc/radiod.conf** can be changed to speed up or slow down the scroll speed of longer display lines. The **scroll_speed** parameter is actually the inter-character delay in seconds. The smaller the value the faster the display scrolls.

The human eye can only discern 10 to 12 images a second limiting the lower range to approximately 0.08 seconds. The optimum speed is 0.2 to 0.3 seconds.

```
# Display Scroll speed 0.08 to 0.6 seconds
scroll_speed = 0.2
```

It is only possible to set the scroll speed between 0.08 and 0.6 seconds.

It may be necessary to adjust the contrast to get a good scrolling display.

Configuring Russian/Cyrillic text

The radio program can display the Russian language either in **Cyrillic** or **Romanized** (convert to Latin) characters. For example, **Радио Пятница** when Romanized becomes **Radio Pyatnica**.

First purchase a character LCD/OLED with a Russian/Cyrillic character ROM. These devices also will display English characters.

To display Russian/Cyrillic text Romanized it is not necessary to change the configuration as this is the default. To display Russian/Cyrillic change the following parameters in **/etc/radiod.conf**.

Change the language to Russian

```
language=Russian
```

Switch off Romanization

```
romanize=off
```

Unless using a HD44780U compatible controller leave the controller setting as it is

```
controller=HD44780U
```

If using an older LCD with an HD44780 (No U at the end) controller set it to HD44780

```
controller=HD44780
```

Leave the codepage setting as 0. This will pick up the correct code page from the language translation file in the **/usr/share/radio/codes** directory.

```
codepage=0
```



The **translate_lcd** parameter must also be set to **on** for Romanization or Cyrillic translation routines to work. Translation is disabled if using an OLED as OLEDs use system fonts.

Configuring European languages

First purchase a character LCD/OLED with a Western European character ROM. These devices also will display English characters. To display Western European text Romanized it is not necessary to change the configuration as this is the default. Any LCD/character OLED can be used for this.

Change the language to European and carry out the same instructions, except for language, as shown in *Configuring Russian/Cyrillic text* on page 191.

```
language=European
```

There is a detailed explanation of LCD code pages and program settings in Appendix B.5 *Cyrillic/European character LCDs/OLEDS* on page 360.

Configuring the Internet Check URL and port

Version 7.0 onwards now regularly checks to see if there is an Internet connection when playing radio streams. It is disabled when playing a USB stick or share which do not require an Internet connection.

This feature it is checking to see if there is a connection to a reliable URL and port number across the Internet which can be contacted such as **google.com** on port number 80 (HTTP). There is a distinction between a network connection and an Internet connection. A network connection is local such as Wi-Fi or a wired network switch. An Internet connection is to a network provider via an Internet router which gives access to Internet resources, in this case, radio streams.

Without it, if the radio fails to load a radio stream because it has no network interruption it attempts to switch to another stream but this is pointless if there is no Internet connection.

This is configured in **/etc/radiod.conf**.

```
internet_check_url=google.com  
internet_check_port=80
```

If Google ever change the **internet_check_url** a new one can be configured.

This feature can be disabled by removing the URL in the configuration

```
internet_check_url=
```

Configuring MPD interface

The Music Daemon listens on port 6600, and this has not changed since the introduction of the Raspbian OS (Now RaspberryPi OS) in 2013. There is therefore no need change it unless the port setting in **/etc/mpd.conf** changes (port "6600").

```
mpdport=6600
```

Configuring the remote-control parameters

The remote-control software has three parameters in **/etc/radiod.conf**:

The first parameter is for the remote control UDP server listen host, either 0.0.0.0 (All interfaces) or localhost.

The second parameter is the remote control communication port Default localhost 5100.

```
remote_control_host=localhost  
remote_control_port=5100  
remote_led=11
```

For information about the remote indicator LED see *Table 12 Remote Control Activity LED* on page 51.

Configuring the radio software UDP server listening port

This allows another host to send UDP messages to the **radiod** UDP server (not the remote control **irradiod** daemon). It is either 'localhost' or the IP address of the remote **radiod** server.

```
remote_listen_host=localhost
```

You should normally never need to change this from the default **localhost**. This parameter is only intended for any new software used to controll the radio from another host using the commands shown in Table 31 UDP messages on page 359. For example:

Remote UDP-Client → UDP network → Radio software UDP-receiver
(IP 192.168.1.66) Commands (192.168.1.203)

```
remote_listen_host=192.168.1.203
```

The remote client would send commands such as 'KEY_VOLUMEUP', 'KEY_CHANNELDOWN' etc. as shown in Table 31 to the **radiod** daemon running on host IP 192.168.1.203 (example).

Configure the I2C bus number

```
i2c_bus=1
```

This only needs to be changed to '0' for the original Model A raspberry Pi's

Configuring character LCD lines and width.

Display width, 0 use program default. Usual settings 16 or 20. Lines are normally 2 or 4.

```
display_width=20  
display_lines=4
```

These parameters are normally set by the **configure_radio.sh** program. However, there may be cases where you might want to change these. They do not affect OLED displays.

Configuring Rotary Encoders

Rotary encoders are configured by the **configure_radio.sh** script.

The settings for the **rotary_class** parameter are **standard**, **alternative**, **rgb_rotary** or **rgb_i2c_rotary**. Some rotary switches do not work well with the standard rotary class so set to "alternative" to use the alternative rotary encoder class.

For rotary encoders with RGB LEDs in the shaft set to **rgb_rotary**

For rotary I2C encoders with RGB LEDs in the shaft set to **rgb_i2c_rotary**

```
rotary_class=standard
```

KY-040 encoders etc have their own physical 10K pull-up resistors and do not need the internal GPIO pull-up resistors so the **rotary_gpio_pullup** parameter is set to **rotary_gpio_pullup=none** otherwise set to "up".

```
rotary_gpio_pullup=up
```

The **rgb_rotary** or **rgb_i2c_rotary** settings use **I2C** and RGB Led settings and should be left to the **configure_radio.sh** script to configure.

Configuring the Menu rotary switch

These settings are not used in the standard RPi Internet Radio and can be ignored for this project. They were used in the Vintage Internet radio supplement to select the menu but are more or less redundant with only a few radios using it.

```
# Menu rotary switch (optional) Normal values are 24,8 and 7 respectively.  
Value 0 disables  
menu_switch_value_1=0  
menu_switch_value_2=0
```

```
menu_switch_value_4=0
```

See <http://bobrathbone.com/raspberrypi/documents/Raspberry%20PI%20Vintage%20Radio.pdf> for more information.

Configuring soft volume (SoftVol)

Some less expensive DACs do not have any controls such as volume. This can be seen by running the **alsamixer** program. It displays the following:

This sound device does not have any controls

Being able to control the volume via the mixer controls is needed for volume control of **Airplay** and **Spotify**. However, it is possible to enable a “soft” volume control (SoftVol). First copy the distribution **softvol** **asound** definition file to **/etc/asound.conf**.

```
$ cp /usr/share/radio/asound.conf.dist.softvol /etc/asound.conf
```

The **/etc/asound.conf** configuration file should look like the following:

```
# Software volume control for card 0
pcm.softvol {
    type            softvol
    slave.pcm      "plughw:0,0"
    control {
        name          "PCM"
        card          0
    }
}

pcm.!default {
    type            plug
    slave.pcm      "softvol"
}
```

Edit **/etc/mpd.conf** file to enable the softvol plugin in and DAC.

```
audio_output {
    type            "alsa"
    name           "My DAC"
    device         "plug:softvol"
    mixer_type     "software"
    mixer_control  "DAC"
    # mixer_device   "default"    # optional
    # mixer_index   "0"         # optional
}
```

Note: “Name” can be anything and doesn’t affect the configuration.

The **alsamixer** program should now display the volume control and should allow the volume to be changed via the mixer control.

Chapter 8 – Operation

Operation of LCD and OLED versions

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. This section is for LCD versions only. For graphical radios see *Operation of HDMI and touch screen displays* on page 200.

Starting and stopping 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|info|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.

info: Show program information

version: Show the version number of the program

To start the radio:

```
$ sudo systemctl start radiod
```

To stop the radio:

```
$ sudo systemctl stop radiod
```

The following System V commands will also work:

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

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

```
$ sudo service radiod status  
● radiod.service - Radio daemon  
   Loaded: loaded (/lib/systemd/system/radiod.service; enabled; vendor  
   preset: enabled)  
     Active: active (running) since Wed 2017-11-08 10:06:19 CET; 3h 55min ago  
       Main PID: 1619 (python)  
      CGroup: /system.slice/radiod.service  
              └─1619 python /usr/share/radio/radiod.py nodaemon  
:  
{The last relevant log entries will be displayed here}
```

To see what version of the software you are running:

```
$ sudo /usr/share/radio/radiod.py version  
Version 7.5
```

To display information about the running program:

```
$ systemctl status radiod  
● radiod.service - Radio daemon  
  Loaded: loaded (/lib/systemd/system/radiod.service; enabled; vendor  
  preset: enabled)  
  Active: active (running) since Wed 2020-09-30 12:03:00 BST; 47min ago  
    Main PID: 1560 (python)  
      Tasks: 3 (limit: 2068)  
     CGroup: /system.slice/radiod.service  
           └─1560 python /usr/share/radio/radiod.py nodaemon
```

The above shows the process ID of the radio, the Music Player Daemon version and operating system details.

Push buttons or Rotary encoders operations

In the following sections there may be an instruction such as “Press left button”. If you are using rotary encoders then the following applies:

Rotary-encoder clockwise = button right
Rotary-encoder anti-clockwise = button left

Rest of page deliberately left blank

Radios with push buttons operation

The original radio has five buttons, four function buttons and one menu button. However, the new design can also support a sixth button which is the mute 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 21 on page 199.

Table 20 Push Button Operation

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



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.

Radios with rotary encoders 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 changes the display mode as shown in the following table.

Table 21 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: Radio, Media playlist, Spotify or Airplay	Volume Up Mute	Volume Down Mute	Cycle up through Airplay, Radio and Media playlists	Cycle down through Airplay, Radio and Media playlists
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 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 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.

Operation of HDMI and touch screen displays

The graphical screen

The HDMI and Touch Screen versions of the program can be started in three separate ways.

1. Automatically when starting the desktop
2. By clicking on the radio icon on the desktop
3. By manually starting the program from the command line

To start the radio from command line run the gradio.py program:

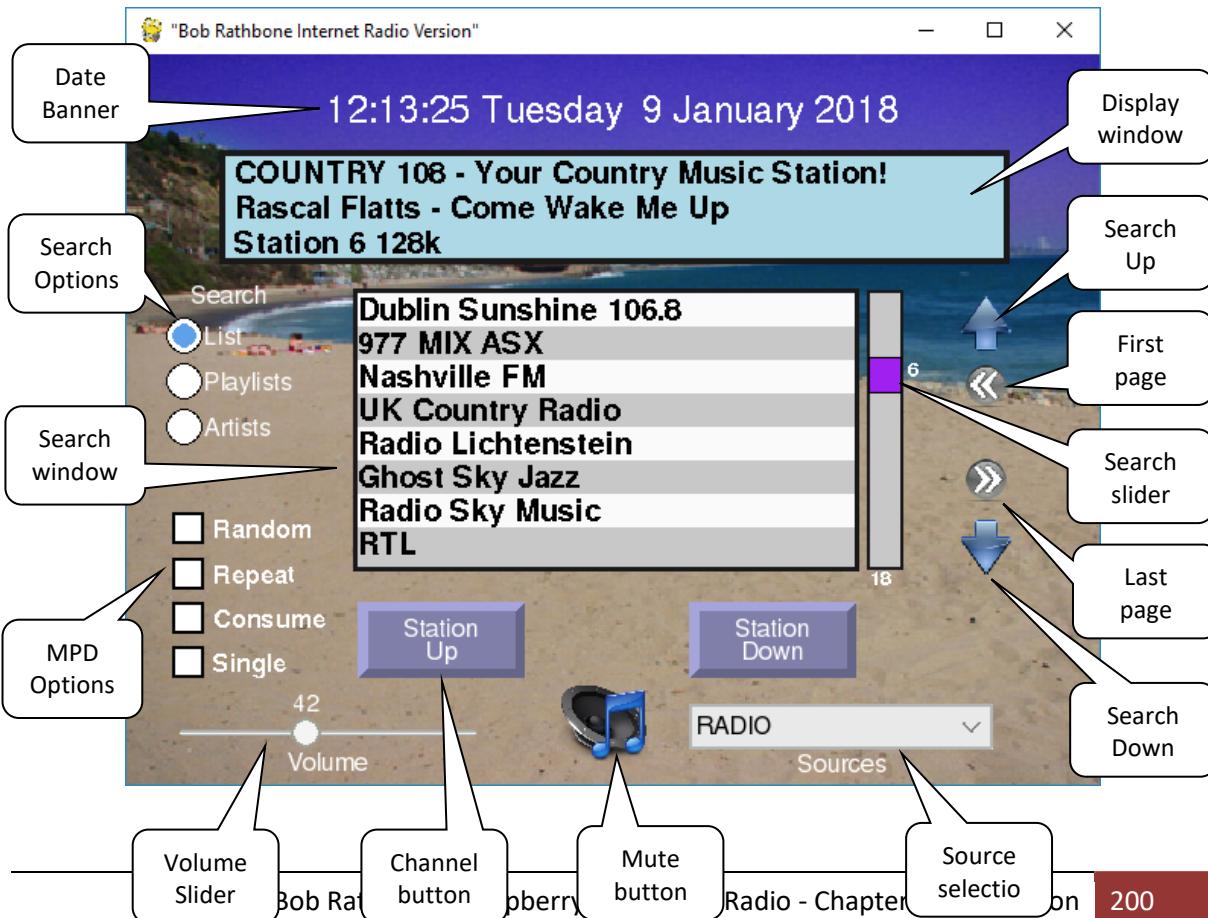
```
$ cd /usr/share/radio  
$ sudo ./gradio.py &
```

Starting the radio from the desktop. Click the icon shown here on the desktop.



In all cases a screen similar to the following will be displayed. In this example **fullscreen=no**.

Figure 197 HDMI and Touch Screen Display



Clicking the mouse on a control such as station Up/Down or touching it do the same thing. In the following description we will only refer to “clicking”. By this, also touching a control is also meant.

The display window

The display window normally displays the Radio station or Media rack that is currently playing. Clicking in the display window changes the third line to display the RSS feed if configured. A second click in the same window displays version details, IP address and hostname.

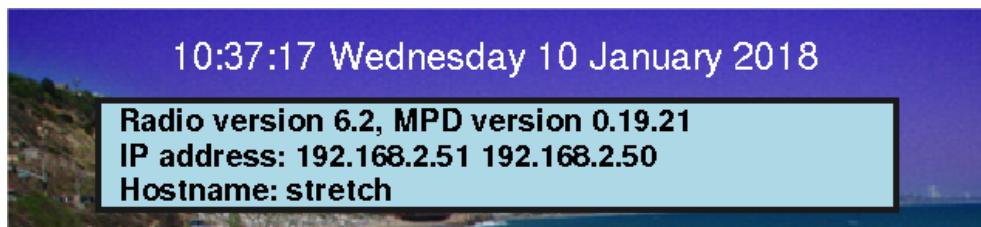


Figure 198 Graphical screen information display

In this example the hostname is ‘stretch’. The version number will be different for later releases. Two IP addresses are displayed (Wireless and Ethernet).

The search window

The search window normally displays the contents of the currently selected playlist.



Figure 199 Graphical radio search window

Click on a station in the list selects it. Clicking in the slider window or dragging the slider re-positions the list. The current position, 16 in this example, is displayed next to the slider. The length of the current playlist, 28 for this playlist, is displayed at the bottom of the slider window.

Clicking on the Up and Down arrows travels up and down the list. Clicking on the left double arrow goes to the first page in the list. Clicking on the right double arrow goes to the last page in the list.



Figure 200 Graphical radio search functions

Clicking on the Playlists radio button selects the available playlists. This shows the playlists for radio or media such as the USB stick or Network share. It also shows ‘airplay’ which is not really a playlist but is a source, but can be selected here. Click on the desired playlist in the search window.



Figure 201 Display playlists

In the following example the USB stick playlist was selected. Once a playlist selected the list is displayed.

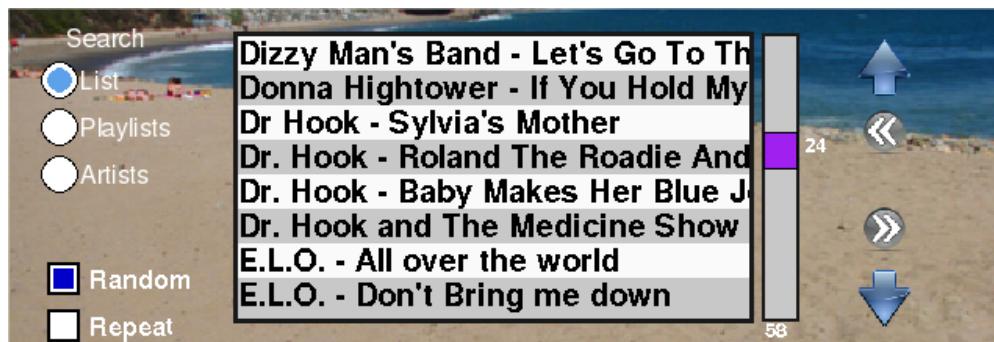


Figure 202 Display of media tracks

Clicking the Artists radio button displays the list of artists in the search window. Once clicked the search window positions on the first song of that artist’s tracks.



Figure 203 Displaying artists

Note that if you click on the 'Artists' radio button when displaying Radio stations, it will always be forced back to the 'List' display as Artist selection is not relevant for Radio stations.

Smaller TFT screens

Screens with a resolution equal to or less than 420 x 320 pixels will display slightly different than previously shown. Only one line will be displayed in the search window. There are no options for Random, Repeat or Consume due to lack of space.



The search list type (Playlist, Station/Track list or Artist) is cycled through by clicking on the Search list type button. All other controls work the same as shown in Figure 197 on page 200.

Artwork display

If the music track has artwork and the **ffmpeg** (See *Setting up the locale* on page 89) package has been installed then the artwork will be displayed. Clicking on any of the radio search buttons will re-display the search window. The artwork cannot be displayed until the track is re-selected.

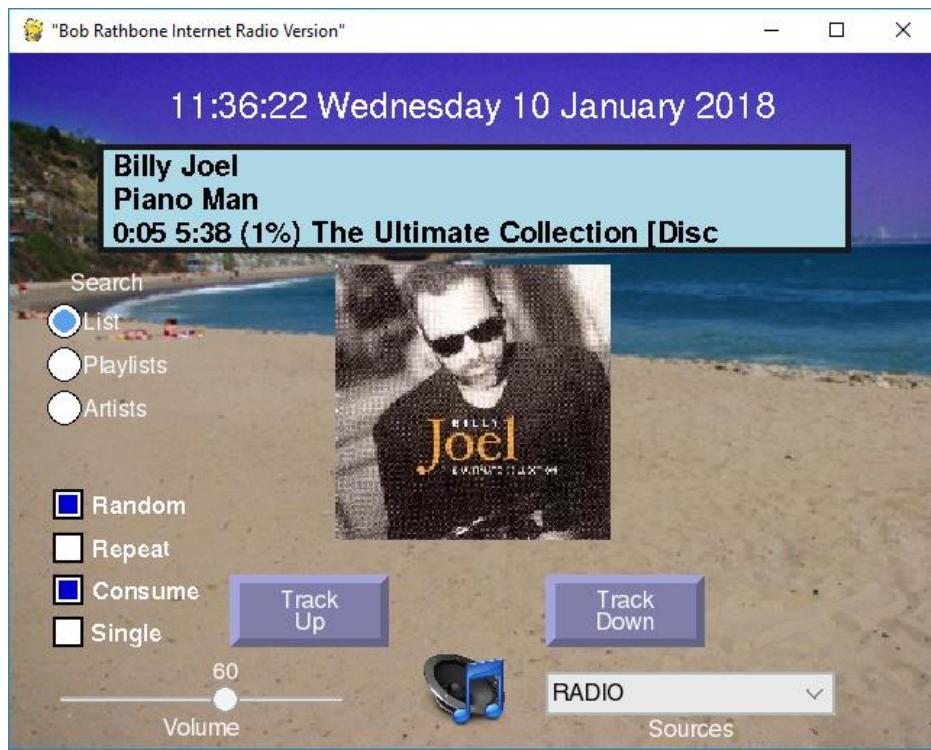
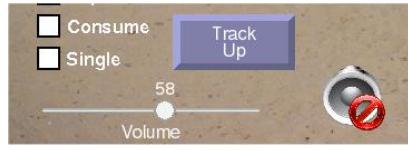


Figure 204 Track artwork display

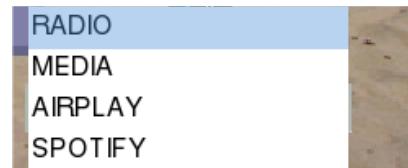
Note that the two grey push buttons now display ‘Track Up/Down’ instead of ‘Station Up/Down’.

Volume and Mute controls



The volume is controlled by a slider at the bottom left of the window. Clicking on the loud-speaker at the bottom of the screen mutes the sound and displays the mute icon as shown on the left. Any volume control change un-mutes the radio.

Source selection



Click on the down arrow on the right of the Source selection to select the Source namely Radio, Media, Airplay or Spotify. The radio will select the first playlist in that source. Re-selecting the same source will select the next playlist for that Source.

Other graphic window controls

Music Player Daemon(MPD) options Random, Repeat, Consume and Single are selected using the square push buttons on the bottom left of the window. Only the Random option is stored for the next time.

Running Airplay on the HDMI touchscreen

Airplay must first of all be installed on the Raspberry Pi. See *Chapter 12 - Setting up Airplay* on page 298 for instructions how to do this. To select Airplay either select it from the Sources drop-down box or from the playlists in the search window.

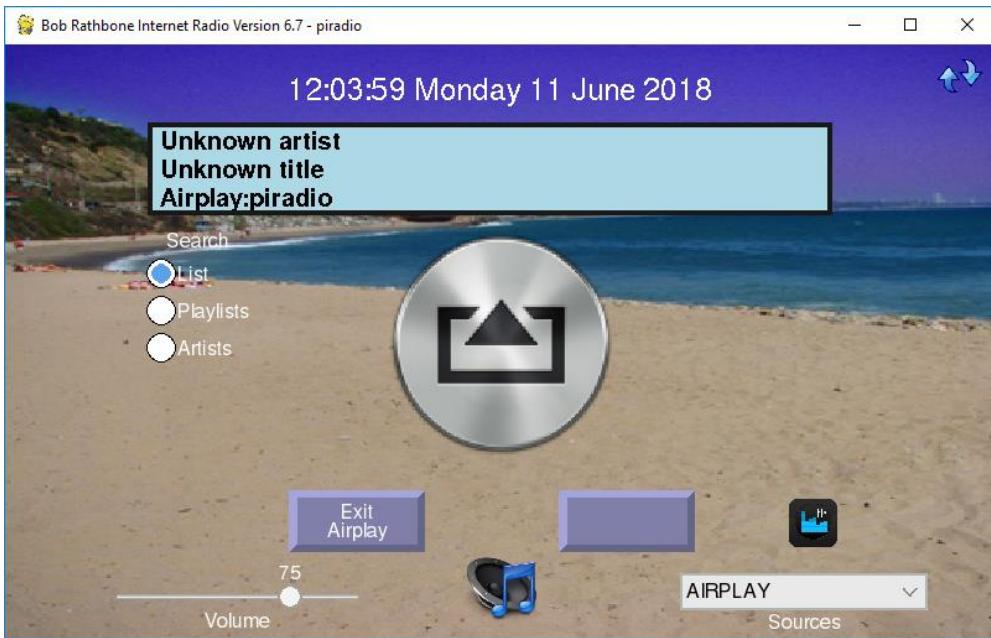


Figure 205 Airplay running on a Graphical screen

Connect to the Raspberry Pi from an Airplay compatible mobile device or run an App such as **CloudBreak**. The hostname to connect to is displayed when Airplay is first opened in the Display Window as shown below:

Unknown artist
Unknown title
Airplay:piradio

In this case the hostname is 'piradio'. To exit Airplay, press the left button at the bottom of the screen. The other button on the right has no label and doesn't do anything in Airplay mode.

Changing the graphical radio theme

The colour scheme and background are largely configurable in the [SCREEN] section of the **/etc/radiod.conf** configuration file. Button colours cannot be configured.

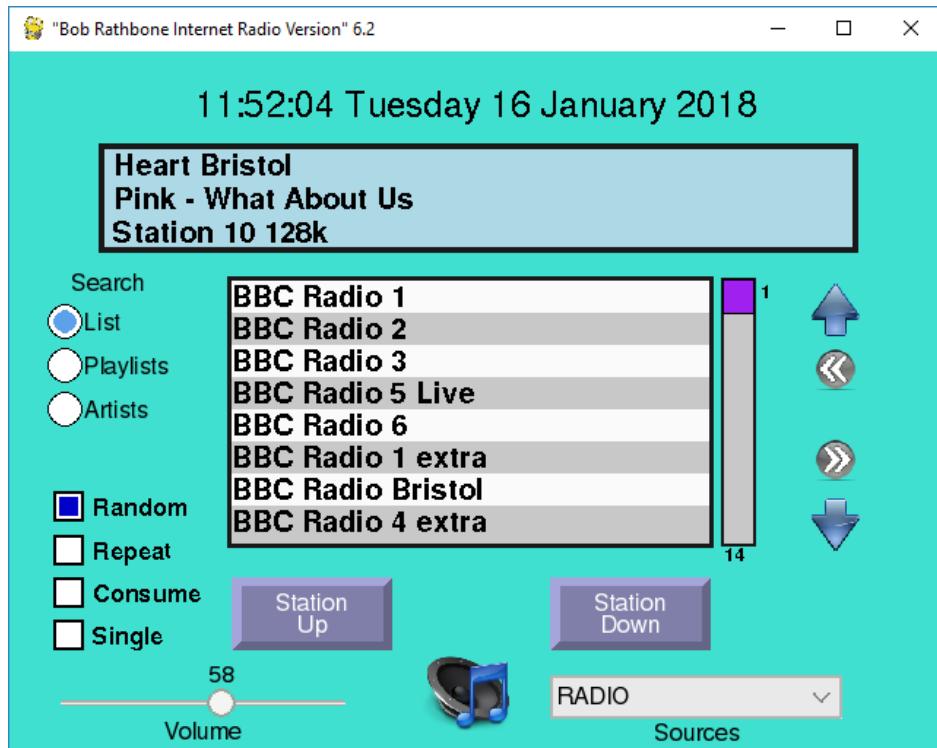


Figure 206 Changing the graphical screen theme

One good personalisation is to use your own favourite holiday picture as the background.

```
wallpaper=<path to your photograph>
```

Window and label colours can be changed to your own preferences. In the above screen the wallpaper option has been disabled, so the **window_color** option is used.

```
# Graphics (touch screen) screen settings
[SCREEN]
fullscreen=yes
window_title="Bob Rathbone Internet Radio Version"
window_color=turquoise
banner_color=black
labels_color=black
display_window_color=lightblue
display_window_labels_color=black
slider_color=purple
display_mouse=yes
switch_programs=yes
screen_saver=0

# Wallpaper backgrounds. See /usr/share/scratch/Media/Backgrounds
#wallpaper=/usr/share/scratch/Media/Backgrounds/Nature/beach-malibu.jpg
# Set date format for graphic screen
dateformat=%H:%M:%S %A %e %B %Y

# The following is specific to the vintage graphical radio
scale_labels_color=white
stations_per_page=40
display_date=yes
display_title=yes
```

Python pygame colour constants

See <https://www.webucator.com/blog/2015/03/python-color-constants-module/>

However, be aware that not all colours are supported on the Raspberry Pi version of pygame.

Graphic screen keyboard controls

The HDMI/Touchscreen version accepts input from the keyboard. It is limited and is only included as a keyboard may be connected to the Raspberry Pi when using an HDMI screen. The normal interface is either touch screen or mouse and not the keyboard.

Table 22 Graphic screen keyboard command

Key	Description	Key	Description
Page Up (PgUp)	Channel/Track Up	Up Arrow	Search Up
Page Down (PgDn)	Channel/Track Down	Down Arrow	Search Down
+ Key	Volume increase	Left arrow	Go to first search page
- Key	Volume decrease	Right arrow	Go to last search page
R	Toggle <u>Random</u>	L	Select Search <u>List</u>
T	Toggle <u>Repeat</u>	P	Select Search <u>Playlists</u>
C	Toggle <u>Consume</u>	A	Select Search <u>Artists</u>
S	Toggle <u>Single</u>	M	Toggle <u>Mute</u> on/off
D	Cycle <u>display window</u>	ESC	Exit program
X	Switch between vgradio and gradio		



You may be interested how the screen-shots in this section were created. A Windows based server called **XMing** was run on the Windows PC. X-Forwarding is then enabled in the SSH program (Putty or Bitvise etc). A SSH terminal session was started and the gradio.py program started which is then displayed on the PC desktop. Then clicking on the graphic radio window at pressing **Alt** and **PrtScn** keys together copies the window to the system clip-board where it can be pasted into a document. Operation is however very sluggish so the method is not recommended for normal use.

The Vintage Graphic Radio

As an alternative to the **gradio.py** program there is a touch-screen version of the radio called **vgradio.py**. This radio program only can play radio stations and not other Media (USB stick for example) or Airplay.



Note: This radio program can only play radio stations and not other Media such as a USB stick or Airplay, nor are there currently any plans to change this. If you want to play media you will need the full feature **gradio.py** program previously described.

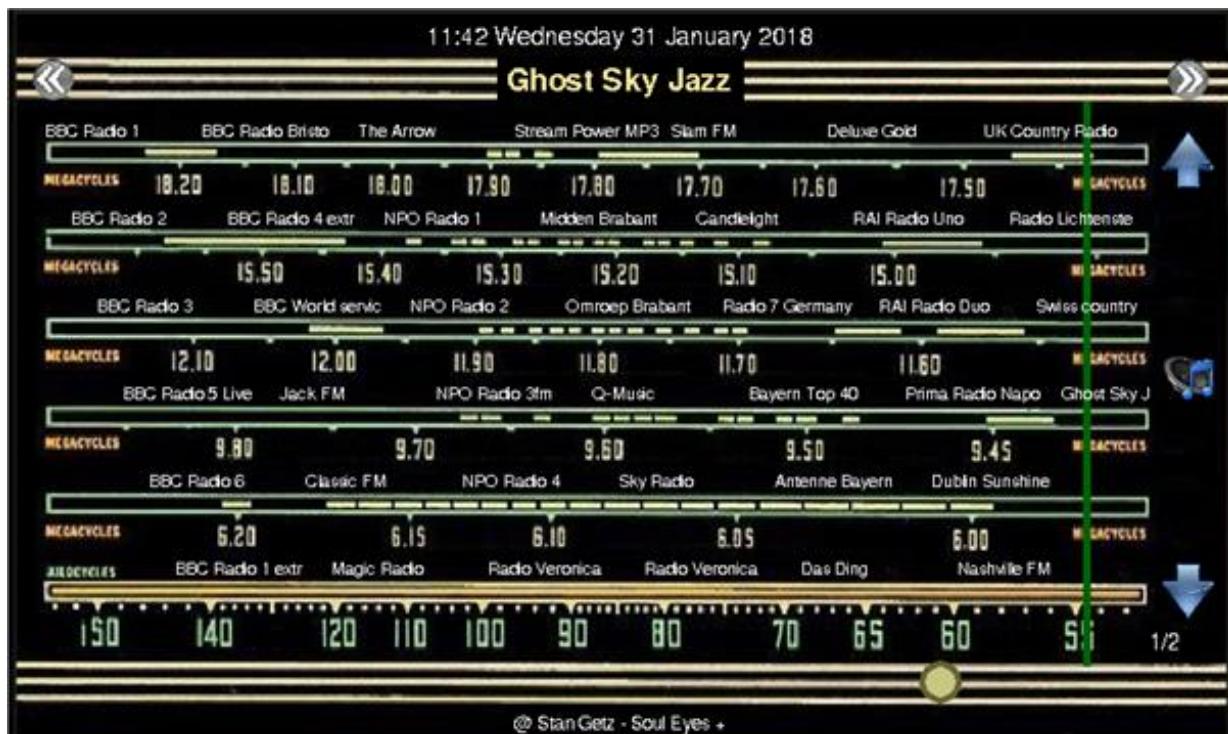


Figure 207 The vintage graphic radio on a touch-screen

This allows a radio to be constructed to look like a vintage radio with a sliding tuning dial. The pages scroll through the stations so hundreds of stations can be added. When you touch the name of a station on the tuner dial the green slider jumps to that location and plays the selected station.

The double arrows at the top of the screen allow you to page through the stations. At the bottom is the round volume slider. Under that is the title of the currently playing song. The blue arrows are used to step through the stations one at a time. The mute button is on the right-hand side of the screen. This design can also be combined with rotary encoders or switches.

To run this radio, either run the **configure_radio.sh** program or amend the **/home/pi/.config/lxsession/LXDE-pi/autostart** configuration file to run **vgradio.py** instead of **radio.py**.

```
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
@xscreensaver -no-splash
@point-rpi
@sudo /usr/share/radio/vgradio.py
```

This radio is designed to work with a single radio playlist. This is normally the **_Radio** playlist. You should configure the radio to start with this playlist by amending the **startup** parameter in **/etc/radiod.conf**.

```
startup=_Radio
```

However, this does not mean that you cannot have multiple radio playlists. If you have more than one radio playlist then by using the page up button (Double right arrow) it is possible to scroll through to the current playlist to the end and then onto the next playlist. In this case the new playlist name will be displayed in the very top-left of the screen.

You cannot currently scroll back to the previous playlist but must continue scrolling through the pages until you reached the desired playlist.

If using the FLIRC remote control dongle then it is only necessary to program the following keys: **pageup**, **pagedown**, **left**, **right**, **up**, **down**.

Switching between graphics programs

It is possible to switch between the full feature graphical radio (**gradio.py**) and the vintage graphical radio (**vgradio.py**). First configure the **switch_programs** parameter in the [SCREEN] section of **/etc/radiod.conf**.

```
switch_programs=yes
```



Restart the program. The switch icon on the left will appear towards the top of the right-hand side of the screen. By clicking on it the program will switch between the two versions of the desktop radio programs. There will be a very short pause in the music stream whilst it is doing the switch-over.

Configuring a screen saver



Note: The **xscreensaver** program described here does not appear to work if the radio program is in full screen mode. This will probably be addressed in a later release.

Modern LCD displays are not as susceptible to screen burnout as the old cathode ray tubes of old. However continuous static screen displays will eventually cause shadowing. It is therefore a good idea to install a screen saver. The standard one for **Raspbian** is called **xscreensaver**. To install it run the following:

```
$ sudo apt install xscreensaver
```

After installation of the screen saver it can be configured in the desktop preferences menu. This allows configuration of time, screen saver or a blank screen. Choose a not too busy screen saver or the blank screen option.

There is also a program called **xscreensaver-command** for command line manipulation of the screen saver. However, the advice is not to use it as, at the time of writing, it causes severe problems with both the console and desktop display.

Playing Media

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)

Playlists for all of the above can be created using the **create_playlist.sh** program.

```
$ cd /usr/share/radio  
$ sudo ./create_playlist.sh
```

See the section on *Creating Media playlists* on page 220 for a detailed description of this program.

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. Run the **create_playlist.sh** program as shown above. Reboot the PI. Once the Radio program is running again, 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 (32/64GB) SD cards now available music can be stored on the SD card. There is already a directory called **/home/pi/Music** where music can be stored.

Using FTP or any other file transfer program, copy the music from a PC to the **/home/pi/Music** directory and reload the library via the options menu. Now run the **create_playlist.sh** program. Select option 3 (SD card). See the section on *Creating Media playlists* on page 220.

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 235. Then goto the section on *Creating Media playlists* on page 220

Organising the music files

The search (find menu) routines get Artist and Track name directly from MPD which in turn get them from the music media file itself. The files should be placed in the top-level directory of the USB stick or in the **/home/pi/Music** directory if using the SD card. Any directory structure can be used. For example:

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

The find menu however will not use the directory structure for Artist/Track names so the directory structure and naming is arbitrary but should relate to the Artist and Track names displayed on the radio display. It is however possible sometimes to change the meta-data ((Artist/Track name) in the media file itself. Search on-line for ways of doing this.



Note: It is possible to create playlists for multiple locations. So, for example you can define one or more playlists for each of the USB Stick, NAS Storage and the SD card and select any of them from the radio or Web interface.

MPD Logging

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

Radio program logging

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

```
2022-03-06 06:44:35,510 INFO ===== Starting radio =====
2022-03-06 06:44:35,511 INFO Initialising radio
2022-03-06 06:44:36,148 INFO Python version 3
2022-03-06 06:44:36,155 INFO Translation code page in radiod.conf = 0
2022-03-06 06:44:36,513 INFO Display code page 0x1
2022-03-06 06:44:36,515 INFO Loaded 'codes.European'
2022-03-06 06:44:36,516 INFO Loaded 'codes.Russian'
2022-03-06 06:44:36,517 INFO Loaded 'codes.English'
2022-03-06 06:44:36,518 INFO Screen LCD Lines=4 Width=20
2022-03-06 06:44:38,293 INFO Romanize True
2022-03-06 06:44:38,811 INFO IP 192.168.1.248
2022-03-06 06:44:40,261 INFO Board revision 2
2022-03-06 06:44:40,286 INFO OS release: Raspbian GNU/Linux 10 (buster)
2022-03-06 06:44:40,299 INFO Linux piboombox 5.10.63-v7+ #1459 SMP Wed Oct 6
16:41:10 BST 2021 armv7l GNU/Linux
2022-03-06 06:44:41,055 INFO Connected to MPD port 6600
2022-03-06 06:44:41,149 INFO UDP Server listening on localhost port 5100
2022-03-06 06:44:41,156 INFO UDP listen:remote 0.0.0.0 port 5100
2022-03-06 06:44:42,851 INFO Radio ['/usr/share/radio/radiod.py',
'nodaemon'] Version 7.5
2022-03-06 06:44:42,853 INFO Radio running pid 1305
```

There are six levels of logging namely CRITICAL, ERROR, WARNING, INFO, DEBUG or NONE.

This is configured in the **/etc/radiod.conf** file. Use DEBUG for more information.

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

Other useful logs

daemon.log A very useful log for various looking at daemon process messages

For instance, if you want to see what a particular daemon is doing use grep <daemon name>.

```
grep -i <daemon name> /var/log/daemon.log
```

For example, to look at Bluetooth processes:

```
$ grep -i bluetoothd /var/log/daemon.log
Jul  8 08:59:55 buster02 bluetoothd[808]: Bluetooth management interface
1.18 initialized
:
```

Daemons of interest are **radiod**, **mpd**, **librespot**(Spotify) and **bluetoothd**.

Installation and Configuration Logs

Installation and configuration logs are stored in directory **/usr/share/radio/logs**. These are:

1. **install.log** – Output from the **configure_radio.sh** script
2. **audio.log** - Output from the **configure_audio.sh** script
3. **stations.log** – Output from **crontab** weekly run of **create_stations.py**

These logs are overwritten every time the above programs are run.

Configuration and status files

The main configuration file is `/etc/radiod.conf`. See section *A.1 Files added to the system* on page 344. This file is normally maintained by the `configure_radio.sh` program. This is run at installation time but can be safely run at any time.

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
language	Espeak language definition file
mixer_volume	Used by Airplay to set mixer volume (See page 298)
mixer_volume_id	Mixer volume ID (Used primarily for Airplay volume control)
rss	RSS feed URL
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
voice	The espeak voice file
volume	The volume setting
equalizer.cmd	The Alsa Equalizer command file called from <code>gradio</code> and <code>vgradio</code>

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 radio or configuration programs. When the radio program starts up it uses the last settings, for example, the volume setting.

Using the Timer and Alarm functions



Note: The Raspbian operating system synchronizes time over the Internet. It does this using the `timesync` service. This service is a light-weight, client only, time synchronisation service, using the Network Time Protocol (NTP). In Bullseye this is implemented the `systemd-timesyncd` service.

There is a timer (Snooze) and alarm function (LCD and OLED versions only). 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" where 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 THEREFORE RELY SOLELY ON THIS ALARM FUNCTION IF YOU HAVE AN IMPORTANT APPOINTMENT OR A PLANE OR TRAIN TO CATCH FOR EXAMPLE. ALSO SEE DISCLAIMER ON PAGE 338.

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 too 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
```

If the following is seen start **mpd** using the **systemctl** command and retry:

```
mpd error: Connection refused  
$ sudo systemctl start mpd
```

To see a list of all available commands, run:

```
$ mpc help
```

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

Table 23 Common 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. When running the radio with an Adafruit RGB plate, it is 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). Note that the program uses the American spelling ‘color’

Shutting down the radio

You can simply switch the power off. This doesn’t normally 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



Note: The creation of playlists has completely changed from earlier 5.x versions of the program. Read the following carefully for an understanding of the new playlists structure. To use old format play-lists, see the section called *Using old 5.x Radio playlists* on page 221.

Previous 5.x versions of the radio only allowed three playlists namely Radio, USB stick or Network share. This has completely changed in version 6.0 onwards. It is now possible to define as many playlists as you wish. For example:

Table 24 Example playlists

Playlist Name	Type	File name	Description
Radio	Radio	Radio.m3u	Playlist with radio stations
BBC stations	Radio	BBC_stations.m3u	Playlist with only BBC radio stations
German stations	Radio	German_stations.m3u	Playlist with only German radio stations
USB stick	Media	USB_Stick.m3u	Playlist with the contents of the USB stick
Network	Media	Network.m3u	Playlist with the contents from a network share
Country	Media	Country.m3u	Playlist with just country music
Rock and Roll	Media	Rock_and_Roll.m3u	Playlist with just Rock and Roll music

You may have as many or few playlists as you like. All playlists are stored in **/var/lib/mpd/playlists** and must have a **.m3u** file extension.



Previous to version 7.3, radio station names began with an underscore “_”. This was to distinguish between RADIO and MEDIA playlists. However, from version 7.3 onwards Radio playlists will be created without an underscore for example **Radio.m3u** instead of **_Radio.m3u**. Old playlists beginning with an underscore will be left untouched.

Creating new playlists

There are four ways to create playlists:

1. Create Radio station playlists with the **create_stations.py** program
2. Create Media playlists from either USB stick or Network share using **create_playlist.sh**
3. Use the Shoutcast (**get_shoutcast.py**) program or Web interface
4. Manual creation of your own Media playlists

The **create stations** program

The **create_stations.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 229. If you have installed the **anacron** package this program will be run on a regular basis in the background in an attempt to filter out any bad or missing stations. The directories and files used by the **create_stations.py** program are shown in the following table:

Table 25 Playlist files and directories

Name	Type	Description
/var/lib/radiod/stationlist	File	The file containing the users list of radio stations
/usr/share/radio/station.urls	File	Initial distribution Radio playlist. This is copied to /var/lib/radiod/stationlist directory during program installation.
/var/lib/mpd/playlists	Directory	Location of MPD playlists
/var/lib/mpd/music	Directory	Location of media files for either a USB stick or a Network share

The **/var/lib/radiod/stationlist** file is the file that should be maintained by you to create Radio playlists. When this *create_stations.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: **(Radio)**

The above will create a playlist called **Radio.m3u** (not _Radio.m3u as previously) 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.aspx>

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



Note: When installing the radio software for the first time a file called **station.urls** will be copied to the **stationlist** file. It will not be overwritten when upgrading or re-installing the software. The user is totally responsible for maintaining the **stationlist** file from then on.

Below is an example of part of a **stationlist** file stored in **/var/lib/radiod** directory. This file is the source of all radio playlists.

```
# Radio stations
(Radio)
# United Kingdom
# The following links are iPhone streams (m3u files)
[BBC Radio 1] http://www.radiofeeds.co.uk/bbcradio1.pls
[BBC Radio 2] http://www.radiofeeds.co.uk/bbcradio2.pls
[BBC Radio 3] http://www.radiofeeds.co.uk/bbcradio3.pls
:
# Dutch stations
(Dutch radio)
[NPO Radio 1] http://icecast.omroep.nl/radio1-bb-mp3
[NPO Radio 2] http://icecast.omroep.nl/radio2-bb-mp3
[NPO Radio 3fm] http://icecast.omroep.nl/3fm-bb-mp3
```

In the above example two Radio playlists are defined by the names in round brackets namely; (Radio) and (Dutch radio). Options can be displayed by entering the program name only.

```
$ ./create_stations.py
This program can only be run as root user or using sudo
Usage: sudo ./create_stations.py [--delete_old] [--no_delete] [--input_file=<input file>] [--help]
      Where: --delete_old   Delete old playlists
```

```
--force      Force update of MPD playlists
--no_delete  Don't delete old playlists
--input_file=<input_file>  Use alternative input file
--help       Display help message
```

The `create_stations.py` program itself is very easy to use. Just run it with `sudo` in the `/usr/share/radio` directory:

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

You may see the following message:

```
Warning: The radio is configured to allow playlist updates by external
clients so overwriting them with this program is disabled.
See update_playlists=yes in /etc/radiod.conf (Add it if it is missing)
Use the --force flag to override this restriction (Current playlists will be
overwritten)
Exiting program!
```

This is because the `update_playlists` parameter in `/etc/radiod.conf` has been set to `yes` to allow external clients such as the Web interface to add, delete and move playlist items. To force playlist creation run the program with the `force` flag.

```
sudo --force ./create_stations.py
```

Either command will create the playlist files in the `/var/lib/mpd/playlists` directory. Using the example shown above this will produce two files called `Radio.m3u` and `Dutch_radio.m3u` in the MPD playlists directory. To create a log file of the program run the following:

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

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

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

```
Processed 46 station URLs from /var/lib/radiod/stationlist
There are 2 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_stations.py --no_delete
```

It is also possible to specify a different file than `/var/lib/radiod/stationlist` using the `--input_file` parameter.

`--input_file=<station list>`

Where `<stationlist list>` is exactly the same format as `/var/lib/radiod/stationlist`.

For example:

```
$ sudo ./create_stations.py --no_delete --input_file=mystation.urls
```

In the above example the stations list is in the `mystation.urls` file.

Finally, there is a help parameter:

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

Changing the character set

The default character set (charset) used by the Music Player Daemon is called UTF-8 and is defined by the following parameter in `/etc/mpd.conf`.

```
filesystem_charset      "UTF-8"
```

This can be changed to LATIN-1 for example for Western European languages.

```
filesystem_charset      "LATIN-1"
```

More information can be found in the man page for charsets.

<https://man7.org/linux/man-pages/man7/charsets.7.html>

However, this is a global change which affects all radio stations. To tell MPD to assume a different character set for one or more specific radio stations, specify it in the charset URL fragment parameter, e.g.:

```
http://radio.example.com/stream#charset=latin-2
```

To set this up define the radio station in the `/var/lib/radiod/stationlist` file with a `charset` definition for your region as shown in the following example:

```
[Espace 2] http://stream.srg-ssr.ch/m/espace-2/aacp_96#charset=latin-1
```

When the `create_stations.py` program is run it will create the following entry in the `/var/lib/mpd/playlists/Radio.m3u` file:

```
#EXTM3U
#EXTINF:-1,Espace 2
http://stream.srg-ssr.ch/m/espace-2/aacp_96#charset=latin-1
```

As many different charsets can be defined as required by the individual radio stations. For example, a German radio station would use **latin-1** and a Polish station would use **latin-2**. If most of your radio stations are in a particular region then it is probably better to set this globally in the **/etc/mpd.conf** configuration file by changing the **filesystem_charset** parameter. For example **LATIN-1**.

```
filesystem_charset      "LATIN-1"
```

Directly creating the Radio.m3u playlist

There is nothing wrong if you chose to directly create your own MPD radio playlist instead of using the **create_stations.py** program. The correct format for a radio playlist is as shown in the following example:

```
#EXTM3U
#EXTINF:-1,BBC - Radio 1
http://stream.live.vc.bbcmedia.co.uk/bbc_radio_one
#EXTM3U
#EXTINF:-1,BBC - Radio 2
http://stream.live.vc.bbcmedia.co.uk/bbc_radio_two
```

Each entry must begin with **#EXTM3U** followed by **#EXTINF:-1,<Title>** on the next line. The third entry must be a valid URL to the radio stream.

You can choose not to use the **stationlist** file and **create_stations.py** playlist generator but there are quite a few advantages and other considerations to using the playlist generator program.

1. The syntax of the **stationlist** file is much simpler than the resulting **Radio.m3u** file.
2. The **create_stations.py** program checks the syntax of the **stationlist** entries and creates a list of errors
3. The **create_stations.py** program checks that the station definitions are valid and that a URL can be accessed over the Internet before adding it to the **Radio.m3u** file
4. A so-called cronjob called **/etc/cron.daily/radiod** runs daily to re-check the **stationlist** entries and overwrites the **Radio.m3u** file with a verified list. If you don't want your **Radio.m3u** file to be overwritten you need to remove **/etc/cron.daily/radiod** file
5. When you come to use the Web Interface you will see that if configured new entries can be added via the **O!MPD** interface and these update the **stationlist** directly and then re-runs the **create_stations.py** program

So, the advice is always to create your playlists in the **/var/lib/radio/stationlist** file and to use the **create_stations.py** playlist generator program.

If you want to define a specific character set (charset) for a station this can be done as shown in the following example below:

```
#EXTM3U
#EXTINF:-1,Espace 2
```

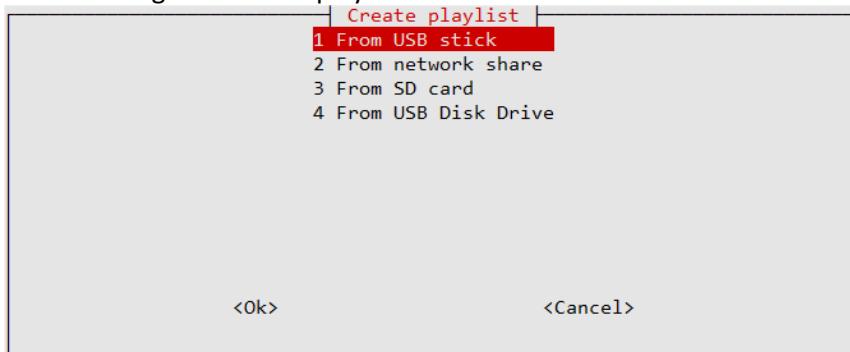
```
http://stream.srg-ssr.ch/m/espace-2/aacp_96#charset=latin-1
```

Creating Media playlists

The radio program comes with a program called **create_playlist.sh**. This creates a single playlist for a USB stick, SD card location or a network share. It can produce as many playlists as required.

```
$ cd /usr/share/radio  
$ sudo ./create_playlist.sh
```

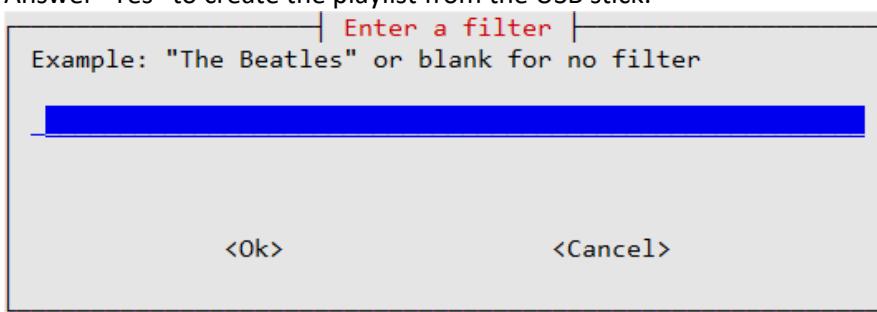
The following screen is displayed.



Select option 1 initially.



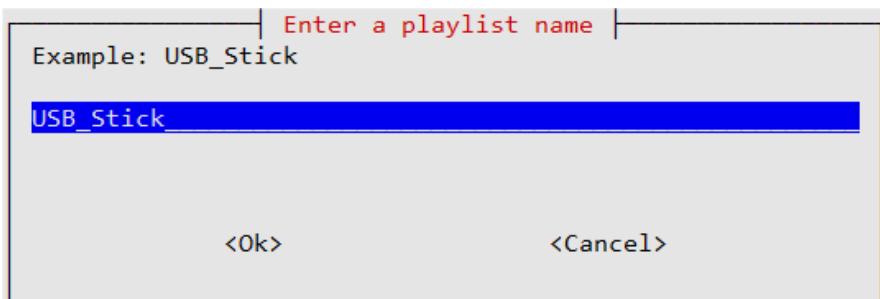
Answer "Yes" to create the playlist from the USB stick.



Enter a filter name or enter for no filter.



Select Yes to continue.

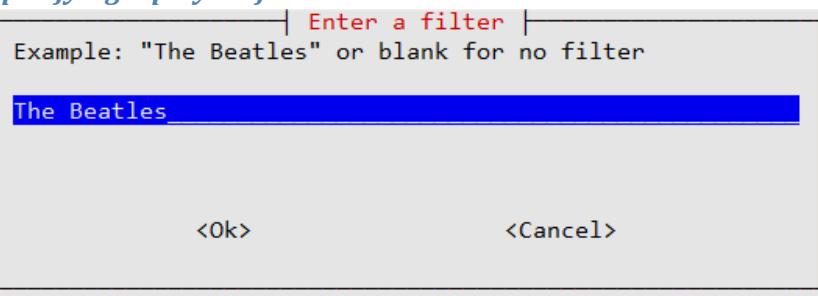


The program will suggest a name for the playlist but you may choose any name (But do not make it too long).

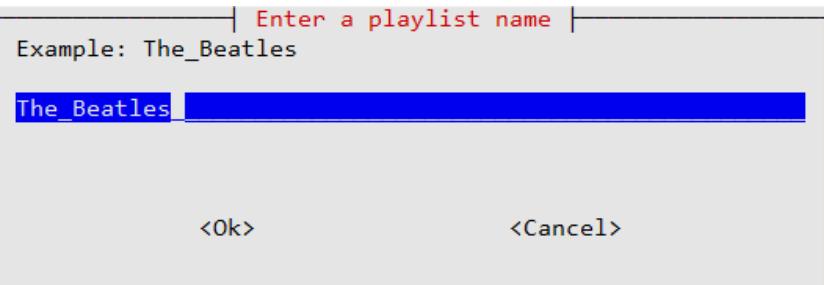
The program continues by creating a playlist called **USB_stick.m3u** in **/var/lib/mpd/playlists**.

```
sudo service radio stop
sudo service mpd stop
Mounted /dev/sda1 on /media
cd /var/lib/mpd/music/
/var/lib/mpd/music
sudo find -L media -type f -name *.mp3 -or -name *.ogg -or -name *.flac >
/tmp/list29073
sudo mv /tmp/list29073 /tmp/USB_Stick
=====
58 tracks found in directory media (No filter)
mv /tmp/USB_Stick.m3u /var/lib/mpd/playlists/USB_Stick.m3u
sudo service mpd start
mpc stop
volume: 58% repeat: off random: off single: off consume: off
mpc update media
Updating DB (#1) ...
volume: 58% repeat: off random: off single: off consume: off
```

Specifying a playlist filter



The program will then suggest the playlist name **The_Beatles**.

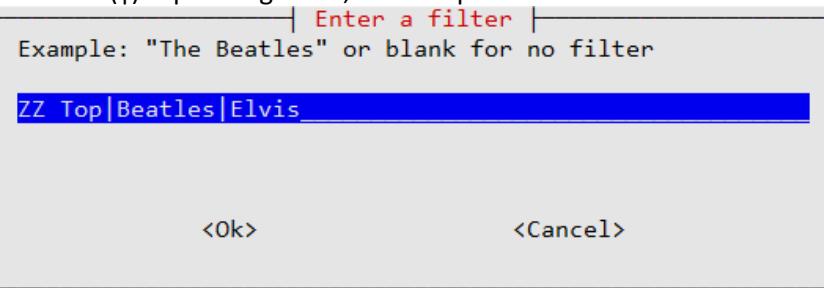


You will note that spaces in the playlist name have been replaced with underscores(_). This is just for the file name. When the playlist is displayed in the radio program the underscores will be converted back to spaces. The program will now create a playlist with the name **The_Beatles.m3u** (or whatever name was given).

```
sudo service radio stop
:
29 tracks found in directory share matching "The Beatles"
mv /tmp/The_Beatles.m3u /var/lib/mpd/playlists/The_Beatles.m3u
:
Updating DB (#1) ...
volume: 58%    repeat: off    random: off    single: off    consume: off
```

Specifying multiple filters

More than one string may be specified in a filter. To do this specify the filter strings with a pipe character (|) separating them, for example:

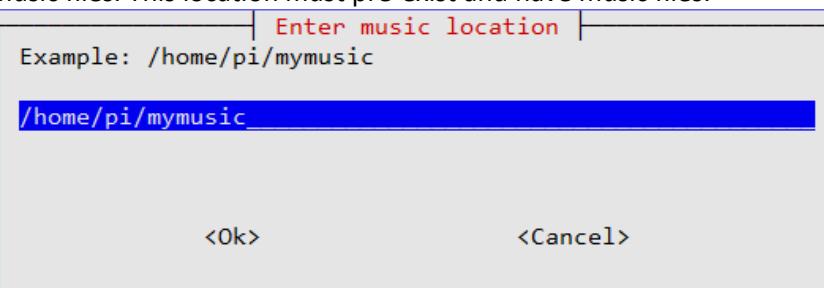


This will filter all songs from ZZ Top, The Beatles and Elvis or any other titles that contain these names. However this may not be what is wanted. Maybe songs by Elvis are wanted and not songs with 'Elvis' in the title. For example *Dire Straits – Calling Elvis*. In such a case use the / character to only look for directory names beginning with 'Elvis'. The above filter becomes:

ZZ Top|Beatles|/Elvis

Restart the radio to reload all new playlists. Using the / character gives a more accurate playlist. Please note that filters are not case sensitive. Filter 'Elvis' and 'elvis' will return the same result.

If you selected option 3 (SD card) you will be prompted for the location where you have installed your music files. This location must pre-exist and have music files.



Maintaining playlists using external MPD clients

From version 7.3 onwards it is possible to add, delete and move playlist items using any external MPD clients capable of doing so. For example the **Snoopy** or **O!MPD** Web clients supplied with the **radiod** Web package. See <https://www.musicpd.org/clients> for a list of MPD clients from the Music Player Daemon Web site.

Before it is possible to do this it is necessary to set the **update_playlists** to **yes** in the **/etc/radiod.conf** configuration file and restart the radio.

```
# Allow updating of playlists by external clients yes/no (Experimental)
update_playlists=yes
```

The current default is set to **no** to maintain backward compatibility with older versions.

The following is an example of adding a new station via the **O!MPD** Web interface.

The recommended format for a new station URL is as follows:

<url>#<name>

For example

<http://relay.181.fm:8098#181.FM Salsa>

In **O!MPD** select the NOW PLAYING tab and at the top of the page click on add. Enter the URL in the format shown above.

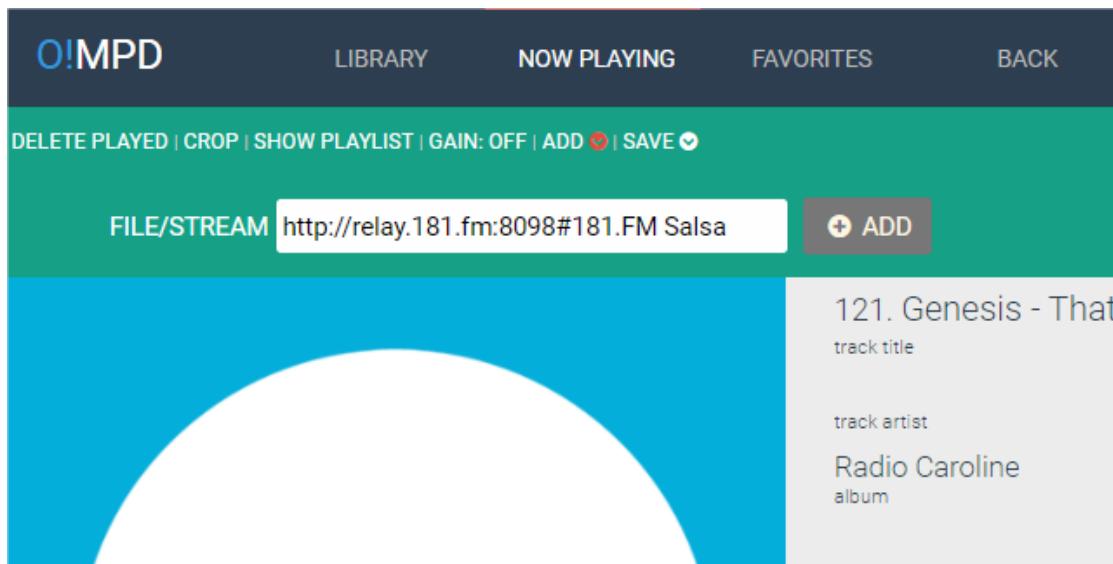


Figure 208 Adding a new station to MPD

Press the ADD button on the right of the URL entry box. The new station will be added to the end of the playlist.

It is possible to add the new station without the **#<name>** part.

For example

<http://relay.181.fm:8098>

In such a case the station will be added with the format New Station <n> where <n> is the position in the playlist. For example **New station 22**.



Figure 209 Adding new station URL in the Web interface

By clicking on the right-hand menu for the new station it is possible to move or delete a URL.

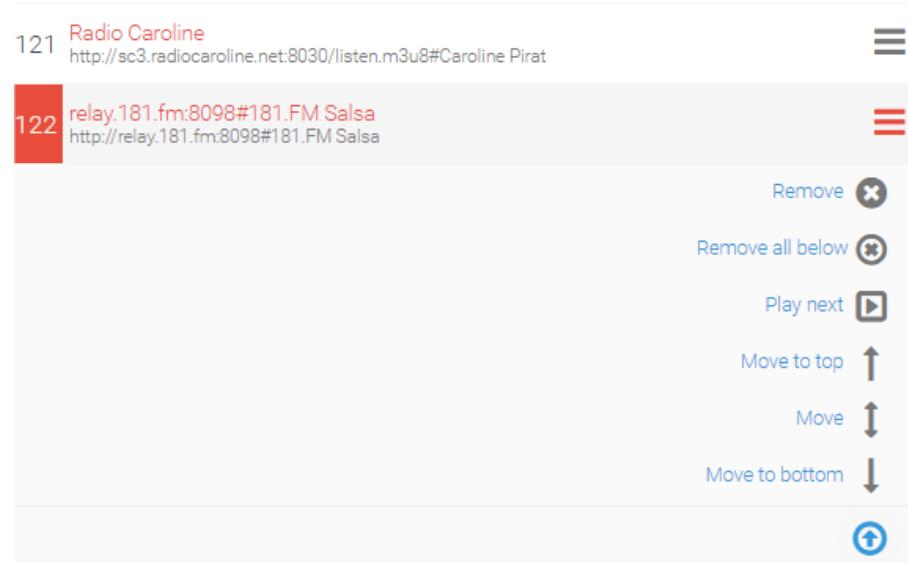


Figure 210 Moving and deleting entries in the Web Interface



Note: The author has been unable to make the double arrow **Move** icon work. Any feedback on this would be appreciated.

There is an important setting in **/etc/radiod.conf** which affects the way the station name is displayed namely **station_names**.

```
# Station names source, list or stream
station_names=list
```

This is normally set to **list** which means that the names displayed by the radio come from the entries contained in the **/var/lib/radiod/stationlist** file as shown below.

```
[181.FM Salsa] http://relay.181.fm:8098
```

In this case station URLs in the playlist use the **<url>#<name>** fomat as previously explained. By setting **station_names=stream** the station name will come from the radio stream its self.

In the above case new URLs can be added without the **#<name>** part as the name will come from the stream.

Accessing Shoutcast

It is possible to create playlists from the Shoutcast database. See <http://www.shoutcast.com>. Shoutcast provide what can best be described as “fringe” radio stations. They do have a few stations by country but not many. If you are hoping, for example, to get all the United Kingdom BBC radio stations you will be disappointed. However their support for radio stations by genre is very good, for example: rock, jazz, country or classical. This version of software provides two methods of creating playlists from the Shoutcast database:

1. Using the `get_shoutcast.py` program
2. Using the shoutcast tab in the radio Web interface.

Using the `get_shoutcast.py` program

Running the program with no parameters will produce the following usage message:

```
$ ./get_shoutcast.py
Shoutcast UDP connect host localhost port 5100
This program must be run with sudo or root permissions!

Usage: sudo ./get_shoutcast.py id=<id> limit=<limit>
search=<string> |genre=<genre> install
    Where: <id> is a valid shoutcast ID.
            <limit> is the maximum stations that will be returned
            (default 100).
            <string> is the string to search the shoutcast database.
            <genre> is the genre search string.
            install - Install playlist to MPD without prompting.

See http://www.shoutcast.com for available genres.
```

If you see the following message then install python-requests as shown below.

```
Traceback (most recent call last):
  File "./get_shoutcast.py", line 19, in <module>
    import requests
ImportError: No module named requests
```

```
$ sudo apt install python-requests
```

The program must be run with sudo. In the following example we want to get fifty jazz stations.

```
$ sudo ./get_shoutcast.py genre="jazz" limit=50
Extracting shoutcast stations: genresearch
Processing URL:
http://api.shoutcast.com/legacy/genresearch?k=anCLSEDQODrElkx1&limit=50&genre=jazz
Abc Lounge
Smoothjazz.com Global
:
:
Created 50 records in /usr/share/radio/playlists/_jazz.m3u
Do you wish to copy this playlist to /var/lib/mpd/playlists [y/n]: y
```

Answer ‘y’ to install the new playlist.

```
Copied /usr/share/radio/playlists/_jazz.m3u to /var/lib/mpd/playlists
Reload playlists: OK
```

This will copy the new **_jazz.m3u** playlist to the **/var/lib/mpd/playlists** directory. The program will also signal the radio program to reload its playlists so that the new playlist can be accessed straight away.

If you answer ‘n’ to the above question your new playlist will be saved in **/usr/share/radio/playlists** repository. You can copy it later, if so wished, to the MPD playlists directory.

```
$ cd /usr/share/radio/playlists
$ sudo cp _jazz.m3u to /var/lib/mpd/playlists
```

You must use **sudo** to do this.

The program requires an authorisation key. This is embedded in the program and it is not necessary to specify it. If it ever changes or expires a new authorisation key must be configured in **/etc/radiod.conf**.

```
shoutcast_key=anCLSEDQODrElkx1
```

You need to get this key directly from <http://www.shoutcast.com>.



Note: Access to the Shoutcast is a free service made available through the goodwill of the folks at Shoutcast. It can be withdrawn at any time if over-used or abused so please do not set up any facility, such as scripting, which will stress their servers.

Using the Shoutcast Web Interface

The radio Web interface now has a Shoutcast tab. Click on Shoutcast tab to open the interface. Fill in the search form and press the Submit button once and wait until the summary page is displayed.



Figure 211 Shoutcast playlist Web page

Select the search type Top 500, Genre or Search from the “search type” drop down box. Select a “limit” for the search then click the “Submit” button. Normally the message “Please wait for selection” is displayed along with a rotating circle wait gif graphic. However, this does not work with **Microsoft Edge** and it is recommended you use the **Firefox** browser.



Figure 212 Shoutcast search selection

The summary page will be displayed. You should see the **Reload playlists: OK** message which means that the new playlist is available in the radio.



Figure 213 Shoutcast playlist summary

Using old 5.x Radio playlists

Old 5.x playlists are not compatible with this version of the radio. However, the **/var/lib/radiod/stationlist** file can still be used. Do the following:

1. Stop the radio
2. Copy the old **stationlist** file to **/var/lib/radiod/stationlist**
3. Remove most of the (title) statements from the **/var/lib/radiod/stationlist** file
4. Remove all old playlists from **/var/lib/mpd/playlists** directory

```
$ sudo rm /var/lib/mpd/playlists/*
```

5. Run the **create_stations.py** program as previously described.

If you find upon running the Radio that you have a lot of radio playlists. Reduce the number by removing title statements in the **stationlist** file as previously mentioned. These are the names in brackets – for example (BBC Radio). Re-run the **create_stations.py** program.

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.

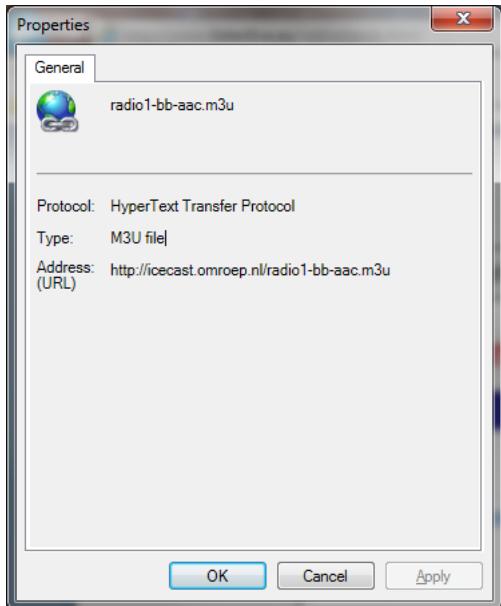
<http://radiomap.eu/>

<http://www.publicradiofan.com>

<http://www.radio-locator.com>

<https://www.internet-radio.com/>

Getting a radio stream from a Web browser



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 illustration on the left 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 be useful when 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 M3U playlist file (MPEG3 URL). This format is used by MPD.
2. A URL pointing to an HLS (HTTP Live Streaming HLS - M3U8) playlist file
3. A URL pointing to a PLS playlist file (Shoutcast Play List)
4. A URL pointing to an ASX playlist file (Advanced Stream Redirector)
5. 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_stations.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.

M3U and M3U8 Files

M3U stands for MPEG3 URL. This is the format that MPD itself uses. The following Wikipedia article explains the M3U and M3U8 file formats:

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

These playlist files have the **m3u** or **m3u8** (UTF-8 encoding) 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. You do not need to create this type of file yourself. Modify the **stationlist** file and run **create_stations.py**.

M3U files may also contain a simple list of file paths to media files. For example:

```
media/Steve Miller Band/Album onbekend/0726 Steve Miller Band - The
Joker.mp3
media/Stories/Album onbekend/Stories - Brother Louie.mp3
:
```

In this version of the radio the program knows that these are media files as opposed to radio playlists because they do not start with an underscore '_' which is the convention that the radio program uses for a radio playlist (It is not a general convention).

Note that in the above example **media** is a directory (or a link to it) in the **/var/lib/mpd/music** directory and that the '/' character is omitted.

PLS file format

A good place to start is the following Wikipedia article:

[http://en.wikipedia.org/wiki/PLS_\(file_format\)](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 always starts 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 is a **File_n**, **Title_n** and **Length_n** where *n* is the entry number.

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&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/>

The screenshot shows a web browser window with the URL <http://www.liveatc.net/search/> in the address bar. The page title is "Airport Detail: EHAM | Live...". The left sidebar contains links for "Site-Wide search", "Browse LiveATC Feeds", "LiveATC Coverage Map", "Top 50 LiveATC Feeds", "Bad Weather Areas", "LiveATC FAQ", "Offer a LiveATC Feed", "Contact LiveATC.net", "Press Inquiries", "LiveATC on iPhone", "LiveATC on Android", "Windows Phone", "Windows 8/10", "LiveATC Mobile (Mobile browser)", "ATC Audio Archives", "Interesting Recordings", "LiveATC Forums", "Twitter | Facebook", and social media icons for YouTube, Google+, and Facebook.

The main content area displays information for "Europe" and "EHAM". It includes the "EHAM METAR Weather" (130925Z 26014KT 230V290 9999 FEW021 BKN027 19/14 Q1021 NOSIG), "EHAM Flight Activity (FlightAware)", "EHAM Webcam: (Airport Webcams)", and "EHAA Radar ARTIP". The "Feed Status: UP" and "Listeners: 1" are shown. There are three "LISTEN" buttons for different media players: Flash, MP3 player, and Windows Media Player. Below this, there is a table for "EHAA Radar ARTIP Audio Archives" with one entry: Facility (EHAA Radar Sector 1 ARTIP) and Frequency (120.5500). Further down, there is another section for "EHAA Radar East Inbound" with similar details and a table for "EHAA Radar East Inbound Audio Archives".

Figure 214 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.

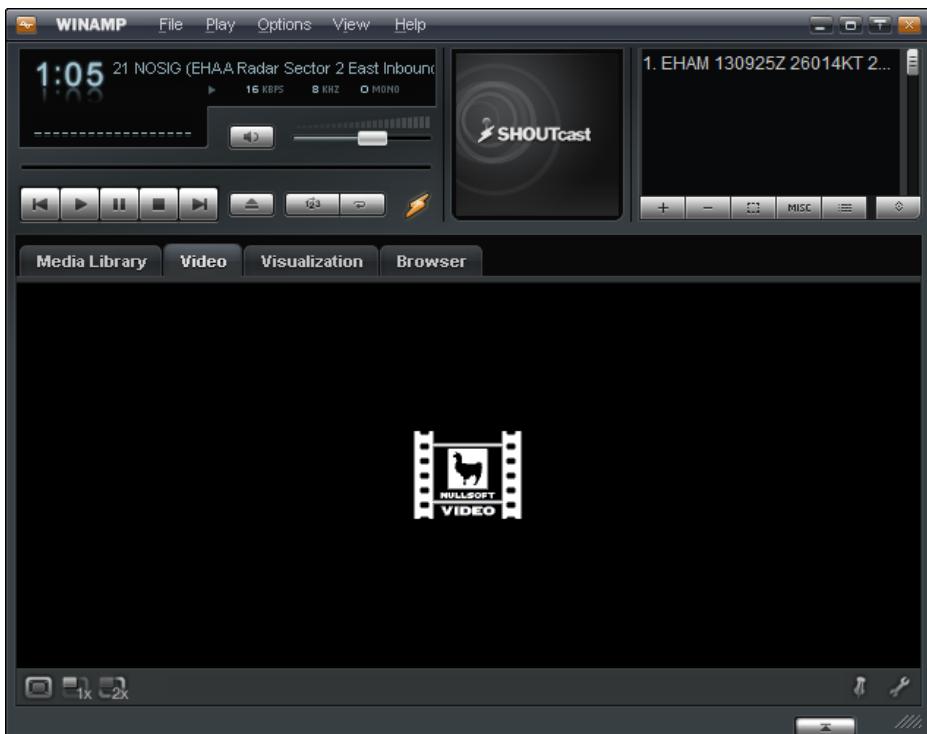


Figure 215 WinAmp playing ATC live feed

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

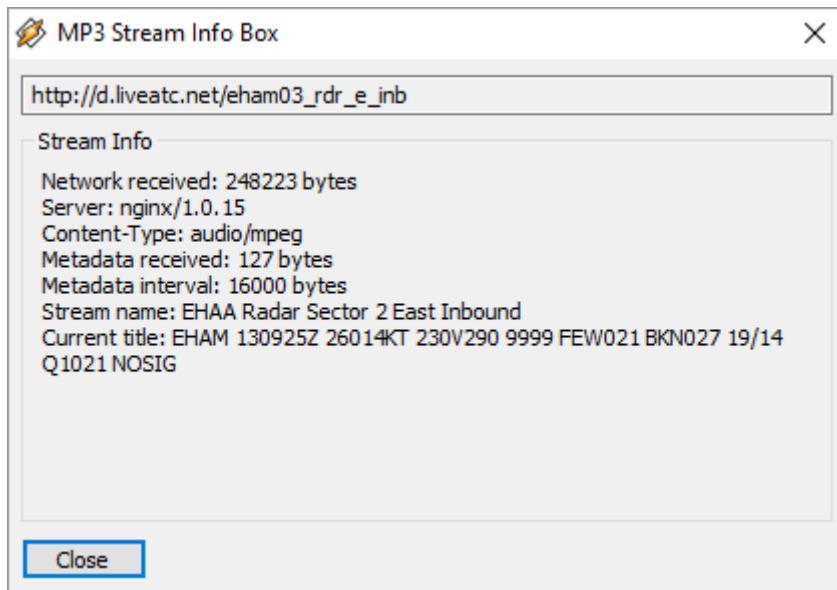


Figure 216 WinAmp station information

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.

```
(Air traffic)
[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/eveh2_app
[CYYT] http://d.liveatc.net/cyyt
[KJFK Gnd Tower] http://d.liveatc.net/kjfk_gnd_twr
[CYYZ Tower] http://d.liveatc.net/cyyz7
```

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

```
$ cd /usr/share/radio
$ sudo ./create_stations.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.dixwx.com/wxdecoding.htm> or search for ATIS/TAF/METAR decode.

In the following example:

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

EHAM	Amsterdam Schiphol, the Netherlands
130955Z	13 th 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. Dew point 14 degrees Celsius
Q1021	Barometric pressure 1021 Millibars (Will be given in Inches Mg in US airports)
NOSIG	No significant weather.

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 Buster 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 were 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,vers=1.0
//192.168.1.6/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 **vers** statement is the CIFS version and can be 1.0, 2.0 or 3.0 depending upon the NAS storage. 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 236.

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 192.168.1.6:/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  
//192.168.1.6/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.

Loading the media playlists

Now run the radio program. The radio stations will normally be loaded. Cycle through the menu by pressing the menu button until **Input Source:** is displayed. Press the channel up or down buttons or rotate the Channel knob to select one of the media playlists previously created using the **create_playlist.sh** program as shown in the section *Creating Media playlists* on page 220.

Once the desired playlist has been selected, press the **Menu** button. The radio program loads the selected playlist and starts playing a track from it. The program does not know the last track number that was played previously for the selected playlist and will select the track number that it finds in the **/var/lib/radiod/current_track** file. If this track number is bigger than the newly selected playlist length it will reset the track number to 1.

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).

Create a Playlist for the share

Now create a playlist for the new network share. See *Creating and Maintaining Playlist files* on page 215. DO NOT FORGET TO DO THIS.

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,vers=1.0  
//192.168.1.6/music /share
```

Further information

Mount points

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. You may also see a link called **sdcards** to the location of the music library on the SD card.

```
$ ls -la /var/lib/mpd/music/total 8  
drwxr-xr-x 2 root root 4096 Jul 7 12:37 .  
drwxr-xr-x 4 mpd audio 4096 Apr 7 11:02 ..  
lrwxrwxrwx 1 root root 6 Jul 7 12:17 media -> /media  
lrwxrwxrwx 1 root root 16 Jul 7 12:37 sdcards -> /home/pi/mymusic  
lrwxrwxrwx 1 root root 6 Jul 7 12:17 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  
$ sudo ln -s /media  
$ sudo ln -s /share  
$ sudo ln -s /home/pi/mymusic sdcards
```

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

Troubleshooting mount problems

See section called *Cannot mount remote network drive* on 258.

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 **MPDdroid** See the following link: <https://github.com/abarisain/dmix/releases> or download from the Android Play Store.

MPDdroid 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 MPDdroid 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.

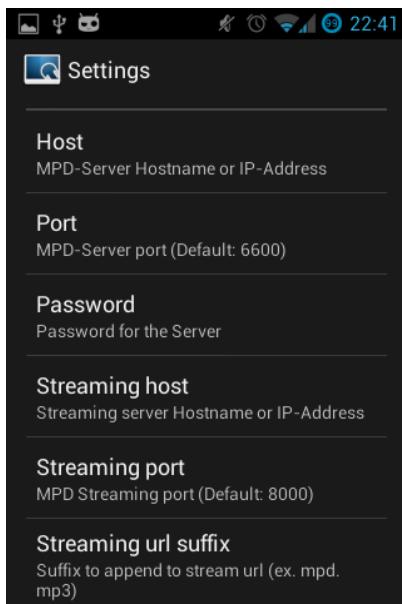


Figure 217 MPDdroid set-up screen

Keep pressing the back button to exit and then restart the MPDdroid App. The play screen should be displayed as shown below. Volume, pause, fast forward/back can all be controlled from this screen.

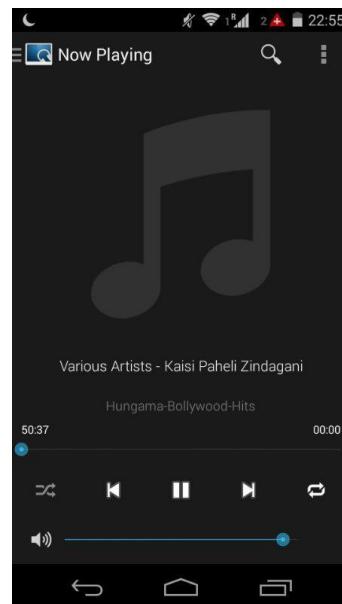


Figure 218 MPDdroid play screen

To switch to the playlist, 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.

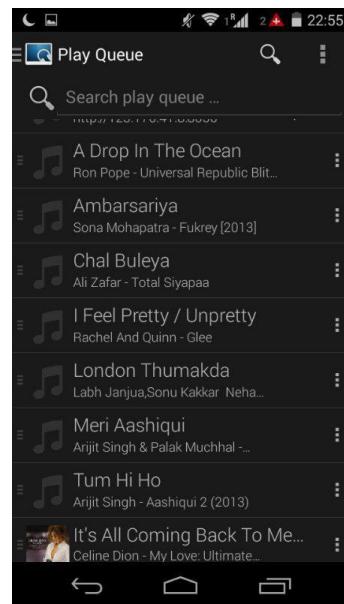


Figure 219 MPDdroid play queue



Note: MPDdroid 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.

Chapter 9 -Troubleshooting

Also see the section called *Using the diagnostic programs* on page 272. If you need to create a log file in DEBUG mode see the procedure for doing this on page 280.

Boot problems

Newly created desktop version of Raspberry Pi OS will not boot

This is a very recent problem. After creating a new SD card using Raspberry Pi OS (32 bit) dated 1st January 2022 the Raspberry Pi will not boot.

This appears to have started in March 2022. It even seems to be trying to reboot several times. This has been experienced on the Raspberry Pi 4B. Other models may also be affected. This appears to be a problem with new DRM VC4 V3D screen driver (**vc4-kms-v3d**).

Insert the SD card into a PC. The PC will display a boot partition as shown in the example below:

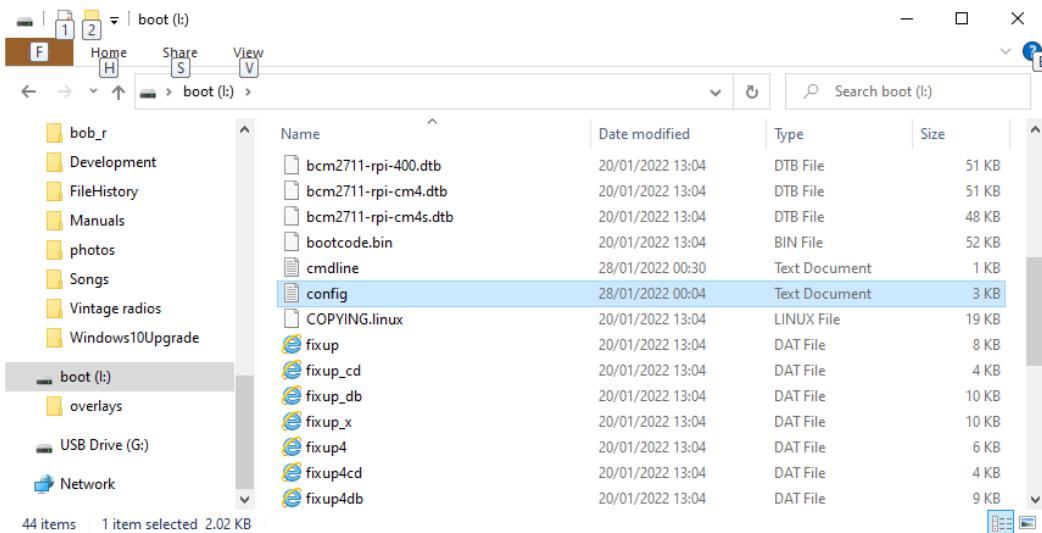


Figure 220 SD card boot partition

In the above example this is on drive I: but will most likely be different on your PC. Find the config file (config.txt) on the boot drive and open it with **Notepad** or any other text editor. Find the entry **dtoverlay=vc4-kms-v3d**.

```
# Enable DRM VC4 V3D driver
#dtoverlay=vc4-kms-v3d
max_framebuffers=2
```

Either disable the driver by putting a # character at the beginning of the line or use the older **vc4-fkms-v3d** **dtoverlay** as shown below

Using the older **vc4-fkms-v3d** **dtoverlay**:

```
dtoverlay=vc4-fkms-v3d
```

Save the file and try booting off the modified SD card. All being well it should reboot OK. If not look for other causes as shown in the rest of this section.

The Raspberry Pi will not boot from a previously good SD card

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 and attach a USB keyboard. 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:~#
```

You are asked to press enter which brings up the dollar \$ prompt

```
Cannot access console, press enter to continue  
$
```

Enter a root password that you can remember.

```
$ sudo passwd root  
Enter new UNIX password:  
Retype new UNIX password:  
Password successfully changed  
$ sudo reboot
```

When reboot has finished you will be asked to enter the root password you just entered which will take you to the # prompt. Then run the following commands:

```
# umount /  
# 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=yes
```

Finally reboot the Raspberry Pi.

```
$ sudo reboot
```

Bad SD card

If experiencing boot problems, a bad SD card cannot be ruled out. If all of the above efforts fail to resolve the problem try installing the **Raspberry Pi OS** on a different SD card.

Installation problems

Missing package dependency problems

If an attempt is made to install the radio software without first installing **mpd** and **python3-mpd** the following maybe be seen:

```
dpkg: dependency problems prevent configuration of radiod:  
  radiod depends on python-mpd; however:  
    Package python3-mpd is not installed.  
  radiod depends on mpc; however:  
    Package mpc is not installed.  
  radiod depends on mpd; however:  
    Package mpd is not installed.
```

The following may also be seen:

```
radiod depends on python2; however:  
  Package python2 is not installed.  
radiod depends on python-configparser; however:  
  Package python-configparser is not installed.
```

To correct this run:

```
$ sudo apt install python3-mpd  
$ sudo apt install python-configparser python2
```

Then run

```
$ sudo apt install --fix-broken
```

Confused or unsure of wiring

All wiring is configurable in **/etc/radiod.conf**. The physical wiring must match the configuration in the configuration file. Run the **wiring.py** program (See page 276). Adapt either the configuration or wiring to match each other. The configuration in **/etc/radiod.conf** uses GPIO numbers and not physical pin numbers. If you are unsure how you have wired a button or rotary encoder this can be confirmed by running the **test_gpios.py** program. See *The test_gpios program* on page 278.

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 ...  
:
```

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

Network problems

Checking the network quality

Connect a screen or a TV and keyboard to the Raspberry Pi. Check the network with the **ip** tool. Run **ip addr** or press the menu button until the information screen is displayed with the IP address(es).

```
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
    default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
    default qlen 1000
        link/ether dc:a6:32:05:36:9b brd ff:ff:ff:ff:ff:ff
        inet 192.168.1.152/24 brd 192.168.1.255 scope global noprefixroute eth0
            valid_lft forever preferred_lft forever
        inet6 fe80::fe6a:9f30:4326:57d1/64 scope link
            valid_lft forever preferred_lft forever
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
    UP group default qlen 1000
        link/ether dc:a6:32:05:36:9c brd ff:ff:ff:ff:ff:ff
        inet 192.168.1.153/24 brd 192.168.1.255 scope global noprefixroute wlan0
            valid_lft forever preferred_lft forever
        inet6 fe80::4c8b:8a00:e1d1:cd27/64 scope link
            valid_lft forever preferred_lft forever
```

The above example shows the Wi-Fi and Ethernet IP addresses.

Display the route to the Router using **ip route**.

```
$ ip route
default via 192.168.1.254 dev eth0 proto dhcp src 192.168.1.152 metric 202
default via 192.168.1.254 dev wlan0 proto dhcp src 192.168.1.153 metric 303
192.168.1.0/24 dev eth0 proto dhcp scope link src 192.168.1.152 metric 202
192.168.1.0/24 dev wlan0 proto dhcp scope link src 192.168.1.153 metric 303
```

The above example shows the route between Wi-Fi and Ethernet interfaces and the IP address of the Router (192.168.1.254).

If the problem is with Wi-Fi check the **/etc/wpa_supplicant/wpa_supplicant.conf** configuration file.

```
$ cat /etc/wpa_supplicant/wpa_supplicant.conf
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=GB

network={
    ssid=<Your-SSID>
    psk=<Your-Router-Password>
    key_mgmt=WPA-PSK
}
```

Checking the Wi-Fi signal strength

If using the Wi-Fi interface, the signal strength received by the Raspberry Pi can be checked either one of the readily available mobile Wi-Fi signal apps or with the **display_wifi.sh** program supplied with the radio software. Good Wi-Fi signal strength is required for the operation of the radio. Run the following commands:

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

This will display the main information about the Wi-Fi signal.

```
Wi-Fi network information  
-----  
Hostname: pizen  
Wi-Fi: wlan0 ESSID:"EE-Hub-4qM4"  
IP address wlan0: 192.168.1.95  
wlan0 Protocol Name:"IEEE 802.11"  
Current Frequency:2.437 GHz (Channel 6)  
Bit Rate=72.2 Mb/s Tx-Power=31 dBm  
Link Quality=70/70 Signal level=-26 dBm  
Power save: on  
0: phy0: Wireless LAN  
    Soft blocked: no  
    Hard blocked: no  
1: hci0: Bluetooth  
    Soft blocked: no  
    Hard blocked: no
```

In the above example the name of the router (ESSID: TP-Link), the Wi-Fi interface (wlan0) IP address, protocol, frequency and channel are displayed. Lastly the really important information is the Tx-Power value in Decibels (dBm). If phy0 is blocked use “rfkill unblock 0” to enable it.

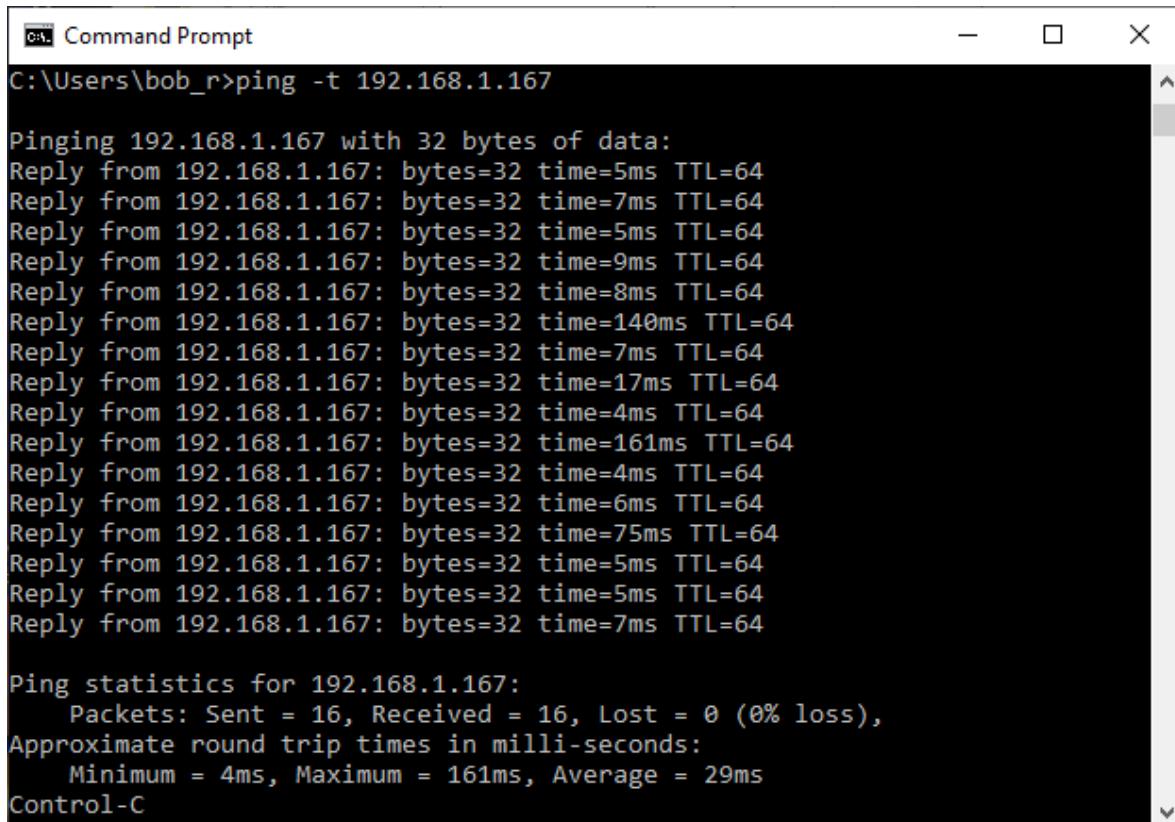
Table 26 Wi-Fi signal strength

Signal strength	Signal quality	Notes
-30 dBm	Maximum signal strength, usually attained if the RPi is very close to the Wi-Fi access point	Excellent for all services
-50 dBm	Anything down to this level can be regarded as excellent signal strength	Excellent for all services
-60 dBm	This is still a good, reliable signal strength	Good for all services
-67 dBm	This is the minimum value for all services that require smooth and reliable data traffic.	Minimum required for streaming services including for the radio
-70 dBm	The signal is not very strong, but mostly sufficient.	Only good for Web browsing, email, and the like
-80 dBm	Minimum value required to make a connection. You cannot count on a reliable connection or sufficient signal strength to use services at this level	The radio will not work reliably at this level
-90 dBm	It is very unlikely that you will be able to connect or make use of any services with this signal strength.	The radio cannot operate reliably at this level.

If the signal strength is poor, try re-positioning the radio nearer the router or use a repeater (TP-Link or similar). Position away from radiators or other large metal objects such as fridges and freezers.

Checking response times with ping

Check the response times using the ping command from a PC or another Raspberry Pi. If possible, connect the PC/RPi via an Ethernet cable to the router rather than via Wi-Fi.



```
C:\Users\bob_r>ping -t 192.168.1.167

Pinging 192.168.1.167 with 32 bytes of data:
Reply from 192.168.1.167: bytes=32 time=5ms TTL=64
Reply from 192.168.1.167: bytes=32 time=7ms TTL=64
Reply from 192.168.1.167: bytes=32 time=5ms TTL=64
Reply from 192.168.1.167: bytes=32 time=9ms TTL=64
Reply from 192.168.1.167: bytes=32 time=8ms TTL=64
Reply from 192.168.1.167: bytes=32 time=140ms TTL=64
Reply from 192.168.1.167: bytes=32 time=7ms TTL=64
Reply from 192.168.1.167: bytes=32 time=17ms TTL=64
Reply from 192.168.1.167: bytes=32 time=4ms TTL=64
Reply from 192.168.1.167: bytes=32 time=161ms TTL=64
Reply from 192.168.1.167: bytes=32 time=4ms TTL=64
Reply from 192.168.1.167: bytes=32 time=6ms TTL=64
Reply from 192.168.1.167: bytes=32 time=75ms TTL=64
Reply from 192.168.1.167: bytes=32 time=5ms TTL=64
Reply from 192.168.1.167: bytes=32 time=5ms TTL=64
Reply from 192.168.1.167: bytes=32 time=7ms TTL=64

Ping statistics for 192.168.1.167:
    Packets: Sent = 16, Received = 16, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 161ms, Average = 29ms
Control-C
```

Figure 221 The ping command

Typical response times are:

Via Ethernet cable - < 1 ms

Via Wi-Fi - 5 to 12 ms with occasional responses < 300 ms

Below is an example of a **bad** network connection.

```
64 bytes from amradio (192.168.1.167): icmp_seq=2923 ttl=64 time=8.62 ms
64 bytes from amradio (192.168.1.167): icmp_seq=2925 ttl=64 time=1127 ms
64 bytes from amradio (192.168.1.167): icmp_seq=2926 ttl=64 time=6.00 ms
64 bytes from amradio (192.168.1.167): icmp_seq=2927 ttl=64 time=14.2 ms
64 bytes from amradio (192.168.1.167): icmp_seq=2927 ttl=64 time=26.6 ms (DUP!)
64 bytes from amradio (192.168.1.167): icmp_seq=2928 ttl=64 time=7.61 ms
64 bytes from amradio (192.168.1.167): icmp_seq=2929 ttl=64 time=6.12 ms
64 bytes from amradio (192.168.1.167): icmp_seq=2945 ttl=64 time=9.00 ms
64 bytes from amradio (192.168.1.167): icmp_seq=2947 ttl=64 time=138 ms
```

A DUP can be seen (packet resent as no response the first one received). Also, the 2nd packet sent took 1.127 seconds to respond.

The above network will give nothing but problems and needs correction. Wired Ethernet connections are very unlikely to give a problem. For a bad Wi-Fi connection consider using a Wi-Fi repeater, such as TP-Link or Netgear, for blind spots in your premises. Put the repeater close to the

Raspberry Pi. Search “bad wi-fi” on the Internet for further advice on how to solve this type of problem.

Reboot your router

If you are getting poor ping responses on a Wi-Fi network it is worth re-setting your Wi-Fi router. Switch off your router for at least a minute and then power up the router again. This has been known to cure poor ping responses and problems with the radio.

If you are using Wi-Fi repeaters then also do a power reset on these devices as well.

Checking the Internet connection

The previous section shows if there is a network connection or not. But this only shows connectivity to the Local Area Network (LAN) which is just the Internet Router and local devices and does not mean there is connection across the Internet itself. This can be easily checked with the **host** command as shown below.

```
$ host google.com
google.com has address 216.58.206.142
google.com has IPv6 address 2a00:1450:4009:811::200e
google.com mail is handled by 20 alt1.aspmx.l.google.com.
google.com mail is handled by 10 aspmx.l.google.com.
:
```

The radio software contains a more sophisticated check which is configured in **/etc/radiod.config**.

```
# Internet check URL. This must be a reliable URL and port number
# which can be contacted such as google.com
# The port number is normally 80 (HTTP).
# The internet_timeout in in seconds
# Disable by removing the URL from internet_check_url= parameter
internet_check_url=google.com
internet_check_port=80
internet_timeout=10
```

The radio program regularly checks for a good Internet connection when playing a radio stream across the network. If you find this check is a problem it can be disabled by removing the URL (google.com)

```
internet_check_url=
```

Wi-Fi is currently blocked by rfkill.

No Wi-Fi network is available and the following message is shown during login:

```
Wi-Fi is currently blocked by rfkill.
Use raspi-config to set the country before use.
```

Running the **raspi-config** network set-up doesn't cure the problem. Run the **rfkill** utility to confirm the problem.

```
$ sudo rfkill list all
0: phy0: Wireless LAN
    Soft blocked: yes
    Hard blocked: no
```

```
1: hci0: Bluetooth
  Soft blocked: no
  Hard blocked: no
```

The above display confirms that the Wi-Fi network is blocked. Unblock it with **rfkill**:

```
$ sudo rfkill unblock 0
```

Wireless network keeps dropping out

The Raspberry Pi powers off the Wi-Fi adapter if it has been idle for some time. Normally the Wi-Fi adapter is in constant use when streaming a radio station but can become idle when muted.

You may see on-line documentation which asks you to add the following line to **/etc/network/interfaces**. However, this does not work for **Buster** or later.

```
wireless-power off
```

To disable Wi-Fi power management, add the following line to the end of **/etc/rc.local** just before the **exit 0** statement.

```
:
sudo iw dev wlan0 set power_save off
exit 0
```

After reboot the Wi-Fi power save state can be displayed with the following command.

```
$ iw dev wlan0 get power_save
Power save: off
```

Cannot find my Raspberry Pi

A very handy tool for checking devices connected to your network is the **Fing** App. This runs on both Android and Apple devices. See section *Finding the Raspberry Pi on a network using Fing* on page 70. More information on Fing can be found at <https://www.fing.com/products/fing-app>.

HDMI/Touchscreen problems

Screen size error

The following may be seen in the logging when running the graphics version of the radio (gradio or vgradio).

```
gradio screen size error: No video mode large enough for 1024x600
```

The problem is that the screen size for the display being used has not been correctly set. There are three places this must be set. In the following example the graphics screen has a resolution of 720 x 480 pixels.

Edit the **hdmi_cvt** parameter in **/boot/config.txt** (may vary).

```
hdmi_cvt=720 480 60 1 0 0 0
```

Make sure that the resolution specified matches your device. This is 720x480 in the above example.

Set the console to the same size.

```
framebuffer_width=720  
framebuffer_height=480
```

Finally edit screen_size parameter in the [SCREEN] section of **/etc/radiod.conf**.

```
screen_size=720x480
```

HDMI/Touchscreen radio does not start

Make sure that the LCD/TFT version of the radio isn't running. Stop and disable it with the following commands:

```
$ sudo systemctl stop radiod  
$ sudo systemctl disable radiod  
$ sudo reboot
```

If it is not starting on reboot then re-run the **configure_radio.sh** program as shown in the section *Configuring the radio* on page 99. Select the option to start the program at boot time.

There is a file called **/home/pi/.config/lxsession/LXDE-pi/autostart**. This is where desktop applications are started from.

```
@lxpanel --profile LXDE-pi  
@pcmanfm --desktop --profile LXDE-pi  
@xscreensaver-no-splash@point-rpi  
@sudo /usr/share/radio/gradio.py
```

The last line starts gradiod.py at boot time. It has been known for the above file to disappear. Re-create it as shown above.

Test the graphic version of the radio

Open a terminal window. The pi user prompt should be displayed. Now run the following

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

The graphical version of the radio should start. If it doesn't, it should display the problem.

```
Traceback (most recent call last):  
  File "./gradio.py", line 46, in <module>  
    from gcontrols_class import *  
  File "/usr/share/radio/gcontrols_class.py", line 22, in <module>  
    from sgc.widgets.base_widget import Simple  
  File "/usr/share/radio/sgc/__init__.py", line 19, in <module>  
    import surface  

```

In the above example the **python-pygame** package is missing. The solution is to install it.

```
$ sudo apt install python-pygame
```

Re-test gradio.py

HDMI/Touchscreen is displaying upside-down

The standard orientation is with the connectors for power and HDMI at the bottom of the screen. However, with the official Raspberry case, it has the cables at the top. If the screen is displaying upside-down then edit the **/boot/config.txt** configuration file and add the following line.

```
lcd_rotate=2
```

Note: you must disable the *DRM VC4 V3D driver* as shown on page 84 for this to work. Reboot the Raspberry Pi for the changes to take effect.

See the following guide:

<https://www.modmypi.com/blog/raspberry-pi-7-touch-screen-assembly-guide>

The touch screen displays a lightning symbol

This is an under-voltage warning. See:

<https://www.raspberrypi.org/documentation/configuration/warning-icons.md>

Use at least a 1.5 ampere (1500 mA) power supply.

The touch screen displays a thermometer symbol

The Raspberry Pi is getting too hot. Improve airflow to the Raspberry Pi. Do not ignore this warning.

Sound is heard but the graphical radio program will not start

This is almost certainly due to the fact that the **radiod** service is running. Either re-run the **configure_radio.sh** program to reconfigure or run the following commands.

```
$ sudo systemctl stop radiod  
$ sudo systemctl disable radiod
```

The HDMI/Touchscreen program only displays a blue screen

This is due to missing scratch files in **/usr/share/scratch/Media** directory. To correct this install scratch as shown in the section called *Install the Scratch package* on page 92.

Cannot launch Graphical radio by clicking on the Desktop Icon

This is almost certainly because the **gradio.desktop** **vgradio.desktop** files in the home Desktop folder are not executable. To correct this do the following

```
$ cd /home/pi/Desktop  
$ sudo chmod +x gradio.desktop vgradio.desktop
```

It should now be possible to launch the radio from either desktop icon by double-clicking it.

Clicking on one of the radio desktop icon asks for an execution option

When attempting to launch the graphical radio programs the following is seen.

```
This text file seems to be an executable script "what do you want to do with it"
```

This is a minor irritation. To run the program just click on Execute.

To disable this behaviour, do the following:

1. On the Windows desktop click on File manager (Folder Icon top left)
2. Move to the home **Desktop** folder by double clicking it
3. Highlight **gradio.desktop** file
4. Select Edit then *Preferences → General*
5. Un-tick the option "Don't ask option on launch executable file"
6. Close the *Preferences* window
7. Highlight **vgradio.desktop** file and repeat the above steps 4 through 7

Retry clicking on either of the Radio desktop icons. The program should be executed straight away.

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
- Incorrect setup of the **/etc/asound.conf**
- If using a sound card, no driver loaded in **/boot/config.txt**
- The Raspberry Pi audio is configured to use the **HDMI** output instead of the output jack or sound card.

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.

Remove the **pulseaudio** package unless required for Bluetooth audio devices.

```
$ sudo apt remove pulseaudio
```

Re-run the **configure_audio.sh** program as shown in the section *Configuring the audio output* on page 109. You must do this to configure MPD to work with pulseaudio.

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 *Installing pulseaudio* on page 96 .

If using a USB or HiFiBerry DAC:

1. Check to see if the DAC is visible using **aplay -l** command
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 other sound devices* on page 124.

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 update or try with -fix-missing?
```

This is due to that an update was not previously carried out as shown in the section called *Update to the latest the packages* on page 84. 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 dependent on good M3U files so check that these are correct as described in the section called *Creating and Maintaining Playlist files* on page 215.

Missing configuration file in mpd.service file

This is normally only a problem with the upgraded version (0.22.x onwards) of MPD.

```
$ sudo systemctl start mpd
Job for mpd.service failed because the control process exited with error
code.
See "systemctl status mpd.service" and "journalctl -xe" for details.
```

Check the status

```
$ systemctl status mpd.service
● mpd.service - Music Player Daemon
  Loaded: loaded (/usr/local/lib/systemd/system/mpd.service; disabled;
  vendor preset: enabled)
  Active: failed (Result: exit-code) since Fri 2020-05-29 19:30:19 BST;
  1min 41s ago
    Docs: man:mpd(1)
          man:mpd.conf(5)
   Process: 1060 ExecStart=/usr/local/bin/mpd --no-daemon (code=exited,
  status=1/FAILURE)
   Main PID: 1060 (code=exited, status=1/FAILURE)

May 29 19:30:19 buster03 systemd[1]: Starting Music Player Daemon...
May 29 19:30:19 buster03 mpd[1060]: exception: No configuration file found
May 29 19:30:19 buster03 systemd[1]: mpd.service: Main process exited,
  code=exited, status=1/FAILURE
May 29 19:30:19 buster03 systemd[1]: mpd.service: Failed with result 'exit-
  code'.
May 29 19:30:19 buster03 systemd[1]: Failed to start Music Player Daemon.
```

In this case `/usr/local/bin/mpd` cannot find `/etc/mpd.conf`. Check that the `ExecStart` statement in `/usr/local/lib/systemd/system/mpd.service` contains the configuration file name.

```
ExecStart=/usr/local/bin/mpd --no-daemon /etc/mpd.conf
```

Missing Libraries cause service start to fail

This is normally only a problem with the upgraded version (0.22.x) of MPD.

A message similar to the following shows that one or more required libraries are missing.

```
$ sudo systemctl status mpd.service
● mpd.service - Music Player Daemon
```

```
:  
Apr 12 01:54:59 buster04 mpd[3460]: /usr/local/bin/mpd: error while loading  
shared libraries: libao.  
Apr 12 01:54:59 buster04 systemd[1]: mpd.service: Main process exited,  
code=exited, status=127/n/a  
Apr 12 01:54:59 buster04 systemd[1]: mpd.service: Failed with result 'exit-  
code'.  
Apr 12 01:54:59 buster04 systemd[1]: Failed to start Music Player Daemon.
```

Solution: Install the pre-requisite libraries as shown in the section called *Installing the MPD upgrade from a Debian package* on page 170.

Reset failed units mpd service units. and recheck.

```
$ sudo systemctl reset-failed mpd.socket mpd.service
```

Restart mpd service.

```
$ sudo systemctl start mpd.service
```

Re-check the mpd service.

```
$ sudo systemctl status mpd.service  
● mpd.service - Music Player Daemon  
   Loaded: loaded (/usr/local/lib/systemd/system/mpd.service; enabled;  
   vendor preset: enabled)  
     Active: active (running) since Mon 2021-04-12 02:06:32 HDT; 2min 29s ago  
       :  
Apr 12 02:06:32 buster04 systemd[1]: Started Music Player Daemon.
```

Finally remove any redundant libraries

```
$ sudo apt autoremove
```

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 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.
```

It would be normal to change the **zeroconf_enabled** parameter in the **/etc/mpd.conf** file to “yes” to overcome this. The Avahi daemon is used in small local networks (such as a home LAN). The avahi-daemon process handles **mDNS** (Multicast DNS), which is used for name resolution and service discovery within the local network. The default for MPD is to disable

Since version 7.1 onwards the following will be seen in **/var/log/mpd.log**:

```
zeroconf: No global port, disabling zeroconf
```

This is not an error. This is because the radio is now using **mpd.socket** in addition to **mpd.service**. The **mpd.socket** service starts the **mpd.service** as required so that multiple applications can access MPD. In the case of the radio this is either **radiod** or the Web service for the radio.

Radiod → mpd.socket → mpd.service

Changing **zeroconf_enabled** to “yes” will have no effect as it will be automatically disabled as **mpd.socket** is being used. By the way zeroconf is the generic term for any service providing mDNS such as Avahi (Linux) or Bonjour (Apple) etc.

LCD Problems

LCD screen not showing anything

Check that the wiring conforms to the *wiring list* on page 22. 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. Make sure the correct Radio variant has been selected. Re-run the **configure_radio.sh** program as shown in the section *Configuring the radio* on page 99.

Run the **test_lcd.py** program to see if the LCD displays anything. This runs independently of any other software and can be used stand alone. Run the **wiring.py** program (See page 276).

The LCD only displays hieroglyphics

This can be caused by incorrect wiring of the LCD. 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 22. In particular check the data lines to pins 11, 12, 13 and 14 (See LCD wiring on page 30). Retest the LCD using the **test_lcd.py** program. If the wiring is correct run the **configure_radio.sh** script to select the correct revision of the board and restart the program. Run the **wiring.py** program (See page 276).

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 *Preventing electrical interference* on page 65.

LCD backlight not working

Check that pins 15 and 16 of the LCD display have +5V and 0V (GND) respectively. See section called *Using KY040 Rotary encoders* on page 31

LCD only displays dark blocks on the first line

This is normal when the raspberry PI starts up. The display should work with the **test_lcd.py** program. If the **test_lcd.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). **configure_radio.sh** script to select the correct radio daemon. Run the **wiring.py** program (See page 276).

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 remove pulseaudio
```

Display scrolling too fast or too slow

Adjust the **scroll_speed** parameter in **/etc/radiod.conf**. Adjust between 0.001 and 0.5 seconds.

```
scroll_speed = 0.2
```

Adafruit LCD backlight problems

When using an AdaFruit Blue/White 2x16 character display, if stepping through the menus the backlight goes off and on. This is because with this type of display the backlight is only using one of the RGB inputs. To solve this problem, set all backlight colours except sleep_color to WHITE.

```
bg_color=WHITE
mute_color=WHITE
shutdown_color=WHITE
error_color=WHITE
search_color=WHITE
info_color=WHITE
menu_color=WHITE
source_color=WHITE
sleep_color=OFF
```

Grove JHD1313 LCD with RGB backlight

When using the Grove JHD1313 LCD the following output is seen from the **systemctl status radiod** command:

```
Sep 18 16:32:57 piradio systemd[1]: Started Radio daemon.
Sep 18 16:32:59 piradio radiod.py[502]: Traceback (most recent call last):
Sep 18 16:32:59 piradio radiod.py[502]:   File "/usr/share/radio/radiod.py", line 29, in <module>
Sep 18 16:32:59 piradio radiod.py[502]:     from display_class import
Display
Sep 18 16:32:59 piradio radiod.py[502]:   File
"/usr/share/radio/display_class.py", line 22, in <module>
Sep 18 16:32:59 piradio radiod.py[502]:     from PIL import ImageColor
Sep 18 16:32:59 piradio radiod.py[502]: ModuleNotFoundError: No module named
'PIL'
Sep 18 16:32:59 piradio systemd[1]: radiod.service: Main process exited,
code=exited, status=1/FAILURE
```

```
Sep 18 16:32:59 piradio systemd[1]: radiod.service: Failed with result  
'exit-code'.
```

Install **PIL** as shown in the section called *Installing the Grove LCD RGB* on page 120.

Pimoroni Pirate Radio problems

Volume UP button (Y button) not working.

Pimoroni are using either GPIO 20 or 24 for button Y (Volume UP). By default, the setting for **volume_up** in **/etc/radiod.conf** is GPIO 20. For boards produced before January 2020 this is GPIO 24:

```
right_switch=24
```

See section *C.5 Pimoroni Pirate Audio wiring* on page 364 for more details. If your card is using GPIO 24 then manually edit the **/etc/radiod.conf** file as shown above.

Playlist problems

The display shows 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.

Solution: Create playlists by running the **create_playlists.py** program as shown in the section called *Creating and Maintaining Playlist files* on page 215.

Restart the radio.

Cannot play newly mounted network share

Cause: Although the share may have been created, there are no playlists found in **/var/lib/mpd/playlists**.

Solution: Create playlists by running the **create_playlists.py** program as shown in the section called *Creating and Maintaining Playlist files* on page 215. Select network share from the menu.

I2C and SMBUS problems

Import errors

The following is seen:

```
$ sudo ./radiod.py nodaemon  
Radio running pid 825  
:  
ImportError: No module named PIL
```

The **python3-pil** package has not been installed as shown in the section *Install libraries for the Olimex OLED* on page 92.

PiFace CAD and SPI problems

PiFace CAD not detected

When running the radio configured for PiFace CAD the following is seen when running the radio in **nodaemon** mode.

```
pifacecad.core.NoPiFaceCADDetectedError: No PiFace Control and Display board  
detected (hardware_addr=0, bus=0, chip_select=1).
```

This is due to the Raspberry Pi being unable to operate with the default speed of the SPI bus. Edit **spi.py** program file as shown in the section *Installing PiFace CAD software* on page 122.

Olimex OLED problems

Radio does not start with Olimex screen

Run the program in no daemon mode as shown in the section called *Running the radio program in nodaemon mode* on page 280.

```
Traceback (most recent call last):  
:  
ImportError: No module named PIL
```

If the above is seen then the libraries for the Olimex display have not been installed. Carry out the instructions as shown in the section called *Install libraries for the Olimex OLED* on page 92.

If the problem is not missing libraries check wiring and connections to the OLED as these can be easily damaged.

OLED Screen is displaying upside down

This can be changed by setting the **flip_vertical** setting in **/etc/radiod.conf** to yes or no.

```
flip_display_vertically=yes
```

Note that this setting only works with OLEDs and not LCDs. Also it does not work with SSD1306 driver. If you need to flip the screen on an SSD1306 device then use the LUMA.SSD1306 driver.

Pimoroni Pirate Audio crashes

Cause 1 SPI not enabled

These devices use an ST7789 OLED display. This requires the SPI interface to be enabled.

A typical crash is seen below when a status check is run.

```
$ sudo systemctl status radiod  
● radiod.service - Radio daemon  
  Loaded: loaded (/lib/systemd/system/radiod.service; enabled; vendor  
  :  
  :  
  "/usr/local/lib/python2.7/dist-packages/ST7789/__init__.py", line 124, in  
  __init__  
Oct 23 18:09:58 raspberrypi radiod.py[356]:      self._spi =  
spidev.SpiDev(port, cs)  
Oct 23 18:09:58 raspberrypi radiod.py[356]: IOError: [Errno 2] No such file  
or directory  
Oct 23 18:09:58 raspberrypi systemd[1]: radiod.service: Main process exited,  
code=exited, status=1/FAILURE
```

```
Oct 23 18:09:58 raspberrypi systemd[1]: radiod.service: Failed with result  
'exit-code'.
```

Solution: Using **raspi-config** to enable the SPI interface. See the section called **Configure the SPI Kernel Module** on page 103.

Cause 2 Memory pointer error

There is a suspected issue with the **PIL** library (Python Imaging Library) also called Pillow which gets a memory fault (Invalid pointer) after about three to four hours of running. It is hoped to fix this in a later release. In the meantime, edit the **/lib/systemd/system/radiod.service** file and un-comment the Restart options as shown below. This will restart the program on failure although it may skip a station.

```
Restart=on-failure  
RestartSec=5s
```

Rotary encoder problems

Testing rotary encoders

Run the test programs shown in the section called *Testing rotary encoders* on page 274. These programs display the configuration found in **/etc/radiod.conf**.

Use the **wiring.py** program to display your configured wiring. Does this match the actual wiring? If not, either correct the wiring or amend the switch settings in **/etc/radiod.conf**.

Check wiring in particular the common pin must be connected to ground (and not 3.3 volts).

Also, you can use the **test_gpios.py** program to confirm how you have wire a specific button or rotary encoder. See *The test_gpios program* on page 278.

Keep getting erroneous button-down events

This can happen with KY040 rotary encoders which have their own pull-up resistors and the VCC pin has not been connected to +3V.

Without the connection to VCC +3V the A, B and switch are all connected together via the 10K resistors and will interfere with each other (crosstalk). Connect the VCC to 3.3V as shown in *See Figure 63 KY-040 Circuit Diagram*.

RGB I2C rotary encoder - Missing module ioexpander

The following message is displayed when using RGB I2C rotary encoders:
`ImportError: No module named ioexpander.`

It is necessary to install **ioe-python**. See section called *Install Pimoroni ioe-python* on page 93.

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/radiod/stationlist` file.

Either correct the radio stream URL or remove it all together. Re-run the `create_stations.py` program. Also check that the file URL is not the pointer to the playlist file (See section Creating and Maintaining Playlist files on page 215).

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(112): Host is down

Cause: Missing or incorrect `vers` statement.

The `vers` parameter can be 1.0, 2.0 or 3.0. The `mount.cifs` man page incorrectly states that version 1.0 is the default. This is correct for the Buster OS. Try adding `vers=1.0` to the `mount` statement in `/var/lib/radiod/share`.

```
...uid=pi,gid=pi,vers=1.0 ...
```

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 noblock' to keep locks local, or start statd.`

`mount.nfs: an incorrect mount option was specified`

Cause:

You need to include the “`-o noblock`” option.

If the error isn't in the above list then search the Web for suggestions.

Sound problems

Re-run the audio configurator program

If experiencing any problems with sound, always re-run the audio configuration program and select the correct option for your sound system, for example Headphones, HDMI or DAC etc.

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

Select the correct sound configuration as shown the section called *Configuring the audio output* on page 109. Reboot the Raspberry Pi.

Check to see if any sound cards are configured

Check first if the card is visible using the **aplay** command. The DAC card should be visible.

```
$ aplay -l  
**** List of PLAYBACK Hardware Devices ****  
card 0: sndrpihifiberry [snd_rpi_hifiberry_dacplus], device 0: HiFiBerry  
DAC+ HiFi pcm512x-hifi-0 []  
Subdevices: 0/1  
Subdevice #0: subdevice #0
```

Re-run the `configure_audio.sh` program as shown in the section *Configuring the audio output* on page 109.



Note: If you have configured Bluetooth speakers then you will not see any cards. This is because Bluetooth devices do not use **alsa** but use **bluealsa** instead.

Use the **bluetoothctl info** command instead. If using Bluetooth devices see the section called *Bluetooth device no sound* on page 260.

HiFBerry or other types of DAC no sound

If the **DAC** card is still 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 correct **dtoverlay** command as shown in *Table 30 Sound card Device Tree overlays* on page 358. The following example is for a HiFiBerry DAC plus.

```
dtoverlay=hifiberry-dacplus
```

Finally modify the **audio_output** section in **/etc/mpd.conf**. The **Device** parameter should point to the correct card.

```
audio_output {  
    type      "alsa"  
    name      "DAC"  
    device    "hw:0,0"  
    #format   "44100:16:2" # optional
```

```
    mixer_device "PCM"
    mixer_control "PCM"
    mixer_type "software"
}
```

Reboot the Raspberry Pi and retest.

Bluetooth device no sound

Check Bluetooth device connected

The following commands are useful for checking that the Bluetooth device is connected. The actual address of your Bluetooth device should be used

```
$ bluetoothctl devices
Device 00:75:58:41:B1:25 SP-AD70-B
```

If no device is seen then rescan and pair your device using the instructions in the section *Connecting a Bluetooth device* on page 129.

Ping the Bluetooth device:

```
$ sudo l2ping 00:75:58:41:B1:25
Ping: 00:75:58:41:B1:25 from DC:A6:32:05:36:9D (data size 44) ...
44 bytes from 00:75:58:41:B1:25 id 0 time 5.72ms
44 bytes from 00:75:58:41:B1:25 id 1 time 4.89ms
44 bytes from 00:75:58:41:B1:25 id 2 time 4.83ms
44 bytes from 00:75:58:41:B1:25 id 3 time 4.91ms
```

Try connecting to it:

```
$ bluetoothctl connect 00:75:58:41:B1:25
Attempting to connect to 00:75:58:41:B1:25
Connection successful
```

Check the HCI UART controller

```
$ dmesg | grep -i Bluetooth
[ 28.274746] Bluetooth: Core ver 2.22
[ 28.274837] Bluetooth: HCI device and connection manager initialized
[ 28.274860] Bluetooth: HCI socket layer initialized
[ 28.274878] Bluetooth: L2CAP socket layer initialized
[ 28.274904] Bluetooth: SCO socket layer initialized
[ 28.285231] Bluetooth: HCI UART driver ver 2.3
[ 28.285249] Bluetooth: HCI UART protocol H4 registered
[ 28.285335] Bluetooth: HCI UART protocol Three-wire (H5) registered
[ 28.285601] Bluetooth: HCI UART protocol Broadcom registered
[ 28.534358] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
[ 28.534367] Bluetooth: BNEP filters: protocol multicast
[ 28.534383] Bluetooth: BNEP socket layer initialized
```

Check MPD and radio configuration for the Bluetooth device

The **configure_audio.sh** program should have amended the **name**, **device** and **format** fields in **/etc/mpd.conf**. The name can be anything. The device definition takes the following format:

```
name      <Your bluetooth device name>
device    "bluealsa:DEV=<Your Bluetooth device ID>,PROFILE=a2dp"
```

```
audio_output {
    type      "pipe"
    name      "SP-AD70-B"
    # device   "bluetooth"
    mixer_type "software"
    command   "aplay -D bluealsa -f cd"
    format    "44100:16:2"
    # mixer_device   "default"    # optional
    # mixer_control   "PCM"       # optional
    # mixer_index     "0"        # optional
}
```

All being well the configuration should match your device if not correct it.

Check that the **bluetooth_device** parameter in the **[RADIOD]** section of **/etc/radiod.conf** has been configured with the Bluetooth device ID of your device.

```
# Bluetooth device ID - Replace with the ID of your bluetooth
speakers/headphones
# Example: bluetooth_device=00:75:58:41:B1:25
# Use the following command to display paired devices
# bluetoothctl paired-devices
bluetooth_device=00:75:58:41:B1:25
```

Amend it if not correct.

Restart the radio or reboot. Music should be heard from the Bluetooth device.

The **/etc/asound.conf** file should look like this except the device address will be different.

```
# asound.conf for Bluetooth device used with MPD type pipe
defaults.bluealsa.service "org.bluealsa"
defaults.bluealsa.device "00:75:58:41:B1:25"
defaults.bluealsa.profile "a2dp"

# 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
    }
}

# Bluetooth speakers
pcm.btdevice {
    type plug
    slave {
```

```

pcm {
    type bluealsa
    service org.bluealsa
    device "00:75:58:41:B1:25"
    profile "a2dp"
}
hint {
    show on
    description "Bluetooth device"
}
}

ctl.btdevice {
    type bluetooth
}

```

Problems pairing the bluetooth device

If you selected a Bluetooth device then reboot the Raspberry Pi. Check that all the Bluetooth daemon is running:

```
$ sudo systemctl status blue*
```

You should see the **bluetooth** daemons and one **bluealsa** daemon running.

- **bluetooth.service** - Bluetooth service


```

      Loaded: loaded (/lib/systemd/system/bluetooth.service; enabled; vendor
      preset: enabled)
      Active: active (running) since Mon 2019-11-04 10:04:54 GMT; 1h 10min ago
        Docs: man:bluetoothd(8)
      Main PID: 888 (bluetoothd)
        Status: "Running"
       Tasks: 1 (limit: 2061)
      Memory: 2.1M
      CGroup: /system.slice/bluetooth.service
              └─888 /usr/lib/bluetooth/bluetoothd --noplug=sap
```
- **bluetooth.target** - Bluetooth


```

      Loaded: loaded (/lib/systemd/system/bluetooth.target; static; vendor
      preset: enabled)
      Active: active since Mon 2019-11-04 11:21:55 GMT; 1min 18s ago
        Docs: man:systemd.special(7)
```

Nov 04 11:21:55 buster8 systemd[1]: Reached target Bluetooth.

- **bluetooth.service** - Bluetooth service


```

      Loaded: loaded (/lib/systemd/system/bluetooth.service; enabled; vendor
      preset: enabled)
      Active: active (running) since Mon 2019-11-04 11:21:55 GMT; 1min 18s ago
        Docs: man:bluetoothd(8)
      Main PID: 838 (bluetoothd)
        Status: "Running"
       Tasks: 1 (limit: 2061)
      Memory: 2.1M
      CGroup: /system.slice/bluetooth.service
              └─838 /usr/lib/bluetooth/bluetoothd --noplug=sap
```
- **bluealsa.service** - BluezALSA proxy


```

      Loaded: loaded (/lib/systemd/system/bluealsa.service; static; vendor
      preset: enabled)
```

```
Active: active (running) since Mon 2019-11-04 11:21:55 GMT; 1min 18s ago
Main PID: 841 (bluealsa)
    Tasks: 6 (limit: 2061)
   Memory: 1.2M
  CGroup: /system.slice/bluealsa.service
          └─841 /usr/bin/bluealsa

:
```

Ignore any warnings about the Sap driver. These should disappear once the audio configuration program has been run.

Check the hcuart daemon.

```
$ systemctl status hcuart
● hcuart.service - Configure Bluetooth Modems connected by UART
  Loaded: loaded (/lib/systemd/system/hcuart.service; enabled; vendor
  preset: enabled)
  Active: active (running) since Tue 2019-11-05 16:17:11 GMT; 16h ago
  Process: 330 ExecStart=/usr/bin/btuart (code=exited, status=0/SUCCESS)
 Main PID: 818 (hcuartattach)
    Tasks: 1 (limit: 1599)
   Memory: 1000.0K
  CGroup: /system.slice/hcuart.service
          └─818 /usr/bin/hciattach /dev/serial1 bcm43xx 3000000 flow -
```

Connection problems

Sometimes a paired device may fail to connect.

```
[bluetooth]# paired-devices
Device 00:75:58:41:B1:25 SP-AD70-B
[bluetooth]# connect 00:75:58:41:B1:25
Attempting to connect to 00:75:58:41:B1:25
Failed to connect: org.bluez.Error.Failed
```

If you have problems connecting to your Bluetooth device try removing it and re-pairing it.

```
[bluetooth]# remove 00:75:58:41:B1:25
```

Repeat the procedure *Connecting a Bluetooth device* on page 129.

No music heard from Bluetooth device

Check that all of the above instructions have been correctly followed.

Check the status of the device using **bluetoothctl info**. Check that it is paired, connected and trusted.

The following is an example using device ID 00:75:58:41:B1:25:. The ‘:’ character means output not shown.

```
$ bluetoothctl info 00:75:58:41:B1:25
Device 00:75:58:41:B1:25 (public)
  Name: SP-AD70-B
  Alias: SP-AD70-B
  Class: 0x00240404
  Icon: audio-card
Paired: yes
Trusted: yes
```

```
Blocked: no
Connected: yes
LegacyPairing: no
UUID: Audio Sink
:
(0000110b-0000-1000-8000-
```

Try changing channels on radio. This causes MPD to retry connecting. Check the Alsamixer.

Run the following:

```
$ bluetoothctl connect <Your Bluetooth Device>
```

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 page 39. Unfortunately, the later 40-pin versions of the Raspberry Pi seem to be more prone to interference. The recommendation is to use a USB DAC or a suitable DAC card.

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 66.

Speaker Tests

There are a number of diagnostics available for testing speakers.

Simple white noise speaker test

Run speaker-test -c2 to generate white noise out of the speaker, alternating left and right. If you have a mono output amplifier, the I2S amp merges left and right channels, so you'll hear continuous white noise:

```
$ speaker-test -c2
```

Simple WAV speaker test

Once you've got something coming out, try to play an audio file with **speaker-test** (for WAV files, not MP3). Note that the following does not work with Bluetooth devices.

```
$ speaker-test -c2 --test=wav -w /usr/share/sounds/alsa/Front_Center.wav
```

You'll hear audio coming from left and right alternating speakers

Simple MP3 speaker test

If you want to play a stream of music, you can install the **mpg123** program and use it to test a live stream. To install **mpg123** run:

```
$ sudo apt install -y mpg123
```

To test a stream.

```
$ mpg123 http://ice1.somafm.com/u80s-128-mp3
```

Any online mp3 can be used if the above is not working.

Cannot change volume when running Airplay

This is most likely caused by the wrong mixer volume ID. In normal radio operation is controlled by MPD. As Airplay is nothing to do with MPD, the volume when using Airplay is controlled through the Alsa mixer. Run the following **amixer controls** command to identify the mixer volume ID.

The mixer volume ID can be identified as shown in the following command example.

```
$ amixer controls | grep -i volume
:
numid=4,iface=MIXER,name='Mic Playback Volume'
numid=8,iface=MIXER,name='Mic Capture Volume'
numid=6,iface=MIXER,name='Speaker Playback Volume'
```

Run the `set_mixer_id.sh` program to set the mixer volume id in `/var/lib/radiod/mixer_volume_id` file.

```
$ cd /usr/share/radio
$ sudo ./set_mixer_id.sh
```

Volume control errors

The following error or similar is seen in the log file:

```
ERROR volume._setVolume error vol=50: [52@0] {setvol} problems setting
volume
```

Set the **mixer_type** to “software” in `/etc/mpd.conf`. The ‘:’ character means output not shown.

```
audio_output {
:
    mixer_type      "software"
:
}
```

Operational problems

When selecting the source, the USB stick isn't shown

You need to create the playlist for the USB stick first. See the section called *Creating and Maintaining Playlist files* on page 215.

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
        inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
          link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
            inet 192.168.2.22/24 brd 192.168.2.255 scope global eth0
              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
```

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.

The radio keeps skipping radio stations

MPD will automatically skip bad stations. Remove any bad URLs from **/var/lib/radiod/stationlist** and re-run **create_stations.py**.

Source selection only shows the radio playlist

When attempting to play music from a USB stick, source selection only shows the radio playlist. This is usually because you have forgotten to run the **create_playlist.sh** program.

See the section *Creating and Maintaining Playlist files* on page 215.

Shoutcast playlist not created

Test the **get_shoutcast.py** program.

```
$ ./get_shoutcast.py
Traceback (most recent call last):
  File "./get_shoutcast.py", line 20, in <module>
    import requests
ImportError: No module named requests
```

If you see the above message, install **python-requests**.

```
$ sudo apt install python-requests
```

A station plays for a few seconds then skips to the next one

This is a known problem with the Music Player Daemon version 0.19.1 on Raspbian Jessie.

There is only one solution and that is to create a new SD card with the latest Raspbian operating system (At the time of writing this was Raspbian Buster) and install the latest version of the radio. This will install Music Player Daemon version 0.19.21 which will cure this particular problem.

The volume keeps getting reset to a 100% when the radio is restarted

This is almost certainly that the mixer volume id is incorrectly set. This is used by the Alsa mixer to set the volume. The radio software stores the mixer volume id in **/var/lib/radiod/mixer_volume_id**. The **mixer_volume_id** is normally set to the correct value by the **configure_audio.sh** program during installation. Check the setting:

```
$ cat /var/lib/radiod/mixer_volume_id  
1
```

Now run the amixer program to determine the numid of the 'Master Playback Volume'.

```
$ amixer controls  
numid=4,iface=MIXER,name='Master Playback Switch'  
numid=3,iface=MIXER,name='Master Playback Volume'  
numid=2,iface=MIXER,name='Capture Switch'  
numid=1,iface=MIXER,name='Capture Volume'
```

Note the **numid** of the mixer volume and update the correct mixer id. In the above example it is 3. Now write the **numid** number to **/var/lib/radiod/mixer_volume_id** and check it.

```
$ sudo echo 3 > /var/lib/radiod/mixer_volume_id  
$ cat /var/lib/radiod/mixer_volume_id  
3
```

Edit the **/etc/radiod.conf** file and set the desired mixer pre-set volume as required.

```
mixer_preset=70
```

Now restart the radio.

Raspberry Pi shuts down when the Menu button is pressed

The radio by default is designed to shut down the Raspberry if the Menu button is pressed in for more than 3-seconds. If this happens immediately upon pressing the Menu button then this is because the incorrect rotary class driver has been selected. Correct this by re-running the configurator and select the correct rotary class driver.

Incorrect clock time displayed

If the time date and time displayed is incorrect for a short while after switch on, this is normal. The Raspberry Pi is not the same as a full featured PC motherboard at misses some of the features that are found on such boards. For example, it does not have a Real Time Clock (RTC). A RTC is a chip with battery back-up which maintains the system time when the device is switched off.

As there is no RTC the only time that the RPi has is the last time the device was used. Shortly after the network becomes active the system synchronises its time with one or more NTP time servers on the Internet using the Network Time Protocol (NTP). This used to be done with the NTP daemon **ntpd**. However, in more recent times this is done with a feature of **systemd** called **systemd-timesyncd**. This should synchronise the time within about 10 to 15 seconds although this can take a minute or more.

If the time is synchronising OK but you want to have the correct time immediately upon switch on see *Adding a Real Time Clock (RTC)* on page 334

IR remote control problems

The irrecord program complains that lircd.conf already exists

When running the following command:

```
$ sudo irrecord -f -d /dev/lirc0 ~/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 135.

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 lircd stop
```

Remote control software does not start up

Check that there are no problems with the **remote_control.py** program (Called by service **irradiod**)

```
$ cd /usr/share/radio/
```

For Bullseye

```
$ sudo ./irradiod.py nodaemon  
remote control running pid 4299
```

For Buster

```
$ sudo ./irradiod.py nodaemon  
IR Remote control listener running pid 1932  
Using lirc module  
Protocols changed to unknown other lirc rc-5 rc-5-sz jvc sony nec sanyo  
mce_kbd rc-6 sharp xmp cec imon rc-mm
```

```
Loaded BPF protocol xbox-dvd  
Flashing LED on GPIO 16  
Listening for input on IR sensor
```

Make a note of the pid (in this example it is 1932). Operate the volume up and down. The following should be displayed:

```
KEY_VOLUMEUP  
KEY_VOLUMEDOWN
```

To stop the program run using the previously noted pid (13299).

```
$ sudo kill -9 1932
```

If the **irradiod.py** program is working OK make sure that the **irradiod** service is enabled to start at boot time. Enable using **systemctl**.

```
$ sudo systemctl enable irradiod
```

Reboot the Raspberry Pi

```
$ sudo reboot
```

Check that the UDP server program is listening on port 5100

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

If not running stop the radio and check the UDP server software

```
$ cd /usr/share/radio/  
$ sudo ./udp_server_class.py  
Starting UDP server on port 5100
```

From another terminal run the following command

For Bullseye

```
$ sudo ./irradiod.py send TEST  
OK
```

For Buster

```
$ sudo ./irradiod.py send TEST  
OK
```

On the server side you should see

```
Data = TEST
```

Stop the **udp_server.py** program with Ctrl-C. Restart the radio and retry.

Running irradiod gives a ModuleNotFoundError for _client

This happens if the instructions for installing lirc were not followed.

```
$ sudo ./irradiod.py status
Traceback (most recent call last):
  File "/home/bob/develop/pi/radio7./irradiod.py", line 37, in <module>
    from lirc import RawConnection
  File "/usr/lib/python3/dist-packages/lirc/__init__.py", line 7, in
<module>
    from .client import get_default_lircrc_path
  File "/usr/lib/python3/dist-packages/lirc/client.py", line 38, in <module>
    import _client
ModuleNotFoundError: No module named '_client'
```

Solution

First reinstall lirc:

```
$ sudo apt reinstall lirc
```

Edit **/usr/lib/arm-linux-gnueabihf/python3.9/site-packages/lirc(paths.py)** and comment out the section which unlinks **_client.so** as shown below:

```
#if os.path.exists(os.path.join(HERE, '_client.so')):
#    try:
#        os.unlink(os.path.join(HERE, '_client.so'))
#    except PermissionError:
#        pass
```

Reboot the Raspberry Pi.

Cannot select source through the Web interface

Make sure that the radiod is running or the vgradio or gradio programs on the desktop.

```
$ sudo systemctl status radiod
```

Run the checks or the UDP server as shown in the previous server. Run the UDP server software. Select a source from the Web interface. In the following example the Radio playlist has been selected.

```
$ sudo ./udp_server_class.py
Starting UDP server on port 5100
Data = PLAYLIST:_Radio
```

Cannot boot from the SD card

See Advanced topics - *Repairing a non-bootable SD card* on page 328.

Using the diagnostic programs

The diagnostic code for testing various components of the radio is contained in the various class code files themselves. For example, **lcd_class.py** (Test LCD display). First stop the radio and then run the relevant class code.

The classes that contain diagnostic code are as follows:

- **lcd_class.py** Test the LCD screen (Directly wired to the GPIO)
- **lcd_adafruit_class.py** Test the Adafruit LCD plate and buttons
- **lcd_i2c_adafruit.py** Test the Adafruit I2C backpack
- **lcd_i2c_pcf8574.py** Test LCD with a PCF8574 I2C backpack
- **lcd_i2c_jhd1313.py** Test Grove JHD1313 LCD RGB I2C
- **lcd_piface_class.py** Test PiFace CAD display and buttons
- **button_class.py** Test push button switches (Directly wired to the GPIO)
- **rotary_class.py** Test rotary encoders (Directly wired to the GPIO)
- **rotary_class_alternative.py** Test rotary encoders using alternative driver
- **rotary_class_rgb_i2c.py** Test RGB I2C Rotary Encoders

A number of other programs are supplied and can also be used for diagnostics.

- **display_model.py** Display Raspberry Pi model information
- **display_current.py** Display current station or track details
- **wiring.py** Display wiring as configured in **/etc/radiod.conf**
- **test_gpions.py** Test state of all GPIOs and connected buttons or rotary encoders
- **display_config.sh** Display the current configuration.
- **display_wifi.sh** Display current Wi-Fi connection 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 unless called by their full pathname.

Cannot start Bluetooth control program (bluetoothctl)

If you see the following message:

```
$ bluetoothctl  
[bluetooth]# scan on  
No default controller available
```

Run the following:

```
$ sudo hciconfig scan  
hci0:  Type: Primary  Bus: UART  
      BD Address: DC:A6:32:E0:50:69  ACL MTU: 1021:8  SCO MTU: 64:1  
      UP RUNNING  
      RX bytes:77809 acl:38 sco:0 events:9464 errors:0  
      TX bytes:11743815 acl:18667 sco:0 commands:122 errors:0
```

If instead of 'UP RUNNING' you see 'DOWN' check the **hciuart** service.

```

$ systemctl status hciuart.service
● hciuart.service - Configure Bluetooth Modems connected by UART
  Loaded: loaded (/lib/systemd/system/hciuart.service; enabled; vendor
    preset: enabled)
  Active: active (running) since Wed 2021-10-13 22:17:07 BST; 2 days ago
    Main PID: 424 (hciattach)
      Tasks: 1 (limit: 2059)
     CGroup: /system.slice/hciuart.service
             └─424 /usr/bin/hciattach /dev/serial1 bcm43xx 3000000 flow -
               b8:27:eb:73:99:d5

Oct 13 22:17:06 pitouch systemd[1]: Starting Configure Bluetooth Modems
connected by UART...
Oct 13 22:17:07 pitouch btuart[341]: Cannot open directory '/etc/firmware':
No such file or director
Oct 13 22:17:07 pitouch btuart[341]: Patch not found, continue anyway
Oct 13 22:17:07 pitouch btuart[341]: bcm43xx_init
Oct 13 22:17:07 pitouch btuart[341]: Set BDADDR UART: b8:27:eb:73:99:d5
Oct 13 22:17:07 pitouch btuart[341]: Set Controller UART speed to 3000000
bit/s
Oct 13 22:17:07 pitouch btuart[341]: Device setup complete
Oct 13 22:17:07 pitouch systemd[1]: Started Configure Bluetooth Modems
connected by UART.
First switch on your Bluetooth device, then switch on scanning. In the
following example this is JVC SP-AD70-B Bluetooth speakers.

```

In the above example it would appear that hciuart cannot find the firmware patches. Check to see if it is present.

```

$ locate BCM4345C0.hcd
/usr/lib/firmware/brcm/BCM4345C0.hcd

```

Now link the above **/usr/lib/firmware** directory to **/etc/firmware**.

```

$ cd /etc/
$ sudo ln -s /usr/lib/firmware

```

Reboot the Raspberry Pi and retry running.

Running test programs

Running the radio program in diagnostic mode

This is one of the first things you should learn to do. Running the **radiod.py** program in **nodaemon** mode will display any errors it encounters. Use Ctrl-C to exit the program.

```
$ sudo ./radiod.py nodaemon
```

Using the LCD test code

```
$ ./lcd_class.py
```

The above program will display the following text on the LCD:

Bob Rathbone
Line 2: abcdefghi

Line 2 scrolls the alphabet followed by 0 through 9.

The **lcd_adafruit_lcd.py** program does the same except it also prints a message on the console screen when a button is pressed.

Testing push buttons program

Test push buttons directly wired to the GPIO pins.

```
$ ./button_class.py
down_switch
up_switch
right_switch
left_switch
menu_switch
Pull Up/Down resistors DOWN
Button pressed on GPIO 17
:
```

Pressing the switches should show on the screen as shown in the above example.

Testing rotary encoders

This program does a simple test of the rotary encoders.

```
$ ./rotary_class.py
Test rotary encoder Class
Left switch GPIO 14
Right switch GPIO 15
Up switch GPIO 24
Down switch GPIO 23
Mute switch GPIO 4
Menu switch GPIO 17
Tuner event 1 CLOCKWISE
Tuner event 2 ANTICLOCKWISE
Volume event 1 CLOCKWISE
Volume event 2 ANTICLOCKWISE
Tuner event 4 BUTTON_UP
Tuner event 3 BUTTON_DOWN
Volume event 4 BUTTON_UP
Volume event 3 BUTTON_DOWN
Volume event 1 CLOCKWISE
```

```
Volume event 2 ANTICLOCKWISE
```

You can also try

```
$ ./rotary_class_alternative.py
```

Testing RGB I2C rotarybencoders

Run the `rotary_class_rgb_i2c.py` program

```
$ ./rotary_class_rgb_i2c.py
You must specify at least one encoder to be tested!

Usage: sudo ./rotary_class_rgb_i2c.py --help
       --test_volume
       --test_channel
       --test_both
       --volume_i2c=<volume_i2c_address>
       --channel_i2c=<channel_i2c_address>

Recommended values for volume and channel I2C addresses are 0x0F and 0x1F
(defaults)
Run "i2cdetect -y 1" to check I2C address
```

You can test either one of the rotary encoders individually or both together. For example:

```
$ sudo ./rotary_class_rgb_i2c.py --test_both
RGB I2C Dual Rotary Encoder Test
Volume rotary encoder I2C address=0xf
Mute switch GPIO 27
Channel rotary encoder I2C address=0x1f
Menu switch GPIO 17
Started
Channel rotary encoder event ANTICLOCKWISE
Channel rotary encoder event ANTICLOCKWISE
Channel rotary encoder event CLOCKWISE
Channel rotary encoder event CLOCKWISE
Channel rotary encoder event BUTTONTDOWN
Volume rotary encoder event CLOCKWISE
Volume rotary encoder event CLOCKWISE
Volume rotary encoder event CLOCKWISE
Volume rotary encoder event ANTICLOCKWISE
Volume rotary encoder event ANTICLOCKWISE
Volume rotary encoder event BUTTONTDOWN
```

The RGB I2C Hex addresses are configured in `/etc/radiod.conf`.

```
# I2C addresses for RGB I2C Rotary Encoders
volume_rgb_i2c=0x0F
channel_rgb_i2c=0x1F
```

However, the encoders themselves are delivered with the default I2C hex address of **0x0F**. One of the encoders must be set to hex 0x1F (Channel Rotary Encoder) using the `rgb_set_i2c_address.py` program.

```
$ ./rgb_set_i2c_address.py --new_i2c_address=0x1F
```

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

file: http://airspectrum.cdnstream1.com:8116/1649_192#Easy hits Florida HD
title: Lou Rawls - You'll Never Find Another Love Like Mine
name: Easy Hits Florida
time: 0
duration: 0.000
pos: 69
id: 181
current_id 70

Status
volume: 35
repeat: 0
random: 0
single: 0
consume: 0
partition: default
playlist: 20
playlistlength: 111
mixrampdb: 0.000000
state: play
song: 69
songid: 181
time: 3002:0
elapsed: 3001.688
bitrate: 192
duration: 0.000
audio: 44100:24:2
nextsong: 70
nextsongid: 182

uptime: 3002
playtime: 3002
artists: 0
albums: 0
songs: 0
db_playtime: 0
db_update: 1644564447
Bit rate 192
```

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>.

The wiring program

The **wiring.py** program displays a wiring list based upon the configuration that it finds in **/etc/radiod.conf**. In the following example the 40-pin wiring (See Table 4 on page 28) has been selected during installation. The first column shows the GPIO setting in **/etc/radiod.conf**. The second column (Pin) shows the physical pin for that GPIO. For example, in the **left_switch** parameter in **/etc/radiod.conf** has been set to GPIO 23 which is physical pin 16 on the 40-pin GPIO header.

The program displays three wiring sections namely:

1. SWITCHES – Rotary encoder and push button wiring
2. LCD – Directly connected LCD wiring

3. OTHER – Remote control and activity LED, I2C backpacks

```
$ cd /usr/share/radio
$ sudo ./wiring.py
Radio wiring scheme based upon configuration in /etc/radiod.conf

----- SWITCHES -----
GPIO  Pin      Switch      Rotary
====  ===      =====      =====
23    16 <----> Left switch   A
24    18 <----> Right switch  B
       6 <----> Gnd 0v        C
4     7 <----> Mute switch    < GND 0V
14   8 <----> Down switch   A
15   10 <----> Up switch     B
       6 <----> Gnd 0v        C
17   11 <----> Menu switch   < GND 0V

Pull Up/Down resistors DOWN

Push button switches must be wired to +3.3V
Rotary push switches must always be wired to GND 0V

----- LCD -----
GPIO  Pin      Function    LCD pin
====  ===      =====      =====
5     29 <----> Lcd data4   11
6     31 <----> Lcd data5   12
12   32 <----> Lcd data6   13
13   33 <----> Lcd data7   14
8     24 <----> Lcd enable   6
7     26 <----> Lcd select   4
       2 <----> VCC +5V     2,15
       6 <----> GND 0V      1,16
10K Pot <----> Contrast   3

----- OTHER -----
GPIO  Pin      Function
====  ===      =====
16   36 <----> Remote led
3    5 <----> I2C Data
2    3 <----> I2C Clock
25  22 <----> IR Remote   (See /boot/config.txt)
```

Currently the wiring for the vintage radio RGB Led and Menu switch are not shown.

In the above output the connections are descriptive. The parameters found in **/etc/radiod.conf** can be displayed with the **-p** option:

```
$ ./wiring.py -p
Radio wiring scheme based upon configuration in /etc/radiod.conf

----- SWITCHES -----
GPIO  Pin      Switch      Rotary
====  ===      =====      =====
23    16 <----> left_switch  A
24    18 <----> right_switch B
       6 <----> GND_0V      C
4     7 <----> mute_switch   < GND 0V
14   8 <----> down_switch   A
15   10 <----> up_switch     B
       6 <----> GND_0V      C
17   11 <----> menu_switch   < GND 0V
```

:

In the above display the **left_switch** label is the parameter found in **/etc/radiod.conf**.

If the wiring shown in the **wiring.py** program then either amend the wiring to match the wiring list shown in the output of the wiring program or amend **/etc/radiod.conf** to match the actual wiring. The program also has a help function (-h option).

```
$ ./wiring.py -h
Usage: ./wiring.py -p -h
Where -p print parameters, -h this help text
```

The test_gpios program

Sometime you may not be sure how you have wired a particular button or rotary encoder or you may have a conflict with another program or a setting in the **/boot/config.txt** configuration file. The **test_gpios.py** program allows both situations to be tested. Running the program with no parameters display the help message.

```
$ ./test_gpios.py
Usage: ./test_gpios.py <--help> <--pull_up> <--pull_down> <--none>
Where --help this help text
    --pull_up Set pull-up-down internal resistors high (+3.3V)
    --pull_down Set pull-up-down internal resistors low (0V GND)
    --pull_off No pull-up-down internal resistors (Use external resistors)
```

The other options set the internal pull up/down resistors to UP, DOWN or OFF.

--pull_up Is the normal parameter for most situations. Use this option if unsure.

--pull_down should be used for older radios with push buttons wired to 0V GND.

--pull_off is used mainly for KY-040 rotary encoders which have their own pull-up resistors.

The following example is for a radio with two normal rotary encoders and an IR receiver configured on GPIO 9.

In this case the program reports that it cannot configure GPIO 9 for testing as it is already configured in **/boot/config.txt** for use by the IR receiver. It attempts to display the conflict.

```
$ ./test_gpios.py --pull_up
GPIO: 2 State:High
GPIO: 3 State:High
GPIO: 4 State:High
GPIO: 5 State:High
GPIO: 6 State:High
GPIO: 7 State:High
GPIO: 8 State:High
GPIO: 9 State:High
Error: GPIO 9 Failed to add edge detection
Check conflict with GPIO 9 in other programs or in /boot/config.txt
dtoverlay= gpio-ir,gpio_pin=9
GPIO: 10 State:High
GPIO: 11 State:High
GPIO: 12 State:High
GPIO: 13 State:High
```

```
GPIO: 14 State:High
GPIO: 15 State:Low
GPIO: 17 State:High
GPIO: 18 State:High
GPIO: 19 State:High
GPIO: 20 State:High
GPIO: 21 State:High
GPIO: 22 State:High
GPIO: 23 State:High
GPIO: 24 State:High
GPIO: 25 State:High
GPIO: 26 State:High
GPIO: 27 State:High
Waiting for input events:
```

Note that GPIO 15 is a rotary encoder input and may be high or low depending upon its position. Same for GPIO 14.

The program waits for events. The Mute button is configured on GPIO 4. The volume rotary encoder A and B inputs are wired to GPIO 14 and 15 respectfully. Pressing the Menu button or turning the volume knob should produce an output similar to the following:

```
GPIO 4 falling
GPIO 4 rising
GPIO 4 falling
GPIO 4 falling
GPIO 11 rising
GPIO 14 falling
GPIO 15 falling
GPIO 14 falling
GPIO 15 rising
```

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=Manufacturer's 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 configuration program

The **display_config.sh** provides diagnostic information how the radio, drivers and MPD have been configured. It also produces a compressed tar file called **/usr/share/radio/config.log.tar.gz** with this information. Send this file to bob@bobrathbone.com.

```
$ ./display_config.sh
Configuration log for pi Tue Jul  4 20:38:27 BST 2023
Release 24th June 2023 - Build 6
IP address: 192.168.1.251

OS Configuration
-----
PRETTY_NAME="Raspbian GNU/Linux 11 (bullseye)"
NAME="Raspbian GNU/Linux":
:
=====
End of run =====
This configuration has been recorded in /usr/share/radio/config.log
A compressed tar file has been saved in /usr/share/radio/config.log.tar.gz
```

```
Send /usr/share/radio/config.log.tar.gz to bobrathbone.com if required
```

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.

```
$ cd /usr/share/radio  
$ sudo ./radiod.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).

In the case of **gradio.py** and **vgradio.py**, these are not daemons and can be run as shown in the example for gradio.py below:

```
$ sudo ./gradio.py
```

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 systemctl stop radiod  
$ sudo systemctl stop irradioid
```

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 or in **/etc/radiod.conf** change **log_creation_mode=truncate**:

```
$ sudo rm /var/log/radiod/radio.log
```

Start the radio and remote control if required:

```
$ sudo systemctl start radiod  
$ sudo systemctl start irradioid
```

Operate the radio including the operation you are having with.

Send the **/var/log/radiod/radio.log** file to bob@bobrathbone.com

Switch off DEBUG mode by editing the **/etc/radiod.conf** file and as shown below.

```
loglevel=INFO
```

Display the kernel details

Display the kernel version using **uname**.

```
$ uname -a
Linux raspberrypi 5.10.63-v7+ #1459 SMP Wed Oct 6 16:41:10 BST 2021 armv7l
GNU/Linux
```

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 the GPIO information

The **gpio readall** command can be used to display the GPIO configuration. Stop the radio first!

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	CEO0	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				

* 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 (Column V is 0).

Displaying information about the Operating system

Display **/etc/os.release** using the **cat** command. Likewise the Debian version

```
$ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 11 (bullseye)"
NAME="Raspbian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
VERSION_CODENAME=bullseye
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"
```

```
$ cat /etc/debian_version  
11.1
```

Chapter 10 - 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 **gmmpc** 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 335.

Installing Icecast

Install **icecast2** using the **install_streaming.sh** script.

```
$ cd /usr/share/radio
$ sudo ./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:

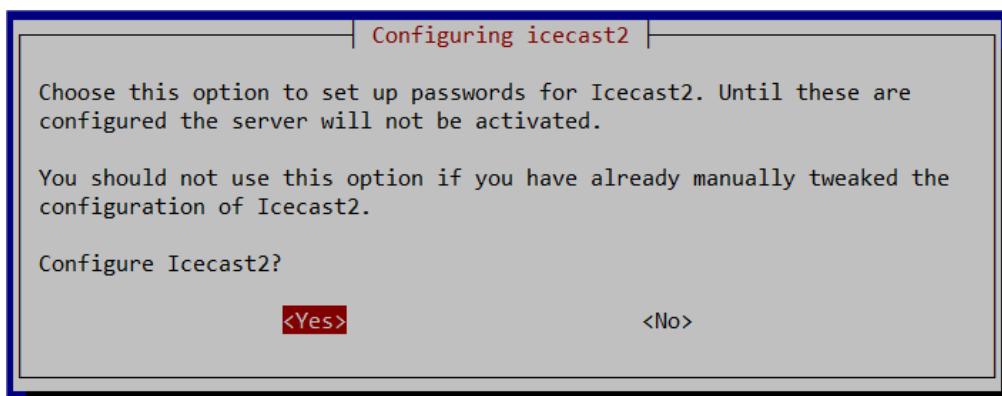


Figure 222 Configuring Icecast2

Answer 'yes' to this. Configure Icecast as follows:

Icecast2 hostname: **localhost** (or the hostname of the Raspberry Pi)

Icecast2 source password: **mympd**

Icecast2 relay password: **mympd**

Icecast2 administration password: **mympd**

It is important that you replace the default password ‘hackme’ with ‘mympd’. The installation program continues configuration. The icecast2 server will be started:

```
Done Configuring icecast2..
Processing triggers for libc-bin (2.19-18+deb8u6) ...
Processing triggers for systemd (215-17+deb8u5) ...
Configuring Icecast2
Copying /etc/icecast2/icecast.xml to /etc/icecast2/icecast.xml.orig
```

Check that the PI Radio stream (Output 2) is enabled

```
$ mpc outputs
Output 1 (My ALSA Device) is enabled
Output 2 (PI Radio MPD Stream) is enabled
```

If not enable it and restart **mpd** and **icecast2**. In this case it is output 2

```
$ mpc enable 2
$ sudo systemctl restart mpd.service icecast2.service
```

Check that MPD has established a connection with the icecast2 server

```
$ netstat -tn | grep :8000
tcp        0      0 127.0.0.1:59096      127.0.0.1:8000          ESTABLISHED
tcp        0      0 127.0.0.1:8000      127.0.0.1:59096          ESTABLISHED
```

The Icecast2 install_streaming.sh script sets the **streaming_on** parameter in **/etc/radiod.conf** to yes.

```
streaming_on=yes
```

It also adds the following configuration to **/etc/mpd.conf**.

```
# MPD Radio Stream
audio_output {
    type          "shout"
    name          "PI Radio MPD Stream"
    description   "MPD stream on Raspberry Pi Radio"
    host          "localhost"
    port          "8000"
    mount         "/mpd"
    password      "mympd"
    bitrate       "128"
    format        "44100:16:2"
    encoding      "mp3"
}
```

This completes the installation of Icecast2 however you may need to configure the clock speed if your Raspberry Pi is an earlier version (See *Overclocking older Raspberry PI's* on page 285)

Now go to *Icecast2 Operation* on page 285.

Overclocking older Raspberry PI's

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. Note that the later versions of the Raspberry Pi cannot be overclocked and are fast enough anyhow.

Run **raspi-config**. Select option 'Overclock'. After a warning screen about over-clocking has been displayed, the following screen will be displayed:

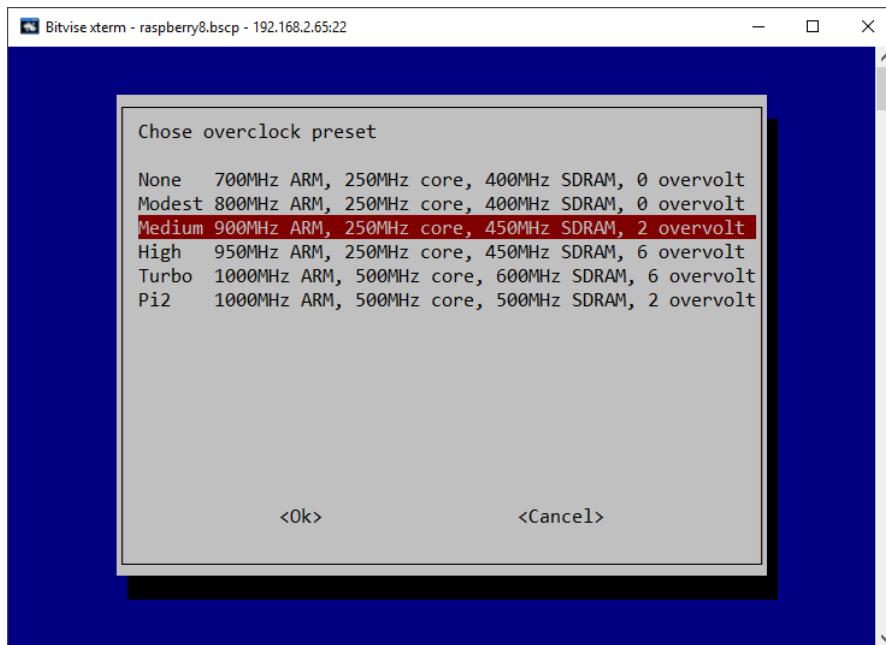


Figure 223 Over-clocking the Raspberry PI

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 normally be enabled. It can be switched on and off in either LCD/OLED versions of the radio and in the full feature graphical version of the radio.

Switching on streaming on and off

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 file is used to enable or disable the Icecast streaming function at boot time.

Checking Icecast2 status

Use the following command:

```
$ service icecast2 status
● icecast2.service - LSB: Icecast2 streaming media server
  Loaded: loaded (/etc/init.d/icecast2; generated)
  Active: active (running) since Sat 2021-09-04 07:27:34 BST; 21min ago
    Docs: man:systemd-sysv-generator(8)
   Tasks: 5 (limit: 4915)
  CGroup: /system.slice/icecast2.service
          └─632 /usr/bin/icecast2 -b -c /etc/icecast2/icecast.xml
```

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 systemctl enable icecast2
```

Playing the Icecast stream on Windows

To play the Icecast2 radio stream on a PC point your Web browser at the IP address of the radio on port 8000 and the **/mpd** mount point. In the following example the IP address of the radio is 192.168.2.8. So, this would be:

<http://192.168.2.8:8000/mpd>

This will normally open the default media player.

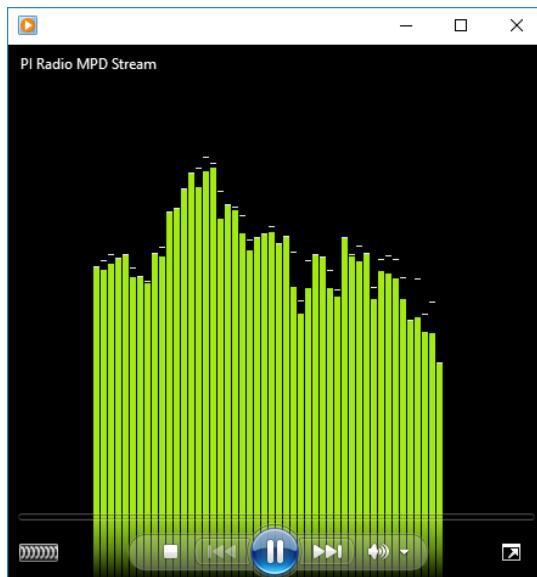


Figure 224 Windows media player

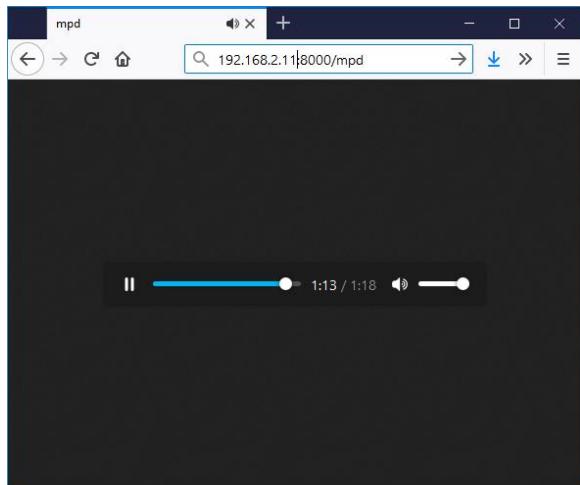


Figure 225 Firefox embedded media player

You will be prompted if you wish to save or open the radio stream. Always select open using the configured media player. The selected radio station or music track should be heard through the PC speakers. At this point you may wish to mute the sound from the radio itself. Simply reduce the volume to almost zero.



Note: It is probably possible to configure Windows 10 Edge or Internet Explorer 11 to use Windows Media player instead of TWINUI. Note: If the mute function is used it will stop the **Icecast** stream.

Running the Icecast Administration Web pages

Using the same IP address as shown in the previous example but without the mount point will bring up the administrator window:

<http://192.168.2.8:8000>

The following screen should be displayed. If not continue to the troubleshooting guide at the end of this chapter:

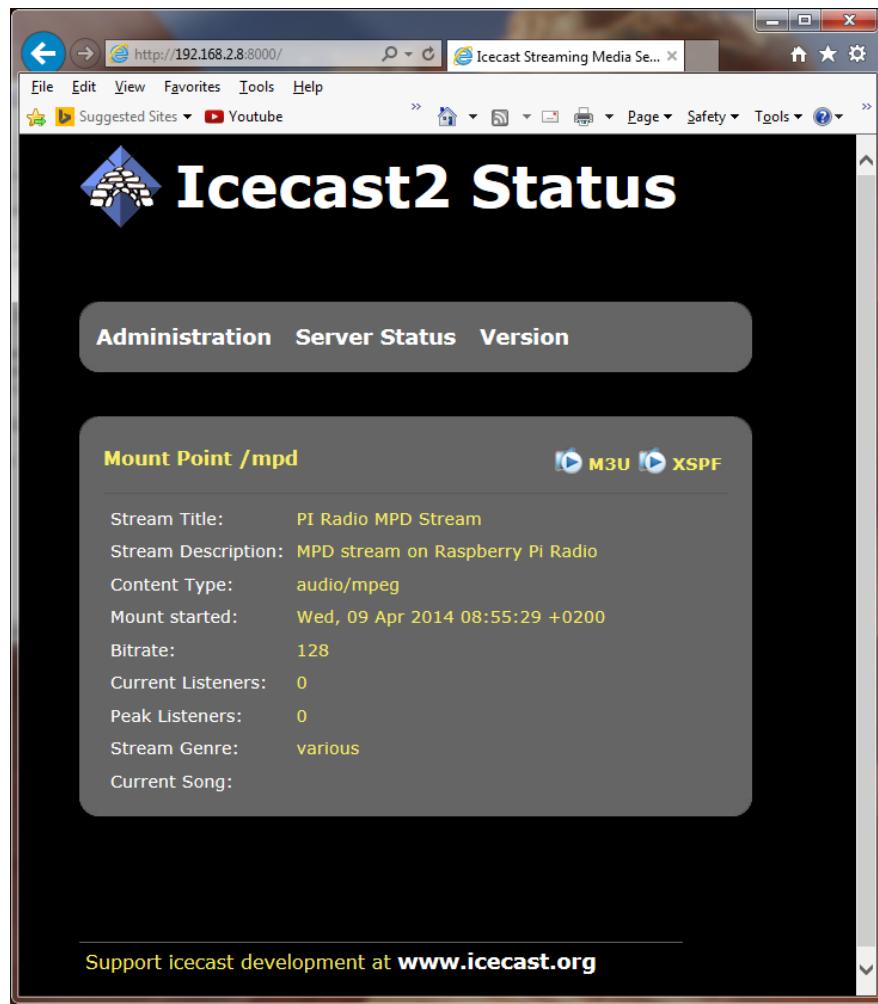


Figure 226 Icecast2 Status

Initially the server status screen is displayed. If you click on the **Administration** tab the you will be prompted for the login credentials.

Log in as admin with user *admin* password *mympd*.



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/mpd>
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

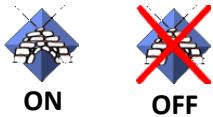
When streaming is switched on an asterix '*' character is displayed as a visual streaming indicator in the LCD display on the Raspberry PI radio. When the '*' character is displayed this indicates that the Icecast2 streaming is switched on.

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 *

Graphical radio streaming operation



Radio versions prior to version 7.3 could only switch streaming on or off in the LCD/OLED versions of the radio. From 7.3 onwards streaming can be switched on and off in the full featured graphical version of the radio (gradio.py). When Icecast2 is installed the Icecast2 ON icon will appear on the right-hand side of the screen next to the display window. Clicking it once only will switch off streaming and the OFF icon will be shown. It does take a few seconds for this to happen so only click on the icon once.

However, it is necessary to first set the **display_icecast_button** in the [SCREEN] section of **/etc/radiod.conf** to 'yes'. Add this parameter if it is missing.

```
# Display Icecast streaming button
display_icecast_button=yes
```

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 Raspberry PI and retry.
- Incorrect IP address or missing port number in the URL.
 - See [Icecast2 Operation](#) on page 285

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.

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 289.

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.

With older versions of the Raspberry Pi, try over-clocking the Raspberry PI using the **raspi-config** program. Medium over-clocking seems to be sufficient. See [Overclocking older Raspberry PI](#) on page 285.



Note: 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.

Chapter 11 - Setting up Spotify



The radio can also be set up as a Spotify receiver. You will still need a Spotify App on your telephone, PC or Tablet. You will also need a Premium Spotify account and not just a free or trial version.

More information at <https://www.spotify.com>

Spotify hardware requirements

Spotify works with the on-board audio output or HDMI audio. However, if using a DAC, the card must be capable of hardware volume control. When the radio is running the volume is software controlled by MPD. When Spotify is running, MPD is stopped and the volume is controlled by the standard **amixer** command. The volume control for Spotify is coded in the **volume_class.py** file. For example, the following command sets the volume to 100%.

```
sudo -u pi amixer cset numid=1 100%
```

The **numid** must match that of the hardware volume control. This is typically numid 1, 2 or 6. The **amixer controls** command will display all the hardware controls for a sound card. To check if your DAC supports Hardware Volume control run the **amixer controls** command.

```
$ amixer controls
:
numid=1,iface=MIXER,name='Digital Playback Volume'
```

DACs using the PCM5122a chip should all support hardware volume control.

However, DACs based upon the PCM5102 chip do not have any hardware volume control so the volume cannot be set by **amixer** commands. It is still possible to use these DACs however volume can only be done from the Spotify client App running on a mobile telephone, tablet or PC.

Spotify installation

The radio software also supports **Raspotify** originally from Dave Cooper which is Spotify for Raspberry Pi OS. First carry out a system update and upgrade as shown in *Update to the latest the packages* on page 84.

Install the **apt-transport-https** package and download and install the **Raspotify** software with the **curl** command into the pi home directory.

```
$ cd
$ sudo apt install apt-transport-https
$ curl -sL https://dtcooper.github.io/raspotify/install.sh | sh
```

This will both download and install **Raspotify**. In the **/etc/default/raspotify** configuration file you will see the OPTIONS line for Spotify account details. It is not necessary to configure your account details as these will be picked up from the connecting **Spotify** App on your PC or Mobile.

```
#OPTIONS="--username <USERNAME> --password <PASSWORD>"
```

More information on Raspotify can be found at <https://dtcooper.github.io/raspotify>

Using sudo edit **/lib/systemd/system/raspotify.service** and disable the restart options.

```
#Restart=always  
#RestartSec=10
```

Set the correct device ID to the **ExecStart** in the same **raspotify.service** file. This must match the **device** setting in the **audio_output** definition in **/etc/mpd.conf**

```
ExecStart=/usr/bin/librespot --device=hw:0,0
```

The configuration in **/etc/mpd.conf** is set up by the **configure_audio.sh** script

```
audio_output {  
    type          "alsa"  
    name          "IQAudio DAC/Zero DAC"  
    device        "hw:0,0"
```

Save the file and update **systemd** with the following command:

```
$ sudo systemctl daemon-reload
```

Finally run the **set_mixer_id.sh** script:

```
$ cd /usr/share/radio  
$ sudo ./set_mixer_id.sh  
mixer_volume_id=1
```

The above **mixer_volume_id** output may vary.

The radio will automatically stop and start Raspotify but it may be started and with the following commands:

```
$ sudo systemctl start raspotify  
$ sudo systemctl stop raspotify
```

You can also check the status of **raspotify** with **systemctl**.

```
$ sudo systemctl status raspotify  
● raspotify.service - Raspotify (Spotify Connect Client)  
   Loaded: loaded (/lib/systemd/system/raspotify.service; enabled; vendor  
   preset: enabled)  
     Active: active (running) since Fri 2022-02-18 10:12:25 GMT; 6min ago  
       Docs: https://github.com/dtcooper/raspotify  
              https://github.com/librespot-org/librespot  
              https://github.com/dtcooper/raspotify/wiki  
              https://github.com/librespot-org/librespot/wiki/Options  
     Main PID: 13033 (librespot)  
        Tasks: 1 (limit: 2054)  
         CPU: 123ms  
        CGroup: /system.slice/raspotify.service  
                  └─13033 /usr/bin/librespot  
  
Feb 18 10:12:25 bullseye2 systemd[1]: Started Raspotify (Spotify Connect  
Client).
```

Disable Raspotify from starting at boot time. Starting and stopping Raspotify is done by the radio.

```
$ sudo systemctl disable raspotify
```

It is necessary to modify **/etc/raspotify/conf** to allow the track title to be displayed.

```
sudo vi /etc/raspotify/conf
```

Comment out the **LIBRESPOT_QUIET** parameter in **/etc/raspotify/conf**. You will need to use your editor with **sudo**. For example, **sudo nano /etc/raspotify/conf**

```
# Only log warning and error messages.  
#LIBRESPOT_QUIET=
```

Failure to do so will mean that track titles will not be displayed.

Raspotify messages during operation can be observed with the following command:

```
$ journalctl --lines 0 --follow _SYSTEMD_UNIT=raspotify.service
```

Finally stop the Raspotify service and restart radiod.

```
$ sudo systemctl stop raspotify  
$ sudo systemctl restart radiod
```



Note: The authors of Raspotify have stated that it is not possible to display the artist's name and never will be. Only the track title can be displayed.

Spotify operation

To start the Radio as a Spotify receiver either select Spotify from the Playlists (LCD or OLED versions) or from the Sources window in the touchscreen version:

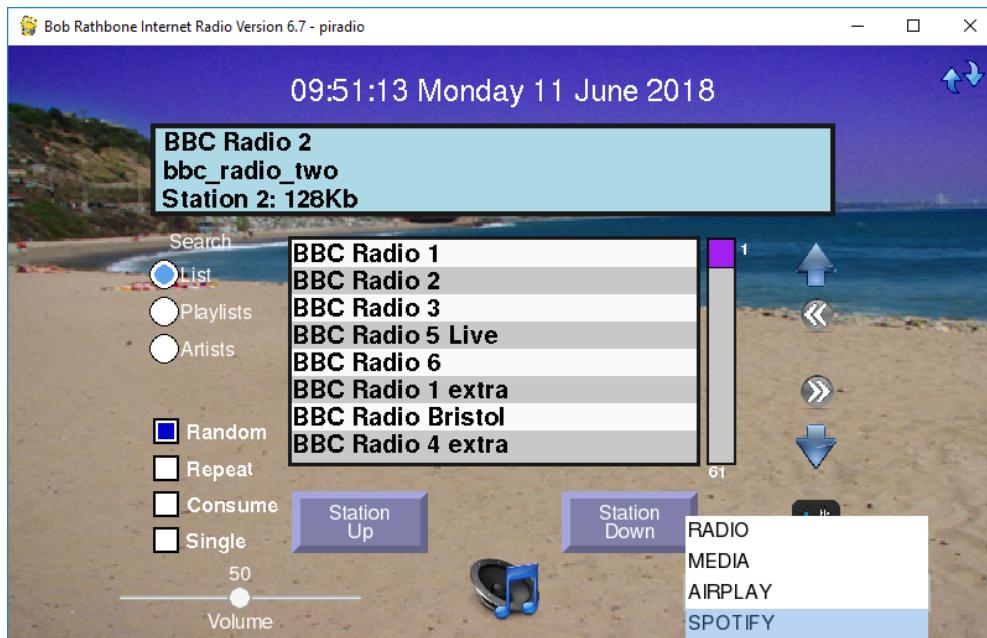


Figure 227 Starting the Spotify Receiver

The following window will appear however a different message may appear on the second line of the display window:



Figure 228 The radio in Spotify mode

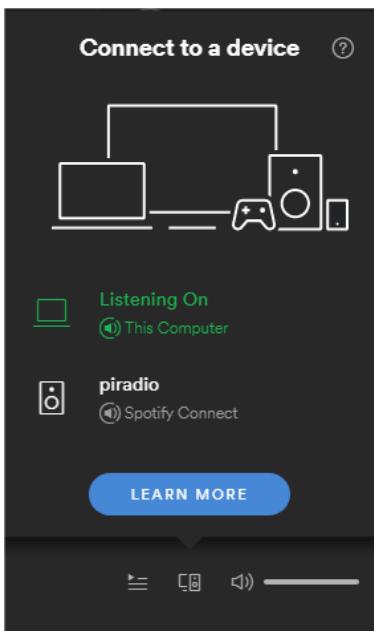


Figure 229 Spotify connecting to the radio

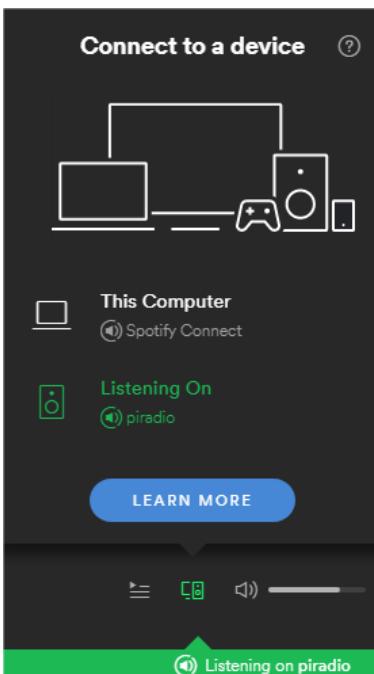


Figure 230 Listening to Spotify on the radio

To use Spotify you will need a Spotify App on your telephone, PC or Tablet.

As previously mentioned you will need a Premium Spotify account and not just a free or trial version.

On the radio, press the Menu button until you come to the sources selection. Press channel Up or Down until you see Spotify displayed.

Press the Menu button again and the radio will stop the MPD player and start raspotify.

Now click on *Connect to a device* in the Spotify application. You should see an entry called Spotify Connect with the hostname of your radio.

Click on the *Spotify Connect* device for the radio (*piradio* in this example).

The Spotify application will switch from the current device (PC, mobile phone or tablet) to the Raspberry Pi radio.

You can control the volume either from the Spotify Application or using the volume control on the radio.

To exit Raspotify on the Raspberry Pi press the Menu button and then select another source such as the radio.

The Spotify application will connect the Raspotify application on the Radio.



Note: If you don't hear any sound then turn the volume up to full.

In the case of the touchscreen version of the program the following screen will be displayed:



Figure 231 Spotify playing a music track

In the case of the LCD or OLED version of the radio the title line above will be displayed on the second line of LCD or OLED screen.



Note: Raspotify unfortunately does not supply the Artist information. Only the track name is supplied by **librespot** which is used by Raspotify. There is currently no solution for this.

Exiting Spotify

For the LCD and OLED versions press the Menu button until the Select source: window is displayed. Select any other source (playlist) to exit.

In the case of the touchscreen version of the radio, press the "Exit Spotify" button at the bottom of the screen.

Troubleshooting Raspotify

Installation problems

If the curl command to install the Raspotify software fails then carry out a system update and upgrade as shown in *Update to the latest the packages* on page 84. Retry the curl command.

Raspotify exits with a 101 error code

This is almost certainly an authentication fault. Check your user name and password are correctly set up in **/etc/default/raspotify** and retry. However, it isn't actually necessary to put a username and password in **/etc/default/raspotify** as Raspotify will be using a Raspotify Premium account running on a PC, tablet or mobile phone.

The client connects to Raspotify but no sound heard

Check that the device ID has been added to the **ExecStart** statement in the same **raspotify.service** file.

```
ExecStart=/usr/bin/librespot . . . . . --device=hw:0,0
```



Note: The author does not directly support Raspotify.
Report Raspotify issues to <https://github.com/dtcooper/raspotify/issues>
Raspotify comes with an MIT licence. See <https://opensource.org/licenses/MIT>

Cannot change Raspotify volume

In some cases, this is normal as many cheaper sound cards (DACs) do not have mixer controls. Mixer controls are required to change the volume on the Raspberry Pi when running Spotify or Airplay. In the case of no mixer controls the sound when running Spotify must be controlled from the App on either your mobile device or from the PC application.

However, it is possible to set up a software mixer (SoftVol).

See section *Configuring soft volume (SoftVol)* on page 195 for further information.

Raspotify initial sound too loud

Edit the **/lib/systemd/system/raspotify.service** file. Locate the following line

```
Environment="VOLUME_ARGS=--enable-volume-normalisation --volume-ctrl linear  
--initial-volume 100"
```

Change to the desired volume setting.

```
--initial-volume 75
```

Chapter 12 - Setting up Airplay

If you have not already done so, carry out a system and firmware upgrade, as shown in *Preparing the Operating System* on page 84. Airplay uses a program called **shairport-sync** from Mike Brady.

Airplay is based upon the procedure in the following link:

<http://www.redsilico.com/multiroom-audio-raspberry-pi>



Do not use the procedure in the above link. Use the procedure described below as it has been greatly modified to work with the radio software.

Installation

Airplay is installed using an installation script called **install_airplay.sh**.

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

This will install both **shairport-sync** and configure the radio to use it.

If the script fails with a dependency error for PHP. Run the following:

```
$ sudo apt -f install
```

Re-run **install_airplay.sh**.

Configuring the Airplay feature

Below are the configuration parameters found in the Airplay section of **/etc/radiod.conf** affecting the Airplay (shairport-sync) function in the radio.

```
[AIRPLAY]  
  
# Airplay activation yes or no  
airplay=no  
  
# Mixer preset volume for radio and media player if using sound card  
# Set to 0 if using onboard audio or USB sound dongle.  
# If using a sound card set to 100% initially and adjust as necessary  
# Old name was mixer_volume  
mixer_preset=100
```



If upgrading from an earlier version of the radio and you selected “Do not update existing configuration” during installation then the [AIRPLAY] section will be missing from **/etc/radiod.conf**. If this is the case then copy the above lines to the end of the file.

Check that Airplay is enabled. This should already be configured by **/etc/radiod.conf**

```
airplay=yes
```

Airplay service check

Check that all is well with D-Bus. The following

```
$ sudo systemctl start shairport-sync  
Failed to get D-Bus connection: Unknown error -1
```

If the above error message is seen install **systemd-sysv**:

```
$ sudo apt install systemd-sysv
```

Re-start and check status.

```
$ sudo systemctl start shairport-sync  
$ sudo systemctl status shairport-sync  
● shairport-sync.service - ShairportSync AirTunes receiver  
  Loaded: loaded (/lib/systemd/system/shairport-sync.service; disabled;  
  vendor preset: enabled)  
    Active: active (running) since Sat 2021-03-20 11:25:15 GMT; 15s ago
```

Reboot the Raspberry Pi.

```
$ sudo reboot
```

See the following section on how to use Airplay.

Using Airplay on the radio

Using Airplay on a HDMI/Touchscreen is described in the section called *Running Airplay on the HDMI touchscreen* on page 204. LCD versions of the radio are described here.



Figure 232 Airplay source selection

Press the menu button until **Input Source:** is displayed.

Turn the channel button (or Up/Down switches on a push-button radio) until **Airplay receiver** appears.

Press the menu button one more time. The word Airplay will be displayed on the bottom line along with 'Unknown artist' and 'Unknown title'.

Now use the Airplay device to connect to the raspberry PI (varies according to device software). Start playing the music tracks and this should start being heard on the radio which also displays the Artist, Track and Album on the LCD display. The volume is adjustable if correctly set-up. The mute also works in the normal way but does not pause or stop the Airplay stream as this can only be done from the device running Airplay.



Figure 233 Running an Airplay device on the radio with Cloudbreak

The above example is using an evaluation copy of CloudBreak running on an Android mobile telephone.

Unfortunately, Cloudbreak is no longer available however there are a number of Airplay Apps available for Android telephones such as “Double twist”.

Chapter 13 - Internet Security



This is a section that probably will not concern most people as their Raspberry Pi is not exposed to the internet. However, with more and more cases of such devices such as Web cams and other Internet connected devices being hacked by unscrupulous hackers, Internet Security is an aspect of home computing that must be taken seriously. These incursions can be used to mount Phishing (harvesting bank details etc.) or Distributed Denial of Service attacks (**DDOS**) on the wider community as a whole. More and more Internet providers are choosing to block compromised user's systems from access to the Internet until the infection is removed.

Some golden Internet Security rules



Always change the user **pi** password from the system installation default. When installed the password for user **pi** is 'raspberry'. It will be the first password that will be attempted by a hacker. The password can easily be changed using the **raspi-config** program (See *Changing the system hostname and password* on page 88). The user **pi** is very dangerous if hacked as with the command **sudo bash** the hacker then has user root privileges and can do anything they want including installing **Phishing** or **DDOS** software. Don't give them the chance!

telnet > _

Never ever use insecure protocols/programs such as **Telnet**, **Rexec** or **FTP** across the Internet. The problem with all such programs is that the login username and password are unencrypted and can be discovered by a hacker using eavesdropping software. The use of such software to access the Raspberry Pi will attract hackers like flies around a honey-pot.

SSH: /#

If access to the Raspberry Pi across a network is required (for example a headless RPi) then use Secure Shell (**SSH**) for terminal access and Secure Copy (**SCP**) for file transfer. On the latest releases of Raspbian, **SSH** is disabled for security reasons. For extra security use **SSH keys** (explained later)



Install **firewall** software such as **fail2ban**. The **Raspbian Buster** operating system has a firewall called **iptables** which can be configured to block or allow access to specific ports from a specific IP address or range. The **fail2ban** software is an enhancement to **iptables** which monitors certain ports for hacking attempts and adds a blocking rule to the **iptables** configuration. More on this later. The fail2ban software is a good defence against so-called brute-force dictionary attacks.

SSH keys installation

Raspberry Pi ssh keys

If using **SSH** across the Internet, changing the password for user **pi** will afford some limited protection. However, a hacker can still access the system with SSH and try to log in as user **pi**. They can still try (often using software) to try and guess the password. By using SSH keys a greater level of protection is afforded as person logging in must be in possession of the SSH keys.

Log in as user **pi** and then run the **ssh-keygen** program. Just press enter when asked any questions.

```
$ ssh-keygen -t rsa -C "raspberrypi"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pi/.ssh/id_rsa):
Created directory '/home/pi/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pi/.ssh/id_rsa.
Your public key has been saved in /home/pi/.ssh/id_rsa.pub.
The key fingerprint is:
c0:54:96:d9:2d:a5:d0:7c:68:91:8a:5f:e2:a5:9d pi@pixelpi
The key's randomart image is:
+---[RSA 2048]---+
| ..=O.*. |
| o .o.X.o |
| o. o.o |
| ..o o |
| oS* . |
| + E |
| |
| |
| |
+-----+
```

Two keys are generated, one public and one private in the **.ssh** directory. The key **id_rsa.pub** is the public key. The **id_rsa** file is the private key.

```
$ ls -la .ssh/
total 16
drwx----- 2 pi pi 4096 Feb 16 12:40 .
drwxr-xr-x 19 pi pi 4096 Feb 16 12:40 ..
-rw------- 1 pi pi 1675 Feb 16 12:40 id_rsa
-rw-r--r-- 1 pi pi 392 Feb 16 12:40 id_rsa.pub
```

Generate a client key

It is also necessary to generate SSH keys on Typically **Putty** and **Bitvise** are a very popular choice for **SSH** clients. There is already so much documentation on the Internet on how to generate SSH keys for both **Putty** and **Bitvise** that it is not repeated here. Search the Internet for instructions.

Add client public key to Raspberry Pi authorised keys

On the Raspberry Pi create or edit the **/home/pi/.ssh/authorized_keys**.

```
$ cd /home/pi/.ssh/  
$ vi authorized_keys
```

Paste or copy the Public key created on the PC to the **authorized_keys** file. (Some output in the following text omitted).

```
ssh-rsa  
AAAAAB3NzaC1yc2EAAAQABJQAAAQEauX+NEQoQECPN2d+Lu+qL2exMT/ICYbrNax6DVWBtKGzTxFOb  
:  
LeiaFbI3tWyi+ZPXg8Swhr1OaPN612E/fnAQPbG12S+YMtcIXknNiwVGL8RB3D8N/Q== rsa-  
key-20170216
```

Finally connect to the Raspberry Pi from the PC client using the **publickey** method. If this works OK disable password login method. Edit vi **/etc/ssh/sshd_config** and disable PAM and Password authentication.

```
PasswordAuthentication no  
:  
UsePAM no
```

Firewall set-up

Linux has a firewall facility built-in to the kernel called **iptables**. Rules may be added to **iptables** to allow or deny access to system services as required. However, **iptables** is static and has to be configured with any new rules. The **fail2ban** program enhances **iptables** by dynamically adding rules as required. For example, if a hacker attempts five unsuccessful logins using SSH then **fail2ban** blocks that IP address from connecting to the SSH port by adding a blocking rule to **iptables**. The following commands install and enable **fail2ban**.

```
$ sudo apt install fail2ban  
$ sudo systemctl enable fail2ban
```

Below is an example of a blocked host (195.22.126.242)

```
$ sudo iptables -L  
Chain INPUT (policy ACCEPT)  
:  
Chain FORWARD (policy ACCEPT)  
:  
Chain OUTPUT (policy ACCEPT)  
:  
Chain fail2ban-ssh (2 references)  
target     prot opt source          destination  
REJECT    all   --  195.22.126.242 anywhere reject-with icmp-port-unreachable
```

However, rules added by fail2ban will be lost if the Raspberry Pi is rebooted. An additional package called **iptables-persistent** can be added so that rules added by fail2ban can be made permanent. The following command allows installation.

```
$ sudo apt install iptables-persistent
```

The following command can be used to make the **iptables** rules persistent after a reboot. Run the following command before rebooting.

```
$ iptables-save
```

Search the internet for more information on **iptables** and **fail2ban**.

Chapter 14 - Frequently asked questions (FAQs)

What is the login name and password?

The default login name is: pi

The default password is: raspberry

You should change this at the earliest opportunity. See *Chapter 13 - Internet Security* on page 301.

How do I change the order of the radio stations?

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 in the `/var/lib/radiod/stationlist` file.

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.aspx
[BBC Radio 2] http://bbc.co.uk/radio/listen/live/r2.aspx
[BBC Radio 3] http://bbc.co.uk/radio/listen/live/r3.aspx
:
:
[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 their 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 show 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. Random always defaults to "off" when the radio is selected. However, when the media library is selected the value stored in the `/var/lib/radiod/random` file is used which is either 1 (Random on) or 0 (Random off). This value can be changed in the selection menu by toggling "Random on/off". So, the radio software will remember the desired random setting for the music library when it is restarted.

Can the volume be displayed as blocks instead of Volume nn?

Yes, it can. Volume is displayed by default as "Volume nn" where nn is 1 to 100. This can be changed as shown in section *Configuring the volume display* on page 181

Why do I see a station number on LCD line 3?

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 179.

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.

Is there a reboot or shutdown option?

There is only a shutdown option. Hold the menu button in for at least three seconds. The radio will stop and should display "Radio stopped" on the display. Wait ten seconds and then power off the Raspberry Pi. Powering back on achieves the same effect as a reboot. Also see *Fitting a wake-up button* on page 49.

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).

What Rotary Encoder can I use for this project?

It is possible to use any so called “Incremental Rotary Encoder” such as the COM-09117 12-step rotary encoders from sparkfun.com or the KY-040 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. These do not work with the radio software.

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 in other projects. See Licences on page 335.

You do not need to ask permission to re-use this code or documentation as it is already permitted in the licenses.

Can I use an Electronic Ink display?

The answer is not at the moment. There are a number of electronic ink displays on the market such as the Pimoroni Inky pHat and the Waveshare ink display. These displays appear to only suitable for static display of information and do not handle dynamically changing screens well. However, looking at on-line demo's the Waveshare device, it can handle partial screen updates however continuous updates appeared to damage the screen after a time. Maybe this will improve over time.

Can you make or sell me a radio?

The answer is NO. I regret I do not make any radios for sale nor do I make radios for other people. The whole purpose of this project is help hobby and enthusiasts to learn computing. If you want one of the radios shown in this manual you need to build it yourself.

Can you recommend the hardware for my project?

No - The author never makes recommendations as to which hardware someone should use for their project. There is a detailed overview, in this manual, of the possible solutions which are supported by the software.

No one can then ever say “well you recommended it” if there is a problem with the hardware selection. The choice of hardware is totally down to the constructor as only they know their exact requirements. Also, the author wishes to remain neutral when it comes to recommending any manufacturer’s hardware offering.

Can you make a change to the software for my project?

Many of the ideas and features in this project have come from the community of enthusiast’s and constructors. Their ideas and contributions have greatly enhanced both the usability and functionality of the project. These suggestions are always welcome and where possible, sooner or later, find their way into the product but not always.

What seems to be a simple change will mean months of testing of a new release. There are at least twelve basic designs plus variations all of which must be validated before the new software can be released.

However, if a constructor wishes to make software changes, help is available although this does not extend to either debugging their software or teaching Python.

What if I want to try different hardware?

Often a constructor comes across other hardware other than that published in this manual and ask whether or not it can be used. This often because it is cheaper, more locally available or better than the current options. The author will give an opinion as to whether or not it might work in their situation but will not take responsibility for the choice made for any given project. The choice of hardware is ultimately the responsibility of the constructor.

Sometimes there are cheaper versions a popular piece of hardware but prove to be disappointing. One such case was a cheap Chinese version of Adafruit’s RGB backplate. This has an RGB backlight. The cheap alternative simply put an RGB LED on the board which did not act as a backlight but rather as an indicator LED. Other cases of cheap alternatives to using the official Raspberry Pi 7-inch touch screen have also displayed start-up problems. So, caution is advised.

Sometimes the suggested hardware would require new or modified software and the author will consider this depending upon the merits of introducing such hardware and has done so many times in the past, thus enhancing the project.

Again, such suggestions and ideas are always welcome but do not guarantee that these suggestions will be supported by the Radio software.

Chapter 15 - Source files and package build

This section is only of interest if you are considering developing your own version of the software or wish to use one of the classes in your own software.

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 317.

The Radio program

The `radiod.py` program is the top-level radio program for the LCD versions of the radio and provides the logic for operating the radio. It is called from the `systemd radiod.service` script in the `/lib/systemd/system` directory.

The Radio Daemon

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

The Display Class

The `display_class.py` is the common top-level display used by the radio programs (`radiod`, `gradio` and `vgradio`) and is responsible for displaying messages on the various types of display. It uses the `display_type` parameter in the `/etc/radiod.conf` configuration file to load the correct software for the display being used. Depending upon the actual device configured it will load one of the following display drivers:

Table 27 Display classes

Display class file	Description	display_type
<code>lcd_class.py</code>	LCDs with a directly connected HD44780U	LCD
<code>lcd_i2c_adafruit.py</code>	LCDs with an Adafruit I2C backpack	LCD_I2C_ADAFRUIT
<code>lcd_i2c_pcf8574.py</code>	LCDs with a PCF8574 I2C backpack	LCD_I2C_PCF8574
<code>lcd_i2c_jhd1313.py</code>	Grove JHD1313 LCD RGB with I2C interface	LCD_I2C_JHD1313
<code>lcd_adafruit_class.py</code>	LCDs with an Adafruit RGB plate	LCD_I2C_ADAFRUIT
<code>lcd_adafruit_class.py</code>	Adafruit LCD I2C RGB plate with buttons	LCD_ADAFRUIT_RGB
<code>lcd_pifacecad_class.py</code>	PiFace CAD 2x16 LCD with push buttons	PIFACE_CAD
<code>oled_class.py</code>	Solomon Systech SSD1306 OLED Display - IQaudIO	OLED_128x64
<code>st7789tft_class.py</code>	Pimoroni Pirate audio with four push buttons	ST7789TFT
<code>ssd1306_class.py</code>	Sitronix SSD1306 128x64 pixel OLED	SSD1306
<code>no_display.py</code>	No display attached (vintage radio/Pirate Radio)	NO_DISPLAY
<code>luma_class.py</code>	LUMA driver for OLEDs with SSD1306, SSD1309, SSD1325, SSD1331, SH1106 and WS0010 chips	LUMA.<DEVICE>

The above settings are either configured by the `configure_radio.sh` program or by modifying the `display_type` parameter directly in `/etc/radiod.conf`. For example, if the PCF8574 I2C backpack is chosen then then the following will be configured in `/etc/radiod.conf`:

`display_type=LCD_I2C_PCF8574`

The Graphical Screen radio programs

The `gradio.py` and `vgradio.py` programs are the radio programs for the HDMI/Touchscreen version of the radio and is launched on the graphical desktop of the Raspberry Pi. It is optionally called from the `Desktop/gradio.desktop` script in the `/home/pi` directory. The `vgradio.py` program gives a vintage radio look and feel to the radio display.

In the case of both programs the `display_type` parameter in `/etc/radiod.conf` is set to GRAPHICAL and is set by the `configure_radio.sh` program.

The Graphics display class

The `graphic_display.py` class performs auxiliary display functions such as scrolling and screen mapping for the `gradio.py` and `vgradio.py` programs.

The Graphics controls class

The `gcontrols_class.py` handles the creation of all graphics controls and widgets for the graphic version of the radio. It also uses the SGC widget routines in the `sgc` sub-directory from Sam Bull and Michael Rochester.

The OLED class

The `cosmic_class.py` is the display interface for the SSD1306 128x64 pixel OLED display supplied with the IQaudIO Cosmic controller. This class is a wrapper for the routines from **Olimex Limited** in the sub-directory `oled`. It drives the OLED screen although not all functions are used by the radio.

The button class

The `button_class.py` detects all button presses from the push button radios (Not the Adafruit RGB nor the PiFace CAD which have their own buttons using I2C and SPI interfaces respectively). It passes button press events up to the event class described later.

The rotary class

The `rotary_class.py` and `rotary_class_alternative.py` detect all rotary encoder events from the radios fitted with rotary encoders. It passes rotary encoder events up to the event class described later.

The Cosmic controller Class

The `cosmic_class.py` is used as the user interface for the IQaudIO Cosmic controller. This provides the interface for three-button and rotary control interface on the controller board.

The Event class

All user interfaces in the radio software generate a largely common set of events. These are handled by the `event_class.py`. The `event_class.py` program accepts events from the following sources:

- The `gradio.py` and `vgradio.py` graphical radio programs
- The `radiod.py` radio program
- The push button interface user interface (`button_class.py`)
- The rotary encoder user interface (Either `rotary_class.py` or `rotary_class_alternative.py`)
- The IR remote control user interface (`irradiod.py` or `ireventd.py`)
- The radio Web user interface running on an Apache Web server

The Menu class

The *menu_class.py* code provides the logic for stepping through the various menus and their options.

The Message class

All messages are generated from the *message_class.py* program. This uses message labels to load the correct text to be displayed or spoken. By using labels and the *language_class.py* software, the radio can be configured to use any language using a Latin character set. It provides messages to display various menu's, time, station and track information.

The Language class

The *language_class.py* provides the text for both the radio display or the **espeak** package. It reads the **/var/lib/radiod/language** file (if present) and passes the text to both the message class and if used. It is used by the message class to deliver messages in the users own language.

The Log class

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

The Volume class

The *volume_class.py* program handles all volume and mixer functions for the radio, Spotify and airplay.

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 will be converted again to a valid character in the standard LCD character set.

The create_stations program

The *create_stations.py* program creates playlist files in the **/var/lib/mpd/playlists** directory using a list of Web links (URLs) with titles as input. This program creates standard playlists for use with with MPD. The operation of the *create_stations.py* program is covered in detail in the section on managing playlist files on page 215.

The update_stationlist program

The *update_stationlist.py* program reads each of the RADIO playlist files in the **/var/lib/mpd/playlists** directory and creates a new **/var/lib/radiod/stationlist** file. This is required to add new station URLs dynamically added by any external MPD clients that are capable of doing so. It is run from the **/etc/cron.daily/radiod** crontab script.

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 or for gaining an insight into the sort of information provided by the MPD daemon.

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 be used in normal operation.

The `configure_radio.sh` script

The `configure_radio.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. It configures the display to be used and the user interface.

The playlist creation program

The `create_playlist.sh` script creates playlists from music directories on either a USB stick or a Network drive such as a NAS. It has the ability to accept filters to make a more selective playlist.

The `configure_audio.sh` script

The `configure_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 `configure_audio_device.sh` script

The `configure_audio_device.sh` script is called by the radio program when it starts up. It is used to select the sound card to be used. This is because Raspbian have recently changed the way audio devices are numbered and these can vary if an HDMI device is attached or not.

It configures `/etc/mpd.conf` and `/etc/asound.conf` with the correct device number depending upon the output of the `aplay -l` command. The script reads the output of the `aplay -l` command and compares each line with the setting of the `audio_out` parameter in `/etc/radiod.conf`.

If the above `audio_out` parameter is missing add it to `/etc/radiod.conf`.

The `configure_audio.sh` script sets the `audio_out` parameter when run during setup. Depending on the sound device selected it sets it to **headphones**, **HDMI**, **USB** or **DAC**

To understand this script run the `aplay` command to display available sound cards (The following is an example only).

```
$ aplay -l | grep -i card
card 0: b1 [bcm2835 HDMI 1], device 0: bcm2835 HDMI 1 [bcm2835 HDMI 1]
card 1: Headphones [bcm2835 Headphones], device 0: bcm2835 Headphones [bcm2835 Headphones]
card 2: IQaudIODAC [IQaudIODAC], device 0: IQaudIO DAC HiFi pcm512x-hifi-0 [IQaudIO DAC HiFi pcm512x-hifi-0]
```

In the above example, three sound cards(devices) are shown. Edit the `/etc/radiod.conf` file and add the following parameter to the `radiod.conf` file.

`audio_out=<device>`

Where `<device>` is any unique string (when compared with other lines)

from the required card definition.

For example:

```
audio_out="Headphones" {Will configure Card 1 On-board audio output jack}
audio_out="IQaudIO DAC" {Will configure Card 2 IQaudIO DAC}
audio_out="HDMI 1" {Will configure Card 0 HDMI 1}
audio_out="DAC" {Will configure Card 2 IQaudIO DAC}
```

The `configure_ir_remote` script

The `configure_ir_remote.sh` script sets up and partially configures both the LIRC (Linux Remote Control) and Kernel Event driver components for an IR remote control. It sets up the correct system service `irradiod.service` or `ireventd.service` depending upon the selection made. See *The remote control daemons* on page 314 for further information.

The `set mixer id` script

The `set_mixer_id.sh` script works out the “Speaker Playback Volume” mixer ID and configures the `mixer_volume_id` in `/var/lib/radiod` directory. This information comes from the `amixer controls` command. This mixer ID (integer) is used to set a default mixer volume for MPD and also is used for volume control when using Airplay. The `set_mixer_id.sh` script is normally called from the radio program (all versions), usually after a reboot, if the `mixer_volume_id` parameter has been removed by the `configure_audio.sh` program. This script also completes configuration of HDMI audio if selected in the `configure_audio.sh` program. It is not normally necessary to run this program separately but can be safely run at any time.

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 19 Remote Control Key names) to a UDP server running in the radio program. It is also used to send commands from the Web Interface to the radio program.

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

The Status LED class

The `status_led_class.py` is called by the vintage radio software. 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 Airplay Class

The `airplay_class.py` file contains the routines for stopping and starting the `shairport-sync` daemon and for getting artist, title and album of the playing track. It is used when Airplay is selected as the source.

The Menu Switch class

The `menu_switch_class.py` code supports an 8-position rotary switch (Not encoder) as an alternative method of operating a simple menu system. It is meant to be used with the vintage radio software but can be used with any variant.

Current_station

The init file

The `__init__.py` file contains a couple of global definitions plus the package version number.

Alsa sound files

The Alsa sound configuration is contained in `/etc/asound.conf` file. You may see mention of the `.asoundrc` file in Alsa documentation but it is not used by the radio project. The `.asoundrc` file does exactly the same as `/etc/asound.conf` file but is specific to one user and is placed in their home directory.

Depending upon the device to be configured a different configuration will be used.

These asound files are contained in the `/usr/share/radio/asound` directory. The correct file is copied to the `/etc/asound.conf` file by the `configure_audio.sh` sound configuration script.

Table 28 Asound configuration files

Asound file	Description
<code>asound.conf.dist</code>	Standard asound.conf
<code>asound.conf.dist.bonnet</code>	Adafruit speaker bonnet Alsa configuration file
<code>asound.conf.dist.pivumeter</code>	Pimoroni pHat with VU meter configuration file.
<code>asound.conf.dist.blue</code>	Bluetooth devices Alsa configuration file
<code>asound.conf.dist.equalizer</code>	Configuration for the Alsa graphic equalizer
<code>asound.conf.dist.pipe</code>	Bluetooth devices Alsa configuration file using a pipe
<code>asound.conf.dist.bonnet</code>	Configuration file for use with the speaker bonnet
<code>asound.conf.dist.softvol</code>	Configuration file for devices that don't have hardware volume controls.

The remote control daemons

There are two remote-control daemons supported with this version.

There is a service start stop is uses `systemctl`. The `udp_server_class.py` program is used for communication between the remote-control daemon and the radio program. The two daemons are:

1. The `ireventd` daemon which is configured using the `ir-keytable` utility
2. The `irradiod` service using the `irrecord` utility program

Method 1 (`ireventd`) should be used by all new users. The `irradiod` service is for legacy support only and is largely redundant. The following table summarizes the methods.

Method	Configurator	System Daemon/driver	Radio daemon	Daemon class	Pidfile
1	IR-KEYTABLE	ir-keytable	evdev (kernel)	ireventd.py	ir_daemon.py
2	LIRC LIRC(Buster)	irrecord " "	lircd (daemon) " "	irradiod.py remote_control.py	ir_daemon rc_daemon " "

It isn't particularly important to remember this table other than there are two methods. It is mainly meant as a reference for any developers.

The IR remote-control ireventd.service

This IR service runs on Bullseye or later. It has not been tested on Buster. Configuration for the IR remote control is done by the **ir-keytable** utility. This is the service that all new users should be using. The command to start the IR **ireventd.service** is:

```
$ sudo systemctl start ireventd.service
```

To stop the IR radio service run:

```
$ sudo systemctl stop irradiod.service
```

The IR Remote Control irradiod.service

The IR Remote Control service is started from the **irradiod.service** definition using the **systemctl** command. It is largely redundant and is only included for legacy support. The actual program it starts depends on the version of the operating system being used.

For Buster the **irradiod.service** starts the **remote_control.py** program. In the case of **Bullseye** or later it starts the **irradiod.py** program. In both cases the command to start the IR radio daemon is:

```
$ sudo systemctl start irradiod.service
```

To stop the IR radio service run:

```
$ sudo systemctl stop irradiod.service
```

When the Remote-Control buttons are pressed it passes the relevant commands to the radio program. The **irradiod.py** or **remote_control.py** program which 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.

Table 29 The irradiod service

OS Version	Service	IR program	IR daemon class	Python
Bullseye or later	irradiod.service.bullseye	irradiod.py	ir_daemon.py	3
Buster	irradiod.service.buster	remote_control.py	rc_daemon.py	2



NOTE: There are no functional differences between the **Buster** and **Bullseye** versions of the **irradiod.service** programs. The only difference is that the **Bullseye** version is written in **Python 3** and the **Buster** version in **Python 2**.

During installation using the **configure_ir_remote.sh** script copies either **irradiod.service.bullseye** or **irradiod.service.buster** to **/usr/lib/systemd/system/irradiod.service**.

In the following text the commands are using the **Bullseye** version. Substitute **irradiod.py** with **remote_control.py** for the **Buster** version.

To start the IR daemon run:

```
$ sudo systemctl start irradioiod
```

To stop the IR daemon run:

```
$ sudo systemctl stop irradioiod
```

Run the **irradioiod.py** or **remote_control.py** program to display the help message.

```
$ ./irradioiod.py
```

This program must be run with sudo or root permissions!

Usage: sudo ./remote_control.py start|stop|status|nodaemon|flash|config|send <KEY>

The start, stop and status commands are obvious. The nodaemon command can be used for diagnostic purposes.

Then run the **remote_control.py** or **irradioiod.py** program for Buster or Bullseye respectively.

For

```
$ sudo ./irradioiod.py nodaemon
IR Remote control listener running pid 1589
Protocols changed to unknown other lirc rc-5 rc-5-sz jvc sony nec sanyo
mce_kbd rc-6 sharp xmp cec imon rc-mm
Loaded BPF protocol xbox-dvd
Flashing LED on GPIO 16
Listening for input on IR sensor
```

Pressing keys on the remote control should display the key names.

```
KEY_VOLUMEUP
KEY_VOLUMEUP
KEY_VOLUMEDOWN
KEY_VOLUMEDOWN
KEY_CHANNELUP
KEY CHANNELDOWN
```

The configuration can be displayed with the config parameter

```
$ sudo ./irradioiod.py config
LED = GPIO 16
HOST = localhost
PORT = 5100
LISTEN = localhost
dtoverlay= gpio-ir, gpio_pin=25
```

The send function can be used to test that the radio daemon is receiving commands OK.

For example, the following command will change channel up.

```
$ sudo ./irradioiod.py send KEY_CHANNELUP
OK
```

Further information on the **irradiod** service file will be found in *Appendix A - System Files used by the Radio Program* in the **/usr/lib/systemd/system/irradiod.service** description on page 349

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 97. You can view the Raspberry PI source at <https://github.com/bobrathbone/piradio6>



Note: This may be out of date compared to the latest version. All source files, including build files, are included in the **radiod** package anyhow so downloading from GitHub isn't necessary.

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_\(software\)](http://en.wikipedia.org/wiki/Git_(software))

Install **git** with the following command:

```
$ sudo apt 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**.

To find out more about **git** and for general support and documentation see:
<http://git-scm.com>

Chapter 16 Advanced topics

Booting from a USB drive or stick

The Raspberry Pi is normally booted from an SD card; however, it is also possible to boot from a USB 3.0 disk drive or stick. The drive can be either a USB drive with a spindle or a Solid-State Drive (SSD) or a USB stick. This procedure works with Raspberry Pi's with USB 2.x or 3.x ports such as models 3B, 3B+ or 4B+. The Pi 400 and 4B may already be USB boot enabled and can usually boot from a USB drive without modifying the firmware.

Before starting this procedure, it is important to understand that there are two types of USB drive protocols of importance with respect to the driver software/hardware/firmware:

1. USB drives that support the **USB Attached SCSI** (UAS) protocol
2. USB drives that support USB Mass Storage **Bulk-Only Transport** (BOT) protocol

UAS drivers generally provide faster transfers when compared to the older **BOT** protocol drivers. When used with an SSD, **UAS** is considerably faster than **BOT** for random reads and writes. For UAS to be enabled the USB Drive, hardware controller, USB driver software and firmware must all support **UAS** for it to work.



Regrettably **Bullseye** does not appear to be properly supporting non-UAS drives and this procedure currently only works with **Buster** for these types of USB drive. This may be a configuration issue and is still being investigated by the author.



Figure 234 Raspberry Pi booting from an Intenso USB 3.0 2.5-inch disk drive

The advantages of booting from a USB disk drive are:

- A USB disk drive is a lot less susceptible to corruption than a SD card
- A USB disk drive generally has greater capacity than a SD card for media files and the like
- USB disk drives are a lot faster than a SD card. SSD disks are even faster but limited to USB 3.0 (or 2.x) speeds.

Disadvantages:

- Drives using a spindle are theoretically prone mechanical failure, however modern solid-state drives or USB sticks don't suffer from this problem.

To make a bootable USB disk drive you will need:

1. A Raspberry Pi with USB 3.0 ports or newer RPis (3B+) with USB 2.0 ports.
2. A spare micro-SD Card
3. A USB disk drive or stick (For example a 2.5-inch USB 3.0 SSD)
4. Raspberry Pi imager software



Note: After carrying out this routine it will still be possible to boot from an SD card. Simply power off, disconnect the USB disk drive, insert a SD card with an OS image and power back on. The Raspberry Pi will now boot from the SD card.



Note: During testing USB sticks were found to be just as vulnerable as SD-cards. It is better to use USB 3.0 2.5-inch SSD drives.

Pi Zero W

Booting from a USB drive on a Pi Zero W is not supported. The Pi Zero W uses a BCM2835, which contains the old SD card-only bootloader. Secondly the USB ports have very limited power availability (Standard 600 mA).

USB Disk Drive power consumption

A small USB disk drive with spindle (250 GByte) takes about 260 Milliwatts. The allowed USB power load for the Raspberry Pi is 1.2A across all four USB ports so there is plenty of power available for other USB devices. Solid State Drives (SSD) take significantly less power and do not suffer from mechanical failure. When the Raspberry Pi is shut down, the power is removed from the USB ports and the disk drive is completely powered off.

Enabling the Raspberry Pi to boot from a USB drive

Making a Raspberry Pi boot from a USB drive or stick depends upon the Raspberry Pi model.

- Method 1 - Models with USB 2.0 ports such as the RPi model 3B+
- Method 2 - Models with USB 3.x ports such as the RPi model 4B

Method 1 - Enabling USB booting on models with USB 2.0 ports

This method Raspberry Pi's which have USB 2.0 ports such as the RPi model 3B+.

To enable the Raspberry Pi's USB boot mode, it is necessary to add a configuration option to the **/boot/config.txt** file. When the RPi is rebooted, this will set a bit in the Raspberry Pi's OTP (One Time Programmable) memory, allowing the device to be booted from a USB mass storage device. After that, we won't need the SD card anymore.

To enable USB boot mode, open a terminal session and run the following command:

```
$ echo program_usb_boot_mode=1 | sudo tee -a /boot/config.txt
```

This adds the config option **program_usb_boot_mode=1** to the end of **/boot/config.txt** file. Reboot the Pi with the SD card still inserted.

```
$ sudo reboot
```

After the reboot, check that boot mode has been enabled with the following command:

```
$ vcgencmd otp_dump | grep 17  
17:3020000a
```



Note: If you are going to use your microSD card with a different Raspberry Pi later on, you might wish to remove the **program_usb_boot_mode=1** line from **config.txt** file, so that the boot mode won't be programmed to that device as well. Edit the file using the command **sudo nano /boot/config.txt**, and remove the above line.

Now power off and proceed to the section called Now carry out the instructions shown in section *Install the Radio Daemon* on page 97.

Method 2 - Enabling USB booting on models with USB 3.x ports

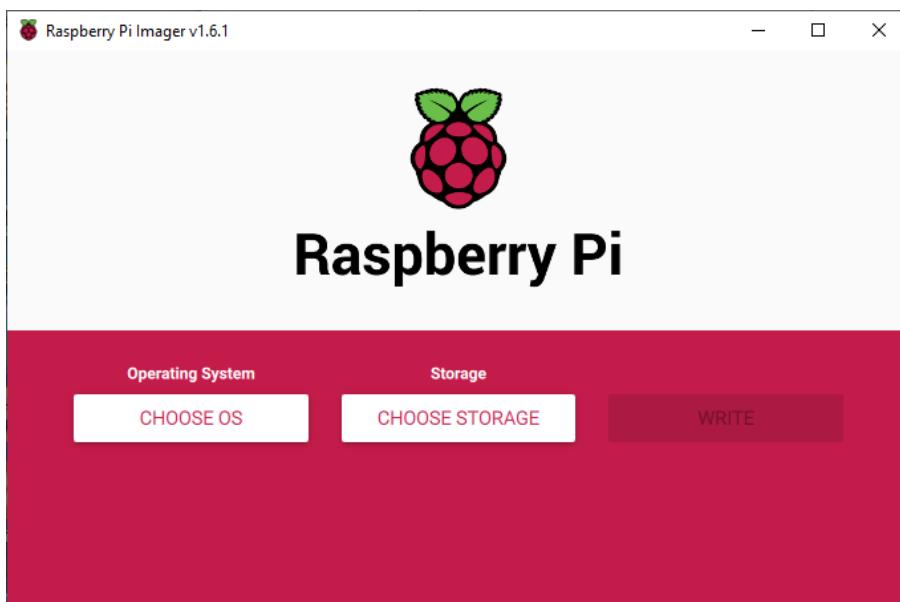
This method Raspberry Pi's which have two USB 3.0 ports and two USB 2.x such as the RPi model 4B or Pi 400. The model 4B and Pi 400 running the latest firmware come already enabled to boot from USB devices. This can be checked with the following command:

```
$ vcgencmd otp_dump | grep 17  
17:3020000a
```

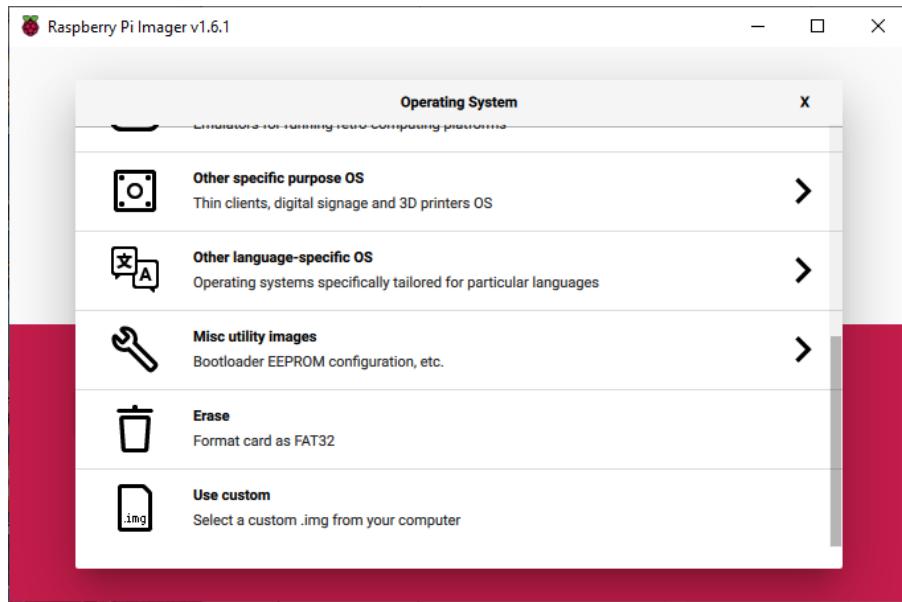
The value shown must be **17:3020000a**. If this is the case then proceed to the section called Now carry out the instructions shown in section *Install the Radio Daemon* on page 97. If not continue below.

The SD card is loaded with some special software to overwrite the EEPROM code to boot from USB. Download the Raspberry Imager software from <https://www.raspberrypi.org/software/>

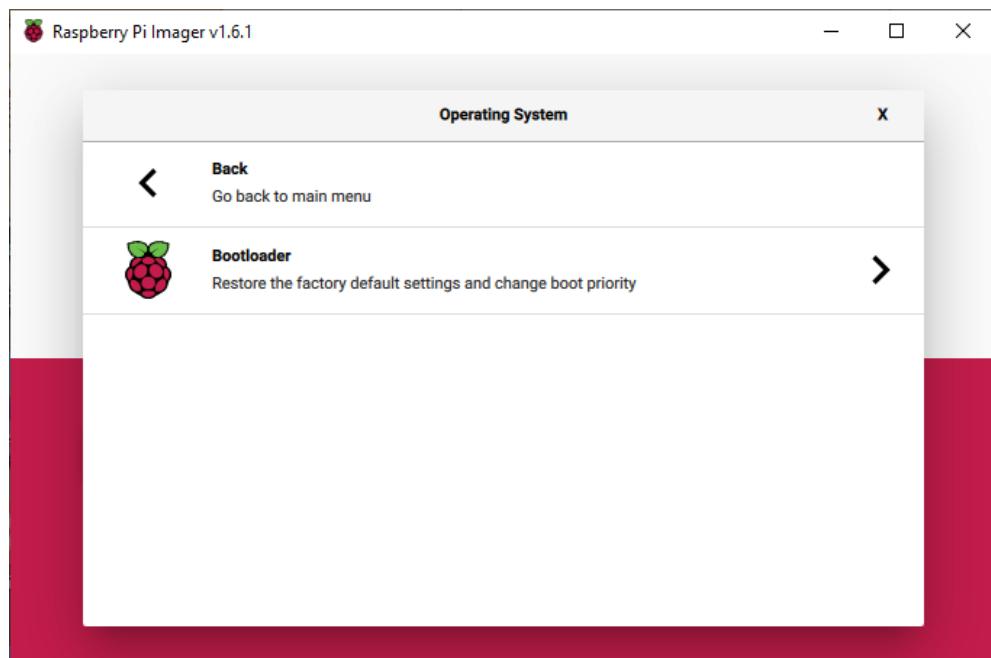
Insert the spare SD card into the PC and start the Raspberry Pi Imager software.



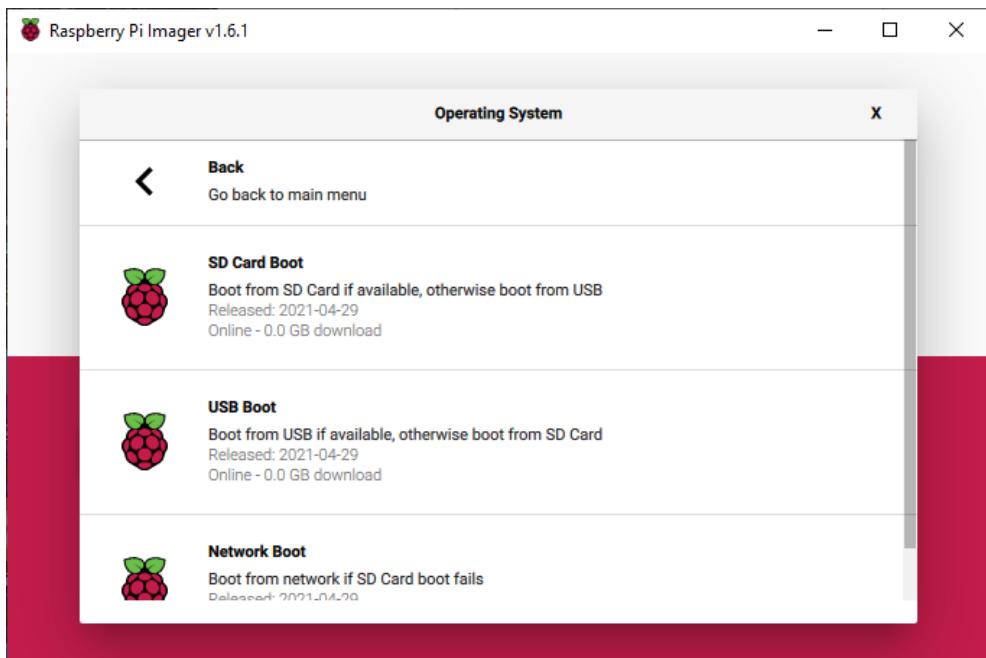
Select “CHOOSE OS”



Use the scroll down and select “Misc Images – Bootloader EEPROM configuration”



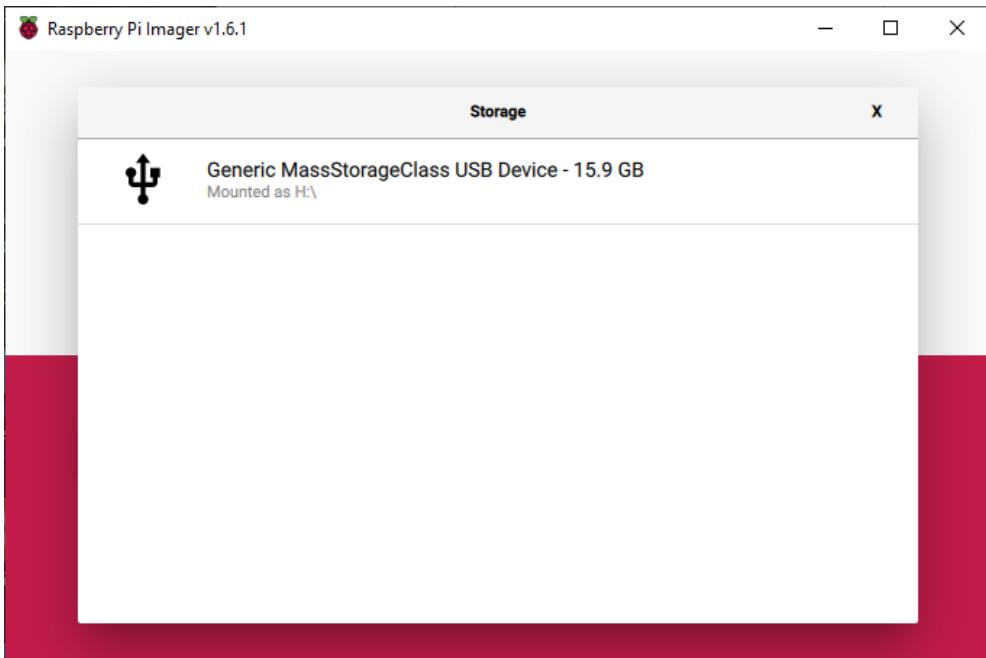
Click the right arrow “>” on “Bootloader” to show the options below.



Select the first or second option depending upon your operational requirements. Typically, this will be the second option “Boot from USB if available, otherwise boot from SD card”.



Click on “Choose storage”



Select your SD card:



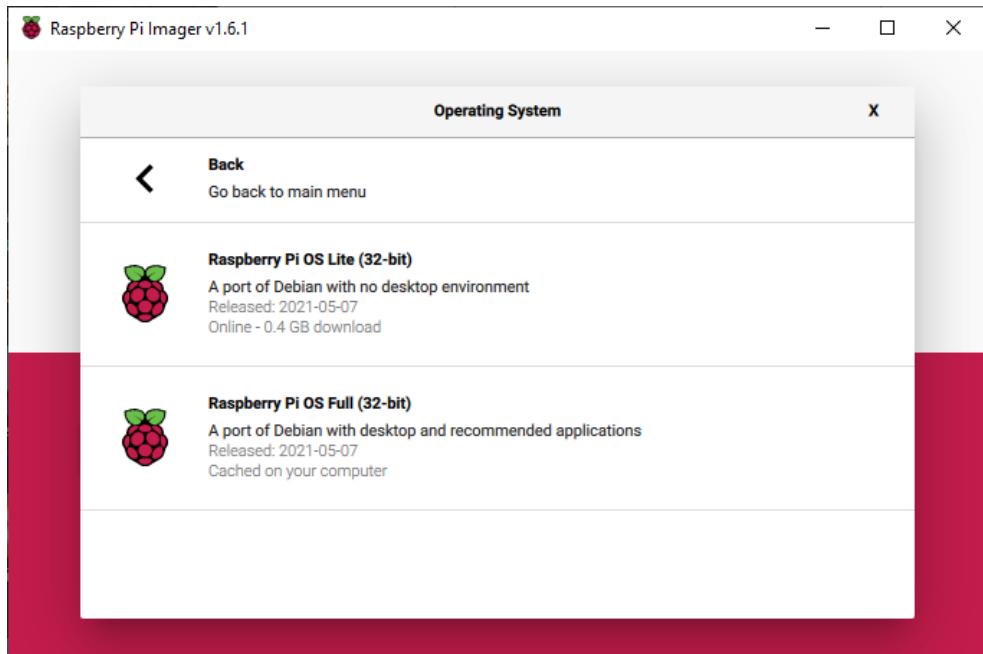
Now press “Write” to create the SD Card image. This will be very quick as there isn’t much to write. Once written, remove the SD card from the PC and label it as USB boot. You will need this later.

Creating the bootable USB drive

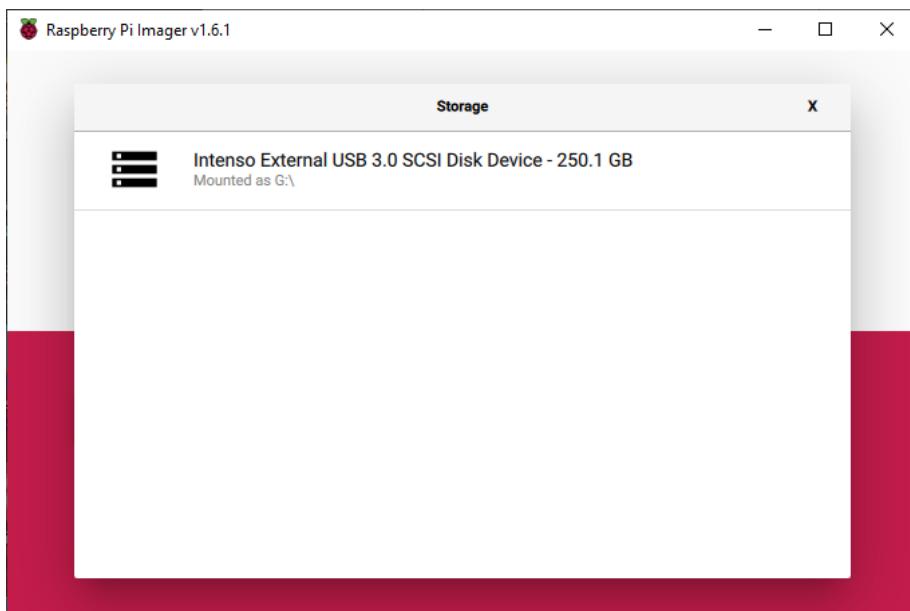
Plug your new USB disk drive into the PC.

Now create the Operating system on the USB disk drive (Not a USB stick). Restart the Raspberry OS imager.

Select the required operating system. Recommended is the “Raspberry Pi OS full (32 bit)”



Now select the USB drive. In the example below this is a USB 3.0 250 Mbyte 2.5-inch disk drive which should be more than enough for most purposes.



Now write to the USB disk as previously shown. Once complete, remove the USB drive from the PC.

Modifying the EEPROM boot order

Take the SD card that you created and labelled as “USB boot” and put it into the Raspberry Pi SD card slot. Power on the Raspberry Pi.

After a few seconds you will see the LED on the Raspberry Pi repeatedly flashing. This means that it has finished writing to the RPi EEPROM. Power off the Raspberry Pi and remove the SD card and put it somewhere safe so that you don’t accidentally put it into any other Raspberry Pi.

USB 2.x SDD disk drive not booting

Disable the VC4 DRM driver.

Edit the /boot/cmdline.txt file using sudo. Disable VC4 driver by commenting it out.

```
#dtoverlay=vc4-kms-v3d
```

Reboot the Raspberry Pi.

USB 3.x SDD disk drive slow booting

Skip this section if you are using a normal disk drive with a spindle.



Note: At the time of writing there is a known problem with some USB 3.0 SDD drives which do not support the UAS protocol and cause them to operate very slowly to a point of being unusable. It is necessary to modify the **/boot/cmdline** file to correct this.

For more information on this problem see the following article:

<https://www.raspberrypi.org/forums/viewtopic.php?f=28&t=245931>

Find the SDD Vendor and Product ID

If using a USB SSD disk drive, it is necessary to first identify the Vendor and Product ID of the USB SDD disk drive and its device. Remove any USB storage devices and boot up the Raspberry Pi from an SD card. Once fully booted up, insert the SDD USB drive into the USB 3.0 port.

Log into the Raspberry Pi and then run the dmesg (ring buffer display) program:

```
$ dmesg
```

This should display details of the newly inserted SSD disk. The following example shows the details for an Intenso 250 Gigabyte SDD drive.

Note the Vendor and Product ID. In this example it is **152d** and **0579** respectively. Also note the device name of the first partition (/boot) of the SSD drive. This is **sda1** in this case.

```
[ 235.244594] usb 2-1: new SuperSpeed Gen 1 USB device number 3 using xhci_hcd
[ 235.276190] usb 2-1: New USB device found, idVendor=152d, idProduct=0579,
bcdDevice= 5.06
[ 235.276213] usb 2-1: New USB device strings: Mfr=1, Product=2,
SerialNumber=3
[ 235.276231] usb 2-1: Product: Portable SSD
[ 235.276249] usb 2-1: Manufacturer: Intenso
[ 235.276266] usb 2-1: SerialNumber: 2020090810405
[ 235.315953] scsi host1: uas
[ 235.317600] scsi 1:0:0:0: Direct-Access      Intenso  Portable SSD
0506 PQ: 0 ANSI: 6
[ 235.319836] sd 1:0:0:0: Attached scsi generic sg1 type 0
[ 235.833551] sd 1:0:0:0: [sdb] 500118192 512-byte logical blocks: (256
GB/238 GiB)
[ 235.833561] sd 1:0:0:0: [sdb] 4096-byte physical blocks
[ 235.833760] sd 1:0:0:0: [sdb] Write Protect is off
[ 235.833769] sd 1:0:0:0: [sdb] Mode Sense: 53 00 00 08
[ 235.834181] sd 1:0:0:0: [sdb] Write cache: enabled, read cache: enabled,
doesn't support DPO or FUA
[ 235.834862] sd 1:0:0:0: [sdb] Optimal transfer size 33553920 bytes not a
multiple of physical block size (4096 bytes)
[ 235.876806] sda: sda1 sda2
```

```
[ 235.879360] sd 1:0:0:0: [sdb] Attached SCSI disk
```

Now mount **/dev/sda1** on the **/mnt** partition. This will be very slow, about one minute, so be patient:

```
$ sudo mount /dev/sda1 /mnt
```

Using sudo edit the **cmdline.txt** file.

```
$ cd /mnt  
$ sudo nano cmdline.txt
```

Add the following line to the front of all of the other options and save the file. Use the Vendor and Device ID previously identified for your device.

```
usb-storage.quirks=152d:0579:u
```

The **cmdline.txt** file should look like the following

```
usb-storage.quirks=152d:0579:u console=serial0,115200 console=tty1  
root=PARTUUID=19de2757-02 rootfstype=ext4 elevator=deadline fsck.repair=yes  
rootwait quiet splash plymouth.ignore-serial-consoles  
systemd.run=/boot/firstrun.sh systemd.run_success_action=reboot  
systemd.unit=kernel-command-line.target
```

Now sync the discs.

```
$ sync;sync
```

Log out and power off and remove the SD card. Leave the drive plugged in and got to the next section.

Boot up the Raspberry Pi from the USB disk drive

Now plug the USB disk drive into one of the USB 3.0 ports and power on. If everything has been correctly done, the Raspberry Pi will boot up from the USB disk drive. If not, check that this procedure has been correctly carried out. This can also be confirmed with the **dmesg** command.

```
$ dmesg | grep -i uas  
[    1.515463] usbcore: registered new interface driver uas  
[    2.161159] usb 2-1: UAS is ignored for this device, using usb-storage instead  
[    2.161289] usb 2-1: UAS is ignored for this device, using usb-storage instead
```

If all is well then continue with installation of the rest of the Operating System as shown *Preparing the Operating System for software installation* on page 84.

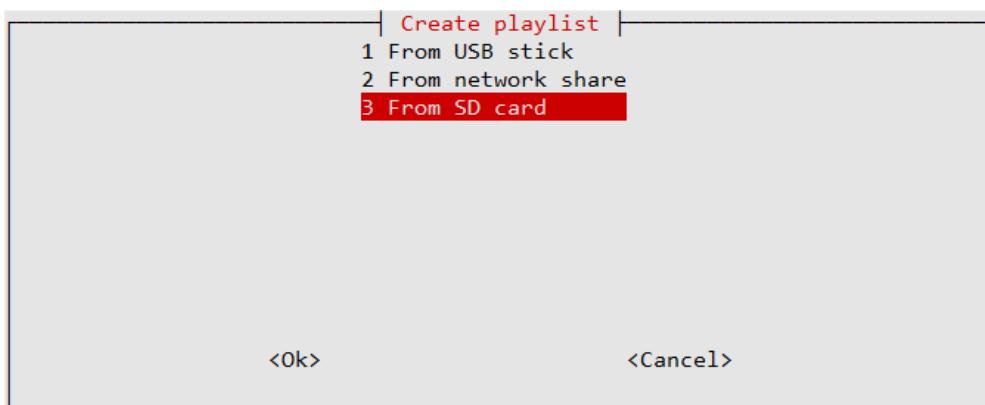
If you wish to boot from an SD card instead of the USB disk drive then remove the USB drive from the USB 3.0 port and insert an SD card configured with the Raspberry Pi OS into the SD card slot and power on. The Raspberry Pi will detect that it does not have a USB drive anymore and will boot from the SD card.

Playing music from the USB disk drive

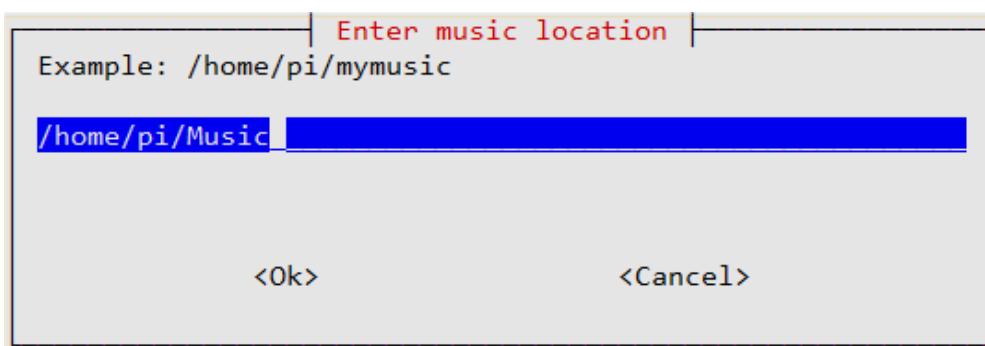
If using a USB disk drive, it makes more sense to play media files from the USB disk drive rather than a separate USB stick.

Using SFTP, copy the music from a PC to the `/home/pi/Music` directory which should already exist and reload the library via the options menu.

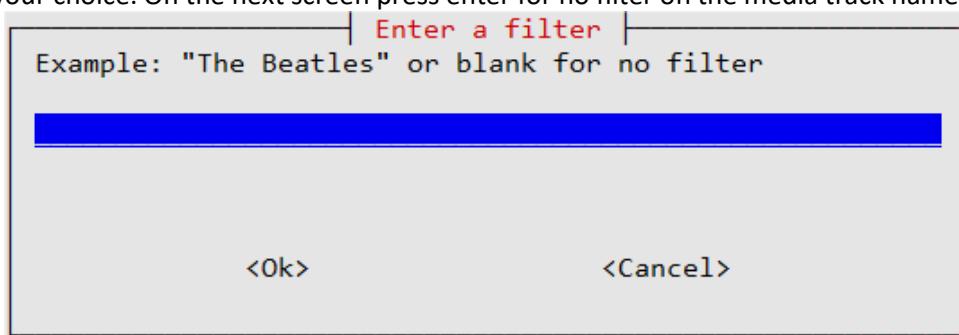
Now run the `create_playlist.sh` program. Select option 3 (SD card).



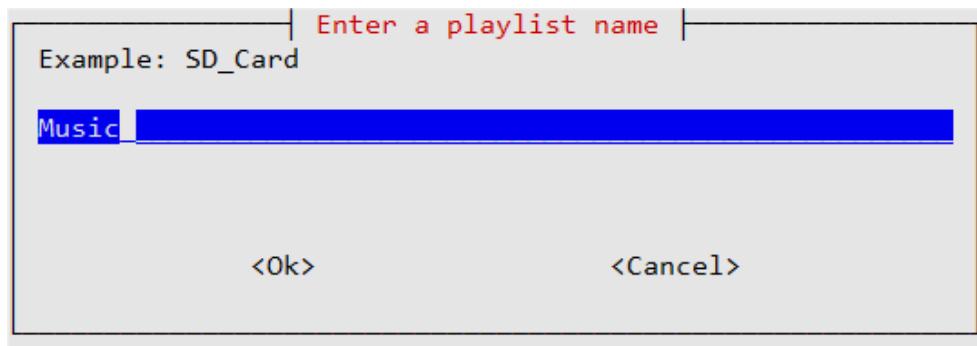
Choose option 3 From SD card even though the music is on your USB drive. Do not select option 1.



Confirm your choice. On the next screen press enter for no filter on the media track names.



Change the default location to `/home/pi/Music`



Press OK. The program will now create the Music playlist. Restart the radio and select the source men to select the Music playlist.

Repairing a non-bootable SD card

One of the most frustrating things is when your radio which has been working fine and suddenly stops booting up. This is extremely frustrating but all may not be lost. It may be possible to repair an SD Card which has become non-bootable. To do this you will need the following:



- A USB SD card reader
- A spare SD card with Raspberry Pi OS

USB SD card readers a very cheap. You should buy one before you need it in an emergency. Also prepare a spare SD card with the Raspberry Pi OS on it.

It is best to do this before you need it in an emergency. See *SD card creation using Raspberry Pi Imager* on page 74

Figure 235 A USB SD card reader

1. Boot up the Raspberry Pi with the good SD card previously created.
2. Log into the Raspberry Pi as user **pi**.
3. Plug the faulty SD card into the USB SD card reader and plug the card reader into the Raspberry Pi.
4. Run the **dmesg** command. This should show the device number of the SD card at the end, similar to that shown below.

```
$ dmesg
:
[ 700.247477] sd 0:0:0:1: [sdb] Attached SCSI removable disk
[ 700.252572] sda: sda1 sda2
[ 700.260025] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

Now run the following **fsck** commands to attempt to repair the file systems on **sda1** and **sda2**.

```
$ sudo fsck -y /dev/sda1
fsck from util-linux 2.33.1
fsck.fat 4.1 (2017-01-24)
0x41: Dirty bit is set. Fs was not properly unmounted and some data may be
corrupt.
Automatically removing dirty bit.
```

```
Performing changes.  
/dev/sda1: 285 files, 98417/516190 clusters
```

/dev/sda2

```
$ sudo fsck -y /dev/sda2  
fsck from util-linux 2.33.1  
e2fsck 1.44.5 (15-Dec-2018)  
rootfs: recovering journal  
Clearing orphaned inode 914 (uid=111, gid=119, mode=0100600, size=0)  
Setting free inodes count to 765822 (was 765872)  
Setting free blocks count to 2122943 (was 2126334)  
rootfs: clean, 149826/915648 files, 1607489/3730432 blocks
```

Run the following to synchronise the disk buffers with the disk.

```
$ sync;sync
```

Now power off the Raspberry Pi and plug the repaired SD card into the Raspberry Pi and power on. The above actions should normally be enough to get the RPi to boot. If not carry out the following:

First repeat the above procedure to make sure that the file system on the SD card is OK. Now mount **/dev/sda1** on the **/mnt** directory. Then change to the **/mnt** directory.

```
$ sudo mount /dev/sda1 /mnt  
$ cd /mnt
```

Check the **cmdline.txt** and **config.txt** file sizes look reasonable.

```
$ ls -la cmdline.txt config.txt  
-rwxr-xr-x 1 root root 144 Sep 27 11:45 cmdline.txt  
-rwxr-xr-x 1 root root 1838 Aug 28 14:00 config.txt
```

Check the **cmdline.txt** file looks something like that shown below.

```
$ cat cmdline.txt  
console=tty1 root=PARTUUID=0b18c756-02 rootfstype=ext4 elevator=deadline  
fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-consoles
```

Check that the **config.txt** file looks normal. This will vary from that shown below.

```
$ cat config.txt  
:  
# Enable audio (loads snd_bcm2835)  
dtparam=audio=on  
  
[pi4]  
# Enable DRM VC4 V3D driver on top of the dispmanx display stack  
dtoverlay=vc4-fkms-v3d  
max_framebuffers=2  
  
[all]  
dtoverlay=hifiberry-dac
```

```
dtoverlay=i2s-mmap
```

If the **config.txt** file looks damaged or is missing, copy it from the good **SD** card. It may be necessary to re-configure the audio system (Run **configure_audio.sh** once rebooted).

Faulty **cmdline.txt** example. The file size in this example is zero bytes size.

```
$ ls -la cmdline.txt config.txt
-rwxr-xr-x 1 root root 0 Sep 27 11:45 cmdline.txt
```

First copy a good **cmdline.txt** file from the good SD card to the faulty one.

```
$ sudo cp /boot/cmdline.txt /mnt
```

However, this one will still not work as the root file system UUID is incorrect as it is for the good SD card and not the root UUID of faulty one.

```
$ cat /boot/cmdline.txt
console=tty1 root=PARTUUID=dfde9f91-02 rootfstype=ext4 elevator=deadline
fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-consoles
```

Find out the PARTUUID of **/dev/sda2** with the **blkid** utility.

```
$ sudo blkid /dev/sda2
/dev/sda2: LABEL="rootfs" UUID="3a324232-335f-4617-84c3-d4889840dc93"
TYPE="ext4" PARTUUID="0b18c756-02"
```

Edit the **cmdline.txt** file and insert the correct PARTUUID

```
$ cat /boot/cmdline.txt
console=tty1 root=PARTUUID=0b18c756-02 rootfstype=ext4 elevator=deadline
fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-consoles
```

Run the following to synchronise the disk buffers with the disk.

```
$ sync;sync
```

Try booting using the repaired SD card. If there are still problems, check other files in the boot partition (**/dev/sda1** mounted on **/mnt**) of the faulty SD card. For example, the kernel files:

```
$ ls -la /mnt/kernel*
-rwxr-xr-x 1 root root 6353376 Aug 15 13:46 /mnt/kernel17.img
-rwxr-xr-x 1 root root 6777440 Aug 15 13:46 /mnt/kernel17l.img
-rwxr-xr-x 1 root root 7904011 Aug 15 13:47 /mnt/kernel18.img
-rwxr-xr-x 1 root root 6006712 Aug 15 13:47 /mnt/kernel.img
```

On the good SD card. The dates may be different but they otherwise look OK

```
$ ls -la /boot/kernel*
```

```
-rwxr-xr-x 1 root root 6353624 Sep 21 14:16 /boot/kernel7.img  
-rwxr-xr-x 1 root root 6779080 Sep 21 14:16 /boot/kernel71.img  
-rwxr-xr-x 1 root root 7904106 Sep 21 14:16 /boot/kernel8.img  
-rwxr-xr-x 1 root root 6006608 Sep 21 14:16 /boot/kernel.img
```

If all of these actions fail to restore the SD card then it is very possible that the SD card is physically damaged, in which case it will be necessary to replace it and install a new OS and Radio software.

Building your own package

If you do modify the code, it may well be that you wish to create your own Raspbian package or create the **radiod** package directly from GitHub. There are several files and scripts required to build the **radiod** package. You need to first download the following files from GitHub.

See <https://github.com/bobrathbone/piradio6>

- **setup.sh** – From 7.2 onwards, script to set-up the build environment and execute **build.sh**.
- **build.sh** – Run this to actually build the package (called from **setup.sh**).
- **piradio** – The package definition file which define the executable and other required files.
- **piradio.preinst** – This is the script that runs before the package files are installed.
- **piradio.postinst** – This is the script that runs after the package files are installed.
- **piradio.postrm** – This is the script that runs if the package is removed to run clean-up tasks.

```
$ cd  
$ git clone https://github.com/bobrathbone/piradio6
```

From version 7.2 onwards run **setup.sh**. Do not use **sudo**. It only needs to be run once, thereafter it is only necessary to run **build.sh**.

```
$ cd piradio6  
$ ./setup.sh
```

If using a version older than 7.2 or building your own package then carry out the following instructions:

Install the build environment packages first:

```
$ sudo apt -y install equivs apt-file lintian
```

The Web interface also has its own build script namely **buildweb.sh** which is a much simpler example.

Study the **piradio** file in particular to glean how to build your own package. Make copies of the package build files and then modify these with your changes. Do not use the original files used to build the **radiod** package as these will be overwritten if the **radiod** package is updated.

To build the package (Example myradio):

```
$ cp -p build.sh mybuild.sh  
$ cp -p piradio myradio
```

Modify the mybuild.sh file to use your package files

```
PKGDEF=myradio  
PKG=myradiod
```

Modify the package name to match the PKGDEF definition and set the initial version in the **myradio** package definition file.

```
Package: myradiod  
Version: 1.0
```

Run the new build script as user **pi**. Do not use **sudo**.

```
$ ./mybuild.sh
```



Note: Most build warnings can be ignored but you should check if these can be easily corrected. Any errors should be corrected.

Start modifying the code with your changes regularly checking the build still runs OK. Good luck with your build.

Using PWM GPIOs for sound output on a Pi Zero

Another method of creating audio on a Pi Zero or Pi Zero W, is to use a GPIOs 13 and 18, PWM-0 (Left) and PWM-1 (Right) as shown in the following diagram. Note you will still need an amplifier for speakers.

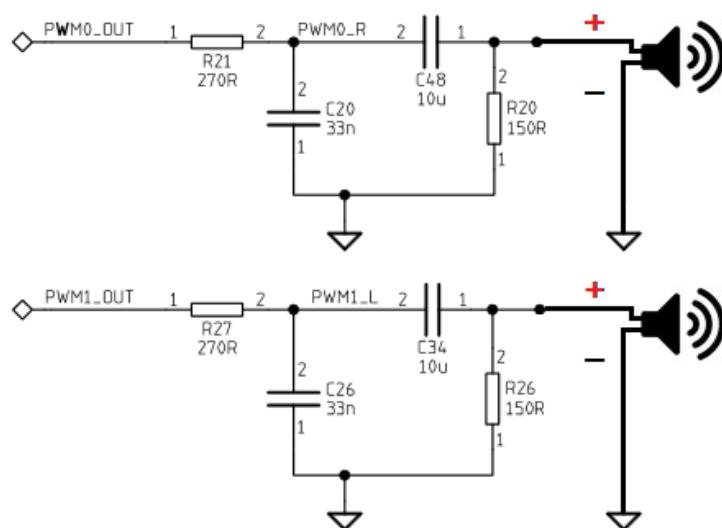


Figure 236 PWM Low band pass filters

This also requires loading the pwm-2chan Device Tree Overlay in **/boot/config.txt**.

```
dtoverlay=pwm-2chan,pin=18,func=2,pin2=13,func2=4
```

See <https://othermod.com/raspberry-pi-zero-audio-circuit/> for the full article how to do this.

Select the internal audio jack to enable audio output.

Fitting a Pimoroni On-Off Shim power switch

Unfortunately, the Raspberry Pi has a tendency to corrupt the SD card if the power is suddenly removed. This risk can be lessened by booting from a USB drive or stick. See *Booting from a USB drive or stick* on page 318. Alternatively, you can use one of the several solutions available on-line such as the Pimoroni power on-off shim.



Figure 237 Pimoroni On-Off shim power switch

The Pimoroni On-Off SHIM plugs into the first twelve GPIO pins of the Raspberry Pi GPIO header. There is a small button in the corner of the board next to USB power connector. This button is the on-off button and connects to GPIO17 (Physical pin 11). This button is also connected to the pins marked **BTN** to allow an external button to be fitted as shown in Figure 238 below.

To operate press the button once to switch on power and boot up the Raspberry Pi. When finished with the RPi simply press and hold the button for one second to initiate a clean shutdown and completely cut the power to the Raspberry Pi.

GPIO pin 17 when held low by the button initiates the standard shutdown command. Once this is completed GPIO 4 (Physical pin 7) is lowered by the software to force to completely cut power to the Raspberry Pi.

For more information on the Pimoroni On-Off shim see:
<https://shop.pimoroni.com/products/onoff-shim>

However, GPIO4 and 17 are used by the radio to connect the mute and menu switch respectively. This means that the conflicting LCD GPIOs need to be reassigned to different GPIOs, for example, GPIO26 and 27.



Figure 238 Raspberry Pi Radio with Pimoroni on-off shim

First edit **/etc/radiod.conf** and change the following definitions from:

```
menu_switch=17  
mute_switch=4
```

to

```
menu_switch=26  
mute_switch=27
```

Physically wire the menu switch to physical pin 37 and the mute switch to physical pin 13.

Pimoroni provide special software on their Web site to support the on-off shim. To install the Pimoroni software run:

```
$ curl https://get.pimoroni.com/onoffshim > onoffshim.sh  
$ sudo ./onoffshim.sh
```

This software must be installed for the Pimoroni on-off shim to work correctly.



Note that the Pimoroni on-off shim cannot be used with the Allo Piano 2.1 DAC with woofer (See *Allo DAC products* on page 61) as the Woofer channel also makes use of GPIO 4 and 17. There is no way around this. Fortunately, most DACs only use a single channel and do not use GPIO 4 and 17.

Adding a Real Time Clock (RTC)

When the Raspberry Pi is switched off for a while and switched on again it will initially have the system time when the RPi was switched off until the time synchronises with one of the Internet's time servers using the Network Time Protocol (NTP). This is because the RPi does not come with a Real Time Clock (RTC) as standard.



A Real Time Clock (RTC) can be added to a Raspberry Pi. There are several solutions available on-line. The illustration on the left shows the RasClock module available from PiHut and which connects to the RPi GPIO Header. The pins used are:

1. 3.3 Volts +
2. Not used (+5V)
3. SDA GPIO 2
4. Not Used (5V+)
5. SCL GPIO 3
6. GND 0V

Figure 239 RasClock RTC V4.2 module

SDA and SCL are the I2C Data and Clock respectively. The RasClock module needs to be fitted with a CR1220 36mA 3.0 Volt button battery (Bought separately). As the module uses the I2C interface it can be also connected to the above pins using jumper wires. Only pins 1,3,5 and 6 are required.

Installation

Plug the module into the first six pins of the Raspberry Pi. Power on the Raspberry Pi and add the following line to **/boot/config.txt** file after the [all] section.

```
dtoverlay=i2c-rtc,pcf2127
```

Reboot the Raspberry Pi. After reboot run the following command:

```
$ sudo hwclock -r  
2022-12-10 12:00:05.664465+00:00
```

This will display the current UTC time. To display local time run **sudo hwclock -l**

Licences, acknowledgements and support

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 cannot 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 or space 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 a 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 338.

Technical support

Technical support is on a voluntary basis by e-mail only at bob@bobrathbone.com. If there are any problems with this email address then also CC r.h.rathbone@gmail.com. Before asking for support, please first consult the troubleshooting section on page 240. 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, sound cards etc)?
- Which program and wiring 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 or Graphics screen?
- Did you run the test programs and what was the result?
- Switch on DEBUG logging as described on page 211, run the program and include the **/var/log/radiod/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.

Run the configuration display and send the **/usr/share/radio/config.log.tar.gz** that it produces to bob@bobrathbone.com. This will save a lot of questions about your configuration.

```
$ cd /usr/share/radio
$ ./display_config.sh
:
This configuration has been recorded in /usr/share/radio/config.log
A compressed tar file has been saved in /usr/share/radio/config.log.tar.gz
Send /usr/share/radio/config.log.tar.gz to bob@bobrathbone.com if required
```



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 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.

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/>

To Adafruit Industries for their excellent LCD plate and I2C code. See <http://www.adafruit.com>.

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-mdp-and-icecast2-on-raspberry-pi/>

To contributors such as Alan Broad who supplied photos of the Lego example of the radio plus code contribution.

To Mike Whittaker for his contribution on how to drive the USB speaker set. To other contributors such as Jon Jenkins 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 Arduino 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 <http://www.allaboutcircuits.com> for Figure 89 Soldering precautions.

Jim Downey from Mobile Alabama, the USA for his article on the backlight for Chinese 1602 I2C LCDs. See http://mbvmc.org/LCD_Backlight.pdf

Tomás González, Sevilla, Spain for his changes to *lcd_adafruit_class.py* (Previously *ada_lcd_class.py*) to switch on the Chinese 1602 I2C LCD backlight.

To the authors of the SGC Widget routines. Copyright (c) 2010-2012, Sam Bull and Michael Rochester

To ModMyPi (<https://www.modmypi.com>) for their excellent Pi products and setup guides.

Icons used in the graphic versions of the radio. Clipart library <http://clipart-library.com> and IconSeeker <http://www.iconseeker.com>

Thanks to **Olimex Limited** for their SSD1306 OLED routines which were used in the oled_class.py program and for their excellent technical support. See <http://www.olimex.com>. Original source <https://github.com/SelfDestroyer/pyMOD-OLED.git>

To **Midas Displays**, Great Yarmouth, UK, for their help and sponsorship for the implementation of Russian/Cyrillic OLED character displays. See <https://www.midasdisplays.com>

Thanks to Gordon Garrity at IQaudIO <http://iqaudio.co.uk> for his help and sponsorship to develop the radio to support the IQaudIO Cosmic Controller and SSD1306 OLED display.

Thanks to **Pimoroni** for their excellent Pirate radio using pHat BEAT software and hardware. See <http://pimoroni.com> for further information.

Thanks to **PiFace UK** for their **PiFace CAD** product. This makes a very easy entry level radio using this software. See <http://www.piface.org.uk/>

Franz-Josef Haffner, from Germany, for his conversion of a Schneider Frères Rondo vintage radio.

To Andrey Gunich (Андрей Гунич) from the Ukraine for his help with the development and testing of Russian/Cyrillic OLED character displays.

To **othermod.com** for their article on using PWM and Low Band Pass filters to produce an audio circuit for the Raspberry Pi Zero (W).

Thanks to **Jim Smith** from Shropshire for his work on optical rotary encoders.

To Syds Post for his work on conversion of the IR Remote Control software from Python2 to Python3 on the Raspberry Pi Bullseye OS. See <https://www.sydspost.nl> (Dutch) in particular his article on the Vintage Internet Radio (Use browser translation facility for English translation).

David Steele for his **comitup** package for configuring Wi-Fi access via a Web Interface. See <https://davesteele.github.io/comitup>

The **ETechnoG Team** <https://www.etechnog.com> for their article on supply voltages.

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.

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 BE LIABLE 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.

Glossary

AC	Alternating Current - In this context 110V or 220V
AP	Application processor (Also see CPU)
API	Application Programming Interface
DC	Direct Current - In this context +5V or +3.3V
A2DP	Advanced Audio Distribution Profile - Bluetooth
AAC	Advanced Audio Coding
ALSA	Advanced Linux Sound Architecture
ARM	Advanced RISC Machine (Type of processor used in Mobile phones and Raspberry Pi)
ASX	Advanced Stream Redirector
ATC	Air Traffic Control
BJT	Bipolar Junction Transistor
BOT	USB Mass Storage Bulk-Only Transport (BOT) – Legacy USB device access (Also see UAS)
CGI	Common Gate Interface – Executable Server-Side scripts
CAD	Control and Display (PiFace) – No longer supported in this version
CD	Compact disc - In this context CD marker pen
CIFS	Common Internet File System
CODECS	Encoder/Decoder for media streams
CPU	Central Processor Unit
DAC	Digital to Analogue Converter (Digital to audio frequency analogue in this case)
DOS	Denial of Service – Attack software aimed at taking down an Internet service (Web etc.)
DOS	Disk Operating System (Microsoft Operating System for PCs)
DDOS	Distributed Denial of Service – DOS attack from hundreds or thousands of computers
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System. Converts a URL such as google.com to an IP address or addresses
DSP	Digital Signal Processing/processor (In this context it is mixer control)
DT	Device Tree (Overlay). Device (Sound cards) configuration in /boot/config.txt in Raspbian

EMI	Electromagnetic Interference (For example fluorescent lighting etc.)
EEPROM	Electrically Erasable Programable Read Only Memory (Typically a Boot PROM)
FBCP	Frame Buffer CoPy – HDMI display mirroring driver
FET	Field Effect Transistor
FLIRC	A USB device and software which maps an IR remote control to the keyboard
HCI	Host Controller Interface. Bluetooth standard for operation with three-wire UART
HLS	HTTP Live Streaming. Media streaming protocol for visual and audio media delivery
HDMI	High-Definition Multimedia Interface for audio and video plus Ethernet interface.
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)
IATA	International Air Transport Association
ICAO	International Civil Aviation Organization
ID3	Metadata standard for MP3 files which provides title, artist, album, track number, and other information embedded in the currently playing stream
IIC	Alternative name used by some manufacturers for I2C
IP	Internet Protocol
IPS	In-Plane Switching, a type of LED (a form of LCD) display panel technology.
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IR	Infra-Red (sensor) for use with infra-red devices such as remote controls
LCD	Liquid Crystal Display, also see OLED
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)
Micro HDMI	Miniaturized version of the High-Definition Multimedia Interface specification
MIPI	Mobile Industry Processor Interface (Used for camera and display interfaces)

MMS	Microsoft Media Server Internet protocol
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MPC	Command line client for MPD
MPD	Music Player Daemon
MPEG	Moving Picture Experts Group
MPEG3	Music encoding standard from MPEG
NAS	Network Attached Storage
NFS	Network File System
NTP	Network Time Protocol
OLED	Organic Light Emitting diode. Also, OLED character displays gradually replacing LCDs
PA	Personal Amplifier (Input to audio stage of a vintage radio)
PIL	Python Image Library. Used by Python to manipulate images and colours
OS	Operating system (Raspbian Buster in this case)
PC	Personal Computer
PCM	Pulse-Code Modulation is a method used to digitally represent sampled analogue signals
PDF	Portable Document Format
PWM	Pulse Width Modulation, is an alternative method for audio circuit design.
PHP	A server-side scripting language designed primarily for Web development
PID	Process ID
PIL	Python Imaging Library – Linux graphics routines – Pillow is a branch of PIL
PLS	MPEG Playlist File (as used by Winamp)
RISC	Reduced Instruction Set Computer – Also see ARM
RPi	Raspberry Pi
RSS	Really Simple Syndication – Web feed usually containing news items
RTC	Real Time Clock – Usually a battery backed up clock chip updating system time
SD	San Disk Memory Card commonly found in cameras and Smartphone's
SPI	Serial Peripheral Interface (Motorola) used by PiFace CAD

S/P-DIF Sony/Philips Digital Interface for short distance audio transmissions.

SCO Synchronous Connection Oriented link – Bluetooth

SDK Software development kit

SSH Secure Shell – Encrypted terminal

SSID Service Set Identifier. An SSID is the public name of a wireless network

SSD Solid State Disk Drive

SVI Standard Volume Indicator. Another name for VU meter/indicator

System V Particular version of UNIX, many features of which have found their way into Linux

TCP/IP The common name for network protocols used by the Internet and computer networks.

TFT Thin Film Transistor – Used in display technology and touch-screens

TTS Text-To-Speech (eSpeak in this case)

TV Television (In this case, with one or more HDMI inputs)

UDP Universal Datagram Protocol. A connectionless network protocol over IP

UART Universal Asynchronous Receiver-transmitter for asynchronous serial communication

UAS USB Attached SCSI protocol – Newer faster USB drive access protocol (Also see BOT)

URL Universal Resource Locator (A link to a Web page for example)

USB Universal Serial Bus

USB 2.0 USB with a maximum signalling rate of 480 Mbit/s (60 MB/s)

USB 3.x USB using full duplex communication at speeds up to 5 Gbit/s (625 MByte/s)

USB-C 24-pin USB connector system which can be inserted either way

USB OTG USB On-The-Go, software to support USB devices (Not supported with this radio)

VCC Common Collector Voltage; the positive supply voltage to a device

VDD Voltage Drain Drain Supply; the positive supply voltage to a device

VEE Voltage Emitter. the negative power supply voltage to a device

VLC Media player used by Pimoroni software (Not used by this radio software)

VSS Voltage Source Supply; the negative supply voltage to a device

VU Volume Unit – Volume meter/indicator also known as SVI (Standard Volume Indicator)

- WEP Wired Equivalent Privacy (WEP) is a security algorithm considered less secure than WPA
- WI-FI Wireless Network typically using the 802.11 Wireless Network protocol
- WPA Wi-Fi Protected Access (WPA) – Also see WPA2
- WPA2 Wi-Fi Protected Access version II, an enhanced, more secure version of WPA.
- XML Extensible Mark-up Language. A Web technology used for transmitting data structures

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. It is mainly configured by the `configure_radio.sh` program.

```
# Raspberry Pi Internet Radio Configuration File
# $Id: radiod.conf,v 1.34 2021/09/27 08:23:29 bob Exp $

# Configuration file for version 7.0 onwards
# Used for both 40 and 26-pin Raspberry Pi versions
#
# Please Note: Configuration of this file, for the most part, is done by
# running the configure_radio.sh program.

[RADIOD]

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

# Logfile creation mode, either truncate or tail
log_creation_mode=truncate

# Startup option either RADIO,MEDIA or LAST a playlist name
#startup=RADIO
startup=_Radio

# MPD client timeout from 2 to 15 seconds default 10
client_timeout=10

# Codecs list for media playlist creation (Run 'mpd -V' to display others)
CODECS="mp3 ogg flac wav wma"

# 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=10

# Volume display text or blocks
volume_display=blocks

# MPD port number (Do not change unless MPD reconfigured)
mpdport=6600

# Remote control UDP server listen host either 0.0.0.0 (All interfaces) or
localhost
remote_control_host=localhost

# Remote control communication host and port Default localhost 5100
remote_control_port=5100

# This allows another host to send UDP messages to the UDP server
# It is either localhost or the IP address of the remote server
remote_listen_host=localhost

# Audio output device - Must match an output using the "aplay -l" command
# The configure_audio.sh program will set this to headphones(default), HDMI,
# DAC or USB
# depending upon the audio device/card selection. You can override this
# setting
```

```

# with your own unique string from the aplay command, for example
"HiFiBerry"
audio_out="headphones"

# Audio config lock stops audio configuration from being dynamically
# changed if HDMI plugged in or out. Default no
audio_config_locked=no

# 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)
# Use GPIO 16 (pin 36) for designs using IQAUDIO DAC cards etc.
# Use GPIO 14 (pin 8) for designs using IQAUDIO Cosmic controller
# remote_led=0 is no output LED
remote_led=0

# Display playlist number in brackets yes or no
display_playlist_number=no

# 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

# Status LED (Typically for vintage radio) Normally 27,22,23 respectively
rgb_red=0
rgb_green=0
rgb_blue=0

# Menu rotary switch (optional) Normal values are 24,8 and 7 respectively.
Value 0 disables
menu_switch_value_1=0
menu_switch_value_2=0
menu_switch_value_4=0

# The i2c_address=0x3C
# Some backpacks use other addresses such as 0x2F, then set i2c_address=0x3C
i2c_address=0x3C

# I2C normaly uses bus 1 on the I2C interface. However the very first
Raspberry
# used bus 0. If you are using a very old Pi then set i2c_bus=0
# Run ./display_model.py to see what model Pi you are running
i2c_bus=1

# Set LCD character translation on or off. Graphic and OLED versions
unaffected
# as they do not need language translation and use system fonts
translate_lcd=on

# Language font translation table to be used.
# Current choices are English(Default), European(Western) and Russian
# Translation tables are contained in the /usr/share/radio/codes directory
# Add other translation tables to the above directory
language=English

# Set LCD/OLED controller being used. HD44780U (default) or HD44780 (Older
LCDs)
controller=HD44780U

# Select LCD code page table 0,1,2 or 3. Default 0

```

```

# 0 = Use codepage parameter specified in primary font file (Selected by
language)
# 1, 2 or 3 Override codepage setting in the primary font file
codepage=0

# Romanize characters (eg convert Cyrillic to Latin characters),
# Set to on or off. Default is on
romanize=on

# Speech for visually impaired or blind listeners, yes or no
# Needs espeak package - sudo apt install espeak
speech=no
# Speech volume as a percentage of the normal MPD volume
speech_volume=75
#Verbose - yes = each station change is spoken
verbose=no
# Speak hostname and IP address
speak_info=no
# Icecast2 streaming yes/no
streaming_on=no

# Set the user interface to 'buttons' or 'rotary_encoder' or 'graphical'
# These can also be used in conjunction with a graphical/touchscreen display
user_interface=rotary_encoder

# Switch settings for Rotary encoders or buttons
menu_switch=17
mute_switch=27
up_switch=0
down_switch=0
left_switch=0
right_switch=0

# Pull GPIO up/down internal resistors (Applies to button interface only).
# Default:down
pull_up_down=up

# Display types
# NO_DISPLAY = No display connected
# LCD = directly connected LCD via GPIO pins
# LCD_I2C_PCF8574 = Arduino (PCF8574) I2C backpack
# LCD_I2C_ADAFRUIT = Adafruit I2C backpack
# LCD_ADAFRUIT_RGB = LCD I2C RGB plate with buttons
# GRAPHICAL = Graphical or touch screen display
# OLED_128x64 = 128x64 pixel OLED
# PIFACE_CAD = PiFace CAD with six push buttons using the SPI interface
# ST7789TFT = Pimoroni Pirate audio with four push buttons using the SPI
interface
# SSD1306 = Sitronix SSD1306 controller for the 128x64 pixel OLED
# LUMA = Luma driver for SSD1306, SSD1309, SSD1325, SSD1331, SH1106, WS0010
#     For example LUMA.SH1106 for OLEDs using the sh1106 chip
# LCD_I2C_JHD1313 = Grove RGB 2x16 LCD (AIP31068L controller)
display_type=LUMA.SH1106

# Display width, 0 use program default. Usual settings 16 or 20
display_width=16
display_lines=4

# LCD GPIO connections for 40 pin version of the radio
lcd_select=0
lcd_enable=0
lcd_data4=0
lcd_data5=0
lcd_data6=0
lcd_data7=0

# Display Scroll speed 0.01 to 0.6 seconds

```

```

scroll_speed=0.2

# Settings are standard, alternative, rgb_rotary rgb_i2c_rotary
# Some rotary switches do not work well with the standard rotary class
# Set to "alternative" to use the alternative rotary encoder class
# For rotary encoders with RGB LEDs in the shaft set to rgb_rotary
# For rotary I2C encoders with RGB LEDs in the shaft set to rgb_i2c_rotary
rotary_class=rgb_i2c_rotary

# KY-040 encoders etc have their own physical 10K pull-up resistors and do
not
# need the internal gpio pull-up resistors.
# otherwise set rotary_gpio_pullup=none otherwise set to "up"
rotary_gpio_pullup=up

# Station names source, list or stream
station_names=list

# Action on exiting radio. Stop radio only or shutdown the system
# exit_action=stop_radio
exit_action=shutdown

# Bluetooth device ID - Replace with the ID of your bluetooth
speakers/headphones
# Example: bluetooth_device=00:75:58:41:B1:25
# Use the following command to display paired devices
# bluetoothctl paired-devices
bluetooth_device=00:00:00:00:00:00

# Action when muting MPD. Options: pause(Stream continues but not processed)
or stop(stream is stopped)
# mute_action=stop
mute_action=pause

# Shoutcast ID
shoutcast_key=anCLSEDQODrElkxl

# Internet check URL. This must be a reliable URL and port number
# which can be contacted such as google.com
# The port number is normally 80 (HTTP).
# The internet_timeout in seconds
# Disable by removing the URL from internet_check_url= parameter
internet_check_url=google.com
internet_check_port=80
internet_timeout=10

# OLED parameters
# Flip display vertically (yes or no) OLED only at present
flip_display_vertically=no

# Splash screen
splash=images/raspberrypi.png

# Allow updating of playlists by external clients yes/no (Experimental)
update_playlists=no

# I2C addresses for RGB I2C Rotary Encoders
volume_rgb_i2c=0x0F
channel_rgb_i2c=0x1F

# Graphics (touch screen) screen settings
[SCREEN]
# Size is in pixels. Supported is 800x480 (7" screen) or 720x480(3.5"
screen)
# or 480x320 (2.8" or 3.5" screen) or 1024x600 (Maximum)
# Also see framebuffer_width and framebuffer_height parameters in
/boot/config.txt

```

```

screen_size=800x480
fullscreen=yes

# Screen save time in minutes, 0 is no screen saver
screen_saver=0

# Title %V = version %H = hostname
window_title=Bob Rathbone Internet Radio Version %V - %H

# Colours - See https://htmlcolorcodes.com/color-names/
window_color=turquoise
banner_color=white
labels_color=white
display_window_color=lightblue
display_window_labels_color=black
slider_color=mediumseagreen
display_mouse=yes

# Wallpaper backgrounds. See /usr/share/rpd-wallpaper/
# More backgrounds in /usr/share/scratch/Media/Backgrounds (Install scratch)
wallpaper=/usr/share/rpd-wallpaper/aurora.jpg

# Set date format for graphic screen
dateformat=%H:%M:%S %A %e %B %Y

# Allow switching between vgradio and gradio
switch_programs=yes

# Display shutdown button. The action of this button is dependent
# upon the exit_action parameter setting (shutdown or stop radio
display_shutdown_button=yes

# The following is specific to the vintage graphical radio
scale_labels_color=white
stations_per_page=40
display_date=yes
display_title=yes

[AIRPLAY]

# Airplay activation yes or no
Airplay=no

# Mixer preset volume for radio and media player if using sound card
# Set to 0 if using onboard audio or USB sound dongle.
# If using a sound card set to 100% initially and adjust as neccessary
# Old name was mixer_volume
mixer_preset=100

```

/etc/logrotate.d/radiod

This file causes the **/var/log/radiod/radio.log** to be rotated so that it doesn't continue to grow and fill the disk.

```
/var/log/radiod/radio.log {
    weekly
    missingok
    rotate 4
    compress
    notifempty
    maxsize 150000
    copytruncate
    create 600
}
```

Old log files are compressed and renamed, for example **/var/log/radiod/radio.log.1.gz**.

/lib/systemd/system/radiod.service

This file is part of the new **systemd** startup services. This file defines the so-called radiod service. It defines how the service will start-up and what services it needs.

```
# Radio systemd script
:
[Unit]
Description=Radio daemon
After=network.target

[Service]
Type=simple
ExecStart=/usr/share/radio/radiod.py nodaemon

[Install]
WantedBy=multi-user.target
```

/usr/lib/systemd/system/irradiod.service

The **irradiod.service** service definition file is part of the new **systemd** startup services. This file defines the so-called **irradiod** service which starts and stops the IR Remote Control software.

There are two versions; one for Buster and the other for Bullseye.

During installation the correct version is copied to the **/usr/lib/systemd/system** directory.
The Buster version starts the **remote_control.py** (Python2) version. The Bullseye version starts the **irradiod.py** (Python3) version.

For Buster

```
# Radio remote control systemd script
# $Id: irradiod.service.buster,v 1.1 2022/02/05 10:44:32 bob Exp $

# This service is for Raspberry OS Buster. See irradiod.service.bullseye for
Bullseye version
# It uses remote_control.py and rc_daemon.py both written Python 2
# This file is copied to /usr/lib/systemd/system/irradiod.service
# by the configure_ir_remote.sh script if the RPi OS is Buster

[Unit]
Description=Radio remote control daemon
```

```

After=network.target

[Service]
Type=simple
ExecStart=/usr/share/radio/remote_control.py nodaemon
ExecStop=/usr/share/radio/remote_control.py stop

[Install]
WantedBy=multi-user.target

```

For Bullseye

```

# Radio remote control systemd script
# $Id: irradiod.service.bullseye,v 1.1 2022/02/05 10:44:32 bob Exp $

# This service is for Raspberry OS Bullseye. See irradiod.service.buster for
Buster version
# It uses irradiod.py and ir_daemon.py (Daemon)
# This file is copied to /usr/lib/systemd/system/irradiod.service
# by the configure_ir_remote.sh script if the RPi OS is Bullseye or later

[Unit]
Description=Radio remote control daemon
After=network.target

[Service]
Type=simple
ExecStart=/usr/share/radio/irradiod.py nodaemon
ExecStop=/usr/share/radio/irradiod.py stop

[Install]
WantedBy=multi-user.target

```

/etc/asound.conf

The source of this file is one of the files found in **the /usr/share/radio/asound directory**. Which file will be copied depends upon the required configuration for the sound device being used

```

.# Set default mixer controls
ctl.!default {
    type hw
    card 0
}

# Set default PCM device
pcm.!default {
    type plug
    slave {
        pcm "plughw:0,0"
        format S32_LE
    }
}

```

The above example is for the on-board audio jack.

Cronjob scripts

The /etc/cron.weekly/radiod script

This script runs once each week and creates playlist files and copies them to the **/var/lib/mpd/playlists** from the stations defined in the **/var/lib/radiod/stationlist** file.

```
#!/bin/sh
:
PATH=/sbin:/usr/sbin:/usr/local/sbin:/bin:/usr/local/bin:/usr/bin

DIR=/usr/share/radio
LOGDIR=${DIR}/logs
LOG=${LOGDIR}/stations.log

mkdir -p ${LOGDIR}
chown pi:pi ${LOGDIR}
sudo ${DIR}/create_stations.py --no_delete 2>&1 >${LOG}
chown pi:pi ${LOG}
```

The /etc/cron.daily/radiod script

Since version 7.3 MPD playlists can be amended by any external MPD client capable of doing so. However, this makes the the **/var/lib/radiod/stationlist** file out-of-date. This daily script updates the **stationlist** file with all the current entries.

```
PATH=/sbin:/usr/sbin:/usr/local/sbin:/bin:/usr/local/bin:/usr/bin

DIR=/usr/share/radio
LOGDIR=${DIR}/logs
LOG=${LOGDIR}/update_stationlist.log
CONFIG=/etc/radiod.conf

mkdir -p ${LOGDIR}
chown pi:pi ${LOGDIR}
grep "update_playlists=yes" ${CONFIG} 2>&1 >${LOG}
if [ $? -eq 0 ]; then
    sudo ${DIR}/update_stationlist.py --update 2>&1 >${LOG}
    sudo ${DIR}/create_stations.py --no_delete --force 2>&1 >${LOG}
fi
chown pi:pi ${LOG}
```



Note: These files require the **anacron** package to be installed to run the above scripts regularly.

/usr/lib/systemd/system/irradiod.service

This is the service start stop script for the remote control daemon. This starts and stops the **/usr/share/radio/remote_control.py** program which handles the remote control for the IR interface.

/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/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 **gpio-ir** dtoverlay. For example:

```
dtoverlay= gpio-ir, gpio_pin=25
```

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
```

The **configure_audio.sh** script configures the audio device to be used in **/boot/config.txt** and sets up the correct **/etc/asound** audio configuration. In the case of DAC configuration, it disables the on-board by changing the **dtparam=audio** parameter from:

```
dtparam=audio=on
```

To:

```
dtparam=audio=off
```

A.3 X-Windows radio desktop files

The lxsession autostart file for the desktop/touchscreen radio

The **/home/pi/.config/lxsession/LXDE-pi/autostart** file is modified to start either **gradio.py** or **vgradio.py** during the desktop start-up if so configured.

```
@lxpanel --profile LXDE-pi  
@pcmanfm --desktop --profile LXDE-pi  
@xscreensaver -no-splash  
@point-rpi  
@sudo /usr/share/radio/gradio.py
```

Desktop radio icon files

The configuration files for the two desktop Icons for graphic versions of the radio are copied into the **/home/pi/Desktop** directory. This displays two radio icons on the X-windows desktop. Clicking either of these starts the appropriate desktop version of the radio.

The Desktop file **gradio.desktop**

```
[Desktop Entry]
Name=Radio
Comment=Internet radio
Icon=/usr/share/radio/images/radio.png
Exec=sudo /usr/share/radio/gradio.py
Type=Application
Encoding=UTF-8
Terminal=false
Categories=None;
```

The Desktop file **vgradio.desktop**

```
[Desktop Entry]
Name=Vintage Radio
Comment=Vintage Internet radio
Icon=/usr/share/radio/images/Vintage.png
Exec=sudo /usr/share/radio/vgradio.py
Type=Application
Encoding=UTF-8
Terminal=false
Categories=None;
```

Appendix B – Technical specification and other notes

B.1 – Technical specification

The specification of the Raspberry Pi internet radio is:

- The software runs on **Raspbian Bullseye** (Debian 11) on all Raspberry Pi's except version 1
- 32-bit and 64-bit software available
- Any sudo enable user can install the software and not just user 'pi'
- Uses the standard Music Player Daemon (MPD)
- The following displays are supported:
 - Raspberry Pi 7-inch touch screen or HDMI screen
 - Most 2.8 and 3.5-inch touch screens
 - 2x16, 2x8, 4x16 or 4x20-character LCD
 - Adafruit LCD plate with 5 push buttons (I2C interface)
 - Adafruit 3.5-inch TFT touch-screen
 - A 128 by 64-pixel OLED display
 - PiFace CAD with 2x16 LCD (Version 1.12 only)
 - Pimoroni Pirate Radio with six buttons and a 3W speaker
 - Pimoroni Pirate Audio radio with a 1W mini-speaker and four push buttons.
 - OLEDs using the Sitronix SSD1306 I2C controller
 - OLEDs supported by the LUMA driver (SH1106, SSD1306 etc)
 - Grove 2x16 LCD with RGB backlight (AIP31068L LCD controller)
- The LCD can be directly interfaced via GPIO pins or using an I2C backpack interface
- Optional IR sensor and remote control using LIRC or FLIRC
- Clock display or IP address display
- Five configurable user interfaces are available:
 - Five push button operation (Menu, Volume Up, Down, Channel Up, Down)
 - As alternative to the above, rotary encoders may be used
 - Touchscreen with Mouse/Keyboard interface
 - IQaudIO Cosmic controller with three push buttons and rotary encoder
 - Pimoroni Pirate radio using pHat BEAT sound card and VU indicator
 - PiFace CAD with six push-buttons (Only five are used). Needs version 6.5.
- Support for Russian/Cyrillic, West European, English (Depending upon LCD capabilities)
- Support for Digital sound cards such as **HiFiBerry**, **IQaudIO**, **JustBoom** or **Pimoroni pHat/BEAT**
- Support for Bluetooth speaker or headphones using BlueAlsa software
- Vintage radio conversion to Internet radio supported
- Timer (Snooze) and Alarm functions (Not touch screen version)
- 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
- Control the radio from either an Android device or **iPhone** and **iPad**
- Plays Radio streams or MP3 and WMA tracks
- Can function as a **Spotify** receiver (Needs a premium Spotify account)
- Optional support for **Apple Airplay** speaker using **shairport-sync**.
- Play output on PC or on a mobile device using ICECAST streaming

- Playlist creation program using a list of URLs (M3U file)
- Playlist creation from the Shoutcast database via command line or Web interface
- Add and delete stations/tracks via any external MPD client capable of doing so
- Fully integrated with mobile apps such as Android **MPDdroid** or Apple **mPod**
- Speech for visually impaired and blind persons using **espeak**
- Support for Russian/Cyrillic and European character sets (Depending upon LCD capabilities)
- Music Player Daemon supports the **ID3** metadata standard. See <https://en.wikipedia.org/wiki/ID3> This provides meta-data such as the title, artist, album, track number, bit-rate, and other information.



Please note that this is not a consumer product. No claims are made to suitability for all users. It is solely intended as a fun construction project. Please also see **Disclaimer** on page 338.

B.2 -Elecrow 7-inch touch-screen notes

Below are some notes on how to set up the Elecrow 7-inch TFT Capacitive touch screen display, with 1024x600 Resolution.



Please note, much of this information was supplied by a third-party and has not been tested by the author. Therefore, any support by the author is limited.

Add the following to **/boot/config.txt** file.

```
hdmi_force_hotplug=1
max_usb_current=1
hdmi_group=2
hdmi_mode=1
hdmi_mode=87
hdmi_cvt 1024 600 60 6 0 0 0
hdmi_drive=1
```

If when running in full screen mode (**fullscreen=yes** in **/etc/radiod.conf**) you see a small window surrounded by a thick black border then in **/boot/config.txt** file amend the following:

```
framebuffer_width=1280
framebuffer_height=720
```

To

```
framebuffer_width=800
framebuffer_height=480
```

Also set the **screen_size** parameter in **/etc/radiod.conf**.

```
screen_size=800x480
```

Both files must be edited using sudo. For example:

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

Further information on the Elecrow touch-screen can be found at:

https://www.elecrow.com/wiki/index.php?title=7_Inch_1024*600_HDMI_LCD_Display_with_Touch_Screen

B.3 Sound card DT Overlays

The following table contains the known Device Tree (DT) overlays for various sound cards. The third column contains the DT overlay statement that needs to be added to the **/boot/config.txt** configuration file. This is either done by running the **configure_audio.sh** program or by directly editing **/boot/config.txt**. Some of the DACs require the **pulseaudio** package to be installed.

Table 30 Sound card Device Tree overlays

Manufacturer	Sound Card	DT Overlay and dtparam	Pulse
Adafruit	3W Stereo Amplifier Bonnet	dtoverlay=hifiberry-dac	Yes
Allo BOSS	384 kHz/32bit DAC PCM5122	dtoverlay=allo-boss-dac-pcm512x-audio	No
HiFiBerry	DAC+ Light/DAC Zero/MiniAmp	dtoverlay=hifiberry-dac dtparam=i2s=on	No
HiFiBerry	DAC+ standard/pro	dtoverlay=hifiberry-dacplus	No
HiFiBerry	Digi/Digi+ all models	dtoverlay=hifiberry-digi	No
HiFiBerry	Amp+	dtoverlay=hifiberry-amp	No
IQaudio	Pi-DAC+	dtoverlay=iqaudio-dacplus	
IQaudio	Pi-DigiAMP+	dtoverlay=iqaudio-dacplus,unmute_amp dtoverlay=iqaudio-dacplus,auto_mute_amp	No
IQaudio	Pi-DACZero	dtoverlay=iqaudio-dacplus	No
IQaudio	Pi-DAC PRO	dtoverlay=iqaudio-dacplus	No
IQaudio	Pi-Digi+	dtoverlay=iqaudio-digi-wm8804-audio	No
JustBoom	Amp, Amp Zero, DAC and DAC Zero	dtoverlay=justboom-dac	No
JustBoom	Digi and Digi Zero	dtoverlay=justboom-digi	No
Pimoroni	pHat Beat or Pirate Audio	dtoverlay=hifiberry-dac dtparam=i2s=on	Yes
Various	PCM5102A or PCM5100A based	dtoverlay=hifiberry-dac dtparam=i2s=on	No

In all cases, disable the on-board sound system by modifying the **dtparam=audio=on** parameter in the **/boot/config.txt** configuration file to off, or by commenting it out.

```
dtparam=audio=off
```

Or

```
#dtparam=audio=on
```

Configuring other audio devices

For other audio devices, the DT overlays can be found in the **/boot/overlays** directory. See the **/boot/overlays/README** file. For example, to enable the Cirrus WM5102 you would add the following line to the end of the **/boot/config.txt** configuration file:

```
dtoverlay=rpi-cirrus-wm5102
```

B.4 UDP messages

This section is only of interest to developers wishing to interface with the radio program. These are messages (events) sent to the UDP listener in the `radio_class.py` program. These are sent from the IR remote control program and from the Shoutcast program & Web interface.

Table 31 UDP messages

Message	Source	Description
MUTE_BUTTON_DOWN	Remote control	Mute button held down
KEY_VOLUMEUP	Remote control	Not used, same as KEY_RIGHT
KEY_RIGHT	Remote control	Volume up or menu up function
KEY_VOLUMEDOWN	Remote control	Not used, same as KEY_LEFT
KEY_LEFT	Remote control	Volume down or menu down function
LEFT_SWITCH	Radio class	Left switch pressed
RIGHT_SWITCH	Radio class	Right switch pressed
KEY_CHANNELUP	Remote control	Not used, same as KEY_UP
KEY_UP	Remote control	Channel up or menu up function
KEY_CHANNELDOWN	Remote control	Not used, same as KEY_DOWN
KEY_DOWN	Remote control	Channel down or menu down function
KEY_MENU	Remote control	Menu function on remote control pressed
KEY_OK	Remote control	OK key on remote pressed
KEY_LANGUAGE	Remote control	Toggle speech facility on/off
KEY_INFO	Remote control	Toggle info on/off
PLAY_<n>	Remote control	Play station or track <n> e.g. PLAY_23
MEDIA	select_source.cgi script	Cycle through Media playlists
RADIO	select_source.cgi script	Cycle through Radio playlists
AIRPLAY	select_source.cgi script	Select Airplay as source
SPOTIFY	select_source.cgi script	Select Spotify as source
INTERRUPT	select_source.cgi script	Not used (Test only)
RELOAD_PLAYLISTS	shoutcast.cgi script	Reload (new) playlists

B.5 Cyrillic/European character LCDs/OLEDS

It is possible to purchase LCDs with a Russian/Cyrillic or other languages including Western European character ROMs.



Figure 240 Russian/Cyrillic character LCD

The radio program can display the Russian language either in **Cyrillic** or **Romanized** (convert to Latin) characters. For example, **Радио Пятница** when Romanized becomes **Radio Pyatnica**.

Romanization of Russian characters

For devices that do not currently support Russian/Cyrillic characters, it is possible to Romanize (convert to Latin characters) Russian text as shown in the following example.

This Romanization is achieved by setting the **romanize** parameter in **/etc/radiod.conf** to **on** which is the default.

```
romanize=on
```

For example, the following Russian text:

```
Низкая цена на нефть все больше давит на рубль
```

Converts to:

```
Nizkaja cena na neft' vse bol'she davit na rubl'
```

Translation (just out of interest):

```
Low oil price puts more and more pressure on the ruble
```

For more information on Romanization of Russian characters see:

https://en.wikipedia.org/wiki/Romanization_of_Russian

Displaying Russian/Cyrillic or European characters

To display the Russian language in native Cyrillic alphabet the following are required:

1. An LCD capable of displaying Russian Cyrillic or Western European characters
2. The **romanize** parameter must be set to **off**
3. The correct code page number in the LCD controller must be selected
4. The correct LCD **controller** must be selected.
5. The **translate_lcd** parameter must be set on unless using OLEDs.

The language in this example is selected by setting the language parameter in **/etc/radiod.conf** to the required language. There are currently only three choices. English (Default), European (Western European) and Russian. In this example Russian has been chosen

```
# Language font translation table to be used.  
# Current choices are English(Default), European(Western) and Russian  
# Translation tables are contained in the /usr/share/radio/codes directory  
# Add other translation tables to the above directory  
language=Russian
```

Font code page selection is achieved by setting the **codepage** parameter in **/etc/radiod.conf**

```
# Select LCD code page 0,1,2 or 3. Default 0  
# 0 = Use codepage parameter specified in primary font file  
# 1, 2 or 3 Override setting in font file  
codepage=0
```

Setting codepage to 0 tells the radio program to lookup the default code page setting found in the selected code page translation file in the codes sub-directory. Otherwise, this can be overridden by setting the codepage specifically as shown in the following table.

codepage	LCD code page	Language characters
0		Select code page from translation file
1	0x0	Japanese & English
2	0x1	Western European & English
3	0x2	Russian & English

Table 32 Character font table selection

All LCDs have one or more-character tables in ROM which provide selection of the correct character font for the character to be displayed. There are usually two code/font ROM pages often known as A0 and A2. However, MC0100 controller devices, for example, may have three. Each table can support only 256 characters, so to support say different language character sets a table must be provided for each one. For example, Midas supply an LCD which supports three languages plus English in each case. These code charts will be found in the controller specification (available from the manufacturer) for the character LCD/OLED device.

In the above table, another manufacturer might use table 0x0 for Russian. This can only be established by looking at the specification for your device.

The final setting for the correct language display is the **controller** parameter in **/etc/radiod.conf**.

```
# Set LCD/OLED controller being used. HD44780U (default) or HD44780 (Older  
LCDs)  
controller=HD44780U
```

Along with the language parameter the controller parameter selects the code translation files to be used.

The codes sub-directory contains the character to code page PROM translation tables. There are two code for each file. These are called Romanized (Convert to Latin characters) and codes (Native characters).

The actual tables to be selected are dependent upon the **language** and **controller** parameters.

Table 33 Code page translation files

Controller	English	Russian	European
HD44780U	European.py Russian.py English.py	Russian.py (codes) European.py (romanized) English.py	European.py (codes) Russian.py (romanized) English.py
HD44780	European_HD44780.py Russian_HD44780.py English_HD44780.py	Russian_HD44780.py (native) European_HD44780.py (romanized) English_HD44780.py	European_HD44780.py (native) Russian_HD44780s.py (romanized) English_HD44780.py

You will notice that **English.py** is always loaded and is last. This means that any codes missed in the other translation files will be caught in **English.py**. The file which relates to the language selection is always the first file and its native codes will be used to display text. The above table can be extended in the future for other languages. The exception is English as all codes will be Romanized.

The **HTMLCodes.py** file found in the **codes** sub-directory is used by the RSS news feeds to translate/strip HTML tags and entities from the RSS feed (in XML). There is also a **README** file in the **codes** sub-directory.



The **translate_lcd** parameter must also be set to on for Romanization or Cyrillic translation routines to work unless using an OLED display.



The author is not a Russian speaker so testing Russian/Cyrillic character LCDs has relied heavily on Russian and Baltic region constructors to test this and is only able to provide limited support on these types of devices.

Below are the settings for the type of display being used:

Table 34 Russian Cyrillic and Romanization display configurations

Type of Display	translate_lcd	romanize_russian
Non-Russian/Cyrillic Latin LCD	on	on
Russian/Cyrillic LCD	off	off
OLED and HDMI displays	n/a	n/a
Raspberry Pi Touch screen	off	off
HDMI Display	off	off

Character Translation routines

The following is only of interest if you wish to modify the existing translation tables or create your own. The original ASCII code chart for English user only had 256 characters using a single byte. All computers could only use the limited character set at the time and Romanization of character sets such as Cyrillic was the only option. As time went on it was decided to encode these additional characters using two or three bytes. A universal encoding system was invented. The translation routines in the radio software (**translate_class.py**) convert these characters to a specific character font position in the LCD ROM. For example:

Character	Encoding	Character ROM
Δ	\xd0\xb4	0xe3 (227)

The actual translation tables are contained in the **/usr/share/radio/codes** directory.

Appendix C – Wiring diagrams and lists

The following tables shows the wiring for the various versions of the radio. These configurations are normally set up by the `configure_radio.sh` program with the exception of the Vintage radio.

See *Configuring the radio* on page 99. It is also necessary to set the `pull_up_down` parameter in `/etc/radiod.conf` depending on wiring.

C1 Push Button and Rotary Encoder 40-pin wiring

The following table shows the wiring for the 40-pin push-buttons or rotary encoders.

Table 35 40-PinPush-buttons/Rotary encoder Wiring

Buttons/Encoders	Pin	GPIO	Configuration	Radio function
Menu button	11	17	menu_switch	Menu
Channel down button/Rotary switch A	16	23	down_switch	Channel down
Channel up button/Rotary switch B	18	24	up_switch	Channel up
Mute button	7	4	mute_switch	Mute sound
Volume down button/Rotary switch A	8	14	left_switch	Volume Down
Volume up button/Rotary switch B	10	15	right_switch	Volume Up

C.2 Push Button and Rotary Encoder 26-pin wiring

The following table shows the wiring for the 26-pin push-buttons or rotary encoders.

Table 36 26-PinPush-buttons/Rotary encoder Wiring

Buttons/Encoders	Pin	GPIO	Configuration	Radio function
Menu button	22	25	menu_switch	Menu
Channel down button/Rotary switch A	19	10	down_switch	Channel down
Channel up button/Rotary switch B	11	17	up_switch	Channel up
Mute button	7	4	mute_switch	Mute sound
Volume down button/Rotary switch A	8	14	left_switch	Volume Down
Volume up button/Rotary switch B	10	15	right_switch	Volume Up

Set `pull_up_down=up` in `/etc/radiod.conf` depending on wiring.

C.3 IQaudio Cosmic Controller wiring

The following table shows the wiring for the IQaudio Cosmic controller.

Table 37 IQaudio Cosmic Controller Wiring

Physical control	Pin	GPIO	Configuration	Radio function
Left hand push button	7	6	down_switch	Channel down
Middle push button	29	5	menu_switch	Menu
Right hand push button	31	4	up_switch	Channel up
Rotary encoder A input	16	23	left_switch	Volume control
Rotary encoder B input	18	24	right_switch	“ “
Rotary encoder Switch	13	27	mute_switch	Mute switch
Left status LED	8	14	rgb_red	Error status
Middle status LED	10	15	rgb_blue	Busy status
Right status LED	36	16	rgb_green	Normal status
IR sensor	22	25	In /boot/config.txt	Remote control

C.4 Pimoroni Pirate Radio wiring

The following table shows the wiring for the Pimoroni Pirate radio (pHat BEAT). Orientation is with the basic push buttons on the left-hand side. The menu button is on the top left-hand side.

Table 38 Pimoroni Pirate radio (pHat BEAT) Wiring

Pimoroni buttons	Pin	GPIO	Configuration	Radio function
On Off button	32	12	menu_switch	Menu
Channel Up button	29	5	up_switch	Channel up
Mute button	31	6	mute_switch	Mute sound
Channel Down button	16	13	down_switch	Channel Up
Volume Up button	36	16	right_switch	Volume Up
Volume Down Button	37	26	left_switch	Volume Down

The On/Off button used in the Pimoroni Radio software re-assigned as the menu switch with the Rathbone radio software. All LCD outputs are set to zero (No display).

Set `pull_up_down=up` in `/etc/radiod.conf`

C.5 Pimoroni Pirate Audio wiring

The following table shows the wiring for the Pimoroni Pirate radio (pHat BEAT).

Table 39 Pimoroni Pirate radio Audio Wiring

Radio buttons	Pimoroni	Pin	GPIO	Configuration	Radio function
Channel Up button	X	36	16	up_switch	Channel up
Channel Down button	A	29	5	down_switch	Channel Down
Volume Up button	Y	18	20/24 (1)	right_switch	Volume Up
Volume Down Button	B	31	6	left_switch	Volume Down

Note 1: Button Y (Volume up) can be GPIO 24 on earlier versions of the Pirate Audio card.

The new settings in `/etc/radiod.conf` are normally set to the following:

```
up_switch=16
down_switch=5
left_switch=6
right_switch=20
```

For earlier versions of the Pirate Audio this is.

```
right_switch=24
```

The Pimoroni Pirate Audio only has four buttons which means there are no buttons available for **Menu** or **Mute**.

For **Mute** press **Volume UP** and **Volume DOWN** together.

For **Menu** press **Channel UP** and **Channel Down** together.

The menu function currently doesn't work that well at the moment but is good enough to get to the Search menu. There is no shutdown function using menu button long press as with other designs.

Set `pull_up_down=up` in `/etc/radiod.conf`

C.6 Vintage Radio Push-button/Rotary Encoder 40-pin wiring

The following table shows the wiring for the 40-pin push-buttons or rotary encoders. This set-up must be manually configured in `/etc/radiod.conf`. It cannot currently be configured by the `configure_radio.sh` script.

Table 40 40-PinPush-buttons/Rotary encoder Wiring

Buttons/Encoders	Pin	GPIO	Configuration	Radio function
Menu button	22	25	menu_switch	Menu
Channel down button/Rotary switch A	19	10	down_switch	Channel down
Channel up button/Rotary switch B	11	17	up_switch	Channel up
Mute button	7	4	mute_switch	Mute sound
Volume down button/Rotary switch A	8	14	left_switch	Volume Down
Volume up button/Rotary switch B	15	10	right_switch	Volume Up

Table 41 Status LED indications

GPIO	Pin	LED	Function
23	16	Red	Error condition, shutdown in progress, IR activity (If configured)
22	15	Blue	Busy condition such as start-up, loading or changing radio stations or tracks.
27	13	Green	Normal operation such as playing stations or tracks.

Table 42 Rotary menu switch

GPIO	Pin	Switch value
24	18	1
8	24	2
7	26	4

Combining the above switch values gives a composite switch value of 0 through 7.

0=Idle, 1=Speak current station/track, 3=Search mode, 4=Source menu, 5=Options Menu 6,7 unused

C.7 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.

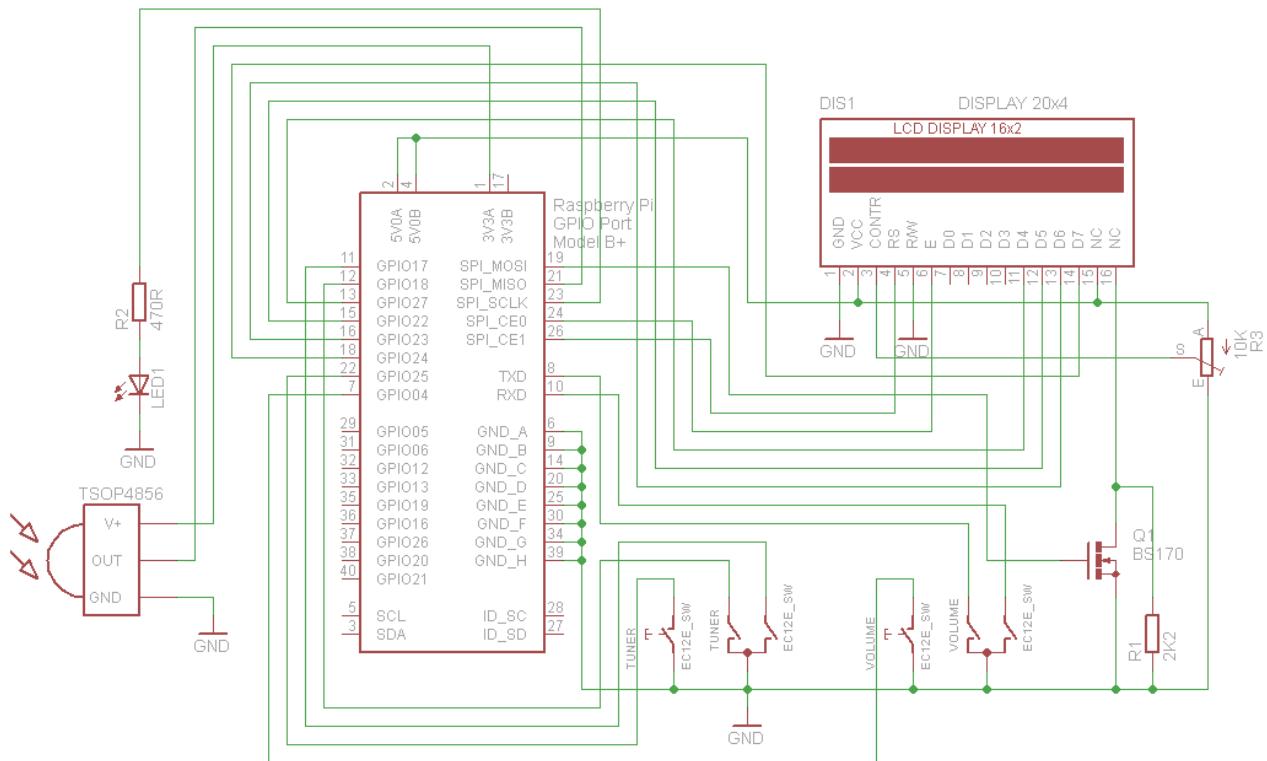


Figure 241 Wiring Raspberry Pi Radio Rotary Encoder version

Appendix D – Using a battery pack

It may be that you wish to make your radio portable. This can be done by using a 5V battery pack.



There are various manufacturers who supply a range of battery 5-Volt packs specifically for the Raspberry Pi. These are charged from any normal 5V charger.

Try to use a battery pack that directly connects to the GPIO header rather than connecting via the micro-USB or USB-C port.

The reason is that if connected by the USB port the internal regulator it will drop the voltage to 4.8 Volts which may give problems with the Wi-Fi connection.

Figure 242 Raspberry Pi 5 Volt battery pack

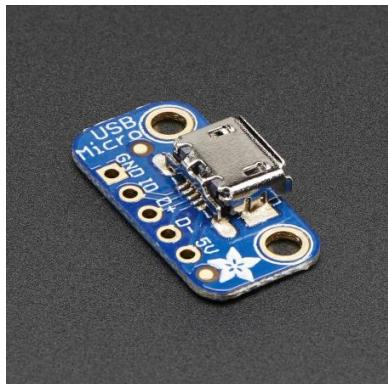


Figure 243 Micro USB breakout board

If the battery pack chosen doesn't connect directly to the GPIO header then purchase a micro-USB breakout board. This allows the 5 volt and GND connections to be connected directly to pins 2 and 6 respectively on the GPIO header.

Appendix E – Explanation of Vcc, Vdd, Vss and Vee

Throughout this guide the terms **Vcc**, **Vss**, **Vdd** and **Vee** may be used. They all apply to the DC voltage applied to a circuit, component or integrated circuit.

Vcc = Voltage Collector Collector

Vdd = Voltage Drain Drain

Vss = Voltage Source Source

Vee = Voltage Emitter Emitter

GND = Ground (0 Volts)

Vcc and **Vdd** are the positive supply voltage to an IC or circuit.

Vss and **Vee** are the negative supply voltage to an IC or electronic circuit.

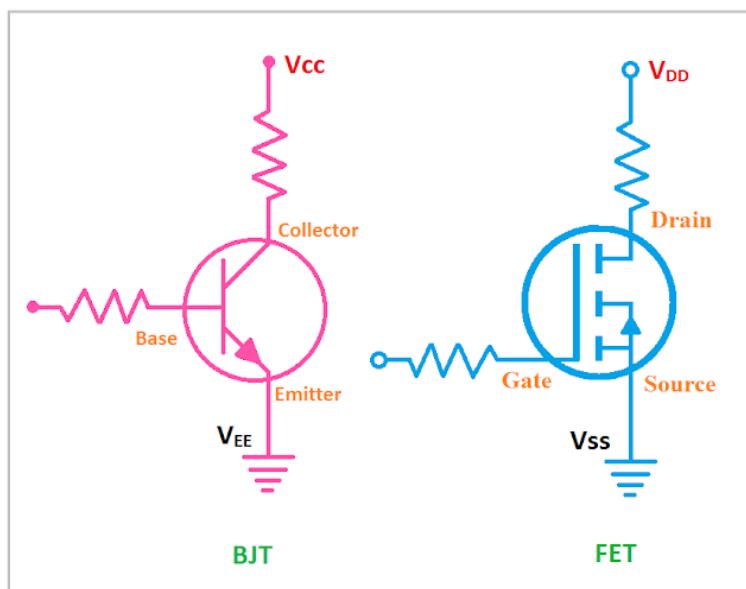


Figure 244 BJT and FET transistors

Which terms are used depends upon the type of circuit being connected to as shown in Figure 244 *BJT and FET transistors*. The terms **Vcc** and **Vee** are invariably used the most. **Vee** and **Vss** are normally tied to the 0V rail but as in the case of some LCDs can actually be a negative voltage.

Table 43 Supply voltage definitions summary

Supply Voltage	BJT (Bipolar Junction Transistor)	FET (Field Effect Transistor)
Positive Supply Voltage	Vcc	Vdd
Negative Supply Voltage	Vee	Vss

Index

1602 I2C LCD, 46
26 way ribbon cable, 40, 41
AAC, 229, 231, 339
activity LED, 8, 26, 51, 151, 178, 179
Adafruit, 6, 7, 26, 37, 38, 41, 45, 46, 47, 48, 49, 51, 68, 110, 135, 178, 180, 182, 214, 254, 272, 336, 337, 356
AdaFruit, 6, 26, 178, 199
AdaFruit industries, 6
AdaFruit RGB plate, 26, 178, 199
airflow, 26
airplay, 202, 298, 311, 313
Airplay, 204, 205, 298, 299, 300, 313, 356
Alarm, 198, 199, 212, 213, 356
Allo BOSS DAC, 358
Allo Piano, 61, 127, 128
alsamixer, 123, 125, 126, 134, 186, 195
amplifier, 8, 39, 41, 64, 66
anacron, 92, 215, 351
aplay, 69, 124, 173, 174, 250, 259, 350
Arduino, 47, 48, 110, 111
asound.conf, 173, 185, 187, 350
ASX, 229, 230, 231, 339
AV, 14
backup, 175
battery, 39, 367
battery pack, 367
Bitvise, 70, 207, 303
Bluetooth device, 126, 129, 261, 262, 263
Bluetooth speakers, 129
bluetoothctl, 129, 272
breakout boards, 44
Buster, 41, 80, 85, 302, 341
CAD, 9, 53, 178, 339
CGI, 160, 339
charset, 218, 219
CIFS, 235, 236, 237, 339
CloudBreak, 205
CODEC, 184
colours, 180, 214, 215
COM-09117 12-step rotary encoder, 30
Comitup, 153
configure_radio.sh, 24, 48, 99, 100, 134, 176, 177, 180, 208, 212, 248, 249, 253, 254, 310, 312, 344, 363
constructor's gallery, 25
cooling fans, 66
Cosmic controller, 24, 52, 92, 180, 310, 356, 363, 365
Cosmic Controller, 135
Cyrillic, 3, 17, 191, 192, 356, 360, 362
Cyrillic character LCD, 191, 192, 360
DAC, 27, 28, 55, 56, 57, 67, 68, 101, 124, 125, 133, 134, 173, 174, 177, 250, 259, 264, 266, 281, 312, 340, 354
daemon, 96, 111, 133, 151, 182, 196, 211, 215, 238, 239, 251, 252, 253, 254, 265, 276, 280, 283, 285, 305, 309, 312, 313, 314, 351
date format, 179, 306
DDOS, 301
Device Tree, 358
DHCP, 235, 339
DNS, 339
dpkg, 98, 160
DSP, 339
DT. *See* Device Tree
Elecrow, 122, 357, 358
Electromagnetic Interference, 65, 253, 340
electronic ink displays, 307
EMI, 65, 253, 340
equalizer, 185, 187
espeak, 11, 149, 172, 173, 174, 200, 212, 311, 337, 357
eSpeak, 2, 342
fail2ban, 302, 303
FBCP, 120
ferrite core, 65
ffmpeg, 91, 203
Fing, 70, 247
firewall, 302
firmware, 298
fsck, 241
FTP, 301
GPIO, 7, 18, 19, 26, 27, 28, 29, 30, 35, 39, 40, 41, 45, 47, 48, 49, 50, 51, 55, 56, 125, 135, 178, 272, 274, 340, 356
GPIO header, 39, 45, 48, 125
GPIO pins, 26, 27, 40, 41, 45, 47, 55, 274, 356
Ground Loop Isolator, 66
Grove JHD1313 LCD, 19

Grove LCD RGB, 19, 120
hciuart, 272
HD4470, 13, 17, 18, 24, 37, 41, 42, 309
HDMI, 15, 69, 124, 173, 241, 250, 312
heat sink, 66, 67
HiFiBerry, 27, 28, 55, 56, 64, 124, 125, 133, 134, 173, 174, 177, 250, 259, 281, 312, 354, 356
hostname, 88, 290
housing the radio, 25
I2C, 18, 19, 27, 28, 41, 45, 46, 47, 48, 49, 51, 65, 110, 111, 128, 135, 337, 340, 356
I2C interface, 18, 24, 25, 45, 47, 49, 52, 356
Icecast2, 212, 283, 284, 285, 286, 288, 289, 290, 336, 337
install_airplay, 298
interface board, 40, 42, 43, 46, 48, 49, 63
Internet Security, 301
ioe-python, 257
ioexpander, 257
iPad, 356
iPhone, 356
iPod 4 pole AV, 14
iptables, 302, 303
iptables-persistent, 304
iptables-save, 304
IPv4, 340
IPv6, 252, 340
IQaudio, 363
IQAudio, 15, 24, 28, 37, 41, 51, 52, 56, 57, 64, 92, 109, 124, 125, 134, 135, 173, 177, 180, 185, 187, 310, 312, 354, 356
IR, 8, 9, 26, 27, 28, 41, 49, 50, 51, 53, 54, 135, 151, 313, 340, 351, 354, 356
IR sensor, 50
IR Sensor, 26, 27, 41, 50, 135
ireventd.service, 143, 144, 150, 315
ir-keytable, 140, 141, 142, 143, 146, 314, 315
irradiod.service, 315, 349, 350, 351
irrecord, 137, 140, 146, 148, 149, 268, 314
Jessie, 74, 85, 86, 96, 235
Jessie Lite, 15, 74, 80, 85, 96, 356
JustBoom, 27, 57, 58, 64, 125, 356
KY-040 Rotary Encoder, 31
language file, 174, 183
LCD, 6, 7, 8, 13, 17, 24, 27, 28, 37, 38, 39, 41, 45, 46, 47, 48, 51, 65, 68, 101, 110, 128, 135, 180, 182, 196, 198, 199, 207, 212, 213, 214, 253, 254, 255, 272, 274, 285, 289, 306, 311, 336, 337, 340, 356, 357
LED Backlight, 37
lirc, 148, 151, 152, 268, 352
LIRC, 340
Locale, 90
M3U, 228, 229, 258, 289, 340, 356
m3u8, 230
MAC, 340
mains filter, 65
metadata, 357
MHS, 23, 118
micro USB, 15
mixer_volume, 212
MP3, 210, 229, 356
MPC, 133, 214, 251, 341
mpd, 95, 133, 134, 170, 211, 214, 215, 217, 237, 238, 239, 251, 252, 253, 255, 258, 266, 276, 283, 289, 290, 306, 311, 337
MPD, 95, 111, 133, 157, 182, 196, 211, 214, 215, 217, 237, 238, 239, 250, 251, 252, 265, 276, 283, 284, 290, 305, 306, 311, 337, 341, 356
MPD Upgrading, 170
MPDdroid, 238
MPDroid, 238, 239, 357
mpeg, 91, 231
MPEG, 229, 341
MPEG3, 229, 340, 341
MySQL, 157
nano, 72, 73
NAS, 210, 212, 236, 341, 356
Network Time Protocol, 212, 341
news feed, 184, 356
NFS, 235, 236, 258, 341
NTP, 212, 341
O!MPD, 157, 160, 165, 223
OLED, 17, 24, 52, 92, 110, 180, 310, 338, 341, 356
Olimex Limited, 310, 338
Optical rotary encoders, 36
Organic Light Emitting Diode, 17
OS, 41, 236, 341
Parameters
 *_color, 180
 client_timeout, 182
 CODECS, 184
 codepage, 192
 controller, 192
 dateformat, 179
 display_lines, 194
 display_type, 309
 display_width, 194
 down_switch, 177

exit_action, 179
flip_display_vertically, 180
i2c_bus, 194
internet_check_port, 193
internet_check_url, 193
language, 191, 192
lcd_data4, 178
lcd_data5, 178
lcd_data6, 178
lcd_data7, 178
lcd_enable, 178
lcd_select, 178
left_switch, 177
menu_switch, 177
menu_switch_value_*, 194
mpdport, 193
mute_action, 184
mute_switch, 177
pull_up_down, 178
remote_control_host, 193
remote_control_port, 193
remote_led, 178
remote_listen_host, 193
rgb_blue, 33
rgb_green, 33, 180
rgb_red, 33, 180
right_switch, 177
romanize, 192, 360
rotary_class, 194
rotary_gpio_pullup, 194
scroll_speed, 191
shutdown_command, 179
speak_info, 172
speech, 172
speech_volume, 172
startup, 181
streaming_on, 284
translate_lcd, 362
up_switch, 177
verbose, 172
volume_display, 181
volume_range, 182
Parameters [AIRPLAY]
airplay, 298
mixer_preset, 267, 298
Parameters [SCREEN]
banner_color, 176
dateformat, 177
display_date, 177
display_icecast_button, 289
display_mouse, 177
display_title, 177
display_window_color, 176
display_window_labels_color, 176
fullscreen, 176
labels_color, 176
scale_labels_color, 177
slider_color, 177
stations_per_page, 177
wallpaper, 177
window_color, 176
window_title, 176
PC, 6, 25, 41, 97, 184, 210, 229, 235, 283, 286, 287, 288, 290, 327, 341, 356
PC speakers., 287
PCF8574, 47, 48, 337
PCM, 56, 174, 341
PCM5102A, 58
PCM5102A DAC, 59
pHat BEAT, 9, 20, 53, 96, 97, 113, 364
Phishing, 301
Pi Pico, 15
Pi Zero, 7, 15
Pi Zero W, 15
PID, 341
PiFace, 9, 53, 339, 351
PiFace CAD, 24, 51, 53, 97, 103, 122, 256, 341, 356
PiFace Control and Display, 9
PIL
 See Pillow, 257
Pillow, 257
Pimoroni, 9, 53, 364
Pimoroni pHat, 56, 126, 358
Pimoroni pHAT, 59
Pimoroni Pirate radio, 113
Pirate Audio, 20, 114, 115
Pirate radio, 9, 20, 53, 364
PiWiz, 94
PLS, 228, 229, 230, 251, 341
potentiometer, 48, 253
power adapter, 39
power supply switch, 39
pulseaudio, 59, 96, 97, 134, 250, 254, 257
Putty, 70, 207, 303
PWM, 332
pygame, 85, 207
radiod package, 331
radiod.conf, 47, 48, 56, 152, 172, 178, 179, 180, 181, 182, 211, 212, 280, 281, 307, 311, 313, 344
Random, 198, 199, 306

Rasbian package, 331
Raspberry Pi, 1, 6, 7, 8, 13, 14, 26, 39, 41, 45, 49, 50, 51, 65, 69, 97, 98, 111, 134, 135, 158, 161, 178, 210, 214, 215, 253, 265, 279, 285, 289, 290, 317, 336, 337, 340, 356
Raspbian Jessie, 74
Raspotify, 291
Real Time Clock, 267, 268, 334, 341
Red Blue Green LED, 313
remote control, 9, 26, 50, 51, 53, 136, 148, 150, 174, 178, 280, 313, 314, 315, 351, 352, 356
Revision 1 board, 30
Rexec, 301
rfkill, 246
Romanize, 360
Romanized, 191, 192, 360, 361
rotary encoder, 6, 7, 8, 24, 26, 27, 29, 30, 31, 41, 46, 48, 49, 51, 135, 182, 198, 199, 200, 213, 272, 307, 356
rpi-update, 85
RSS, 183, 184, 198, 199, 212, 311, 341, 356
RTC. *See* Real Time Clock
Russian, 3, 17, 18, 191, 192, 356, 360, 361, 362
Russian, 191
Russian, 360
screen saver, 209
SD, 341
SD card, 175
Secure Shell. *See* SSH
Serial Peripheral Bus interface, 9
service mpd, 133
service radiod, 111, 184, 196, 280
SH1106, 19, 24, 97, 105, 119, 309, 346, 356
shairport-sync, 298, 299, 313, 356
Shoutcast, 162, 215, 225, 226, 227, 228, 229, 230
smbus2, 110
softvol, 195
SoftVol, 195, 297
Solid State Drives, 319
sourceforge, 146, 147, 173, 174
speech, 172, 174
speech_volume, 172
SPI, 9, 47, 53
SPI interface, 24, 53, 103
Spotify, 291, 294, 295, 296, 356
SSD, 319, 325
SSD1306, 18, 24, 92, 97, 105, 114, 115, 119, 309, 310, 338, 346, 356
SSH, 80, 214, 301, 342
SSID, 342
Stretch, 14
systemd, 349
TCP/IP, 313, 342
Telnet, 301
TFT, 6, 22, 24, 120, 121, 342, 356
timeout, 182
timesync, 212
timezone, 86, 87, 88
tone control, 11, 67
touch screen, 5, 21, 22, 24, 25, 356
TSOP38238, 41, 50
TSOP382xx, 26
type of radio, 24
UDP, 152, 313, 315, 342
URL, 184, 198, 199, 212, 229, 230, 231, 258, 283, 289, 290, 305, 340, 342
USB, 7, 15, 39, 41, 46, 65, 210, 238, 264, 266, 337, 342, 356
USB adaptor, 7
USB disk drive, 75, 318, 319, 323, 326, 327
USB stick, 210, 238, 356
USB to Ethernet adapter, 7, 15
USB-C, 14, 39
Vcc, 368
Vdd, 368
Vee, 368
version 1.0 boards, 29
vi, 72
vintage radio, 11, 25, 178, 313
Vintage radio, 51, 64, 178, 356
Vss, 368
wake-up button, 49
Waveshare, 22, 117, 307
Web interface, 157, 161, 169, 235, 305, 306, 356
WEP, 343
wget, 97, 98, 160, 231
WIFI, 88, 343
Wi-Fi signal strength, 244
Win32DiskImager, 175
wiring, 18, 19, 27, 30, 37, 41, 48, 68, 179, 253, 254, 257
wiring diagram, 366
WMA, 210, 356
WPA, 152, 343
WPA2, 343
XML, 230, 343
xscreensaver, 208, 209
xscreensaver-command, 209

