Raspberry Pi Internet Radio

Constructors Manual

A comprehensive guide to building Internet radios using the Raspberry Pi

Bob Rathbone Computer Consultancy

www.bobrathbone.com

29th of January 2020

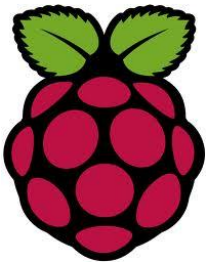
Version 6.12

Contents

Figures

Tables

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 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.

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

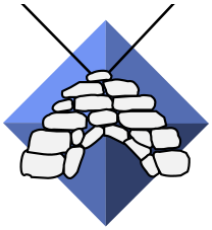
This manual contains instructions how to run the radio software with Bluetooth speakers or headphones. Bluetooth versions 1.x through 5.x supported depending upon the Raspberry Pi model used.

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 allows multiple playlist creation from Shoutcast radio stations by genre or country which can selected through the radio 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.

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, PiFace CAD 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.

A full specification can be found in *Appendix C – Technical specification* on page 253.

Document Overview

This document is organised into the following sections.

Quick links

Please note that there is also a full document index on page 260 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.

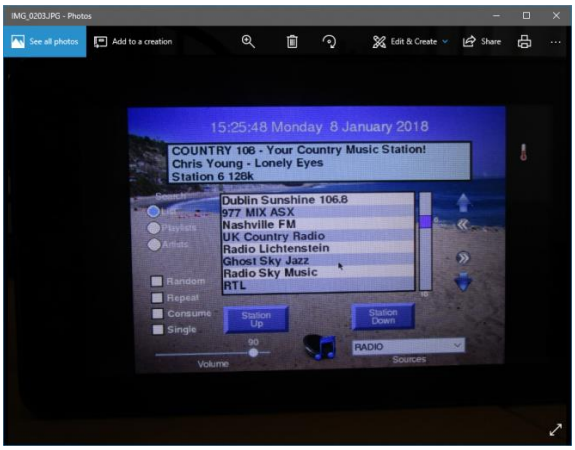


Figure 1 Raspberry pi 7-inch touchscreen radio

The latest design 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.

Figure 2 HDMI Television running the radio

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.

Figure 3 Vintage tuning touch-screen radio

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 4 Adafruit 3.5 inch TFT

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 *Construction using the Adafruit 3.5-inch TFT* on page 41.

Figure 5 Radio using the Adafruit LCD plate

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

Figure 6 Lego Internet Radio

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

Figure 7 Pi radio using rotary encoders

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

Figure 8 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 9 Transparent Perspex Radio

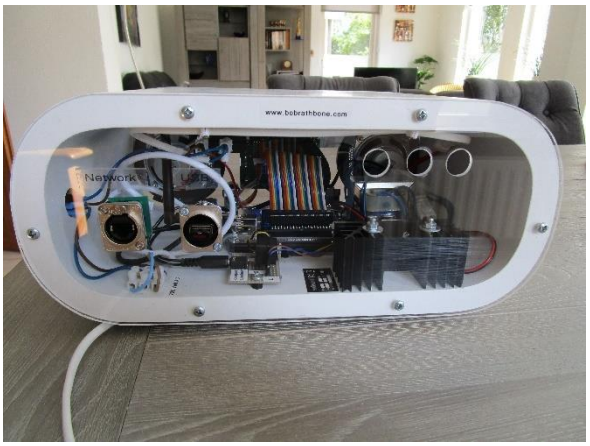


Figure 10 Perspex radio rear view

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.

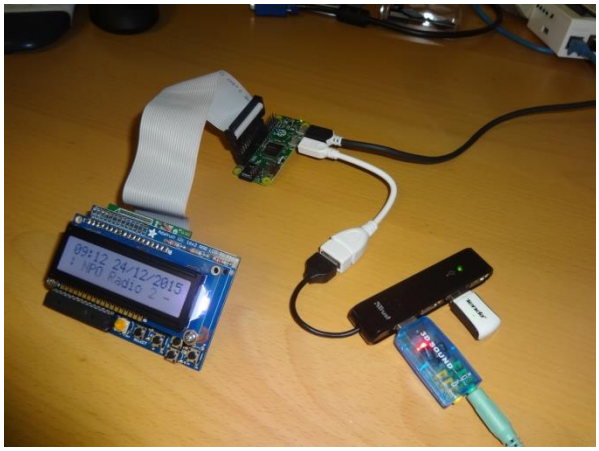


Figure 11 The Radio running on a Pi Zero

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 12 Boom Box radio front view

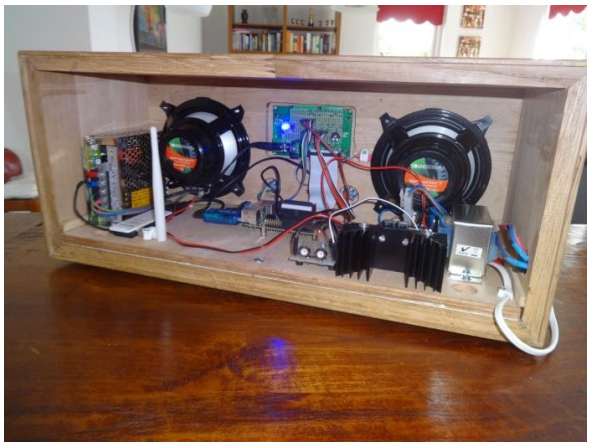


Figure 13 Boom Box Radio rear view

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 14 Raspberry Pi Wine Box radio

Figure 15 Wine box internet radio internal view

Figure 16 Very small radio using the Cosmic Controller

Here is a really cute radio made using the IQaudIO 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.

The lenses over the three LEDs and IR sensor apertures are simply stick-on transparent furniture protectors available from most hardware stores.

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 17 RPI radio with two-stage valve amplifier

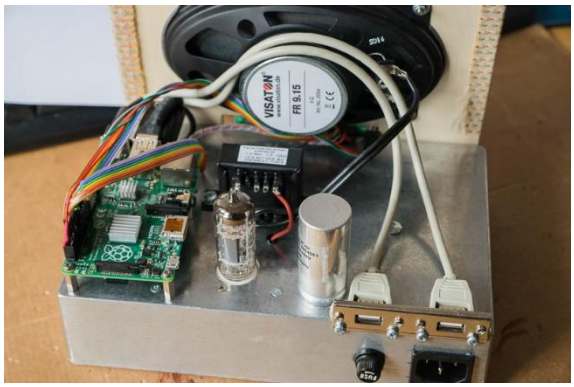


Figure 18 The RPi valve radio chassis view

Figure 19 Pimoroni Pirate Radio

From version 6.9 onwards the 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

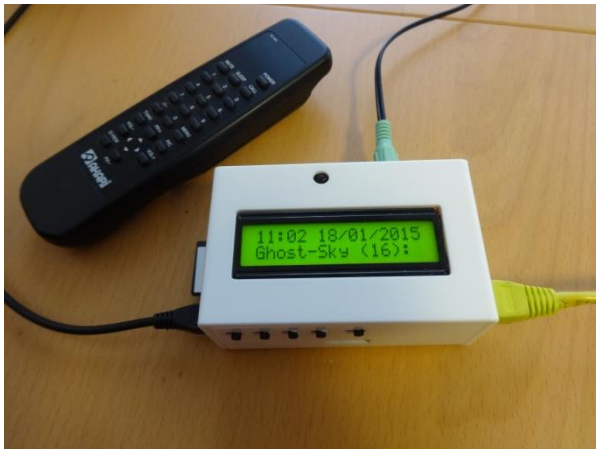


Figure 20 PiFace CAD Radio with IR Remote Control

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. It also has inbuilt support for a remote control. It has one drawback in that the push buttons are on the bottom of the unit. The PiFace CAD uses the Serial Peripheral Bus interface (SPI) on the Raspberry Pi.

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 21 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 22 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

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 HD44870 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 originally with the intention of promoting the teaching of basic computer science in schools. It has however become immensly popular with hobbyists and engineers.

The Raspberry Pi model 3B

Figure 23 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
If you are new to the Raspberry PI try the following beginners guide. http://elinux.org/RPi_Beginners

Raspberry Pi model 4B

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

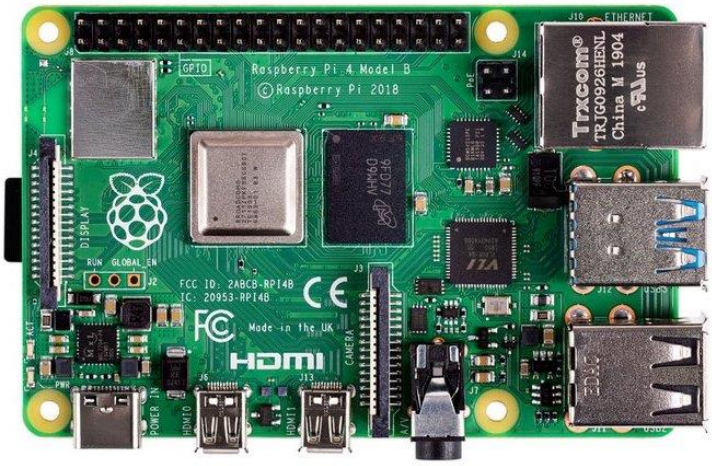


Figure 24 Raspberry Pi Model 4B Computer

Note: The Raspberry Pi 4B requires **Raspbian Buster** SD card. It will not work with **Raspbian Stretch**.

Figure 25 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 26 Rasberry Pi MicroHDMI cable

The model 4B also requires an official Raspberry Pi MicroHDMI cable for each of the micro HDMI ports (This is not the same as the PiZero 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 3.55mm AV cable. To complicate matters: not all of these cables are the same! However existing audio jack plugs are compatible with the new AV connector.

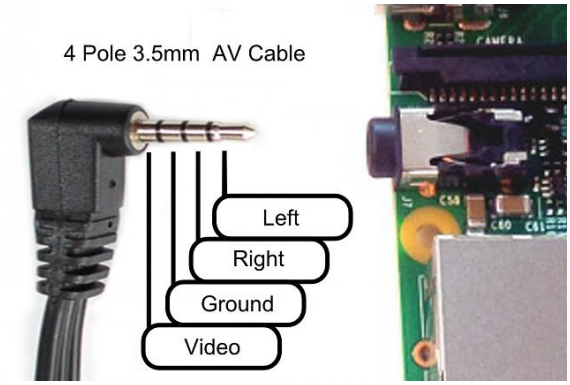


Figure 27 Raspberry PI B+ AV cable

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 28 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 an micro USB to standard USB adapter. The PiJack Ethernet adapter board is not currently supported. The Pi Zero W has onboard Wifi and Bluetooth and is a better choice for the radio.

Figure 29 USB Ethernet adapter

Note: 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 11) or one of the Pi Zero DAC boards available from manufacturers such as **IQaudIO**, **HiFiBerry**, or **JustBoom**.

Alternatively, from version 6.12 use Bluetooth speakers. See *Connecting a Bluetooth device* on page **Error! Bookmark not defined.**

The HD44870 LCD display

Figure 30 The HD44870 LCD display

The HD44870 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 HD44870 LCD controller so there is a wide choice of these displays.

Figure 31 OLED 4 x20 LCD display

The latest displays use OLED displays (Organic Light Emitting Diode) and give very good results.

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

For pin-out details see LCD pin outs on page 26.

The Midas HD44780 LCD display

Figure 32 The HD44780 LCD display

The HD44780 LCD from Midas is an alternative to the HD44870. It is pin compatible with the HD44870 except for pin 15 (VEE) which is a negative voltage output.

For pin-out details see LCD pin outs on page 26.

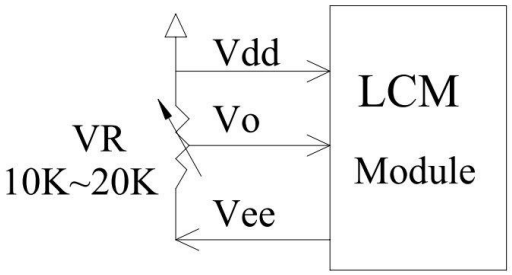


Figure 33 HD44780 potentiometer wiring

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.

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 34 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 3.5-inch TFT

From version 6.7 onwards the radio software supports the Adafruit 3.5-inch TFT touch screen (720 x 480 pixels). The small size can make the controls difficult to use but will still work.

Figure 35 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.

Warning: During testing, a touch screen using USB-OTG (USB On-The-Go) technology was tried but was unsuccessful. Also attempts to enable USB OTG software seemed to disable the normal USB ports. USB-OTG technology allows USB devices to act as a host for other USB devices such as USB storage or cameras. The author regrettably cannot give any support on these types of screens.

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

```
lcd_rotate=2
```

Reboot the Raspberry Pi for the change to take effect.

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 *C.2 -Elecrow 7-inch touch-screen notes* on page 254.

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 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.

Pimoroni Pirate Radio

Figure 36 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 88.

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	Àdafruit 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	PiFace CAD 2x16 LCD

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 3.5 TFT (480x720 pixels)
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	PiFace CAD 6-buttons (Five used)

Any type of HD44870 LCD display can be used with any user interface. The HD44870 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 screen 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

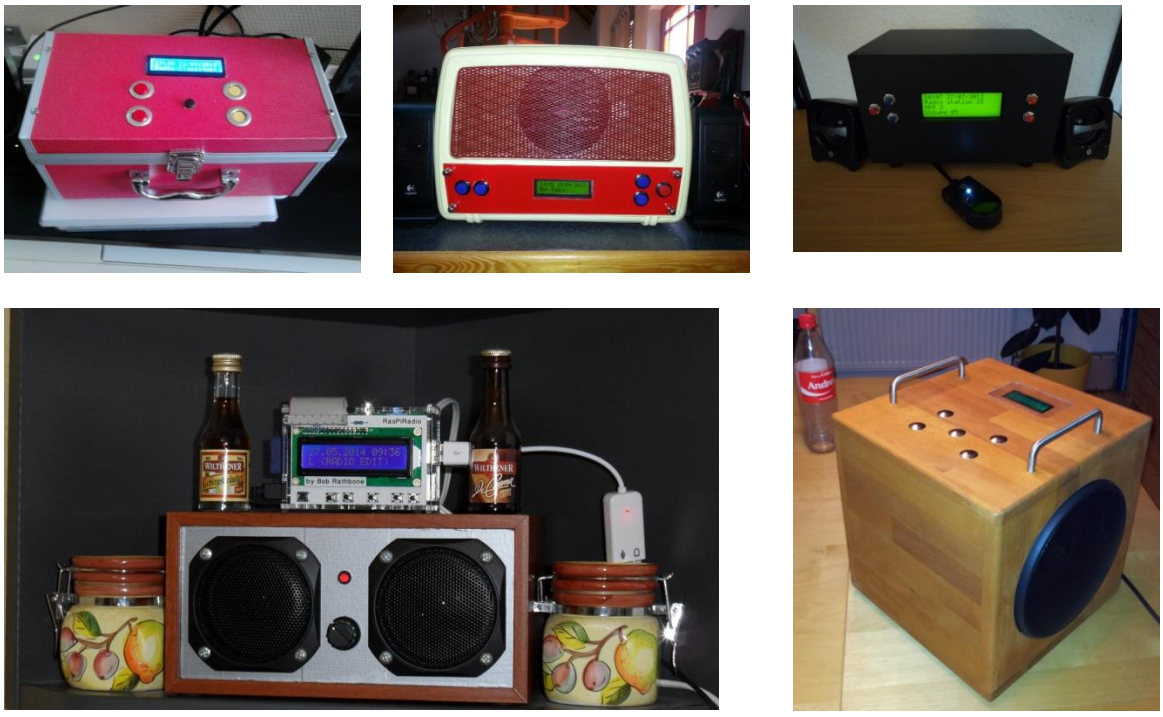
The first choice uses more wiring but is the cheapest option. The second choice uses an I2C backpack which is an extra component to be purchased. However, I2C backpacks are reasonably cheap.

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 37 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 WiFi dongle with an aerial mounted externally to the case. Also, the case must be earthed at the main supply both for safety reasons and to prevent interference with sound and/or the LCD screen

Building in a IR sensor and remote control

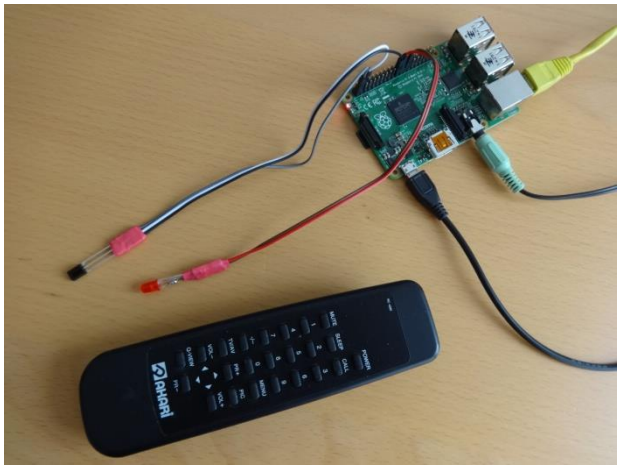


Figure 38 IR Sensor and Remote control

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.

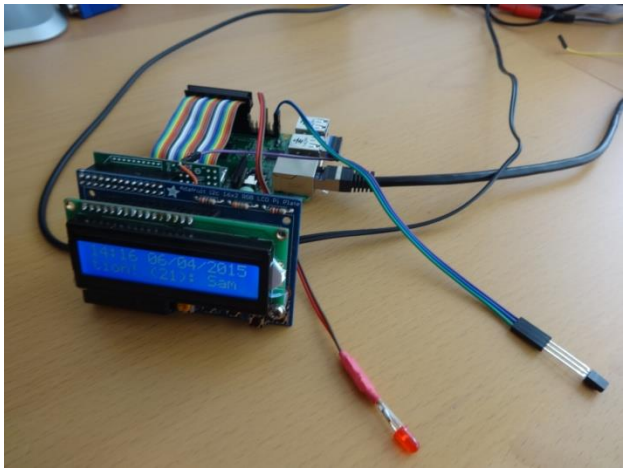


Figure 39 Adafruit and IR sensor and activity LED

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 52 on page 32.

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

Table 4 and Table 5 on the following pages 23 and 24 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 4 and Table 5 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.

This is where it can get a little confusing. The radio components (LCD, buttons, rotary encoders etc.) can use either 26-pin or 40-pin wiring as shown in the two tables. The 26-pin version wiring can also be used on a 40-pin Raspberry Pi. The 40-pin version of the wiring in Table 5 guarantees that there will be no wiring conflicts with DAC components.

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

Table 3 Radio wiring conflicts

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

Colour Legend | Radio | Conflict | Alternative wiring |

You are strongly advised to use the alternative 40 pin wiring scheme so that IQaudIO and **HiFiBerry** products and similar can be used either at the outset or at a later date.

The configuration for the radio is contained in a file called **/etc/radiod.conf**. By default, this is configured for the 40-pin scheme. Every component of the radio is configurable in this configuration file. There is a program called **configure_radio.sh** that is used to configure the settings in the **/etc/radiod.conf** file.

The original 26-pin Raspberry Pi's had a GPIO conflict with GPIO 18 which was used by DACs. The down switch now uses GPIO 10 as a result.

```
down_switch=18
```

Changed to:

```
down_switch=10
```

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).

If using DAC products do not use the 26 pin version of the wiring but use the wiring shown in Table 5 on page 24.

Table 4 Controls and LCD wiring 26 pin version

Pin	Description	Radio Function	Name	LCD pin	Push Buttons	Encoder (Tuner)	Encoder (Volume)
1	3V3	+3V supply			COMMON		
2	5V	5V for LCD		2,15			
3	GPIO2	I2C Data*	I2C Data				
4	5V						
5	GPIO3	I2C Clock*	I2C Clock				
6	GND	Zero volts		1,3*,5,16		Common	Common
7	GPIO 4	Mute volume			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	+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-CE0	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 5 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	+3V supply	+3V	+3V		+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	+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-CE0		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 40 below.

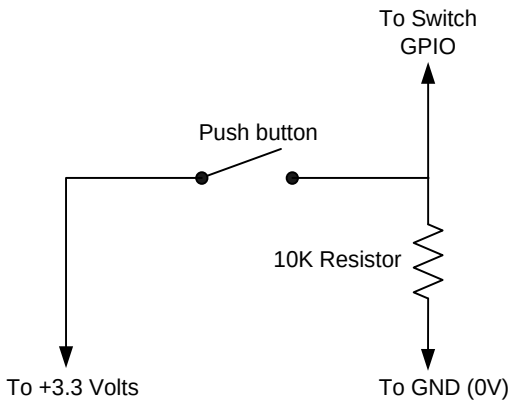


Figure 40 Push-button Wiring version 1 boards

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 5 on page 24. As from version 6.9 onwards the other side of the switches can be wired to either +3.3V (Original 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.

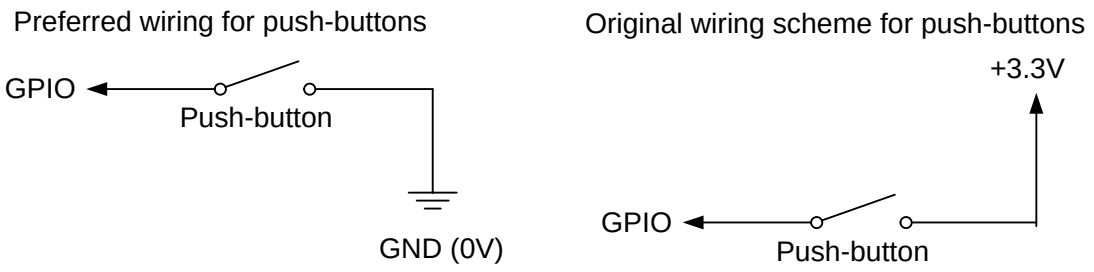


Figure 41 Push-button wiring version 2 onwards boards

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 126.

Rotary encoder wiring

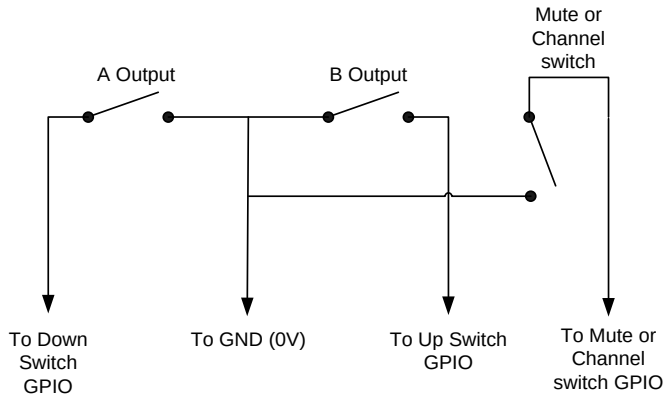


Figure 42 Rotary Encoder Diagram

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 4 on page 23. 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 43 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 are used in this project.

The rotary encoders used in this project are wired with the COMMON or GND pin in the middle and the A and B outputs either side. However, some rotary encoders are wired with A and B as the first two pins and GND (COM) as the third pin. 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.

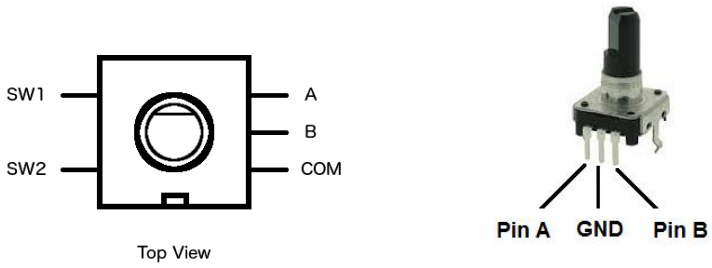


Figure 44 Rotary encoder pin-outs

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

LCD Module Wiring

The following shows the wiring for a directly wired HD44870 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 4 and Table 5 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 6 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 5).

Note 1: If using the Midas HD44780 display do not connect pin 15 to the +5V supply or you will damage the display. Connect pin 15 as shown in the section called *The Midas HD44780 LCD display* on page 15.

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 58.

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 *The Midas HD44780 LCD display* on page 15.

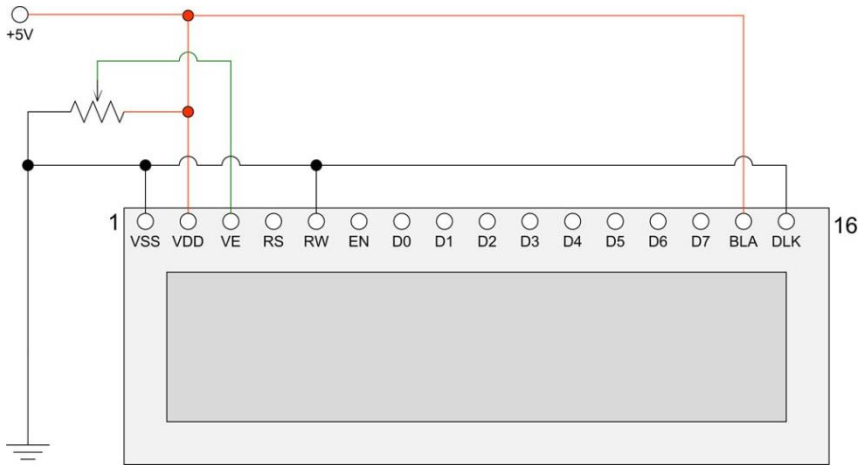


Figure 45 HD44870 LCD electrical circuit

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 58.

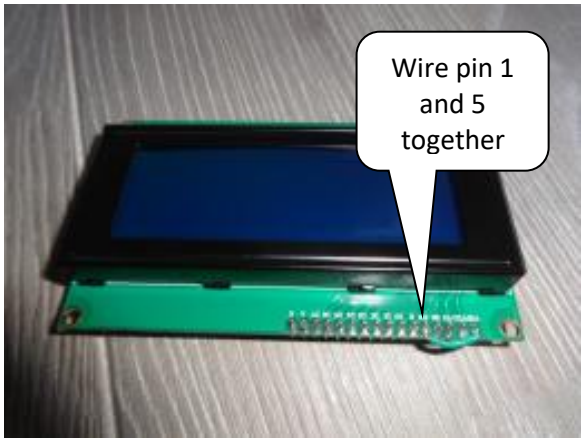


Figure 46 Wire LCD pin 1 (GND) and 5 (RW) together

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 HD44780 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 *The Midas* HD44780 *LCD* display on page 15 for further information.

If using a 16x4 LCD although not directly supported then set the **lcd_width** statement in **/etc/radiod.conf** to 16 (lcd_width=16).

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).

From version 5.9 onwards there is a new program called wiring.py which will display physical the wiring required for the settings found in **/etc/radiod.conf**. See page 197.

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 such as the Ciseco Humble PI can provide power in this way.
Visit https://www.modmypi.com
This interface board can be ordered with an optional 5-volt regulator. If using the Humble PI try with regulator use 6v to 7v as the power input to the Humble PI 5v regulator (Not the Raspberry PI). Trying to use 9v or more will mean that the 5-volt regulator will get far too hot.

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

Things not to do:

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

Things to do:

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

You should try to use a single power supply switch for the radio. 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 30.

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 235.

Selecting an audio amplifier

There is a wide range of amplifiers that can be used with the radio which fall into four 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 and amplifier from manufacturers such as **IQaudIO**, **HiFiBerry** or **JustBoom**
4. The audio stage of an existing (vintage) radio. Use the PA or record player input (Usually mono only)

Figure 47 PC speakers

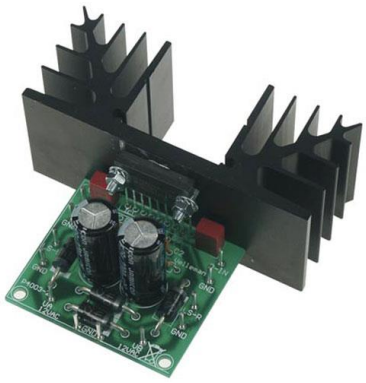


Figure 48 Velleman 30W stereo amplifier

Figure 49 IQaudIO DAC and 20W Amplifier

Figure 50 Vintage radio PA input

The above manufacturer's boards are examples only and do not imply any specific recommendations.

GPIO Hardware Notes

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

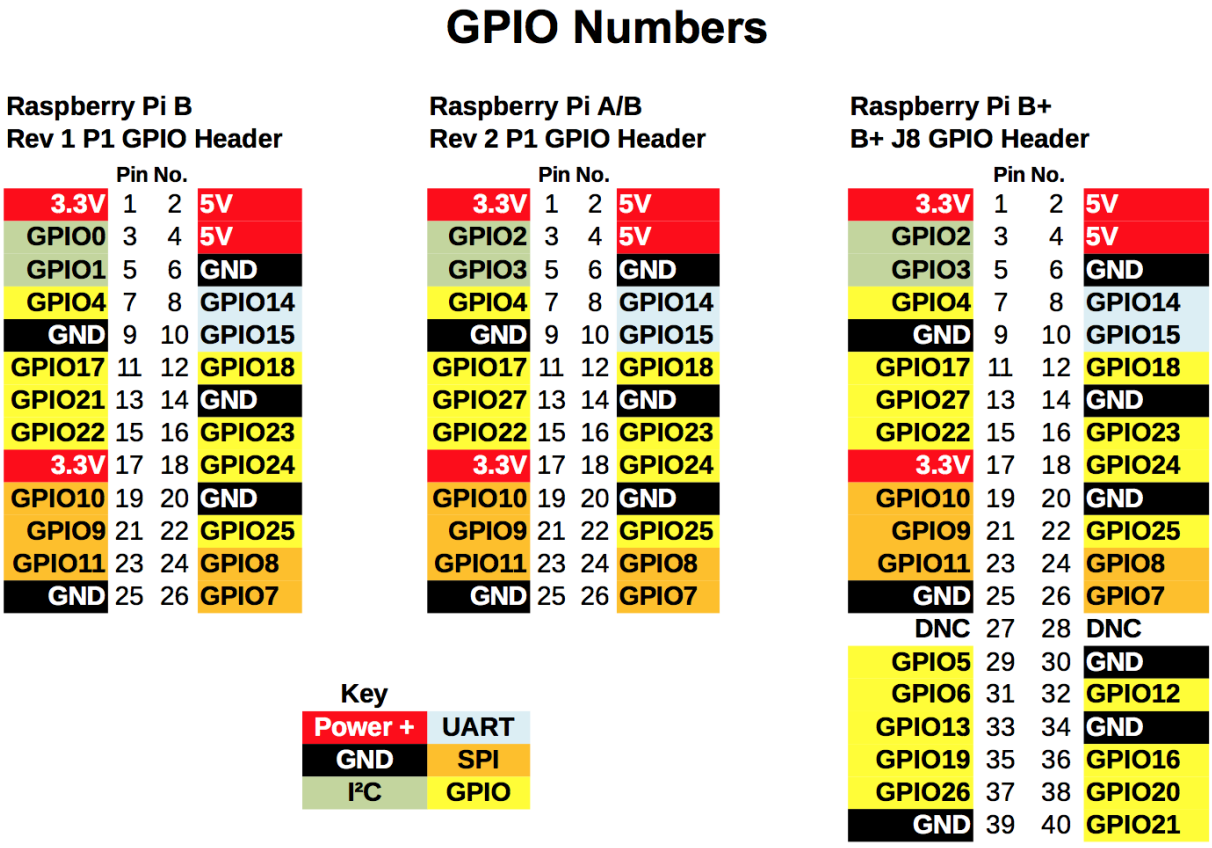
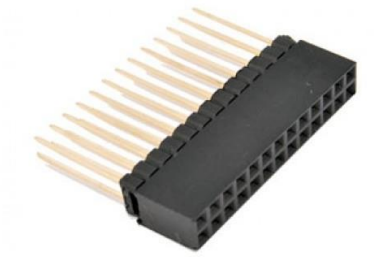


Figure 51 GPIO Numbers

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



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

Figure 52 26 pin header extender

Parts List

The following table shows the parts list for the Raspberry PI Internet Radio. This list is for the version using the HD44870 LCD directly connected to the GPIO pins. If using the Adafruit five button LCD Plate then don't order the parts marked with an asterix (*)

Table 7 Parts list (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 20
1	Raspbian Buster 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 HD44870 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	Four port USB hub (Revision 1 & 2 boards only)	Any PC supplier
1	External power supply for USB hub (1200 mA)	Any PC supplier
1	26 way PCB mount male connector	Any electronics shop
1	26 way GPIO extender (model B+ boards only)	ModMyPi and others
1	Mains cable	Hardware shop
1	Double pole mains switch with neon	Farnell Element 14
5	Male 2 pin PCB mount connectors	Any electronics shop
2	Female 4 pin PCB connectors	Any electronics shop
1	Female 2 pin PCB connectors	Any electronics shop
1	16 pin male in-line PCB mount connector	Any electronics shop
1	Stereo jack plug socket	Any electronics shop
1	Panel mount Ethernet socket	Any electronics shop
1	Adafruit I2C LCD interface Backpack ***	http://www.adafruit.com/
1	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 53 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.

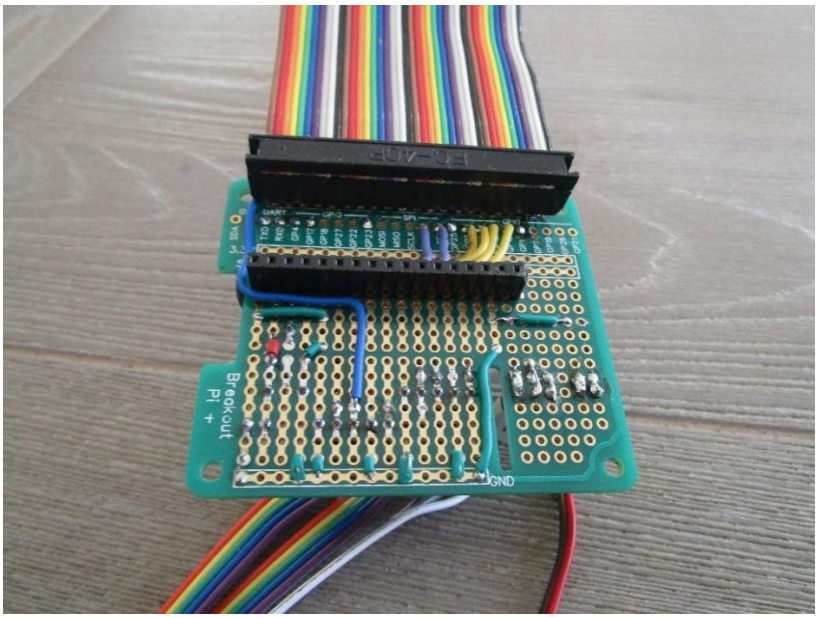


Figure 53 40-pin Interface board with ribbon cable

The figure below shows a 4x20 HD44870 LCD plugged into the interface board.

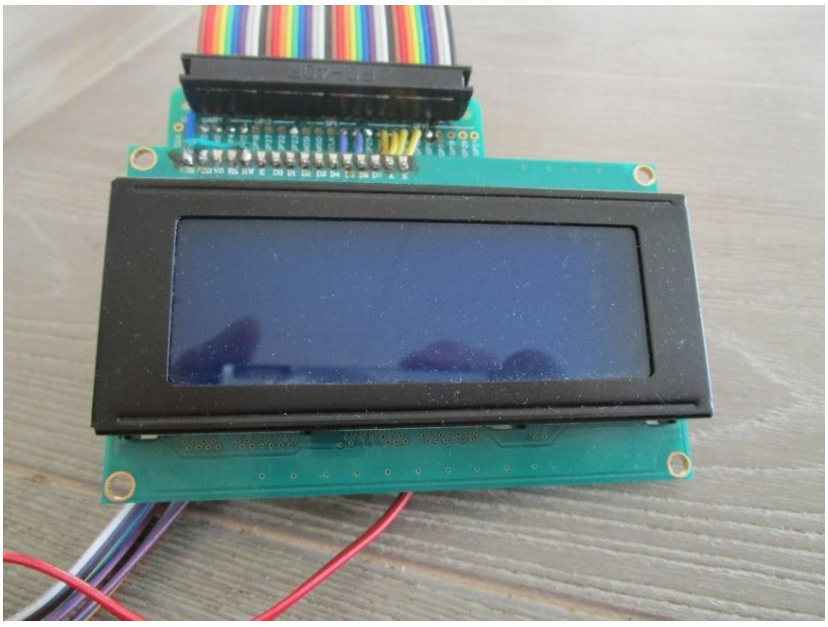


Figure 54 Interface board with LCD screen attached

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

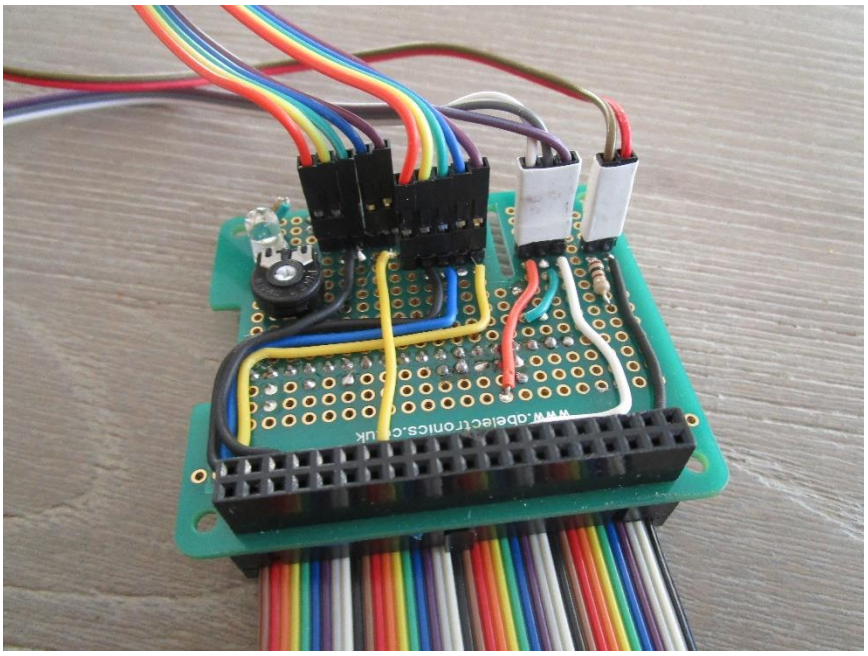


Figure 55 Radio controls connections

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

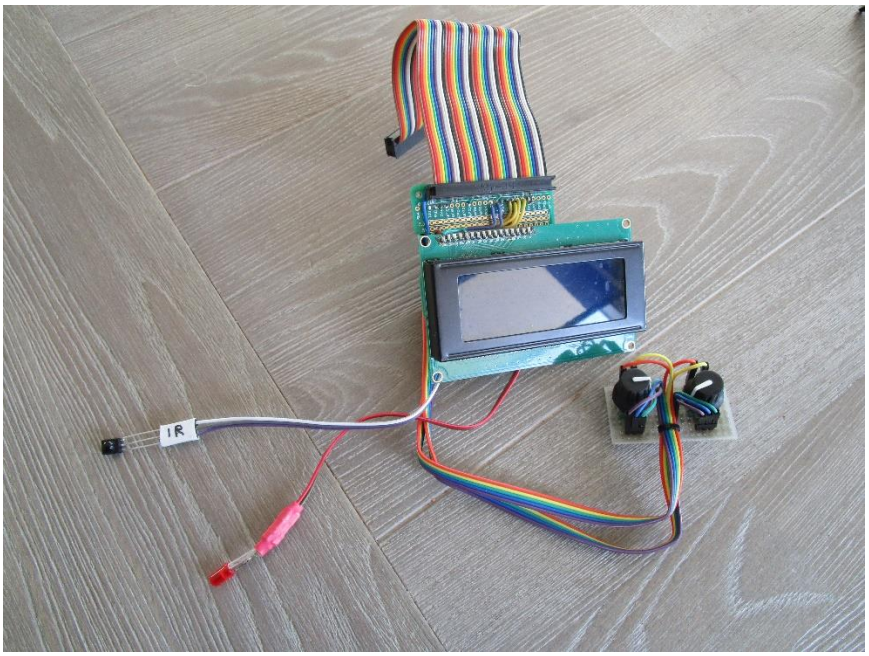


Figure 56 Interface board overview

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.

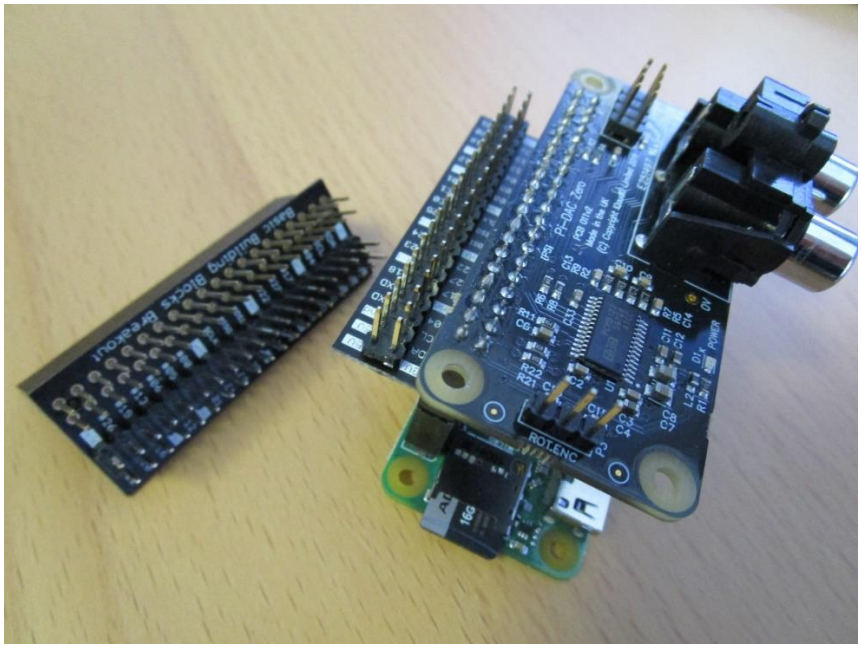


Figure 57 GPIO header breakout board

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.

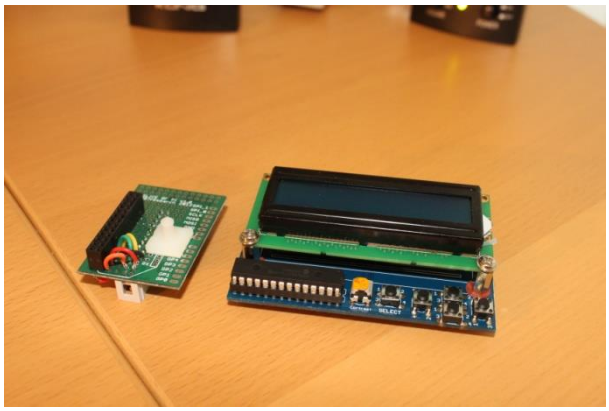


Figure 58 Adafruit LCD plate

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.

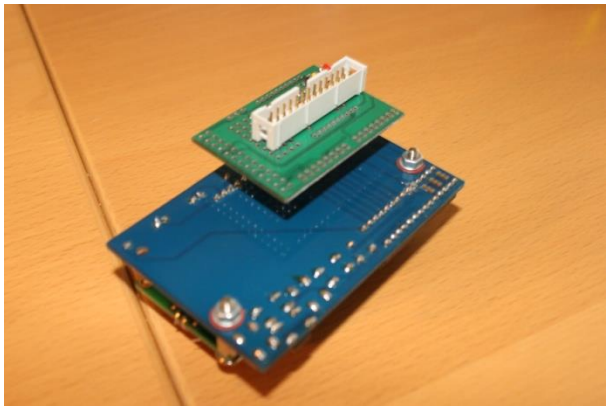


Figure 59 Adafruit LCD plate with ribbon cable adapter

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 from version 5.10 onwards as the backlight is software controlled.

Figure 60 Chinese 1602 I2C LCD

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.

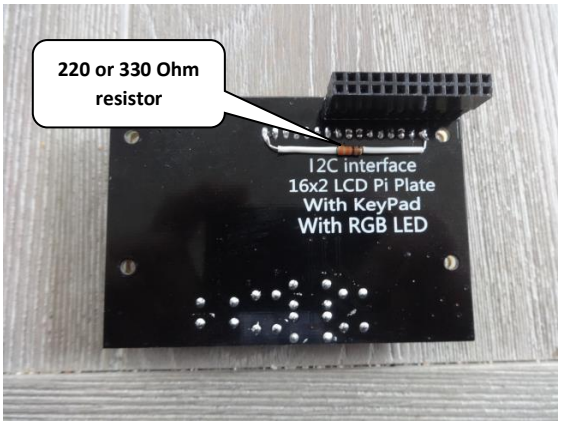


Figure 61 Enabling the backlight.

If running the radio version 5.9 or earlier then solder a 220 or 330 Ohm resister 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.

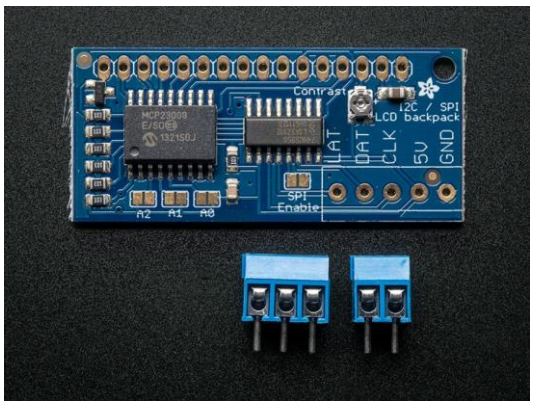
The radio software also supports the more common PCF8574 chip based backpack popular with the Arduino hobby computer may also be used. See http://www.play-zone.ch/en/i2c-backpack-pcf8574t-fur-1602-lcds-5v.html for example.

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.

Figure 62 Adafruit I2C Backpack

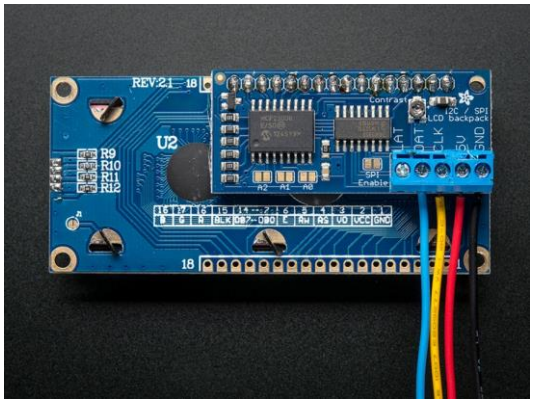


Figure 63 LCD connected to an Adafruit I2C backpack

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

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

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

Arduino PCF8574 I2C backpacks

These types of backpack are popular with Arduino users. The device address is usually hex 0x27. Another manufacture may use hex 0x37. This is configurable in the radio configuration program.

Figure 64 Arduino I2C backpack

The wiring From top to bottom is:

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

The blue potentiometer on the right is the contrast adjustment.

```
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.

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 Rasberry Pi. The Ciseco Humble Pi prototype board shown in Figure 65 has been discontinued.

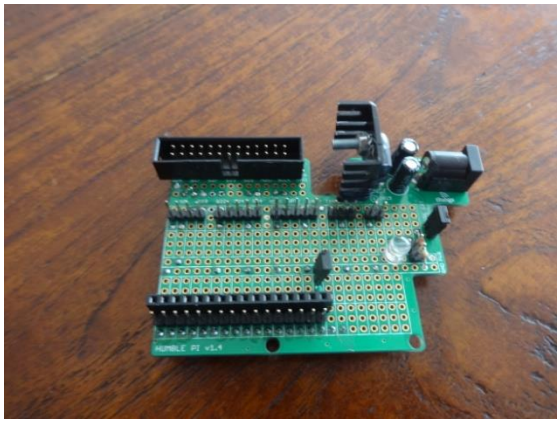


Figure 65 Ciseco Humble PI I2C interface board

Figure 66 The I2C backpack 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.

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

Construction using the Adafruit 3.5-inch TFT

Note: There are two types of TFT touch-screen available screen available from Adafruit namely Capacitive or Resistive. The one used in this project was the 3.5-inch Resistive type.

The 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. Below is the easy installation script from the manual. The supported resolution is 720 x 480 pixels. Smaller resolutions are not supported.

```
$ wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-
Scripts/master/adafruit-pitft.sh
$ chmod +x adafruit-pitft.sh
$ sudo ./adafruit-pitft.sh
```

The TFT screen should be calibrated using the **evtest** program

```
$ sudo apt-get install evtest tslib libts-bin
$ sudo evtest /dev/input/touchscreen
```

See the Adafruit manual for the full procedure.

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).

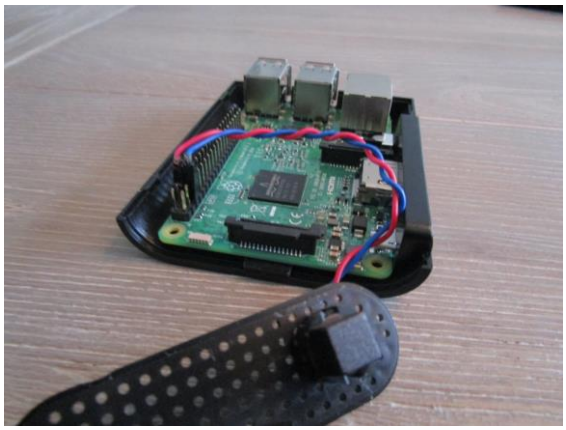


Figure 67 Wake-up button

On the left is the Raspberry Pi unit which is running the radio on a TV with HDMI inputs as shown in Figure 2 on page 4. 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: 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 47.

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 11 IR Sensor Pin outs* on page 102 for the GPIO pin connection.

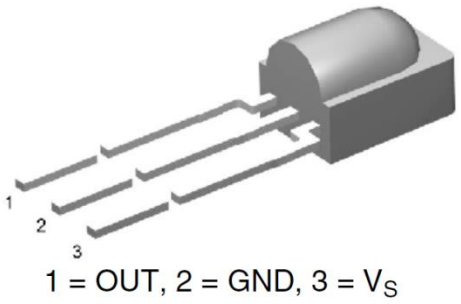


Figure 68 TSOP38238 IR sensor

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 11 IR Sensor Pin outs* page 102.
** Caution; Do not accidently 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:

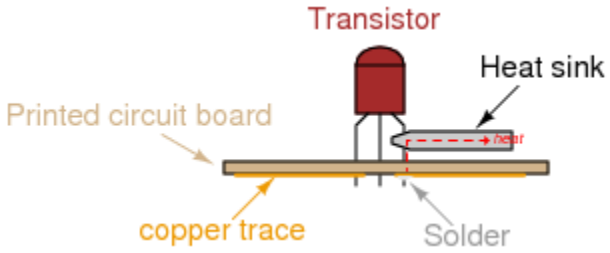


Figure 69 Soldering precautions

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 102.

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.

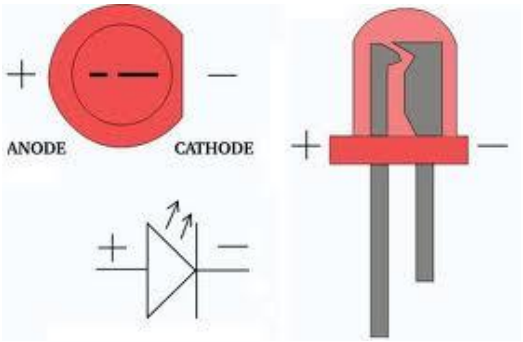


Figure 70 LED polarity

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 8 Remote Control Activity LED

Radio Type	Pin	GPIO	Type of Raspberry PI
Activity LED not fitted	none	n/a	Not applicable
Two or Four line LCD with Push Buttons	23	**11**	26 or 40 Pin version
Two or Four line LCD with Rotary encoders	23	**11**	26 or 40 Pin version
Two or Four line LCD with I2C backpack	23	**11**	26 or 40 Pin version
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 126.

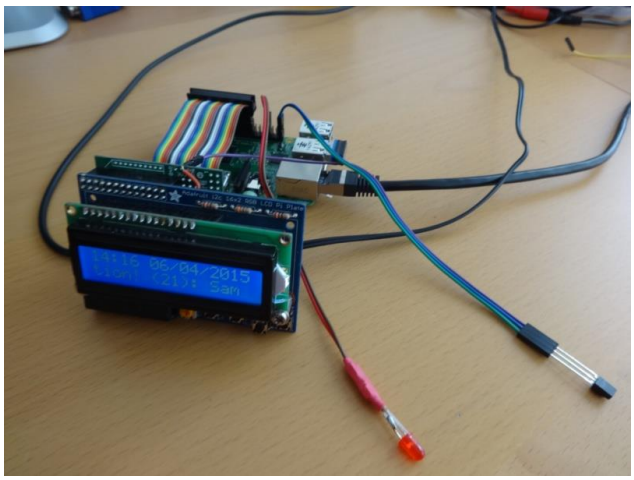


Figure 71 Adafruit plate with IR sensor and activity LED.

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/

From version 6.6 onwards 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 72 IQaudIO Cosmic controller and OLED display

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

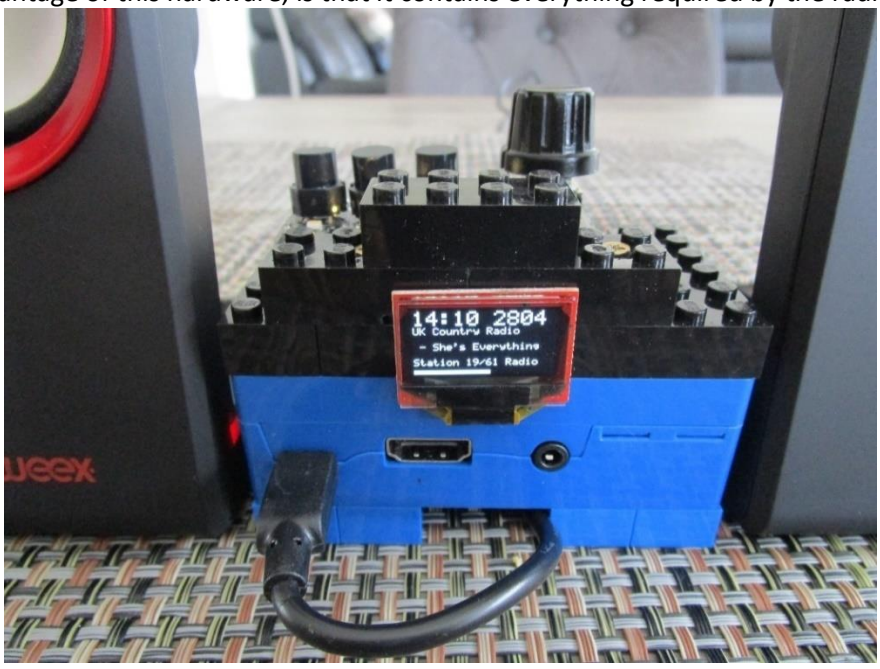


Figure 73 Lego radio with IQaudIO Cosmic controller and OLED

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 page 127.

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 74 PiFace CAD and Rasberry PI

Figure 75 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 87 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 43.

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 website: 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 102.

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

```
$ sudo apt-get 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-get update... done.
Installing apt-transport-https... done.
Installing /etc/apt/sources.list.d/flirc_fury.list...done.
Running apt-get 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 135.

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 21 Sound card Device Tree overlays* on page 255.

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

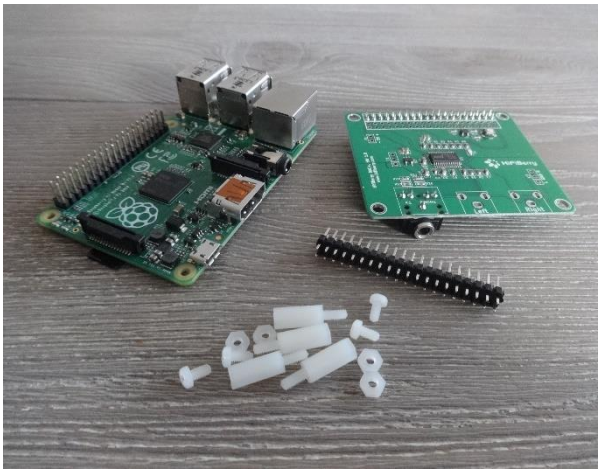


Figure 76 HiFiBerry DAC Plus

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

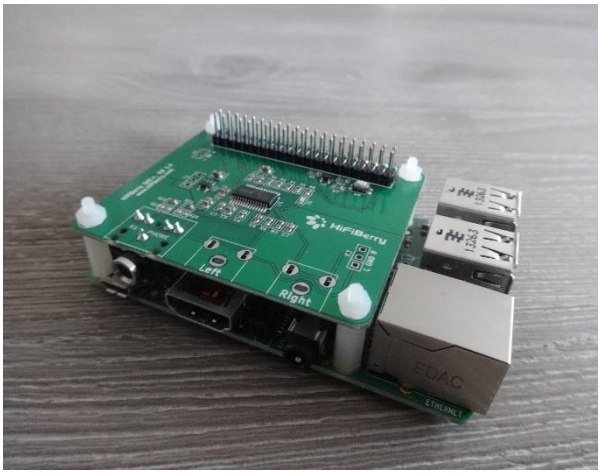


Figure 77 HiFiBerry mounted on the Rasberry Pi

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

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

IQaudIO 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

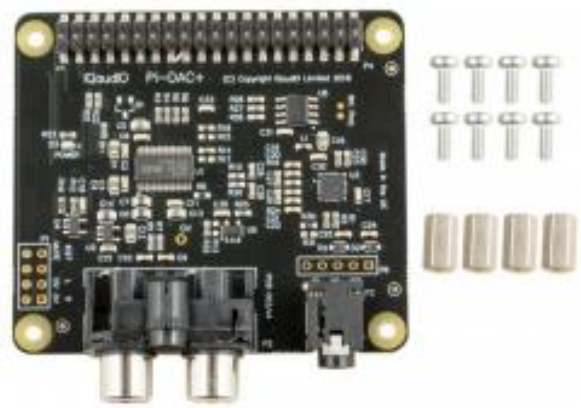


Figure 78 IQaudIO DAC plus

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 79 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 5 Radio and DAC* devices 40 pin wiring and NOT the 26 pin wiring version shown in Table 4. The **/etc/radiod.conf** configuration file must also be configured to support these devices by running the audio configuration program.

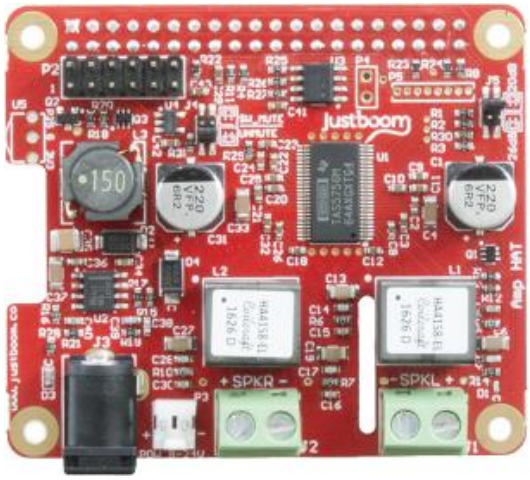


Figure 80 JustBoom Amp HAT

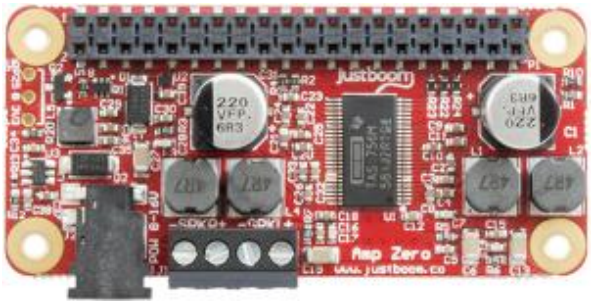


Figure 81 JustBoom Amp Zero pHAT

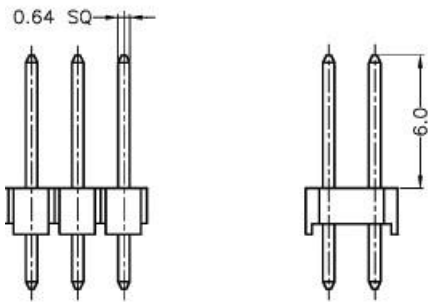


Figure 82 JustBoom Zero stacker requirements

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 83 Using the 40-pin stacker

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.

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

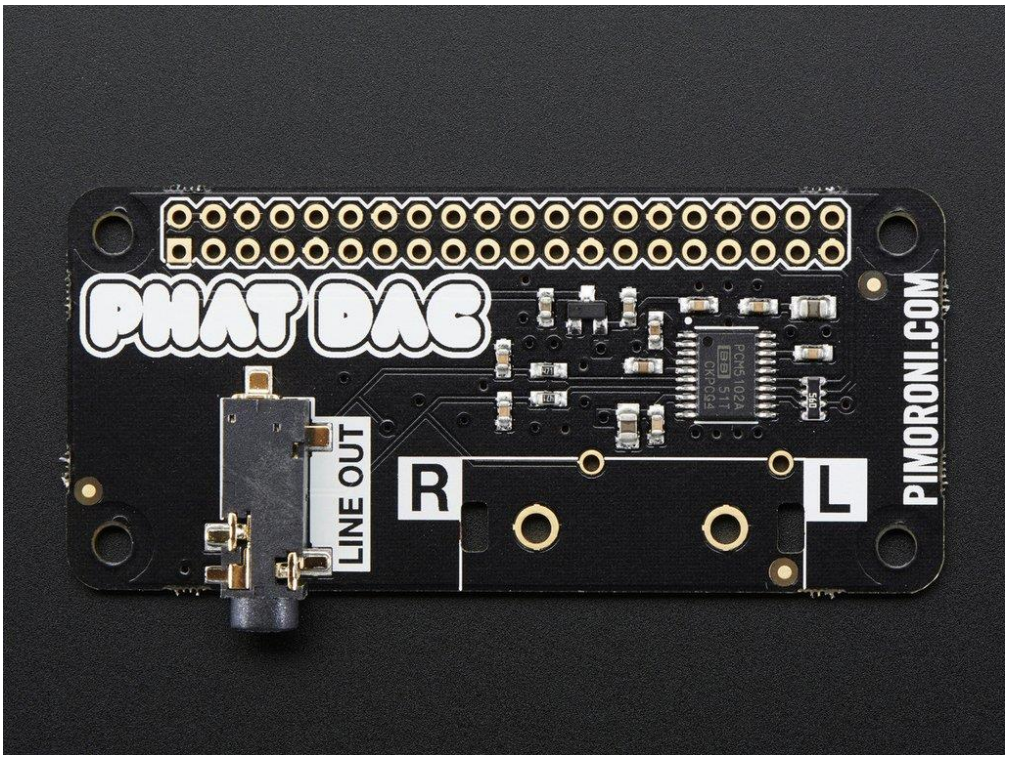


Figure 84 Pimoroni pHat DAC

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 HiFiberry 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 74. The Pimoroni pHat DAC will run fine without any mixer controls.

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 85 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 86 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 centimetres.

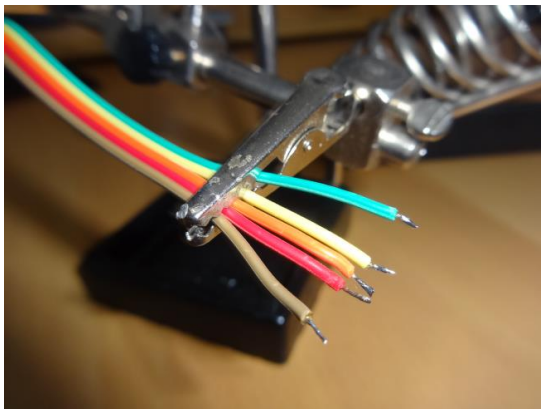


Figure 87 Tinning the wires with solder

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.

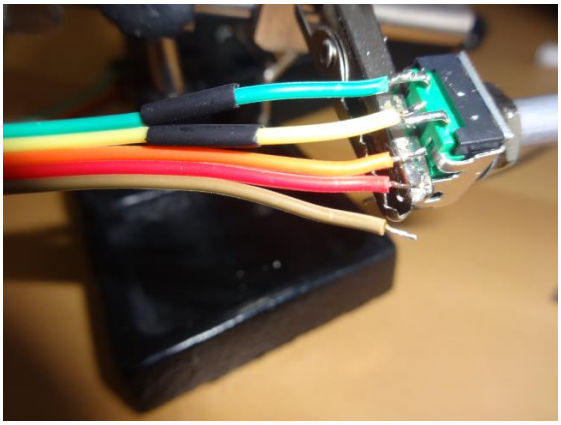


Figure 88 Soldering up the switch

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.

Figure 89 Shrink shrink-wrap with a hair dryer

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.

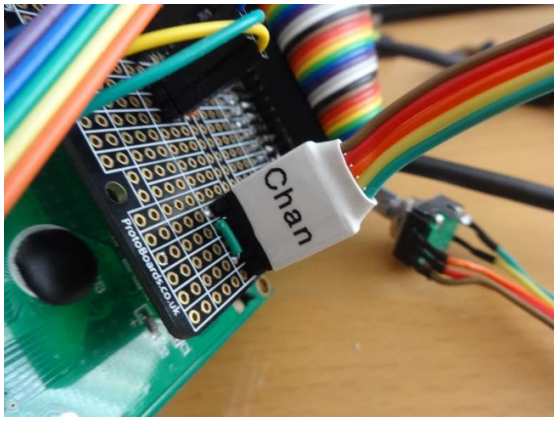


Figure 90 Connecting the rotary encoder an interface board

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.

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 91 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 92 Loop +5V supply around the core

Fit a mains filter

Figure 93 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 94 Integrated mains socket and filter

Use an I2C LCD backback

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 39 for further information.

Preventing ground loops

Figure 95 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 Rasberry Pi or sound card and the amplifier input.

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

Connecting up a USB power adapter

Figure 96 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 97 below. Even with a heat sink good ventilation is necessary.

Figure 97 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 using a heat sink first.

Figure 98 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

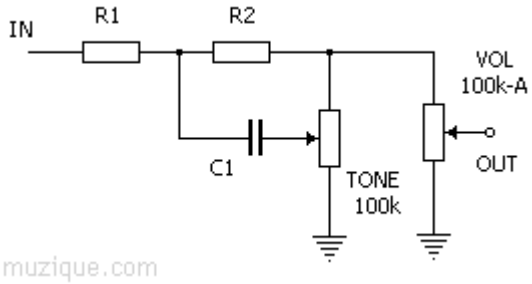


Figure 99 Simple tone control circuit

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

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

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.

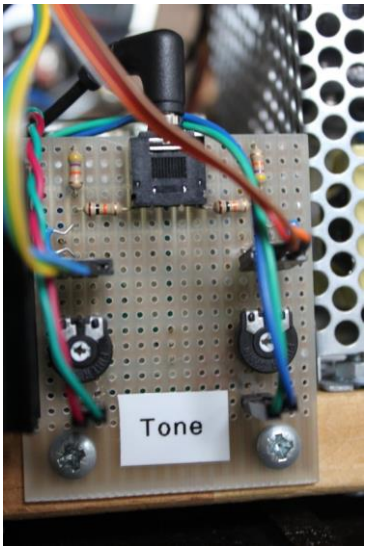


Figure 100 Tone control board

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 9 Adafruit backlit RGB display wiring

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

Figure 101 IN4148 diode

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

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

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

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

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

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

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

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

END OF EXAMPLE COMMANDS.

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/game
s:/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.

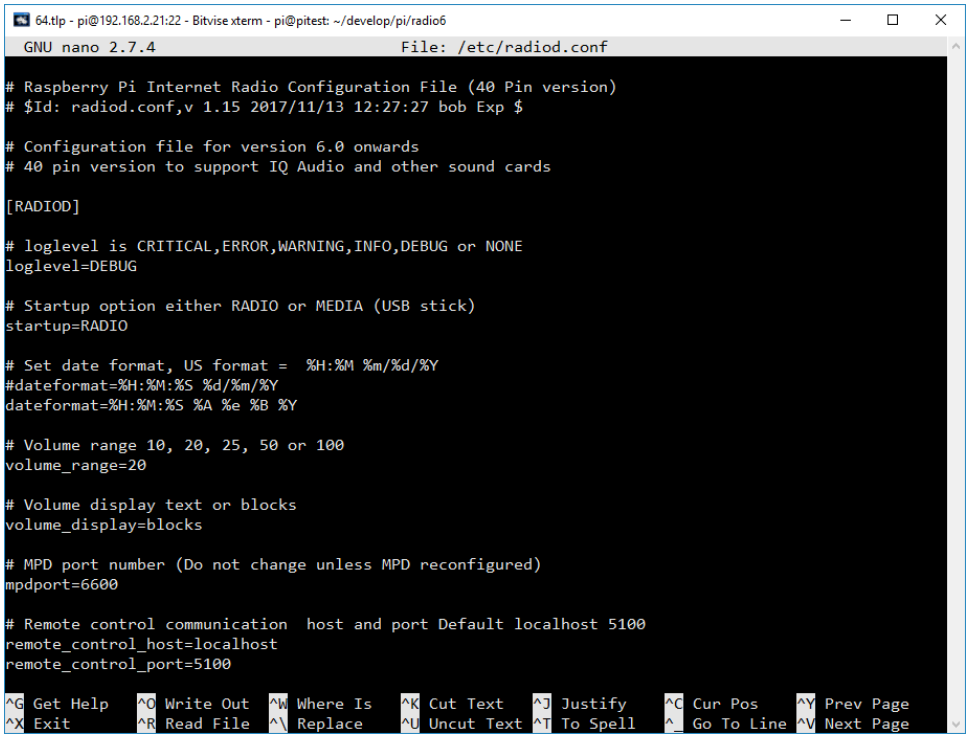


Figure 102 The nano file editor

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

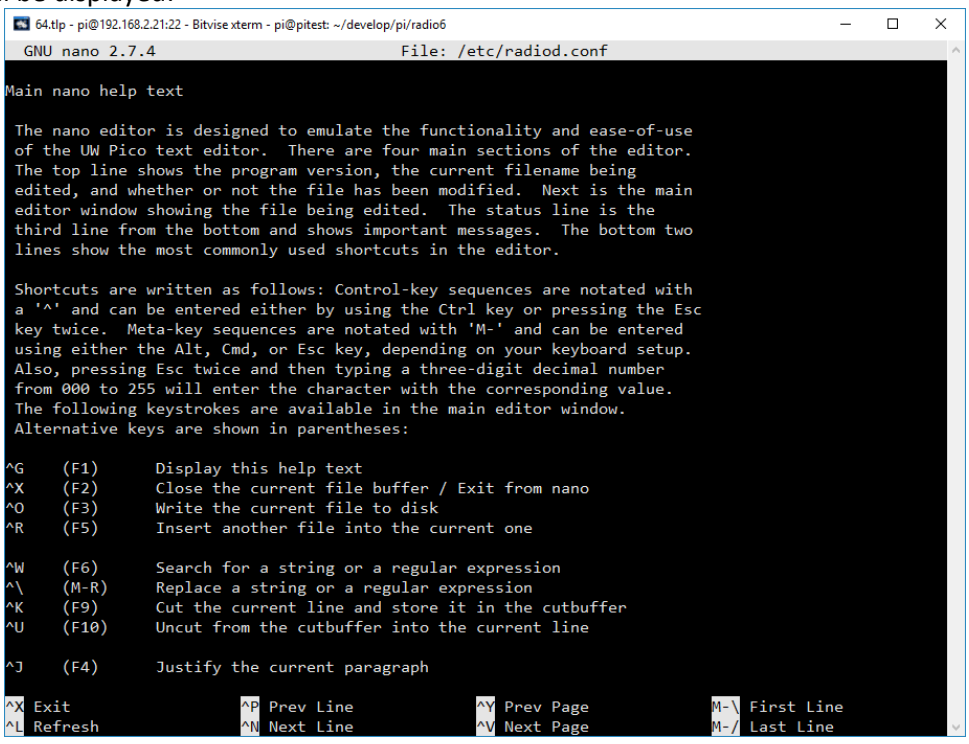


Figure 103 The nano editor help screen

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

There is a cheat sheet in *Appendix B – Cheat sheet* which contains a list of installation instructions. A lot of very useful Raspberry Pi documentation will be found at:

https://www.raspberrypi.org/documentation

The latest version of Raspbian is called Buster. Create a new SD card with **Buster** or **Buster Lite**. There is a "Full" version of Buster however this is unnecessary for this project.

Note: The touch-screen or HDMI TV version of the software requires a desktop version of the operating system so use **Raspbian Buster** and not the **Lite** version. Although it is possible to upgrade Lite to use a desktop, it is beyond the scope of this manual. Only use **Lite** for LCD versions of the radio.

Note: The Raspberry Pi 4B released in June 2019 needs Raspbian Buster from June 2019 or later. The Pi 4B will not boot with earlier versions of the operating system.

Raspbian Buster download (recommended)
http://www.raspberrypi.org/downloads

SD card creation

Use at least an 8 Gigabyte Card for Buster Lite or 16 Gigabyte for Buster Desktop/Full. Create an SD card running the latest version of **Raspbian Buster** or **Buster Lite**. See the *Image Installation Guides* on the above site for instructions on how to install the **Raspbian** operating system software.

Log into the system

Boot up the Raspberry Pi with the new SD card with the **Debian Buster** operating system. If you have used **Buster** 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. With **Buster 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).

Using SSH to log into the Raspberry PI

If using a PC or MAC it is possible to use SSH (Secure Shell) to log into the Raspberry Pi. You will need to install either **Putty** or **Bitvise** on a PC to provide an SSH client. However, the latest version of **Buster** requires SSH to be enabled first. Due to increasing security concerns SSH is disabled by default on Raspbian images. See https://www.raspberrypi.org/blog/a-security-update-for-raspbian-pixel/. There are two ways to enable SSH:

1. Add a file called **ssh** to the boot sector of the SD card
2. Use the **raspi-config** program to enable SSH.

Add a file called ssh to boot sector

The boot partition on a Pi should be accessible from any machine with an SD card reader, on Windows, Mac, or Linux. If you want to enable SSH, all you need to do is to put a file called **ssh** in the **/boot** directory. The contents of the file don't matter: it can contain any text you like, or even nothing at all. When the Pi boots, it looks for this file; if it finds it, it enables SSH and then deletes the file. Insert the SD card into the SD card reader. On a Windows PC run the **cmd** program by typing

"cmd" in the Windows search box. Now change to the drive letter where the SD card is. In the following example this is D: Type "D:" then type "echo 0 > ssh".

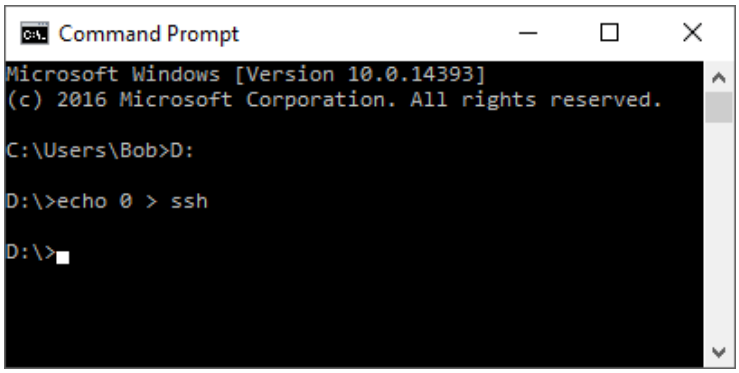


Figure 104 Enabling SSH on the boot sector

Boot the Raspberry Pi with this SD card.

Enabling ssh in raspi-config

Using a keyboard and HDMI screen connected to the Raspberry Pi log into the system as user pi.

After logging in run **raspi-config**. Select "Advanced options" and select option A4 to enable SSH.

```
$ sudo raspi-config
```

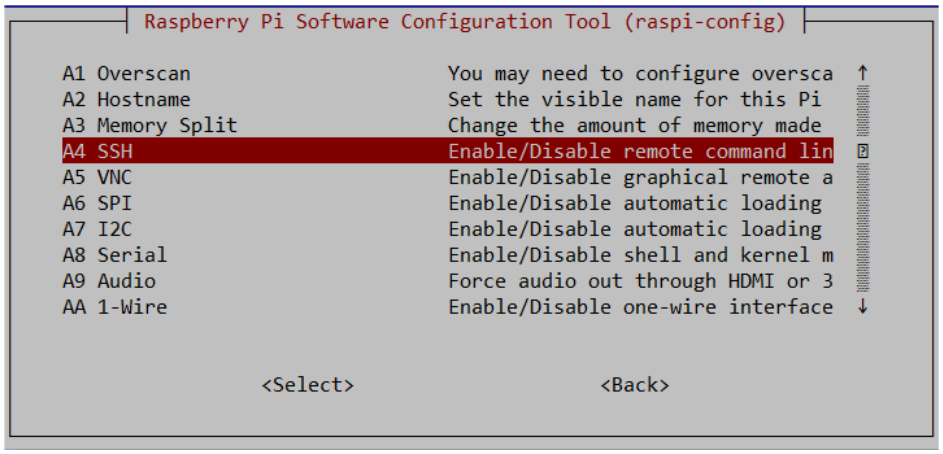


Figure 105 Enabling SSH

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

```
$ sudo reboot
```

After logging back in the following message is displayed.

```
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 219 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-get update
```

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

```
$ sudo apt-get upgrade
```

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

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

Run the following command

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

If you have an older version of the Raspberry Pi then update the firmware.

```
$ sudo rpi-update
```

Reboot the Raspberry Pi.

```
$ sudo reboot
```

Important: After upgrading the system the repository locations may no longer be valid. Re-run **apt-get 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-get 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 versions of the Radio and takes a lot of processing power. It is enabled by default in **Buster** but is not installed with **Buster Lite**. If you are not planning to use the touch-screen or HDMI version of the radio then disable it. Select option **3 Boot options**

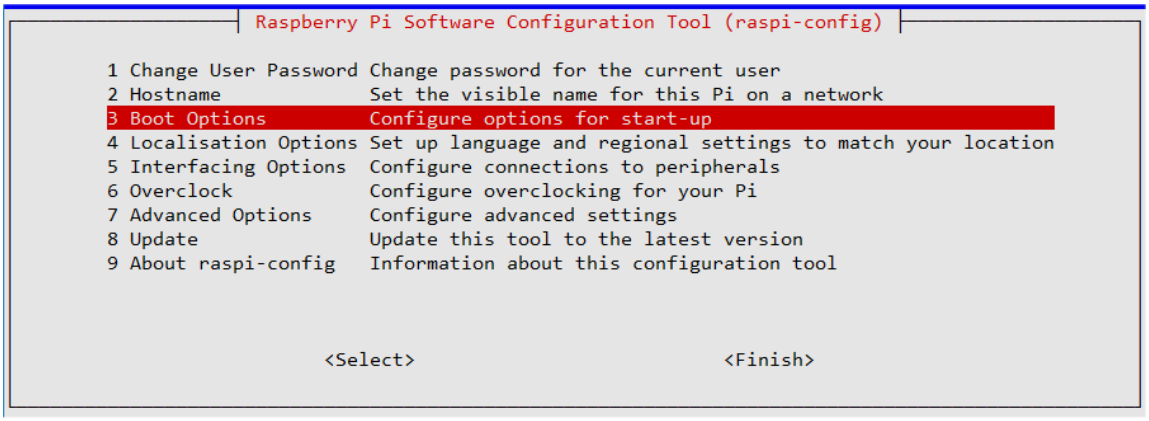


Figure 106 Disabling the graphical desktop

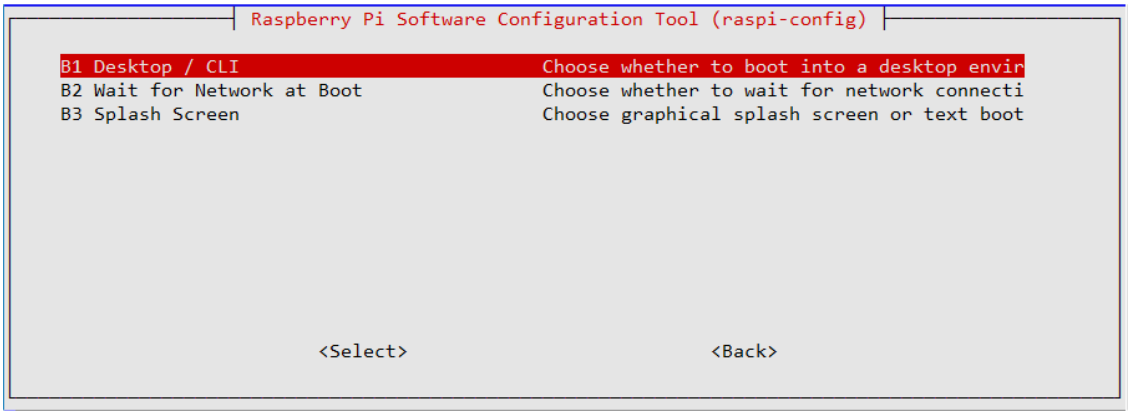


Figure 107 Desktop enable/disable selection

Select option **B1** Desktop/CLI.

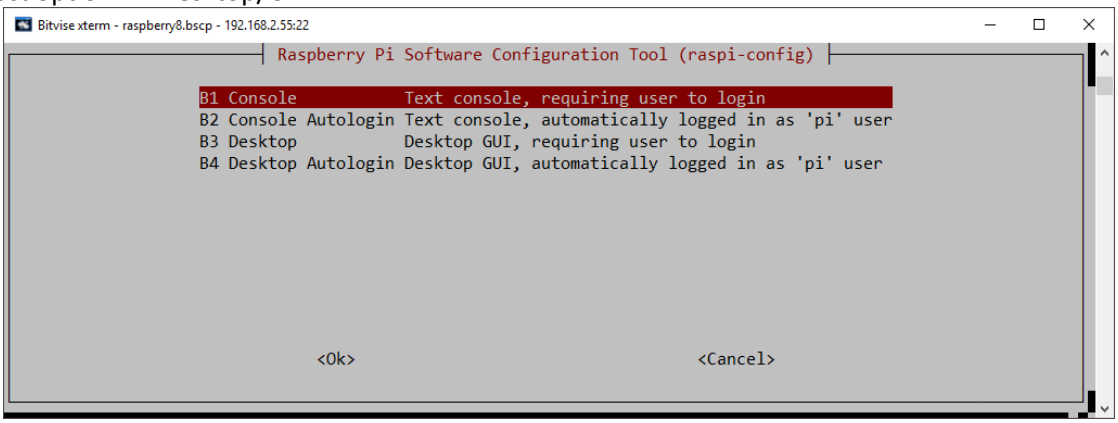


Figure 108 Console login selection

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

Setting the time zone

The **Raspbian Buster** operating system is usually set to UK time. The easiest way to set the time zone for your country if you are in a different time zone is to use the **raspi-config** program and select "Localisation Options":

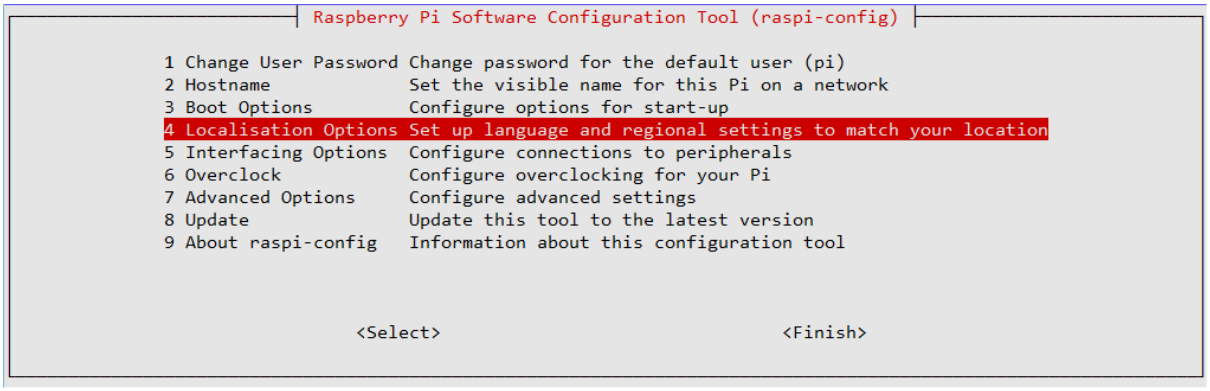


Figure 109 Setting the time zone

Select option 4 "Localisation Options", Use the tab key to move to <Select> and press enter.
The above screen is using the Bitvise SSH client program (See Putty on the web).

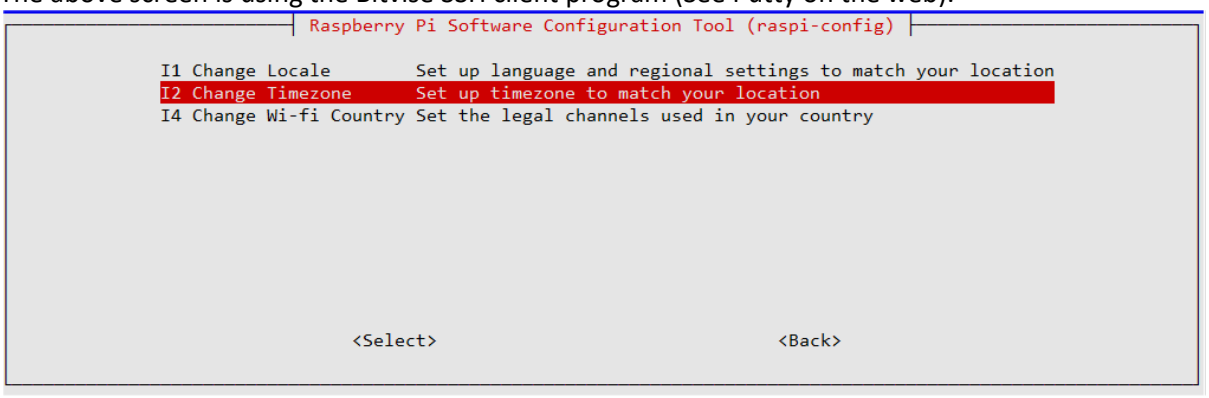


Figure 110 Selecting the time zone

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

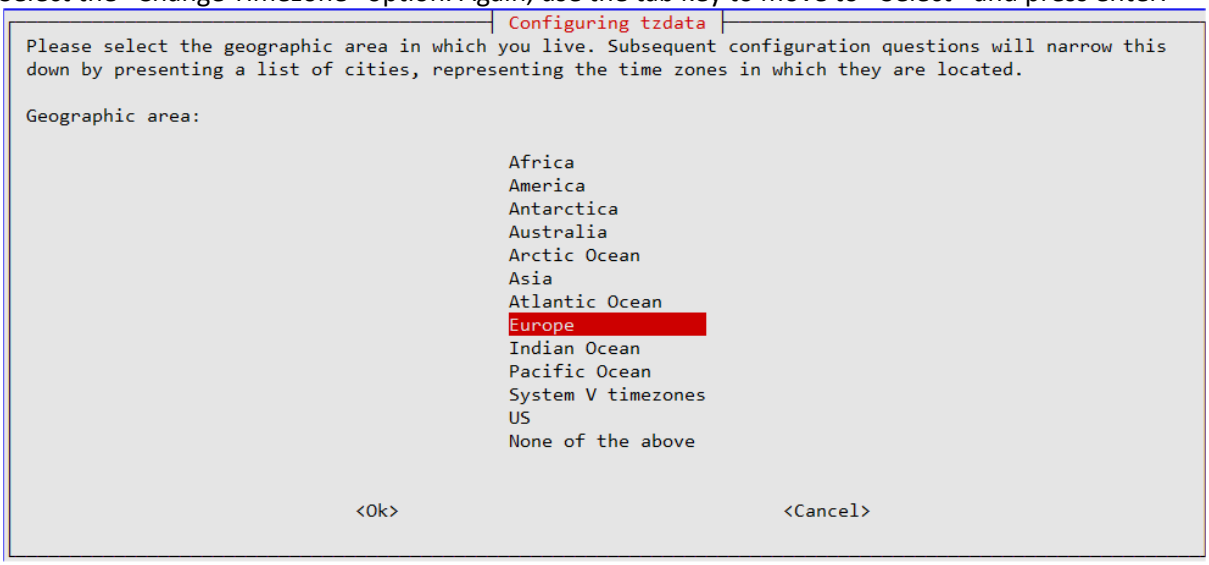


Figure 111 Saving the timezone

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

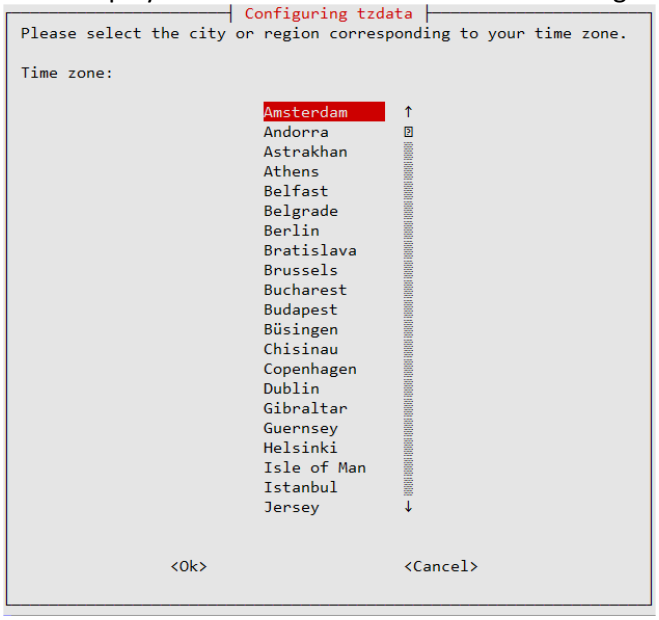


Figure 112 Time zone country selection

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

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

Both the password and hostname can be changed using the **raspi-config** program.

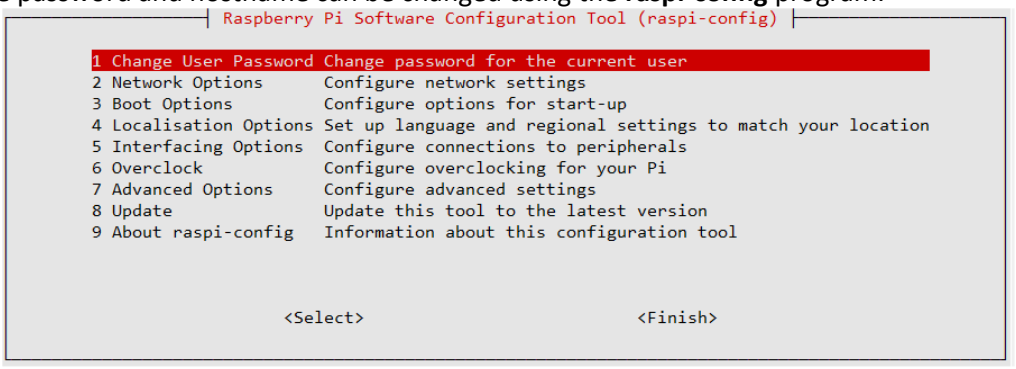


Figure 113 Changing the Raspberry PI password

Option 1 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 2 Network Options:

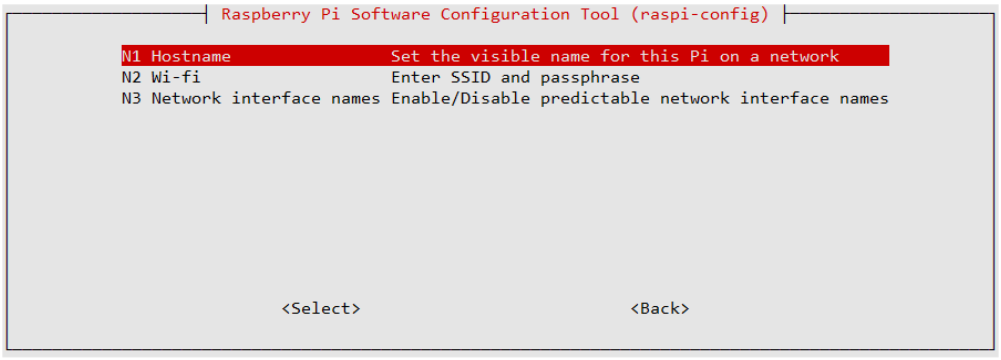


Figure 114 Changing the hostname

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

As the password is entered nothing is displayed. This is normal. You will be asked if you wish to reboot the system. After you reboot the system you will see the new hostname at the login prompt.

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

In the above example the new hostname is **piradio**. Once the hostname has been changed the program will ask if you wish to reboot. Answer "yes" to reboot.

Configuring the Wi-fi Connection

You will be asked to select the country you are in for legal WiFi channel selection. Select option N2:

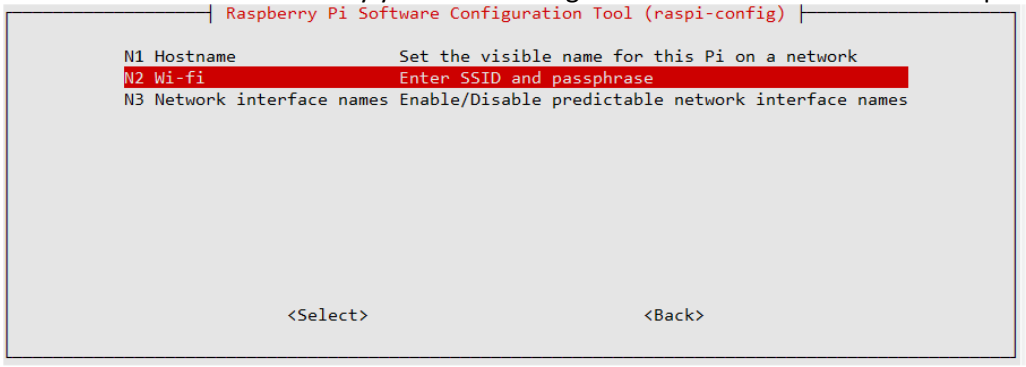


Figure 115 Setting up the Wi-Fi in raspi-config

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 *Configuring a wireless adaptor* on page 112.

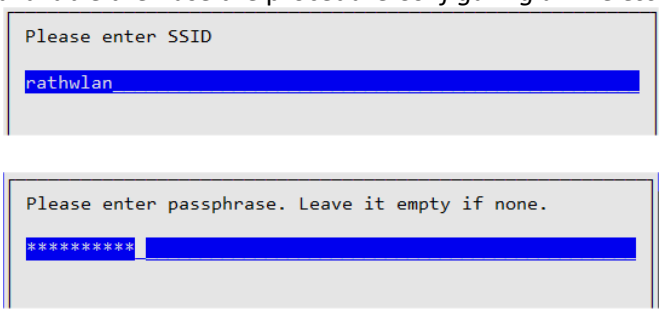


Figure 116 Entering Wi-Fi credentials

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 **rasp-config**. Select localisation options and select I4 to set Wi-Fi country:

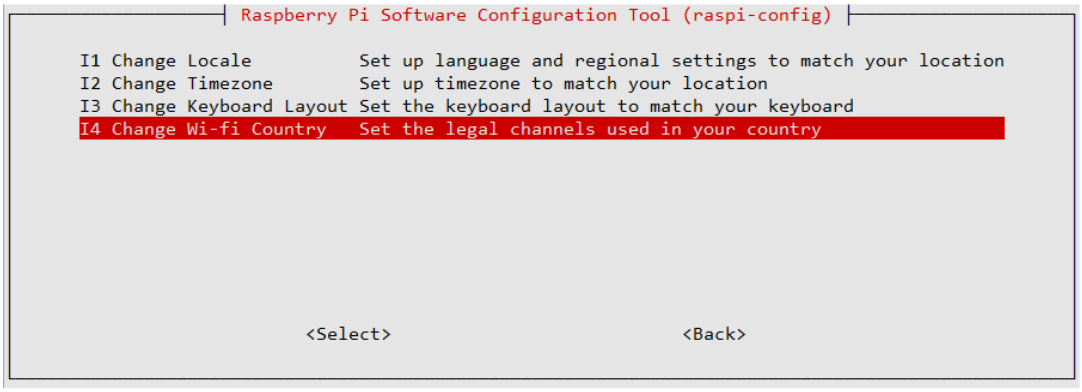


Figure 117 Setting up the Wi-fi country

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.

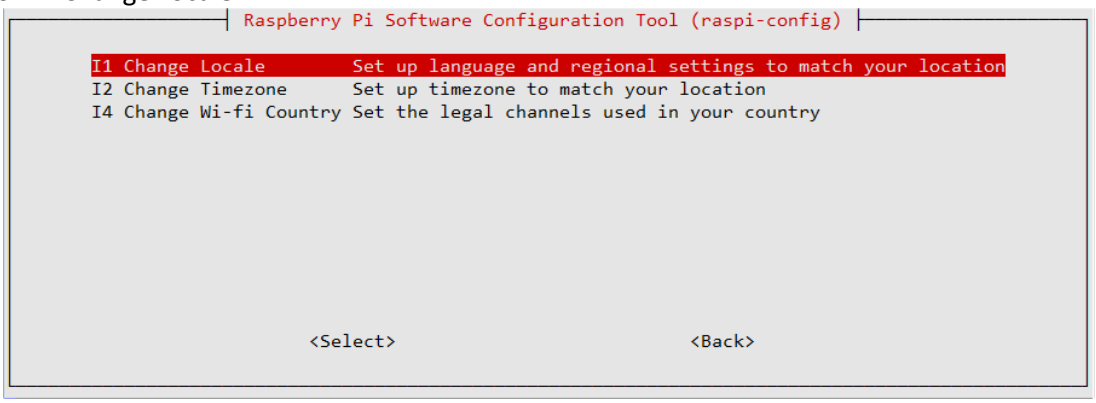


Figure 118 Selecting change locale

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).

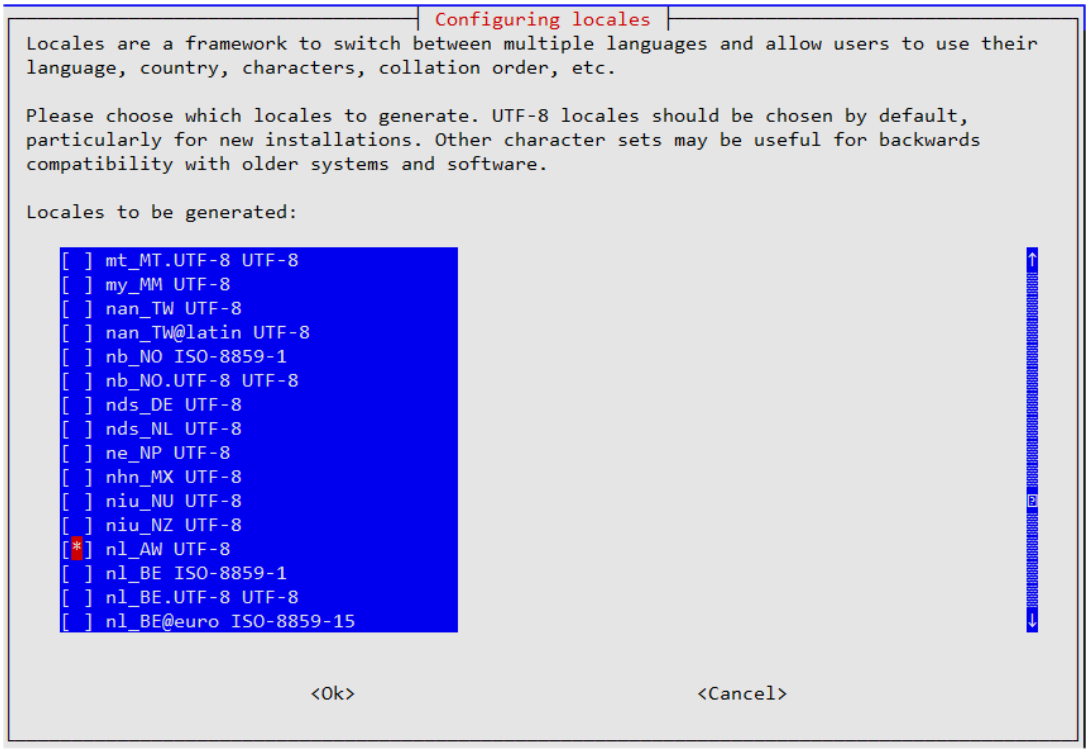


Figure 119 Generating the locale.

Warning: Do not be tempted to generate “All locales” as this is not necessary 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:

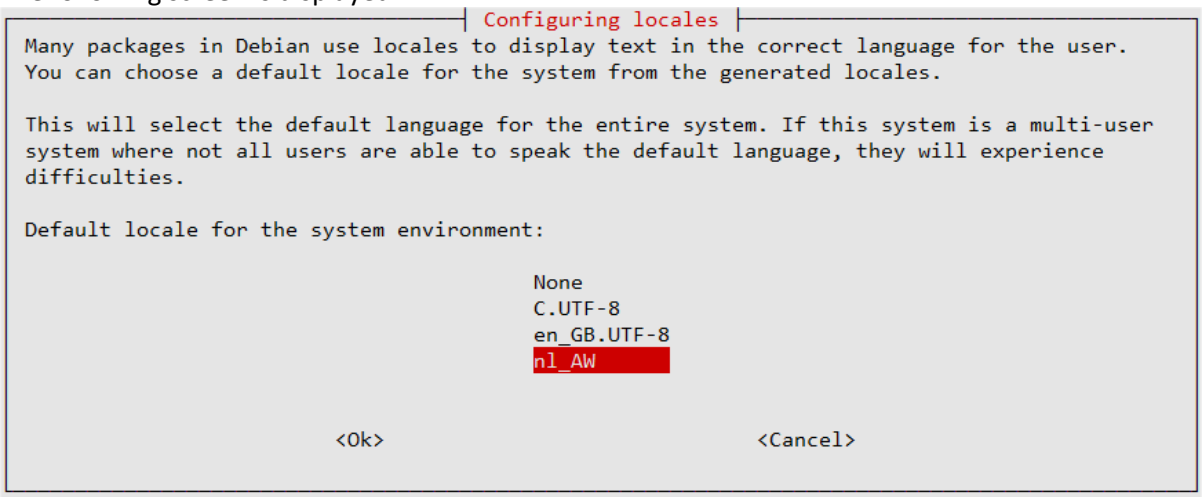


Figure 120 Selecting the locale

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

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:

```
$ sudo apt-get 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-get update** command to update the library list and retry installing **ffmpeg**.

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

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

```
$ man ffmpeg
```

Install python-requests

If you are running **Buster Lite** it is necessary to install **python-requests** for the Shoutcast program as this is not loaded by default. This step should not be necessary if using the **Buster** desktop operating system.

```
$ sudo apt-get install python-requests
```

Install libraries for the Olimex OLED

If using the IQaudIO Cosmic controller and 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-get install libffi-dev
$ sudo apt-get install build-essential libi2c-dev i2c-tools python-dev
```

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

```
$ sudo apt-get install python-pil
```

Install the anacron package

The **anacron** package is required to run the **/etc/cron.weekly/radiod** script. This script runs the **create_stations.py** program to update the playlists on a weekly basis. This is done so that any bad radio streams are removed from the radio playlists that might otherwise cause a problem.

Install it with the following command.

```
$ sudo apt-get install anacron
```

Install the Scratch package

This section is only applicable if the HDMI/Touchscreen version of the radio is to be used. 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 version of Raspbian Desktop it isn't.

Scratch is not actually used by the radio software but the graphic files in the **/usr/share/scratch/Media/Backgrounds** directory are used for the background wallpaper of the full feature graphical radio.

The background wallpaper is set by the following parameter in **/etc/radiod.conf** and can be amended to use a different background.

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

To install scratch run the following:

```
$ sudo apt-get install scratch
```

Chapter 6 - Installing the radio Software

Music Player Daemon Installation

If you haven't already done so upgrade the system packages as shown in *Update to the latest the packages* on page 65.

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

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

Answer yes 'y' when asked to continue.
If the installation says it cannot find the above packages then re-run the update command to update the library list and retry installing.

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

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

Note: 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.

Installing pulseaudio

The **pulseaudio** package may or may not 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**. 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 10 PulseAudio installation options

Type of radio installation	Install pulseaudio
Using espeak	No
Pimoroni Pirate Radio with pHat BEAT	Yes (Installed by phatbeat)
Adafruit speaker bonnet	Yes
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	No

The radio installation program will configure the Music Player Daemon to use **pulseaudio** instead of **alsa** if **pulseaudio** is installed.

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

To install **pulseaudio**:

```
$ sudo apt-get install pulseaudio
```

To remove **pulseaudio**:

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

Also remove unwanted libraries.

```
$ sudo apt autoremove
```

Install the Radio Daemon

Note: If you are installing the software for a Pimoroni Pirate radio with pHat beat it is necessary to first install the Pimoroni software as show in the section *called Installing Pimoroni Pirate Radio (pHat BEAT)* on page 88.

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:

```
$ wget http://www.bobrathbone.com/raspberrypi/packages/radiod_6.12_armhf.deb
```

Run **dpkg** to install the package.

```
$ sudo dpkg -i radiod_6.12_armhf.deb
```

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

```
(Reading database ... 131542 files and directories currently
installed.Preparing to unpack radiod_6.12_armhf.deb ...
Raspberry PI internet radio installation
Stopping radiod service
Unpacking radiod (6.12)
```

Configuring the radio

Once that is done the installation will run the **configure_radio.sh** script. This update 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 124.

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.

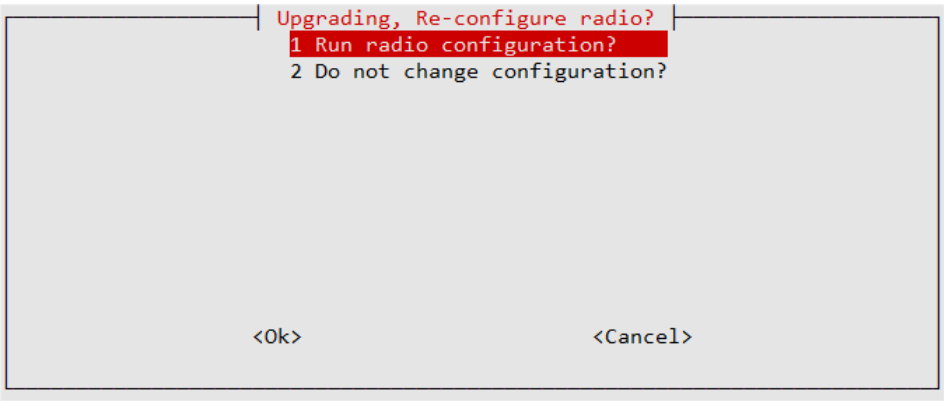


Figure 121 Configure radio – Upgrade

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:

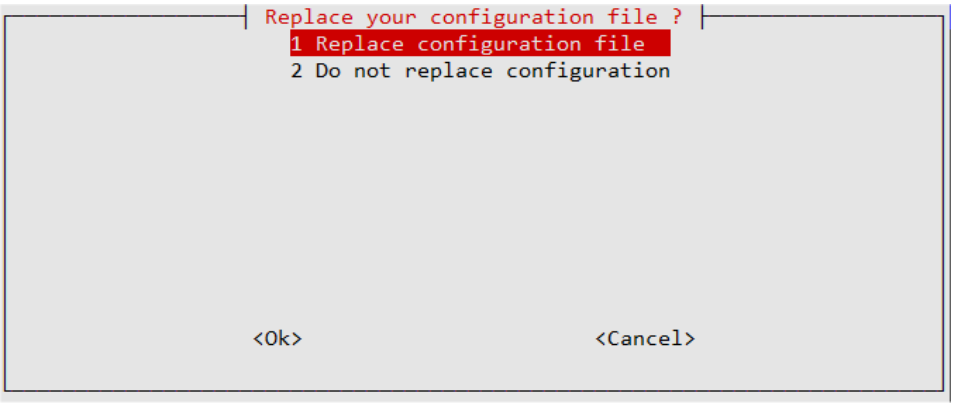


Figure 122 Replace configuration file

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:

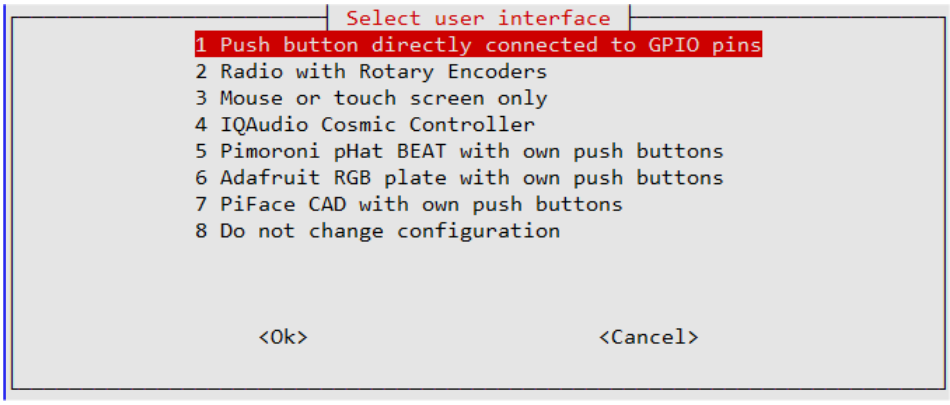


Figure 123 Configure radio - User interface selection

Even if you are using a touch screen or HDMI display you still can select option 1 or 2. Only select option 3 if you are only going to use a HDMI or touch screen display with no buttons or rotary

encoders and with or without a mouse. pHat BEAT and PiFace CAD come with their own push-buttons so select option 5 or 6 respectively.

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. This screen and all following screens have the option to not change the configuration. The program will then ask you

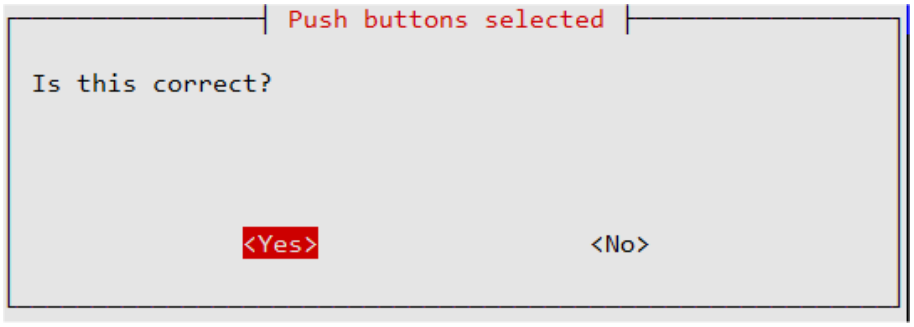


Figure 124 Configure radio - Confirmation screen

If the push-button interface is selected, the program will ask how they have been wired. As from version 6.9 push-buttons can be wired to either +3.3V (The original scheme) or GND(0V).

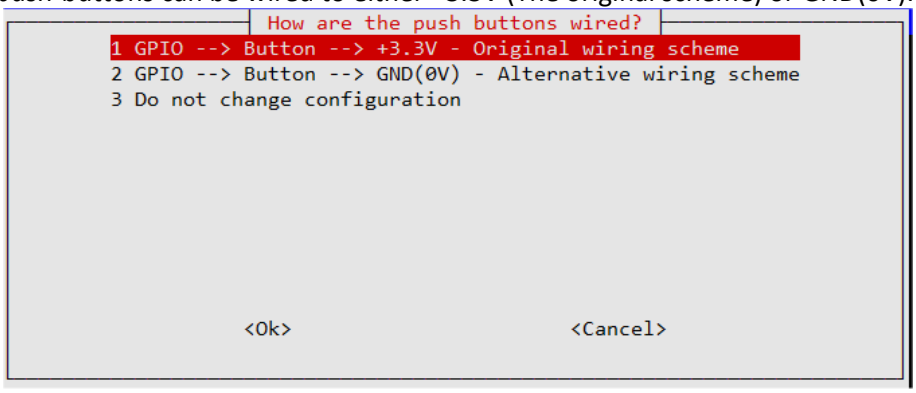


Figure 125 Push-button voltage selection

Select the correct voltage that the push-buttons have been wired to, either GND(0V) or +3.3V. The program will now ask which version of the wiring has been used during construction:

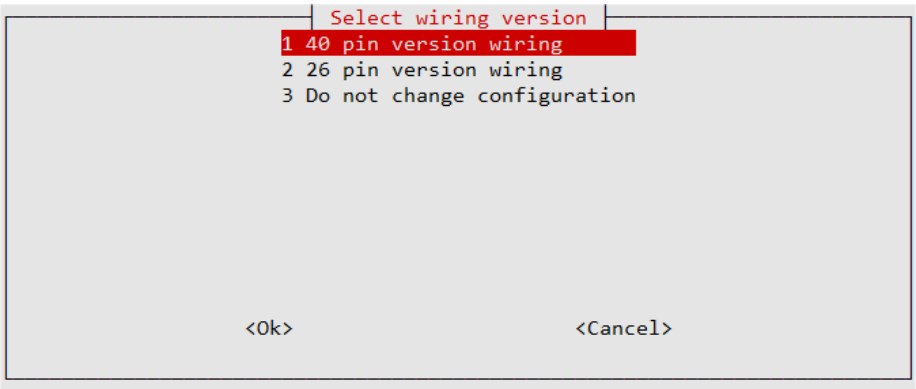


Figure 126 Configure radio - wiring selection

Normally select the 40-pin option unless you have used the 26-pin wiring scheme. See *Table 4 Controls and LCD wiring 26 pin version* and *Table 5 Radio and DAC devices 40 pin wiring.* Confirm selection and continue to the next screen:

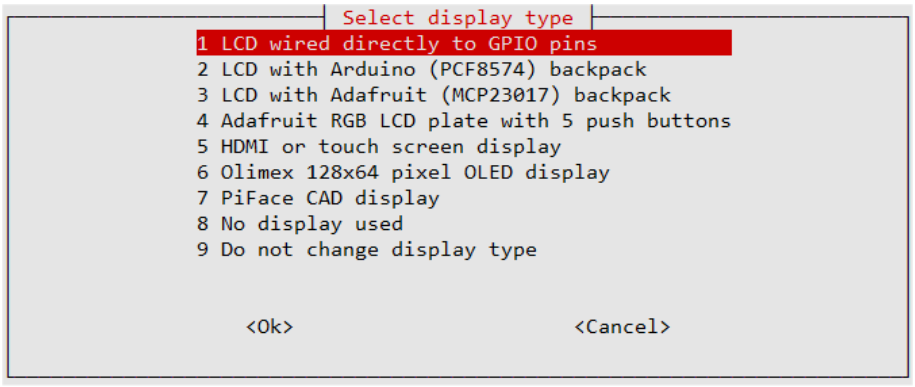


Figure 127 Configure radio - Display interface selection

Select the correct option for the display interface and confirm selection. Again, it is possible leave the configuration unchanged.

If option 5 – 'HDMI or touch screen display' was selected the go to the section called *Installing the HDMI or* touch screen on page 83.

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 82.

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

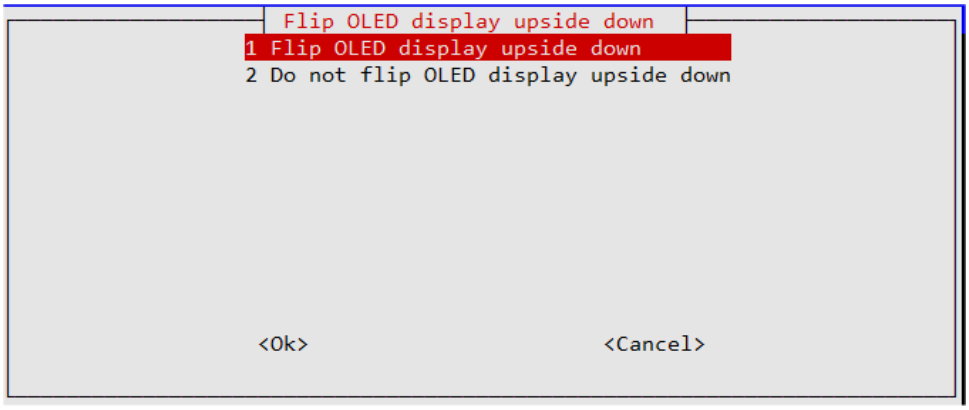


Figure 128 Olimex OLED flip display

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.

Configure SPI Kernel Module

Skip this section unless installing PiFace CAD.
If installing the radio on PiFace CAD it is necessary to enable the SPI interface. If PiFace CAD was selected the following message will be seen:

```
The PiFace CAD display 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:

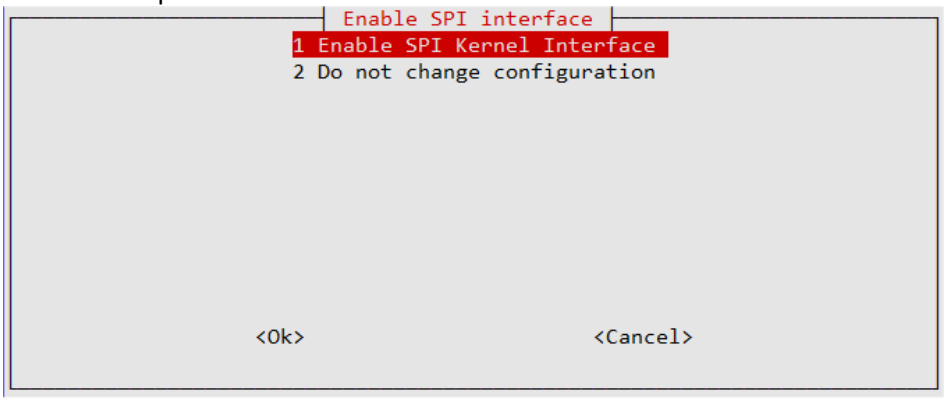


Figure 129 Enable SPI Kernel Module

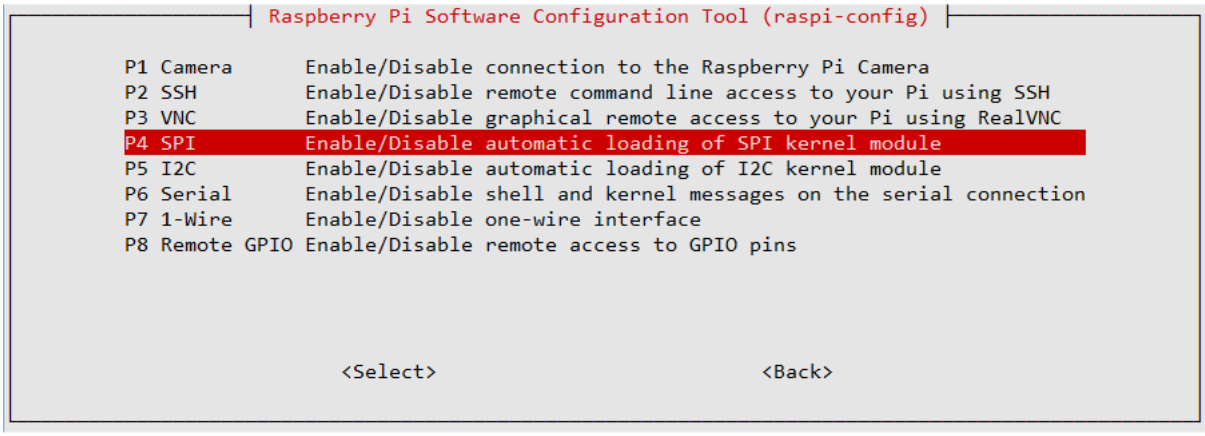
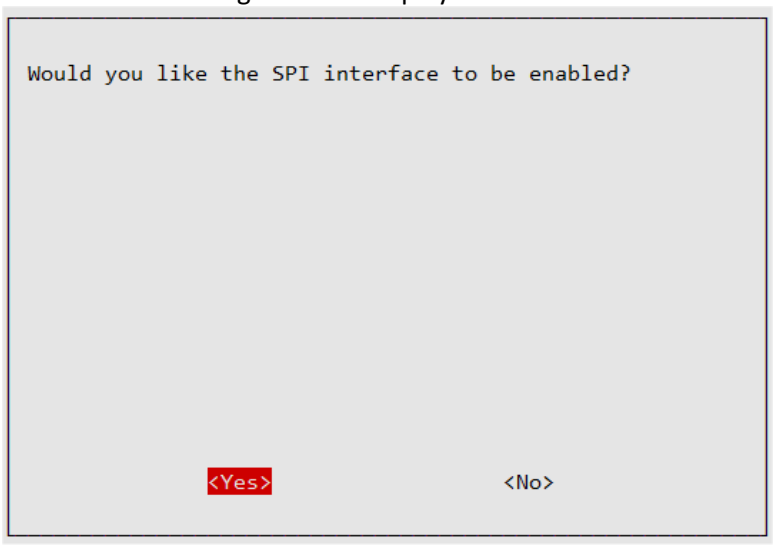


Figure 130 Enable SPI kernel module option

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



Select Yes to enable the SPI kernel module. Select "Finish" to exit.

Configure the I2C interface

Skip this section unless using a device which uses the I2C interface. If the OLED display or one of the I2C backpacks or Adafruit RGB plate was selected then the following screen will be seen:

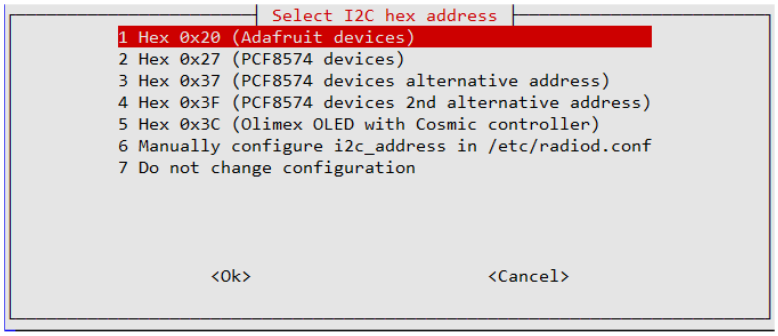


Figure 131 Configure radio - I2C interface hex address

At this stage you may not know what the Hex address for PCF8574 devices is so simply select option 1 or 2 depending upon whether you are using an Adafruit or Arduino backpack. These use different I2C integrated circuits namely MCP23017 (Adafruit) or PCF8574 (Arduino). For the OLED display select option 4. Once you have installed the I2C libraries (described later) you can run the **configure_radio.sh** program again to change to the correct hex address.

The following text will be displayed if I2C libraries are required

```
The selected display interface type requires the
I2C kernel libraries 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
   P5 Enable/Disable automatic loading of I2C kernel module

Press enter to continue:
```

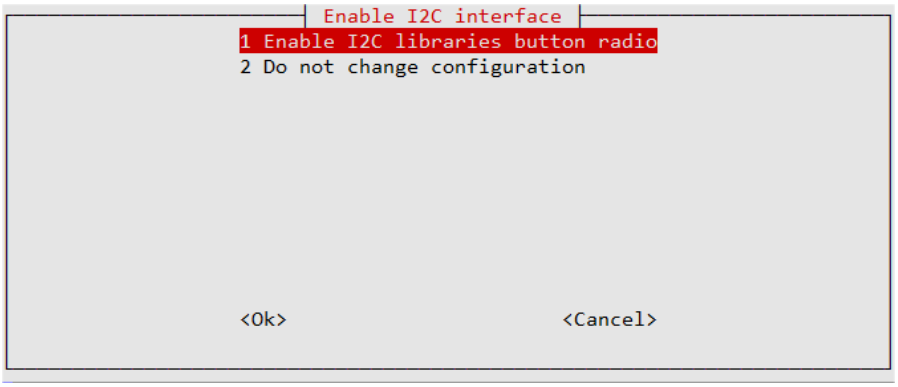


Figure 132 Configure radio - Enable I2C libraries

The **raspi-config** program now runs. Select 5 “Interfacing options” then option P5:

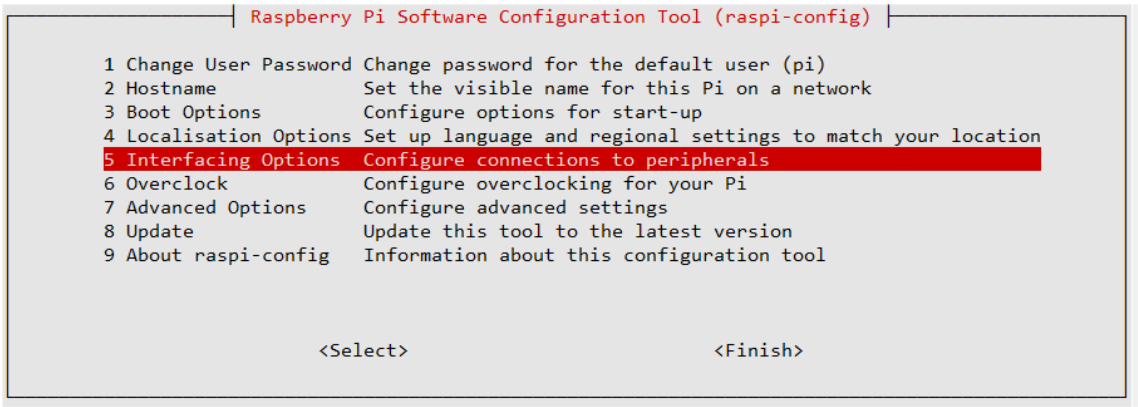


Figure 133 Selecting interfacing options in raspi-config

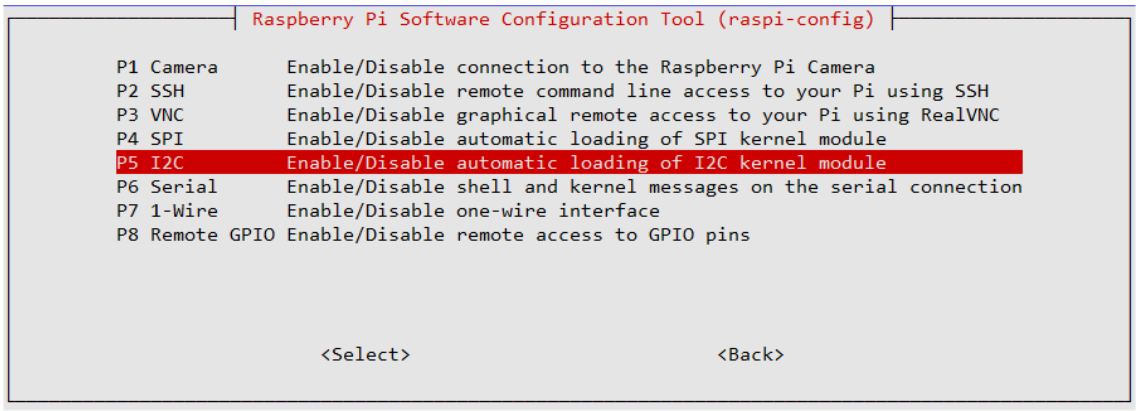
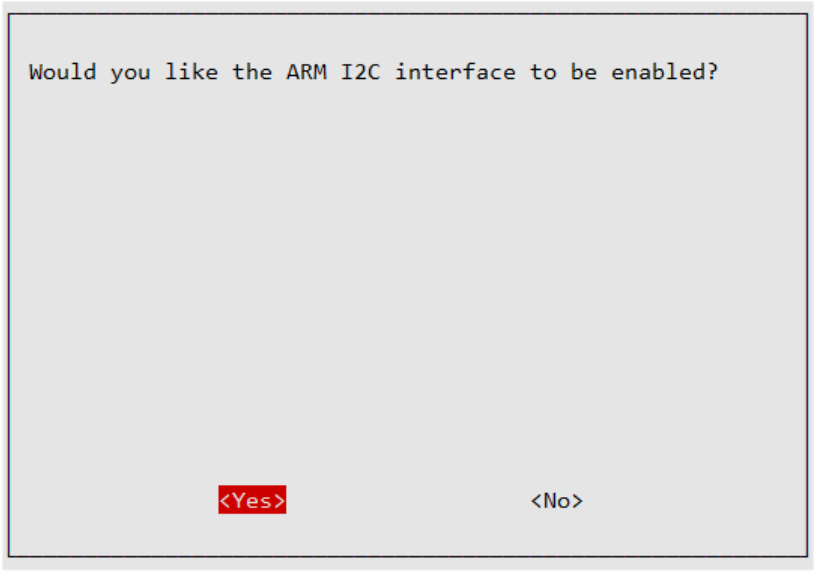


Figure 134 Select I2C libraries in raspi-config

Select - P5 I2C Enable/Disable automatic loading of I2C kernel module. The following screen will be displayed. Select yes to enable the ARM I2C interface.



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.

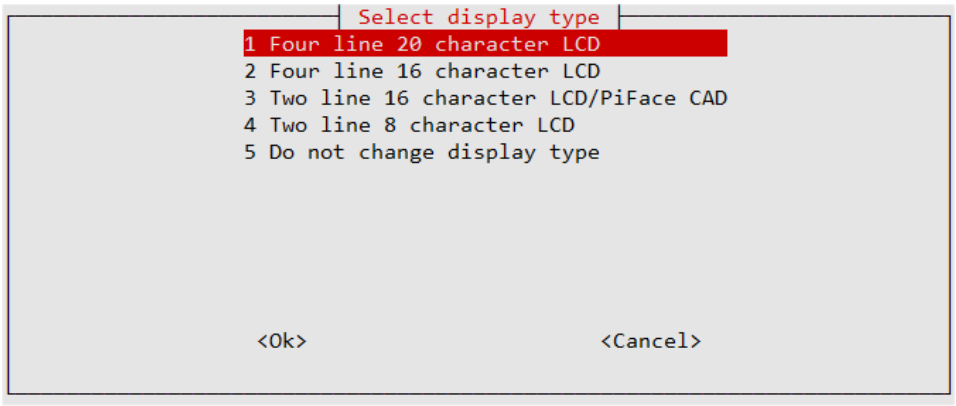


Figure 135 Configure radio - Display type selection

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

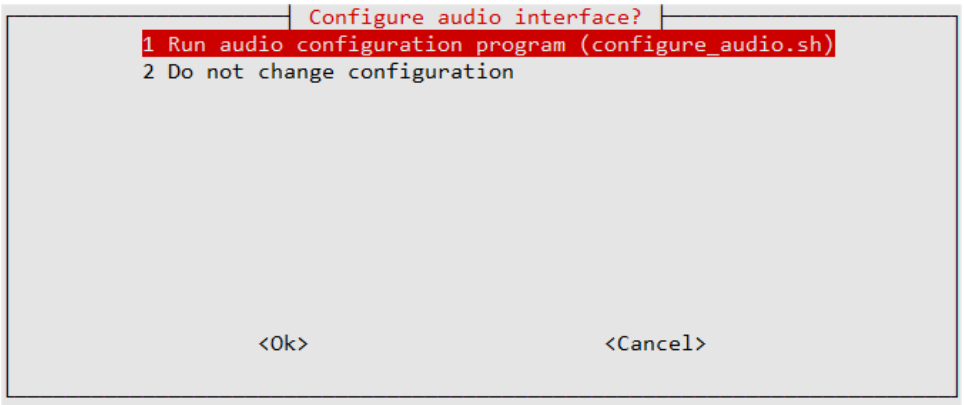


Figure 136 Configure radio audio output option

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 84.

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:

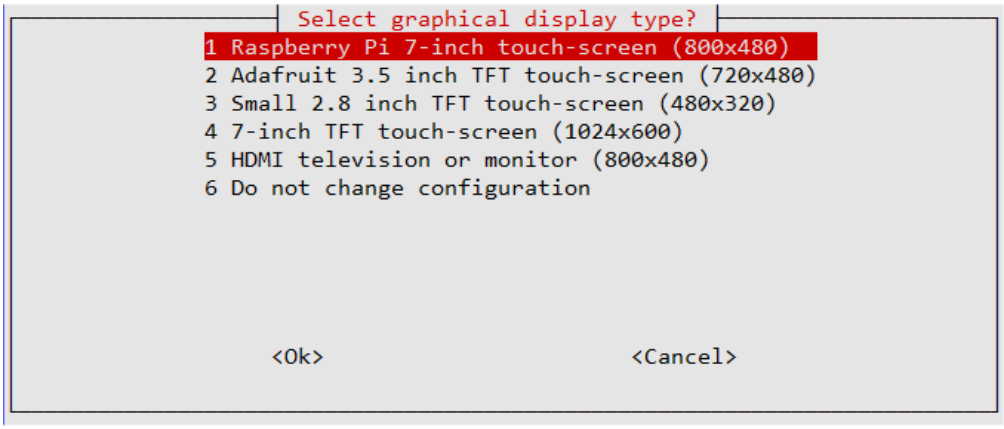


Figure 137 Touchscreen selection

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

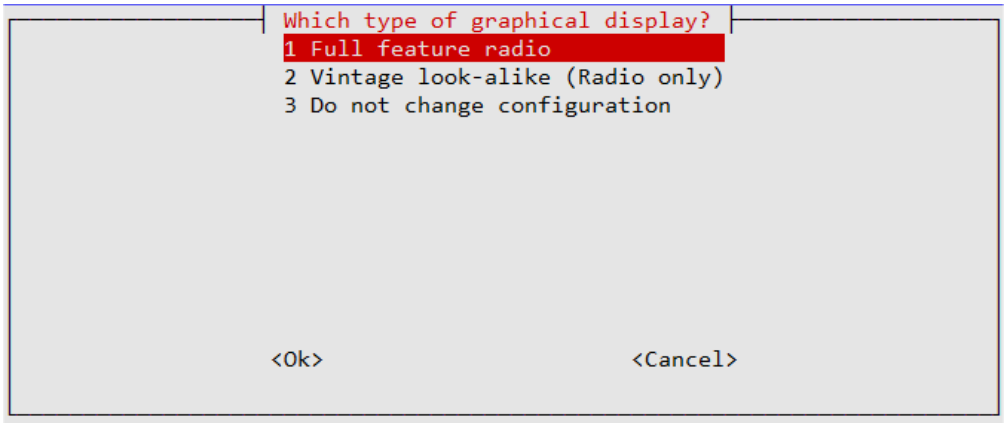


Figure 138 Selecting the type of radio display

Select which radio program to start-up. . See *Figure 1 Raspberry pi 7-inch touchscreen radio* and *Figure 3 Vintage tuning touch-screen radio* on page 4.

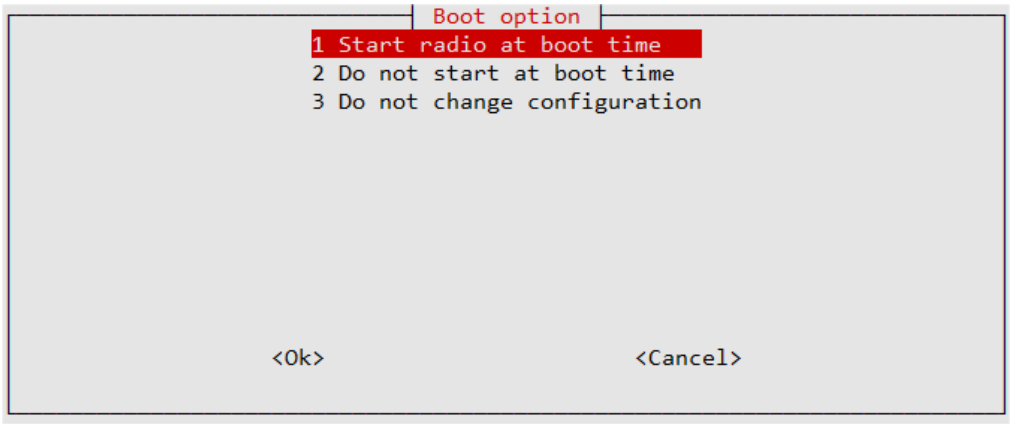


Figure 139 Configuring the HDMI or touch screen display startup

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.

```
[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;
```

There is also a similar file created called **gradio.desktop** for the vintage graphical radio.

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).

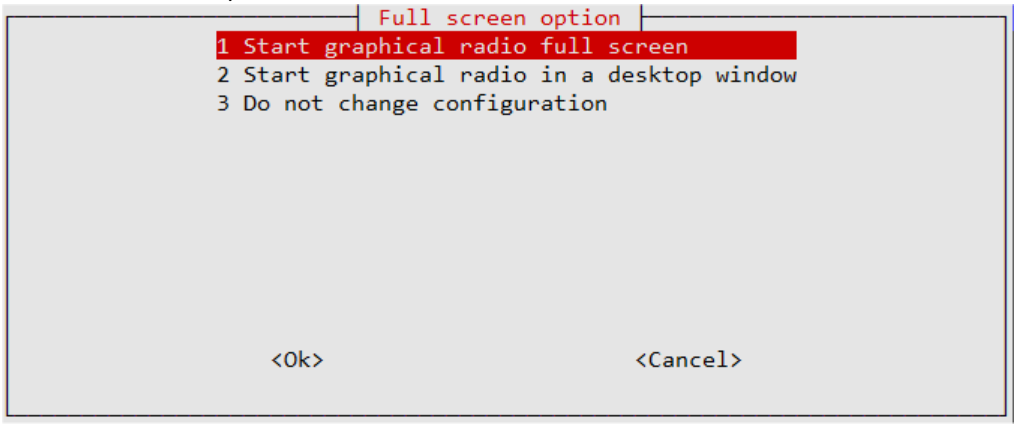


Figure 140 Configuring the graphical radio full screen

Configuration of the HDMI/Touch screen is shown in the section *Configuring the HDMI or Touch Screen* on page 124.
Operation of the HDMI/Touch screen is shown in the section called *Operation of HDMI and touch screen displays* on page 144.

Now go to the next section *Configuring the audio output* on page 84.

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 installation will automatically run the **configure_audio.sh** script.

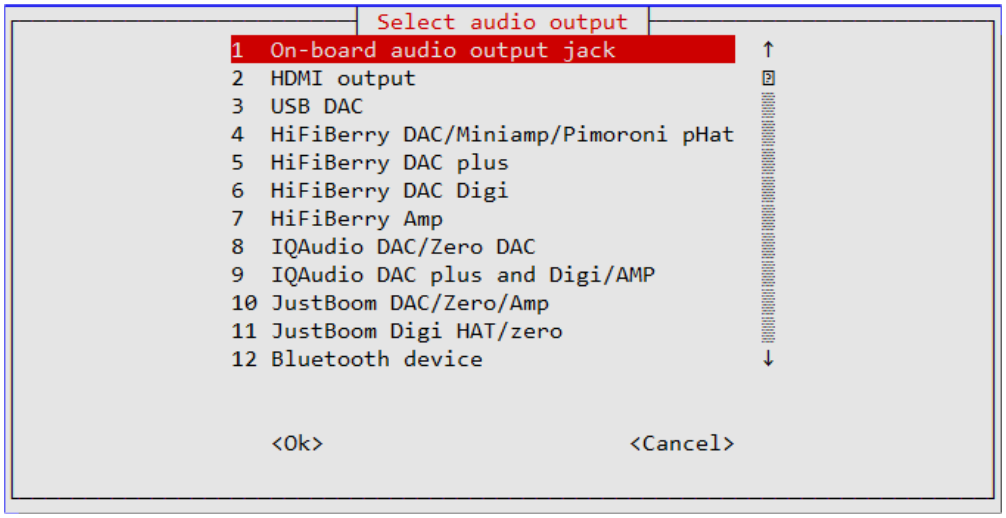


Figure 141 Selecting the audio output device

Please note the scroll bar on the right side of the above screen. There is another selection. after option 12 (Manually configure). 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
$ sudo ./configure_audio.sh
```

Installing the Python I2C libraries

This needs to be carried out for all displays which use I2C (OLED and Adafruit RGB plate) or backpacks (Arduino or Adafruit). Enter the following commands to add SMBus support required for Python I2C support:

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

Note: **python-smbus** may already be installed on **Raspbian Buster** but not the **Lite** version
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 (Old boards) carry out the following:

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

This will search **/dev/i2c-0** (Old RPIs) or **/dev/i2c-1** (Later 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 142 *The I2C bus display using the i2cdetect program*.

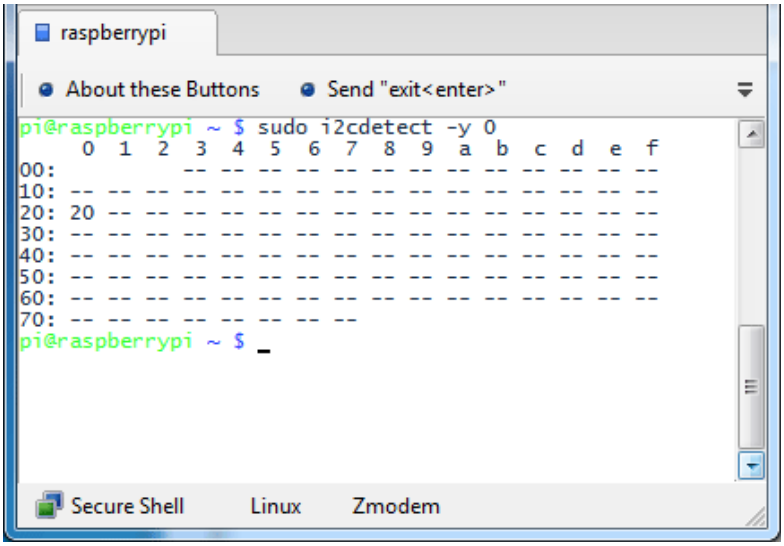


Figure 142 The I2C bus display using the i2cdetect program

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
```

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 overides 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 177.

Installation logs

A log of the changes made by the radio configuration program will be written to the **/usr/share/radio/install.log** file. For the audio configuration program changes will be written to the **/usr/share/radio/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 177.

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

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

This will also stop and start the MPD daemon.

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

```
$ sudo systemctl disable radiod
```

To re-enable it:

```
$ sudo systemctl enable radiod
```

Installing PiFace CAD software

Before running the radio on the PiFace CAD it is necessary to install the PiFace CAD Python library. Run the following command:

```
$ sudo apt-get install python-pifacecad
```

The SPI maximum frequency has changed to 125000000 after kernel 4.9.43, however the **pifacecad** software can't support the default frequency. You can read about this at the following link: https://github.com/raspberrypi/linux/issues/2165

To correct this problem, it is necessary to modify **/usr/lib/python2.7/dist-packages/pifacecommon/spi.py** to limit the SPI frequency.

```
$ sudo vi /usr/lib/python2.7/dist-packages/pifacecommon/spi.py
```

It is very likely that this problem may be fixed in a later release of the kernel and that the following patch will not be necessary.

Edit the “spi transfer struct” from

```
    # create the spi transfer struct
    transfer = spi_ioc_transfer(
        tx_buf=ctypes.addressof(wbuffer),
        rx_buf=ctypes.addressof(rbuffer),
        len=ctypes.sizeof(wbuffer)
    )
```

To:

```
    # create the spi transfer struct
    transfer = spi_ioc_transfer(
        tx_buf=ctypes.addressof(wbuffer),
        rx_buf=ctypes.addressof(rbuffer),
        len=ctypes.sizeof(wbuffer),
        speed_hz=ctypes.c_uint32(100000)
    )
```

Warning. Do not forget to add the comma (,) to the end of the previous line. Restart the **PiFace CAD** radio. Also run update the Raspberry Pi firmware to the latest version.

```
$ sudor pi-update
```

Installing Pimoroni Pirate Radio (pHat BEAT)

Note: The current version of the Pimoroni does not currently install on Rasbian Buster. This has been reported as an issue to Pimoroni. See: https://forums.pimoroni.com/t/vlcradio-installation-scripty-fails-on-rasbian-buster/12282 The following procedure is a temporary workaround and will change once the issue has been fixed by Pimoroni.

Install Pimoroni software

To use the Pimoroni Pirate radio with **pHat BEAT** it is necessary to install the Pimoroni software and test the radio using the Pimoroni software first. Do this before installing the Rathbone radio software. The Pimoroni software uses the **VLC** media player and not **MPD**.

Once the Pimoroni software is installed and tested it is then necessary to disable the Pimoroni radio software and install the Rathbone radio software as shown in *Chapter 6 - Installing the radio Software* on page 74. Only the VU meter software is used by the Rathbone software.

To install the Pimoroni software carry out the following:

The following instructions are based on the following link:
https://github.com/pimoroni/phat-beat/blob/master/projects/vlc-radio/README.md

Run the following command from the pi user home directory:

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

The following will be displayed.

```
:
```

```
If you want to see what this script does before running it,
you should run: 'curl https://get.pimoroni.com/vlcradio'
:
Do you wish to continue? [y/N] y
```

Enter **y** to continue.
The installation will initially fail with the following message:

```
The VLC Radio installer
does not work on this version of Raspbian.
Check https://github.com//
for additional information and support
```

To correct this edit /home/pi/Pimoroni/phatbeat/projects/vlc-radio/setup.sh

```
$ cd /home/pi/Pimoroni/phatbeat/projects/vlc-radio
$ nano setup.sh
```

Find the raspbian_check() routine and add the two lines to check for Buster as shown below

```
raspbian_check() {
    IS_SUPPORTED=false
    IS_EXPERIMENTAL=false

    if [ -f /etc/os-release ]; then
        if cat /etc/os-release | grep -q "/sid"; then
            IS_SUPPORTED=false && IS_EXPERIMENTAL=true
        elif cat /etc/os-release | grep -q "stretch"; then
            IS_SUPPORTED=false && IS_EXPERIMENTAL=true
        elif cat /etc/os-release | grep -q "jessie"; then
            IS_SUPPORTED=true && IS_EXPERIMENTAL=false
        elif cat /etc/os-release | grep -q "wheezy"; then
            IS_SUPPORTED=true && IS_EXPERIMENTAL=false
        elif cat /etc/os-release | grep -q "buster"; then
            IS_SUPPORTED=true && IS_EXPERIMENTAL=false
        else
            IS_SUPPORTED=false && IS_EXPERIMENTAL=false
        fi
    fi
}
```

Now run the **vlc-radio setup.sh** script

```
$ sudo ./setup.sh
```

Now re-run the original Pimoroni installation script:

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

The installation script should now run OK but will take some time to install and build the software. Reboot the Raspberry Pi and test the Pirate radio.

```
$ sudo reboot
```

After rebooting to start playing the radio, press the play button on the pHAT BEAT. The buttons are labelled on the pHAT BEAT board itself and on the front of the Pirate Radio. The play button is the second one down from the top.

Note: This author does not provide any support for the Pimoroni software or VLC media player. If you have problems getting the Pimoroni software to run then contact https://forums.pimoroni.com/c/support

Disable Pimoroni software

First disable the Pimoroni software run the following commands:

```
$ sudo systemctl disable vlcd phatbeatd
$ sudo reboot
```

Install the Rathbone Internet radio software

Do the following:

1. Now carry out the instructions shown in *Chapter 6 - Installing the radio Software* on page 74.

2. Since the Pirate Radio does not have a screen it is necessary to install **espeak** as shown in the section called *Installing the speech facility* on page 120 to hear choices when using the menu button.

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.

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

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, 6.12 in this case.

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

For example:

radiod-patch-6.12-1.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-6.12-
1.tar.gz
$ tar -xvf radiod-patch-6.12-1.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.

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:

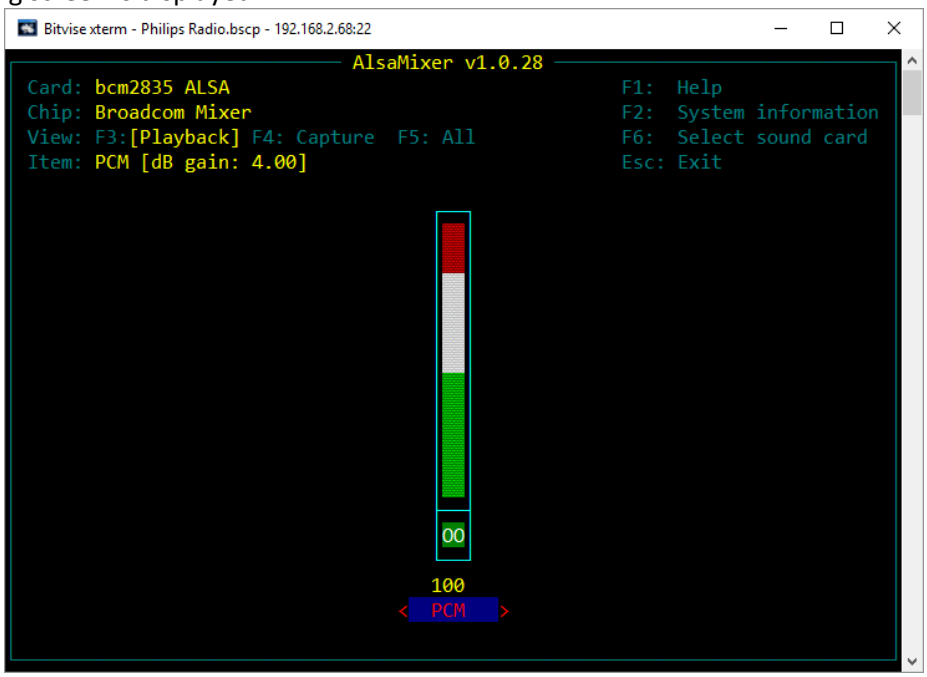


Figure 143 Basic Alsa sound mixer

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 93)
- Sound cards such as **HiFiBerry**, **IQaudIO**, **JustBoom** and **Pimoroni pHat** DAC and DAC+ products (See page 94)
- Bluetooth speakers or headphones (See page 97).

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
```

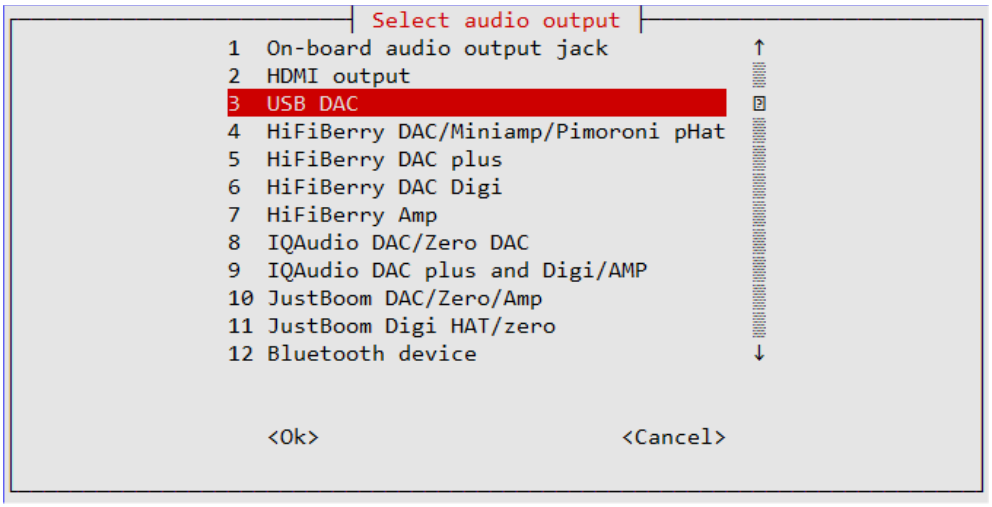


Figure 144 Configure USB DAC

Reboot when prompted. After rebooting the Raspberry Pi run the **alsamixer** program.

```
$ alsamixer
```

The following screen is displayed:

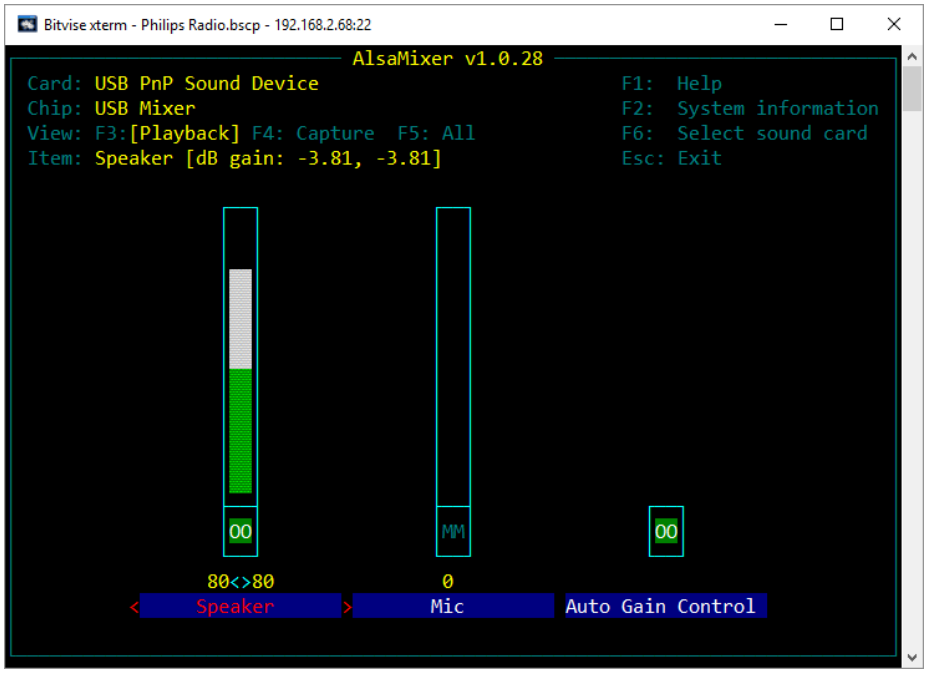


Figure 145 The USB PnP Alsa Mixer

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

Configuring a 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
```

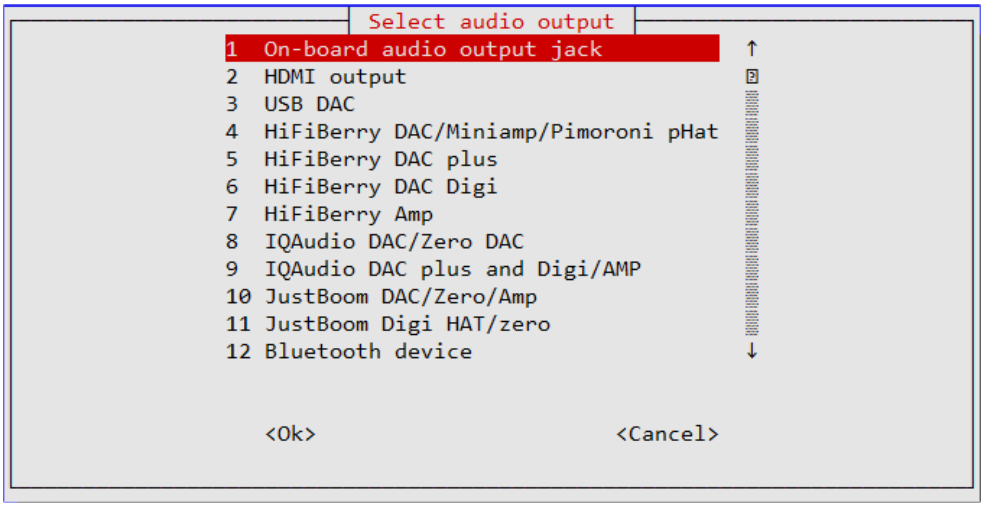
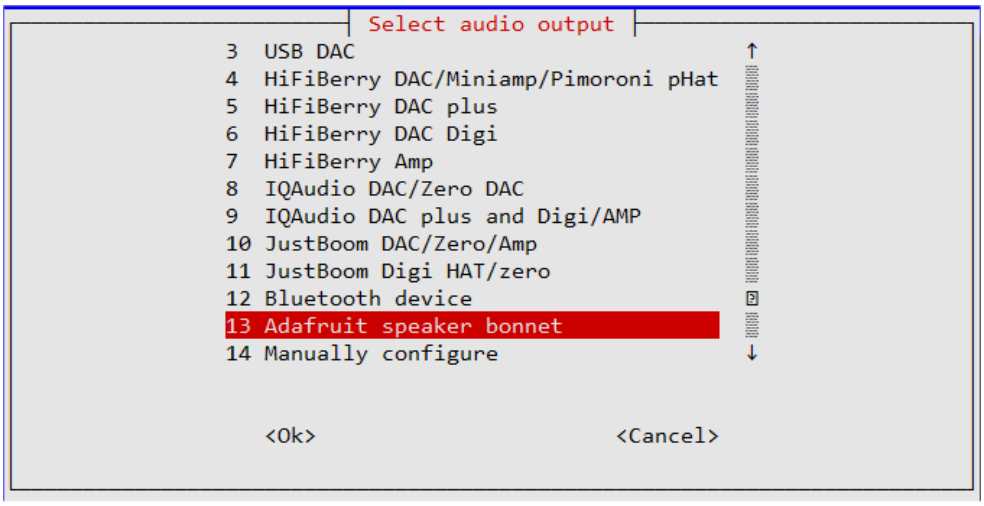


Figure 146 Configuring add-on DAC sound cards

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 compatable 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%.

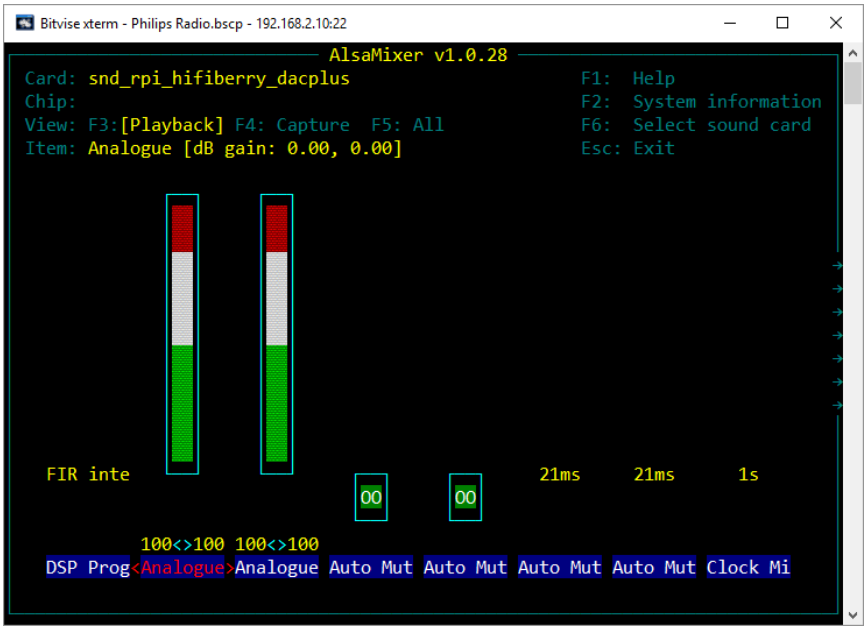


Figure 147 Set mixer analogue volume

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

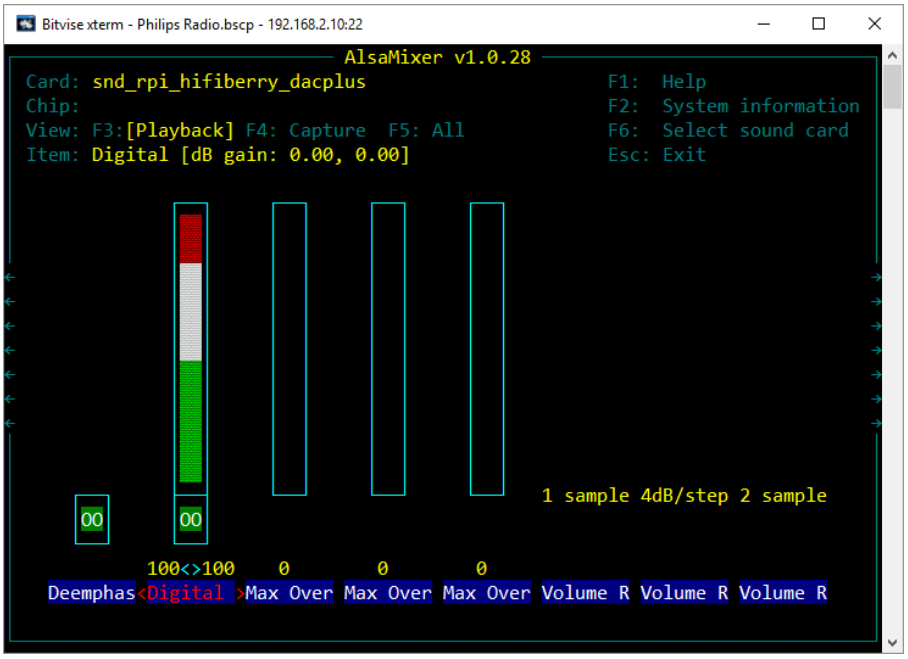


Figure 148 Set mixer digital volume

Connecting a Bluetooth device

Install the Bluetooth software

Usually all necessary Bluetooth is installed for the full versions of Rasbian but may be missing for the Lite version. To make sure all required software is installed run the following:

```
$ sudo apt-get install bluez bluez-firmware pi-bluetooth bluealsa
```

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 are using Rasbian Lite you may see the following message:

```
$ bluetoothctl
[bluetooth]# scan on
No default controller available
```

To overcome this run **bluetoothctl** with **sudo**.

```
$ sudo bluetoothctl
```

Next 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
```

```
[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
```

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 following displays all available commands:

```
$ bluetoothctl help
```

Now re-run the configure_audio.sh configuration script and select bluetooth.

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

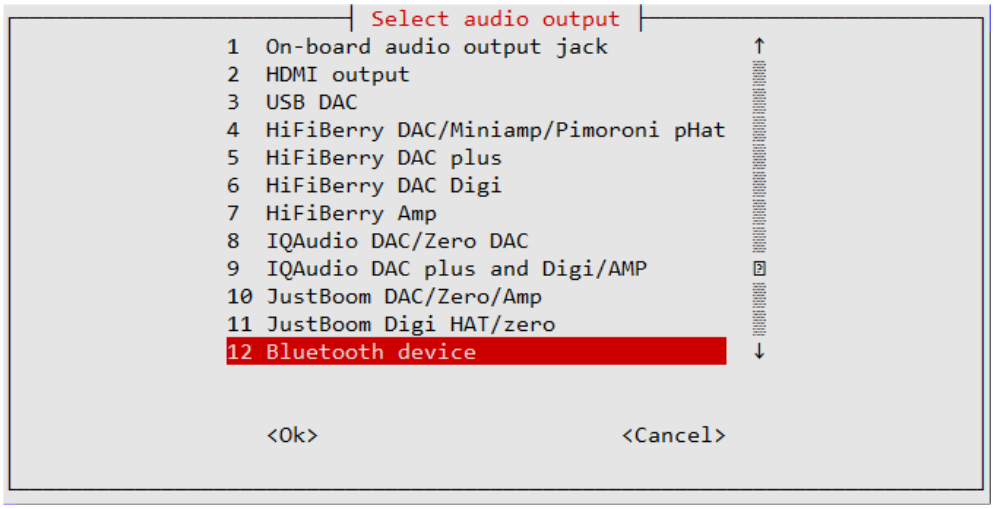


Figure 149 Configuring bluetooth devices

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:

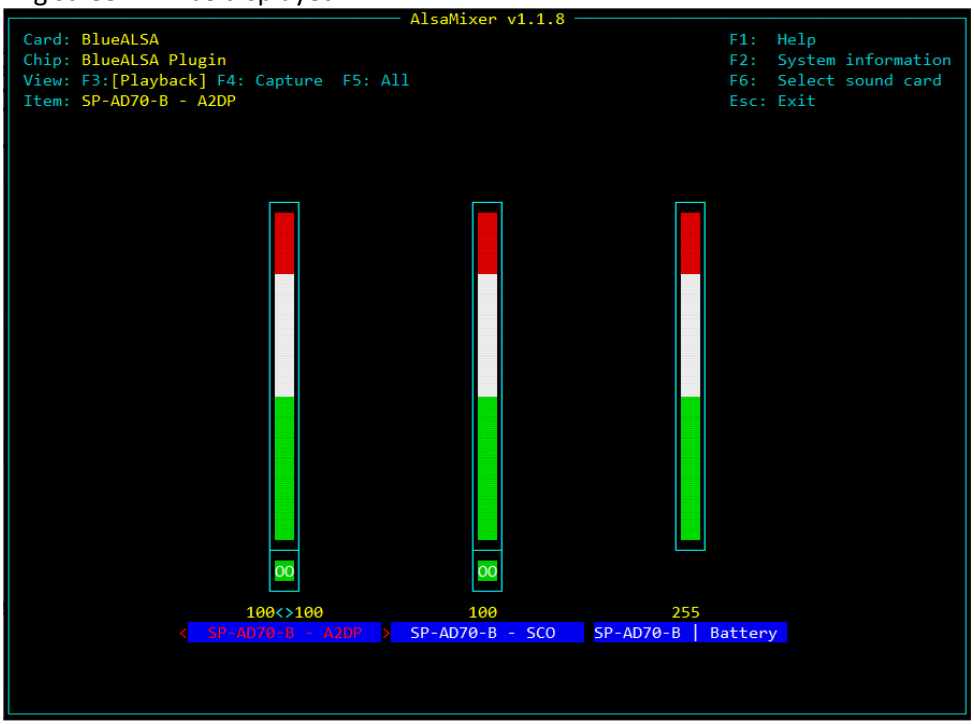


Figure 150 Alsamixer using Bluetooth devices

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 158.

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-get install pulseaudio
```

Either re-run the **configure_radio.sh** program or manually change the **audio_output** type statement in /etc/mpd.conf to **pulse**.

```
audio_output {
        type            "pulse"
        name            "IQAudio DAC+"
        device          "hw:0,0"
        mixer_type      "software"
        :
}
```

Reboot the Raspberry Pi to restart the radio.

```
$ sudo reboot
```

Installing the Infra-Red sensor software

In Raspbian Stretch and Buster released from April 2019 onwards, the **lirc-rpi** kernel device overlay which previously provided the driver for IR devices has been replaced with two new kernel device overlays called **gpio-ir-tx**. and **gpio-ir** which handle IR transmission and reception respectively. As a result, a new procedure needs to be described here, namely the **gpio-ir** kernel module installation procedure. The **gpio-ir-tx** transmitter kernel module is not used in this project.

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 11 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

Install the lirc software

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

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

Update the Raspberry Pi firmware. This not necessary if your version of **Buster** is September 2019 or later as it is done automatically when **Buster** is installed.

```
$ sudo rpi-update
```

Run the IR remote installation program:

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

The following screen will be displayed.

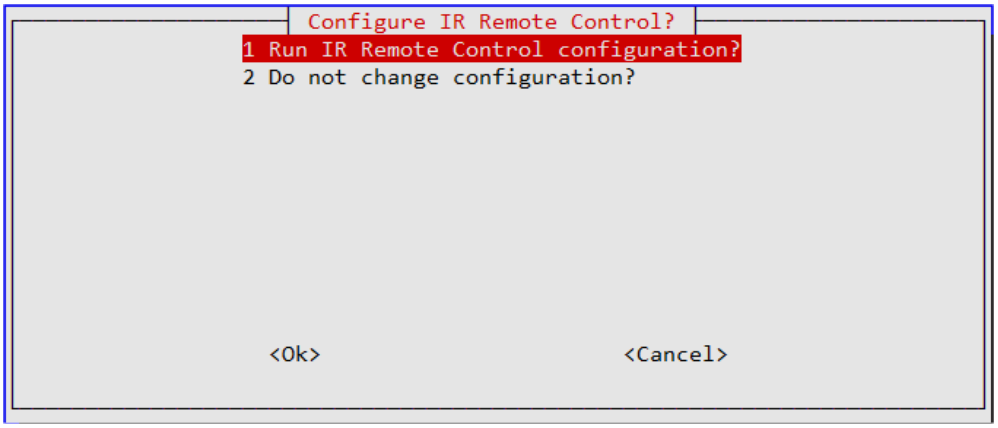


Figure 151 IR Remote Installation program

Run the configuration program and select the operating system being used. Either Buster or Jessie.

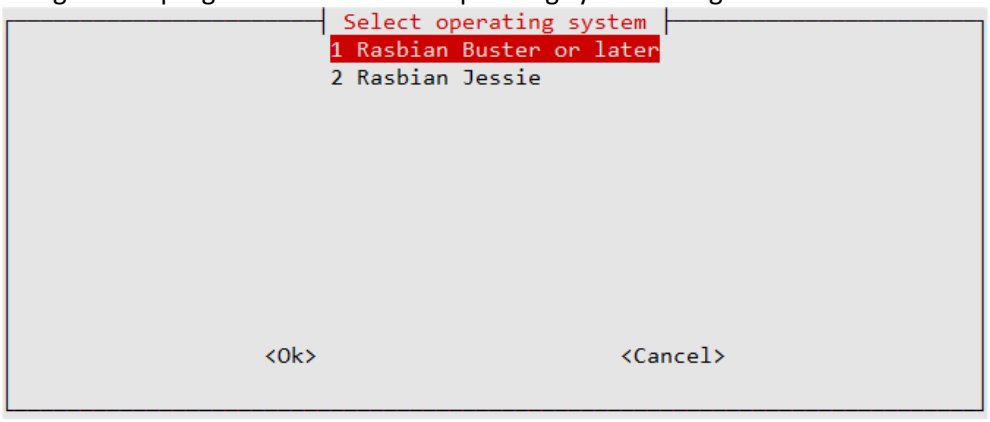
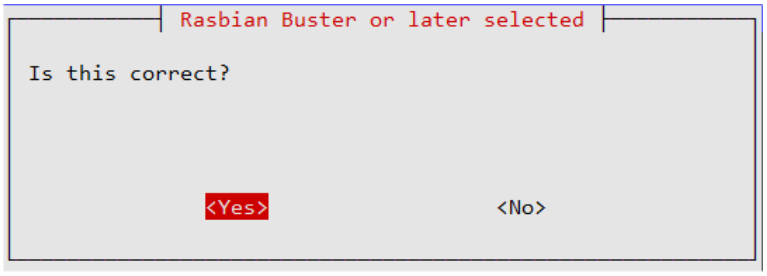


Figure 152 IR configuration OS selection

The program will ask for confirmation.



It is now necessary to select which GPIO is to be used for the IR sensor. This is either 9, 16 or 25.

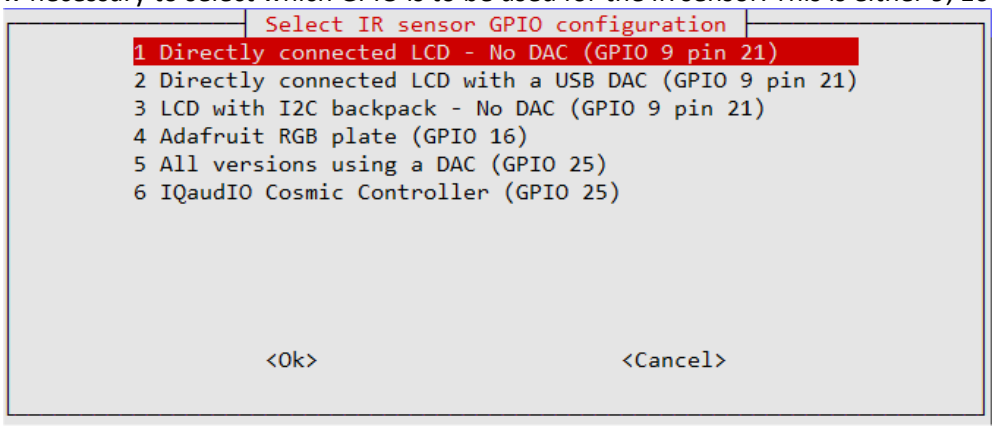


Figure 153 IR configuration IR sensor GPIO selection

If you selected Jessie as the operating system it will display the following message:

```
You have selected Jessie as the OS version you are using.
Please note the release date of the kernel you are using from the line
below:
    Linux 4.19.75-v7l+ #1270 SMP Tue Sep 24 18:51:41 BST 2019
Enter to continue:
```

Note the date of the Kernel release. Select correct Kernel date from the following screen:

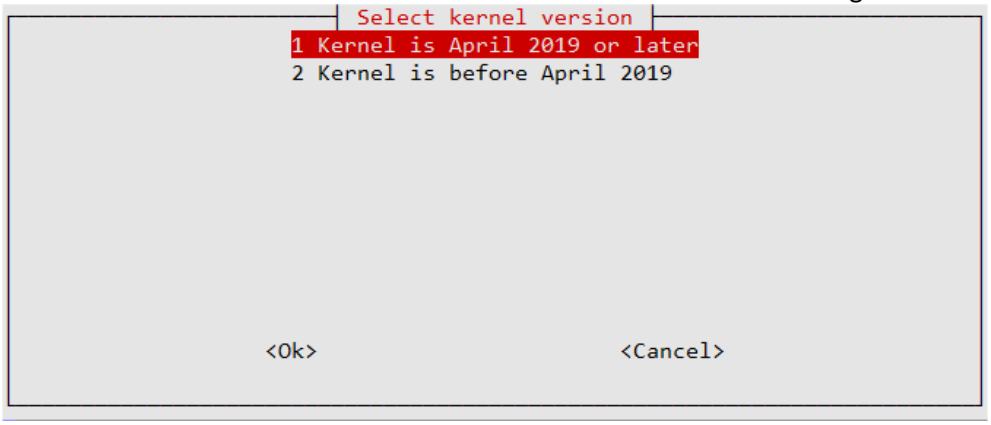


Figure 154 IR Configuration - Kernel relase date selection

Now select the Remote Activity LED GPIO. This is either GPIO 11, 13 or 14.

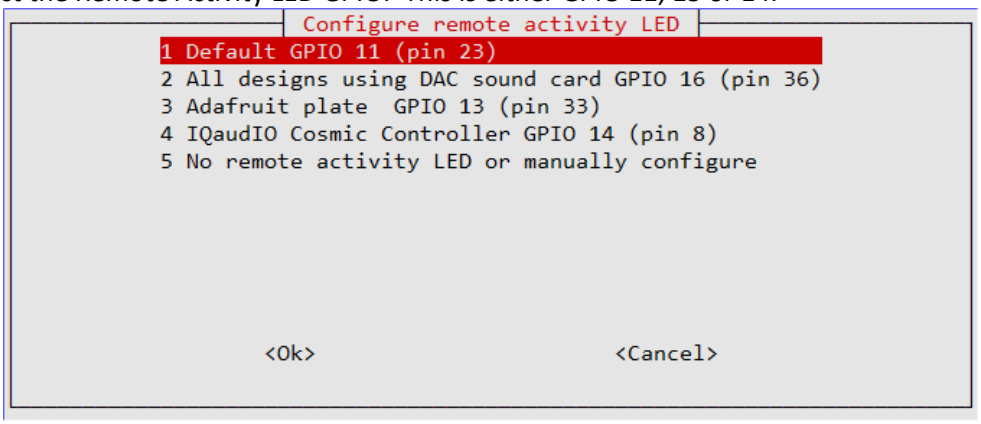


Figure 155 IR configuration Activity LED GPIO selection

Once the Activity LED selection has been made the program will install LIRC components and configure the **/boot/config.txt** file.

```
./configure_ir_remote.sh configuration log, Sat 26 Oct 12:07:48 BST 2019
Selected GPIO is 9
Remote activity LED is GPIO 11

Added following line to /boot/config.txt:
dtoverlay=gpio-ir,gpio_pin=9
sudo dtoverlay gpio-ir gpio_pin=9

Configured remote_led=11 in /etc/radiod.conf

Installing lirc
:
sudo cp /lib/udev/rc_keymaps/rc6_mce.toml /etc/rc_keymaps/rc6_mce
sudo cp /usr/share/radio/lircrc.dist /etc/lirc/lircrc
```

The program will display the following instructions to complete the set-up process.

For Buster

```
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

A log of this run will be found in /usr/share/radio/install_ir.log
```

For Jessie

```
:
Then copy your configuration file (myremote.conf) to  /etc/lirc/lircd.conf
          sudo cp myremote.conf /etc/lirc/lircd.conf
:
```

The program adds the **gpio-ir** dtoverlay to the **/boot/config.txt** file for Buster.

```
dtoverlay=gpio-ir,gpio_pin=25
```

or for Jessie with a kernel released before April 2019:

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

Reboot the radio

```
$ sudo reboot
```

After reboot check that **lircd** is running.

```
$ sudo systemctl status lircd
● lircd.service - Flexible IR remote input/output application support
   Loaded: loaded (/lib/systemd/system/lircd.service; enabled; vendor
preset: enabled)
   Active: active (running) since Sat 2019-10-26 11:45:25 BST; 35min ago
     Docs: man:lircd(8)
           http://lirc.org/html/configure.html
 Main PID: 3316 (lircd)
    Tasks: 2 (limit: 2061)
   Memory: 1.0M
   CGroup: /system.slice/lircd.service
           └─3316 /usr/sbin/lircd --nodaemon

:
oot
```

Now test the remote control. Run the test program

```
$ sudo mode2 -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.

Configuring the IR remote control

There are a number of ways to do this. Only two are described here.

1. Download the configuration from **sourceforge.net**
2. Create one using the **irrecord** utility program

Method 1

To download a configuration from **sourceforge** go to
http://lirc-remotes.sourceforge.net/remotes-table.html

Find your remote control in the list and click on it. Click on the caption **Download this file** and save it to your PC and copy it to the Raspberry Pi. Alternatively copy the link from the caption and use **wget** to download it directly to the Rasberry 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 log as it ends in **.conf**

```
$ mv 3F14-00048-180.lircd.conf?format=raw myremote.lircd.conf
```

For Buster

Now copy it to the /etc/lirc/lircd.conf.d directory

```
$ sudo cp myremote.lircd.conf /etc/lirc/lircd.conf.d/.
```

For Jessie

```
$ sudo cp myremote.lircd.conf /etc/lirc/myremote.lircd.conf
```

Method 2

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 12 Remote Control Key names and functions*.

Table 12 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.

Notes: 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.**

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.
On completion of the key assignments the following is displayed:

```
Successfully written config file myremote.lircd.conf
```

You may occasionally 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.
Terminate the program by pressing Enter when prompted for the next key.

If this problem persists create separate myremote files with a few keys each and copy them into **/etc/lirc/lircd.conf.d**. For example, myremote1, myremote2, myremote3….

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.

```
$ sudo systemctl restart lircd
```

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

Testing the remote control

Now 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 seem to work with some versions of Buster. Use the above **irw** program.

Enable and start and check the irradio 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 44.

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

```
$ sudo systemctl enable irradiod radiod
$ sudo systemctl start irradiod radiod
```

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 ./remote_control.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 a wireless adaptor

This section is only needed for earlier versions of the operating system. The wireless adaptor is now configured using raspi-config. See section *Configuring the Wi-fi Connection* on page 69.

If you need to manually configure a wireless adaptor for the radio then follow this procedure. Choose a wireless adapter that has been approved for the Raspberry PI or use the model 3B or later with in-built WiFi adapter. See the following link for approved Raspberry PI peripherals: http://elinux.org/RPi_VerifiedPeripherals

Install the wireless adapter

Switch off the Raspberry PI and plug in the adaptor into one of the USB ports. Power the PI back on and log in. Check to see if your wireless adapter has been recognised by running the **lsusb** command.

```
pi@raspberrypi:~$ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 148f:5370 Ralink Technology, Corp. RT5370 Wireless
Adapter
```

The above shows a Ralink (Tenda) wireless adaptor but this will vary depending on the adapter that has been installed. All newer Raspberry Pi's have an inbuilt WiFi interface that will not be seen with the above command.

Configure the network adaptor

The file to be amended is **/etc/wpa_supplicant/wpa_supplicant.conf**.
Edit this file. It will only contain a couple of lines.

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
```

Add the following after the above lines:

```
country=GB
network={
    ssid="YOUR_SSID"
    psk="YOUR_KEY"
}
```

Substitute YOUR_SSID and YOUR_KEY with the actual SSID and key for your wireless router. The above configuration is for a router using WPA encryption. If your router is using the older WEP encryption then you will need to adapt the configuration to use WEP. See next section.
Change the country identification to your counties code, for example NL for the Netherlands.

Explanation of the network fields

Table 13 WiFi network configuration

Field	Description
ssid	your WIFI (SSID) name
psk	Your WIFI password

The only problem with the above configuration is that the **psk** key is in plain text and can be read by anyone who has access to the Raspberry PI. It is possible to increase security by generating a so-

called passphrase with the **wpa_passphrase** command. For example, if your **ssid** is *mywlan* and the WIFI password is *abcdef1234* then use the following command to generate the passphrase.

```
# wpa_passphrase mywlan abcdef1234
network={
        ssid="mywlan"
        #psk="abcdef1234"
        psk=53a566e0ccf03ec40b46e6ef4fc48b836e428fb0fd5e0df95187ba96e60ce7ce
}
```

Copy and paste the passphrase into the psk parameter into the **/etc/wpa_supplicant/wpa_supplicant.conf** file. Do not include any quotes around it. Optionally remove the comment line which shows the original key (#psk="abcdef1234").

Operating the wireless interface

If configured correctly the wireless adapter will start up when the Raspberry PI is rebooted. The adaptor can be started and stopped with the following commands:

For Raspbian Buster and later:

```
pi@raspberrypi~ $ sudo ip link set wlan0 up
```

and

```
pi@raspberrypi~ $ sudo ip link set wlan0 down
```

For earlier versions of Raspbian

```
root@raspberrypi:/home/pi ifup wlan0
```

and

```
root@raspberrypi:/home/pi# ifdown wlan0
```

To see what **SSIDs** are available run the **iwlist** command as shown in the following example:

```
pi@raspberrypi~ $ sudo  iwlist wlan0 scanning | grep ESSID
                    ESSID:"mywlan"
                    ESSID:"VGV751926F4B9"
                    ESSID:"prime"
                    ESSID:"Sitecom6A212C"
```

Troubleshooting the wireless adapter

Problem – Starting the wireless adapter gives the following message when using the older **ifup** and **ifdown** commands:

```
$ sudo ifup wlan0
wpa_supplicant: /sbin/wpa_supplicant daemon failed to start
run-parts: /etc/network/if-pre-up.d/wpasupplicant exited with return code 1
Failed to connect to wpa_supplicant - wpa_ctrl_open: No such file or
directory
wpa_supplicant: /sbin/wpa_cli daemon failed to start
run-parts: /etc/network/if-up.d/wpasupplicant exited with return code 1
```

This is due to an incorrect **/etc/wpa_supplicant/wpa_supplicant.conf** file. The problem is due to an incorrect configuration. For example, a space after the **ssid=** directive as shown below.

```
network={
    ssid= "homelan"
    psk="d762c954df"
}
```

Solution: Correct the error and run the **ifup wlan0** command.
Problem: The following is seen:

```
pi@raspberrypi~ $ sudo ifup wlan0
ifup: interface wlan0 already configured
```

Solution: This isn't actually an error. Just run the **ifdown wlan0** command and retry the **ifup wlan0** command. It should then work.

Configuring a static IP address

The Raspberry PI Ethernet network interface comes configured out of the box to use DHCP (Dynamic Host Configuration Protocol) when using Raspbian Buster or a similar operating system. This means it is given an IP address by the (home) router for a particular lease period. This also means that if the Raspberry PI is switched off for any length of time it may get a different IP address from the one it had previously. This may not be very convenient especially if you are using the web interface or a mobile app to control the radio. There are two possible choices here:

1. Configure the home router so that DHCP delivers a fixed address based upon the Raspberry Pi's MAC address. This is called DHCP IP address reservation.
2. Configure the Raspberry Pi with a static IP address.

Configuring DHCP in a router

It is beyond the scope of this manual to show how to reserve DHCP IP addresses and will vary anyway between routers. Also, it isn't always possible configure the router. Take a look at the following link for a good example of how to do this: http://lifehacker.com/5822605/how-to-set-up-dhcp-reservations-so-you-never-have-to-check-an-ip-address-again
Alternatively search the internet with the name of your router and the term "DHCP IP address reservation".

If DHCP IP address reservation isn't an option then it is possible to configure a so-called static IP address which will not change between reboots. To do this you must find out the IP address of your router (gateway) and netmask for your network. Do this with the **ip route** command as shown below:

```
$ ip route
default via 192.168.1.254 dev eth0 proto dhcp src 192.168.1.152 metric 202
192.168.1.0/24 dev eth0 proto dhcp scope link src 192.168.1.152 metric 202
```

In the above example the IP address of the gateway is 192.168.1.254. Your router's IP address will almost certainly be different. Take a note of the gateway IP address and the subnet (Genmask) of your network. In this case that is 255.255.255.0. You will need to use these values to configure the interface. The next thing you need is a free IP address in your network. There will be lots of them but approximately 50 to 100 of them will be claimed for the DHCP pool. The only way to know what DHCP is using is to log into your router and look at the configuration. If this isn't possible or you are

unsure of what to do then pick one somewhere in the middle of the network range. For example, in the above network you could choose 192.168.1.125. Check that it isn't already in use somewhere else in your network. Check it with the **ping** command. If you see 100% packet loss then the IP address you have chosen isn't in use in your system so you can use it.

```
$ ping -c 4 192.168.1.125
PING 192.168.1.125 (192.168.1.125) 56(84) bytes of data.
From 192.168.1.8 icmp_seq=1 Destination Host Unreachable
From 192.168.1.8 icmp_seq=2 Destination Host Unreachable
From 192.168.1.8 icmp_seq=3 Destination Host Unreachable
From 192.168.1.8 icmp_seq=4 Destination Host Unreachable

--- 192.168.1.125 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3012ms
```

Ethernet static IP configuration

There are two methods of configuring static IP addresses:

1. Edit the **/etc/dhcpcd.conf** file.
2. Create a network interface file in the **/etc/network/interfaces.d** directory

Only method 1 is described here. Method 2 is the **System V** method of configuration and is rarely used in **Rasbian**. If you want to know more about creating network files in /etc/network/interfaces.d then refer to the man page:

```
$ man interfaces
```

Edit the **/etc/dhcpcd.conf** file using **sudo nano** or **sudo vi**. Go to the bottom of the file to see an example of a static IP address (commented out).

```
# Example static IP configuration:
#interface eth0
#static ip_address=192.168.0.10/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
#static routers=192.168.0.1
#static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1
```

Remove the # character from the interface definitions and modify the IP addresses as required. However, use an IP address which is clear of the DHCP pool. An IP address 150 or higher will probably be safe.

```
interface eth0
static ip_address=192.168.0.150/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1 8.8.8.8
```

Note: IP 8.8.8.8 is the IP address of Googles public DNS (Domain Name System) servers. The host command will display the DNS entry.

```
$ host 8.8.8.8
8.8.8.8.in-addr.arpa domain name pointer dns.google.
```

Save the file and reboot the system. After reboot; log into the Raspberry PI using the new IP address. If unable to log in connect a keyboard and screen and reboot. Log in and troubleshoot the problem.

Installing the Web interface

MPD has several web clients. See the following link: https://www.musicpd.org/clients/ The one used in this example is called *Snoopy* (No longer listed) is used in this version the radio software.

Install Apache

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

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

Re-run the update to refresh package lists.

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

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

Run the following command:

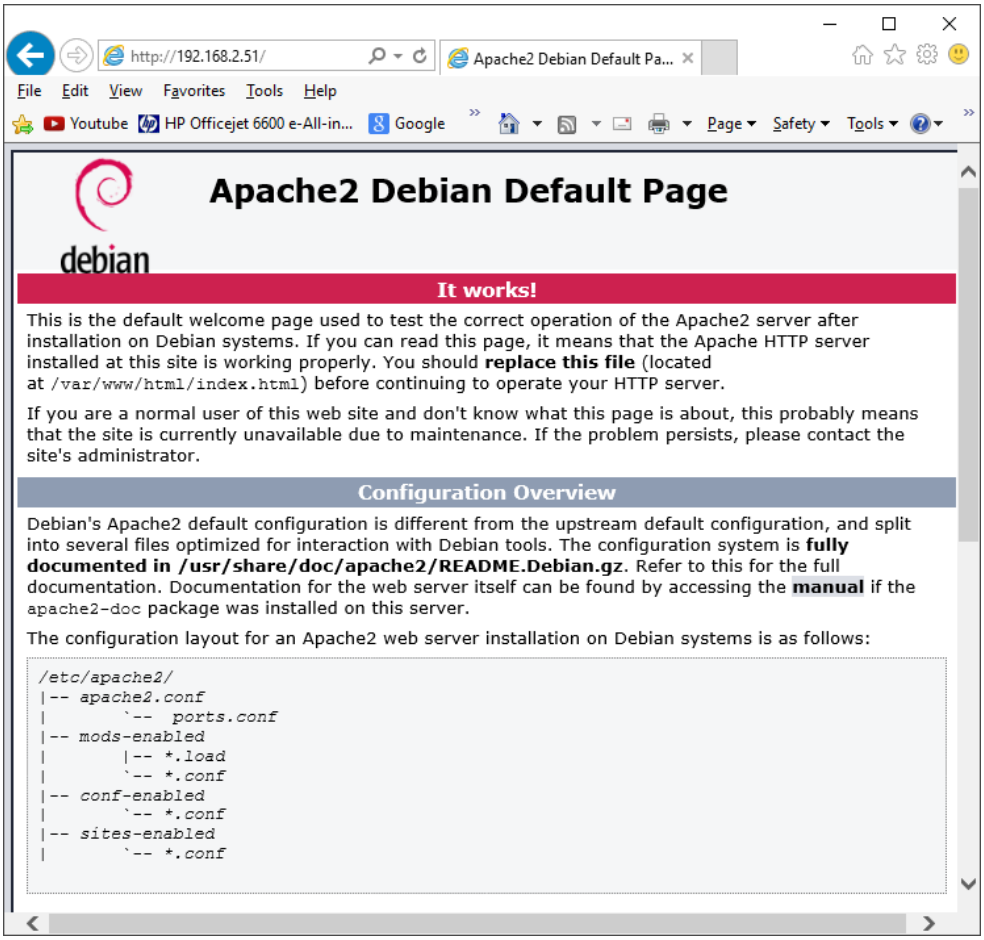
```
$ sudo apt-get install apache2 php libapache2-mod-php
```

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

```
$ sudo apt-get -f install
```

Test the Apache web browser

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



Install the Web Browser server pages

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

For **Raspbian Buster** run the following:

```
$ wget http://www.bobrathbone.com/raspberrypi/packages/radiodweb_1.7_armhf.deb
```

Now run:

```
$ sudo dpkg -i radiodweb_1.7_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://192.168.2.11 .
You should see the following display:

Figure 156 Radio web interface

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

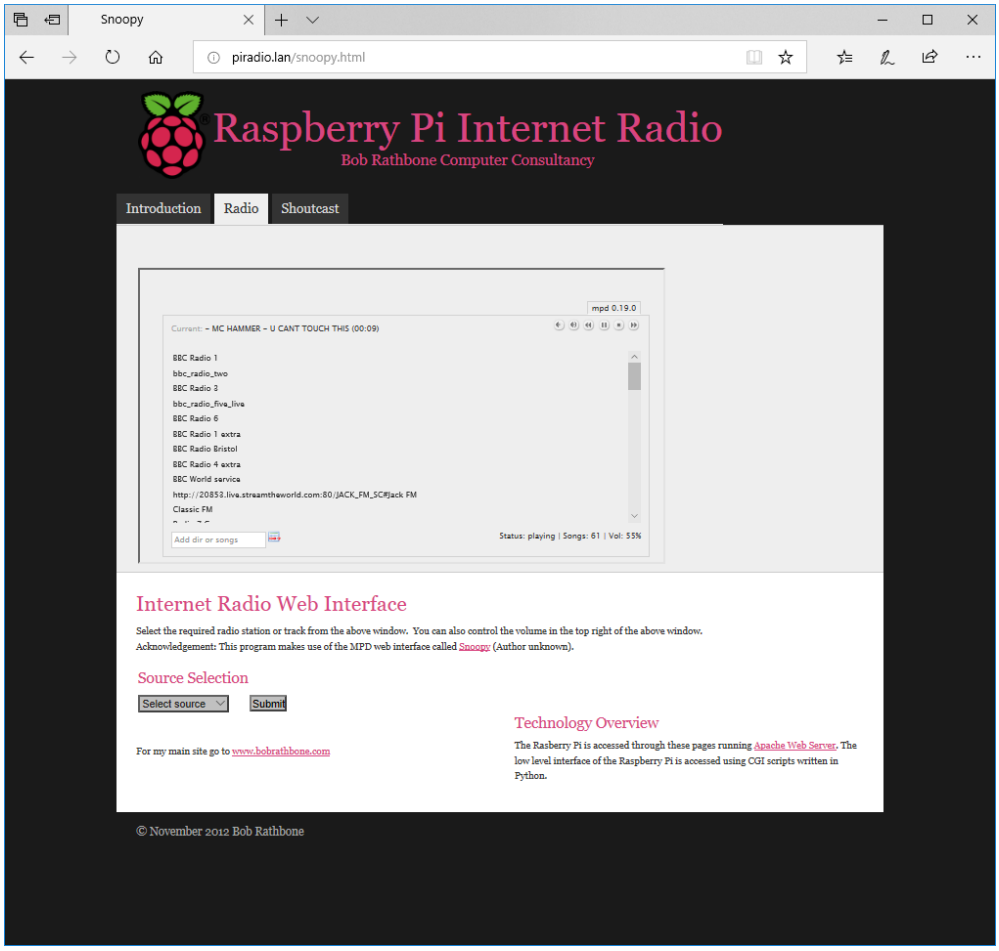
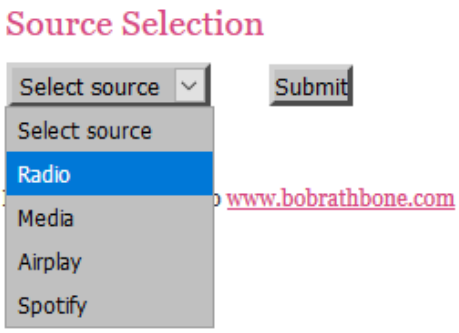


Figure 157 Snoopy web interface

Click on any station in the list to select a station. After a short pause the station should start playing if it is online.



Currently there are four music sources that can be selected namely Radio, Media, Airplay and Spotify.

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 164.

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.

Changing the Web Interface Radio photo

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

```
<img border="0" src="images/radio.jpg" width="780" height="580"></td>
```

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 236). This facility requires installation of the **espeak** package.
See 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 131

Install the **espeak** package:

```
$ sudo apt-get install espeak
```

Enable the speech facility in **/etc/radiod.conf** and restart the radio.

```
# Speech for visually impaired or blind listeners, yes or no
speech=yes
```

The verbose setting speaks the station or track details every time it is changed. However it can take a long time to move through the tracks or stations whilst speaking. Usually set this to no.

```
verbose = no
```

To get the right balance between speech volume and the normal radio volume adjust the **speech_volume** parameter percentage (10-100%)

```
speech_volume=30
```

The /var/lib/radiod/voice file

The **/var/lib/radiod/voice** file contains the **espeak** command (or part of it).

```
$ espeak -ven+f2 -k5 -s130 -a
```

Where -v is the voice (en+f2 = English female voice 2), -k is capitals emphasis, -s is the voice speed and -a is amplitude (0-200), the -a parameter is filled in by the radio program.

Testing espeak

You can test **espeak** with the following command (Stop the radio first).

```
$ espeak -ven+f2 -k5 -s130 -a20 "Hello Bob" --stdout | aplay
```

To see the capabilities of **espeak** see the website http://espeak.sourceforge.net/ or run:

```
$ espeak –h
```

If no sound is heard then then test using the **aplay** program. The **espeak** system will not work if **aplay** is not working. Test with **aplay** and a suitable wav file.

```
$ sudo mpc pause
$ aplay /usr/share/scratch/Media/Sounds/Vocals/Singer2.wav
```

If still no sound check what devices are configured using aplay.

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

In the above example there are two devices namely *bcm2835 ALSA* (normal audio jack output) and *Generic USB Audio Device*. If using either a HiFiBerry DAC or IQaudIO device then create the **/etc/asound.conf** file using nano:

```
$ 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 102.

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 ann 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 =/usr/share/scratch/Media/Backgrounds.
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 4 on page 23. 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

From version 6.8 onwards 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 (14, 15 and 23 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 44. Which pins you connect to will depend on the type of radio you are building. Table 8 on page 44 shows the required LED connections. Boards such as the 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 44.

```
# 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 **remote_control.py** program.

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

Or use the service command:

```
$ sudo service irradiod flash
```

The **irradiod** script calls the **remote_control.py** program. The activity LED should flash about six times. If not then check that the **remote_led** parameter in the **/etc/radiod.conf** configuration file is correctly set and that the activity LED is correctly wired (See LED wiring on page 44).

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.

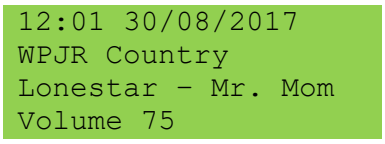
```
#startup=RADIO
startup=USB_Stick
```

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

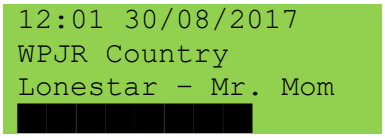
The volume can be displayed as either text or as a series of blocks. This is configured in **/etc/radiod.conf** using the **volume_display** parameter. The default is text.

```
# Volume display text or blocks
volume_display=text
```



To display the volume as a series of blocks change this to 'blocks':

```
volume_display=blocks
```



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 x 5 = 50%.

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

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

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

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. From version 6.12 onwards this is configurable from three to fifteen seconds using the **client_timeout** parameter in **/etc/radiod.conf**. The default is five seconds.

```
# MPD client timeout from 2 to 15 seconds default 5
client_timeout=5
```

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 service radiod restart
```

Configuring Music Player Daemon CODECs

From version 6.10 onwards 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

THIS PROCEDURE IS NOT CURRENTLY WORKING WITH THIS VERSION OF THE OPERATING SYSTEM AND SHOULD NOT BE CONFIGURED AS IT STOPS THE ALSA SOUND SYSTEM FROM WORKING. THE AUTHOR DOES NOT CURRENTLY HAVE A SOLUTION FOR THIS PROBLEM BUT HOPES TO PROVIDE A NEW PROCEDURE SOON. THE ONLY SOLUTION AT THE MOMENT IS A COMPLETE RE-INSTALL.

Note: The author of the radio software does not currently have a configuration that is compatible with either **Pulseaudio** or **Airplay**. If you wish to use either **Pulseaudio** or **Airplay** you cannot currently use the Alsa equalizer . This may change in a future release.

Install the Alsa plugin with **apt-get**:

```
 $ sudo apt-get 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         "hw:0,0"
        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";
```

Reboot the Raspberry Pi.

```
$ sudo reboot
```

After reboot run the Alsa Equalizer as user **mpd**. It will not work if called as either user pi or root (sudo).

```
$ 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:

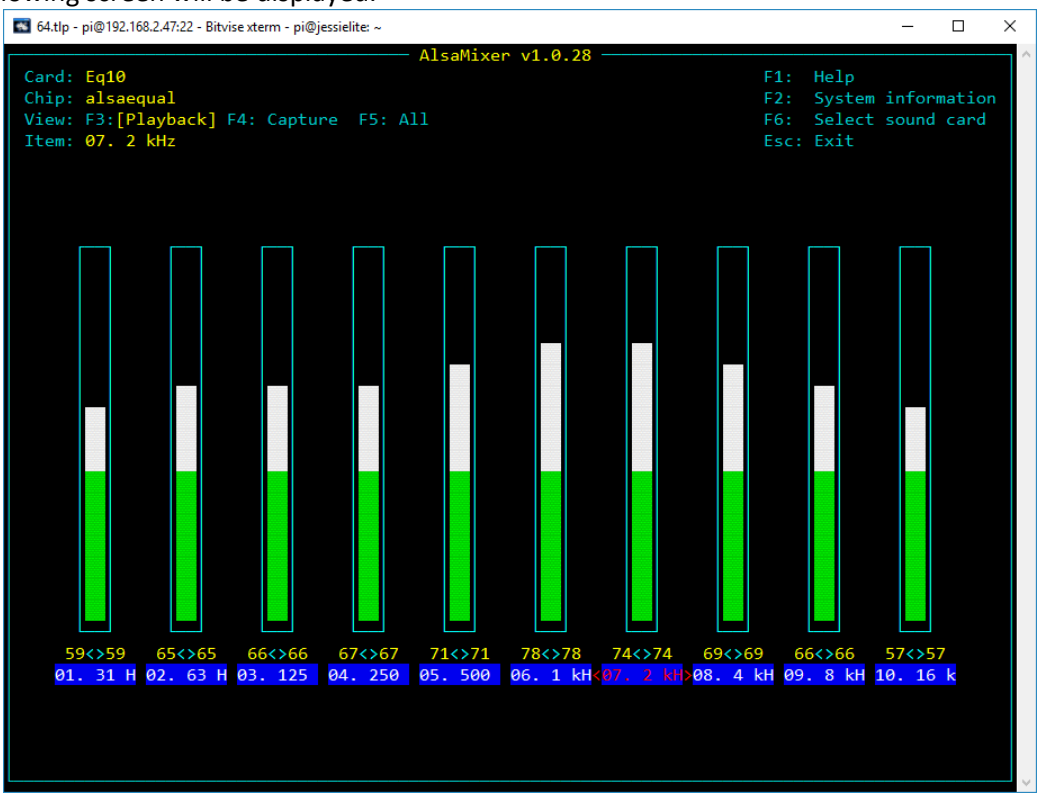


Figure 158 The Alsa

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 102.

First of all install the **FLIRC** software as shown in the section called *Installing the FLIRC USB remote control* on page 47.

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 136. The first time you run this program it may ask you if you want to upgrade the firmware. Always upgrade the firmware:

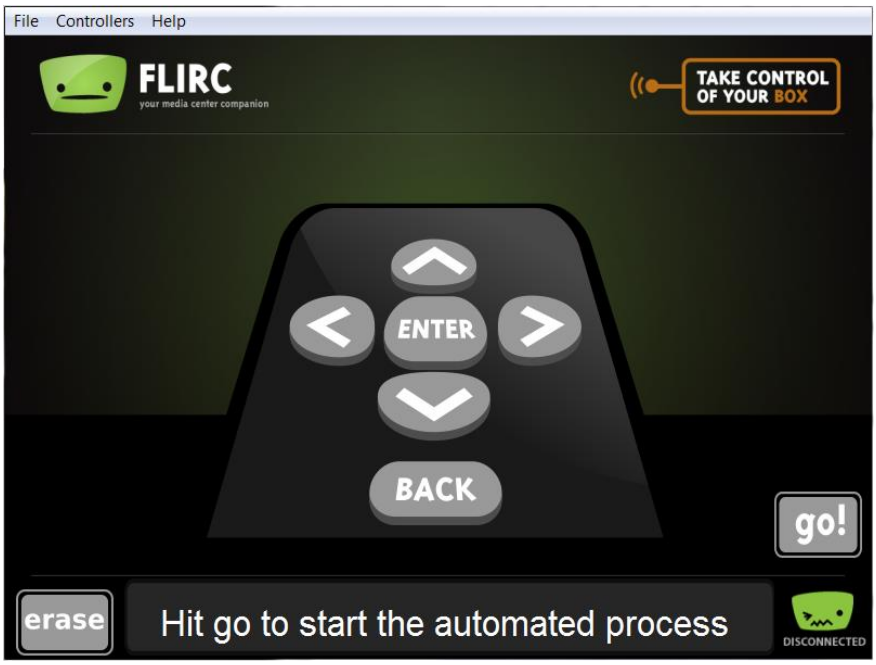


Figure 159 FLIRC setup program

On the **Controllers** drop down menu select the **Full Keyboard** controller.

Figure 160 FLIRC keyboard controller

Now map the buttons on the remote control to the keys shown in Table 16 Graphic screen keyboard command on page 150. 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
```

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 144.

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 start 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 displyed here}
```

To see what version of the software you are running:

```
$ sudo service radiod version
Version 6.12
```

To display information about the running program:

```
$ sudo service radiod info
radiod.py Version 6.12
radiod.py Running PID 1383
Music Player Daemon 0.21.5 (0.21.5)
Raspbian GNU/Linux 10 (buster)
Linux buster2 4.19.63-v7l+ #1249 SMP Thu Aug 1 16:31:35 BST 2019 armv7l
GNU/Linux
```

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

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 15 on page 142.

Table 14 Push Button Operation

	Volume buttons		Channel buttons	
LCD Display Mode	Right button	Left button	Up button	Down button
Mode = TIME **Line 1: Time** **Line 2: Station or Track**	Volume Up	Volume Down	Station/Track up	Station/Track down
Mode = SEARCH **If source = RADIO** **Line 1: Search:** **Line2: Radio Station**	Volume Up	Volume Down	Scroll up radio station	Scroll down radio station
Mode = SEARCH **If source = MEDIA** **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 or Media playlist 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 ,Alarm Time Set (Hours), Alarm Set (Minutes), Streaming:**	Toggle selected mode on or off. Set timer and Alarm	Toggle selected mode on or off. Set timer and Alarm	Cycle through Random, Consume, Repeat, Reload Music, Timer, Alarm , Alarm Time Set and Streaming	Cycle through Random, Consume, Repeat, Reload Music, Timer, Alarm Time Set and Streaming:
Mode = RSS (1) **Line 1: Time** **Line 2: RSS feed**	Volume Up Mute	Volume Down Mute	Station/Track up	Station/Track down
MODE = IP address **Line 1: IP address** **Line 2: Station or Track**	Volume Up Mute	Volume Down Mute	Scroll up through track or radio station	Scroll down through track or radio station

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

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 (Tuner knob depressed) changes the display mode and the functions of the clockwise and anti-clockwise operation of the knobs as shown in the following table.

Table 15 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 or Media playlist 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	Toggle selected mode on or off. Set timer and Alarm	Cycle through Random, Consume, Repeat, Reload Music, Timer, Alarm Time Set, Streaming and Background colour(1)	Cycle through Random, Consume, Repeat, Reload Music, Timer, Alarm , Alarm Time Set, Streaming and Background colour(1)
Mode = RSS (2) **Line 1: Time** **Line 2: RSS feed**	Volume Up	Volume Down	Station/Track up	Station/Track down
MODE = IP address **Line 1: IP address** **Line 2: Station or Track**	Volume Up	Volume Down	Scroll up through track or radio station	Scroll down through track or radio station

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

Mute function

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

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

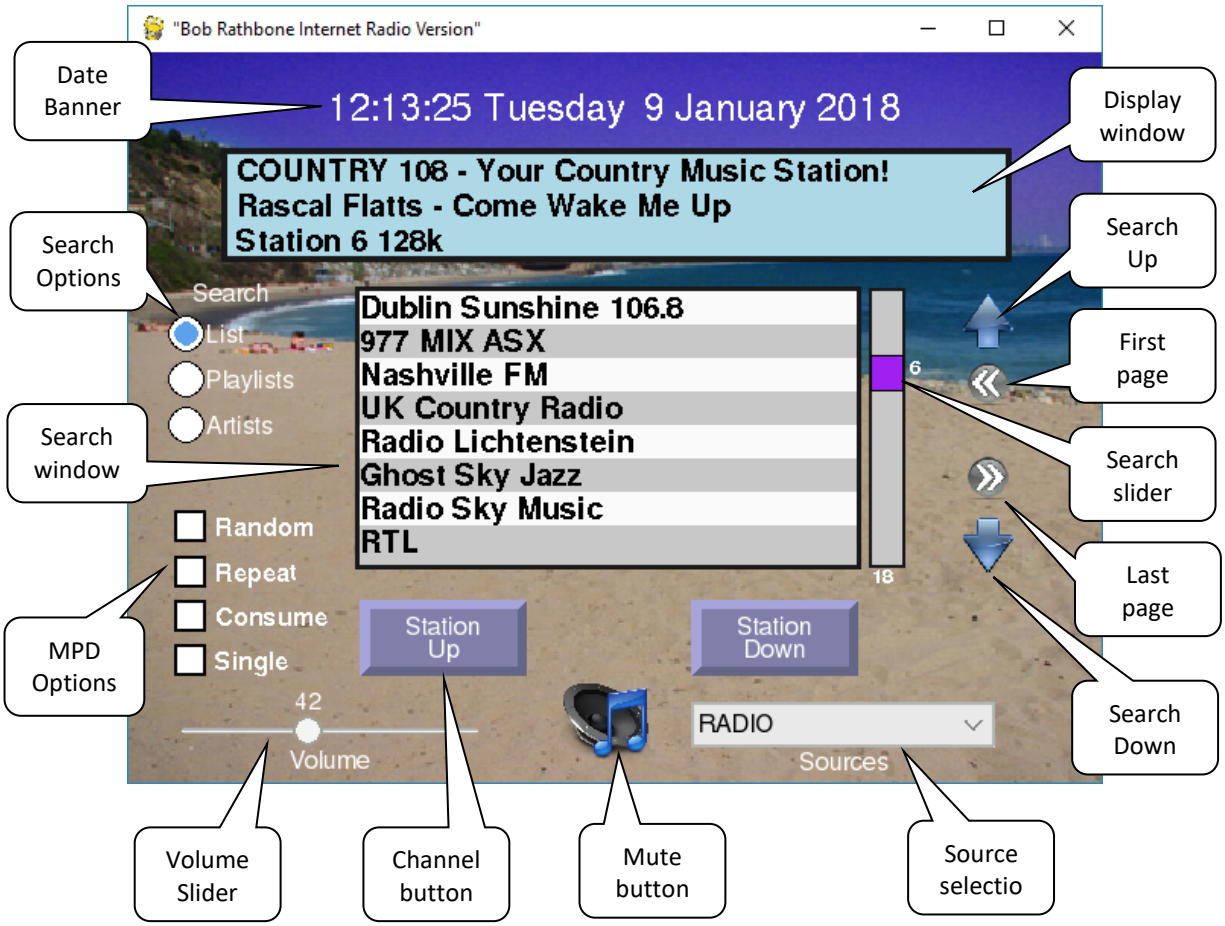
To start the radio from command line run the gradio.py program:

```
$ 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 161 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.

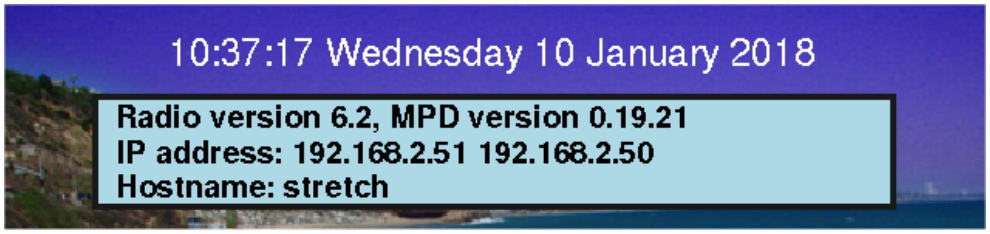


Figure 162 Graphical scree information display

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 163 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 164 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 165 Display playlists

In the following example the USB stick playlist was selected. Once a playlist selected the list is displayed.

Figure 166 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 167 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 161 on page 144.

Artwork display

If the music track has artwork and the **ffmpeg** (See *Setting up the locale* on page 69) 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.

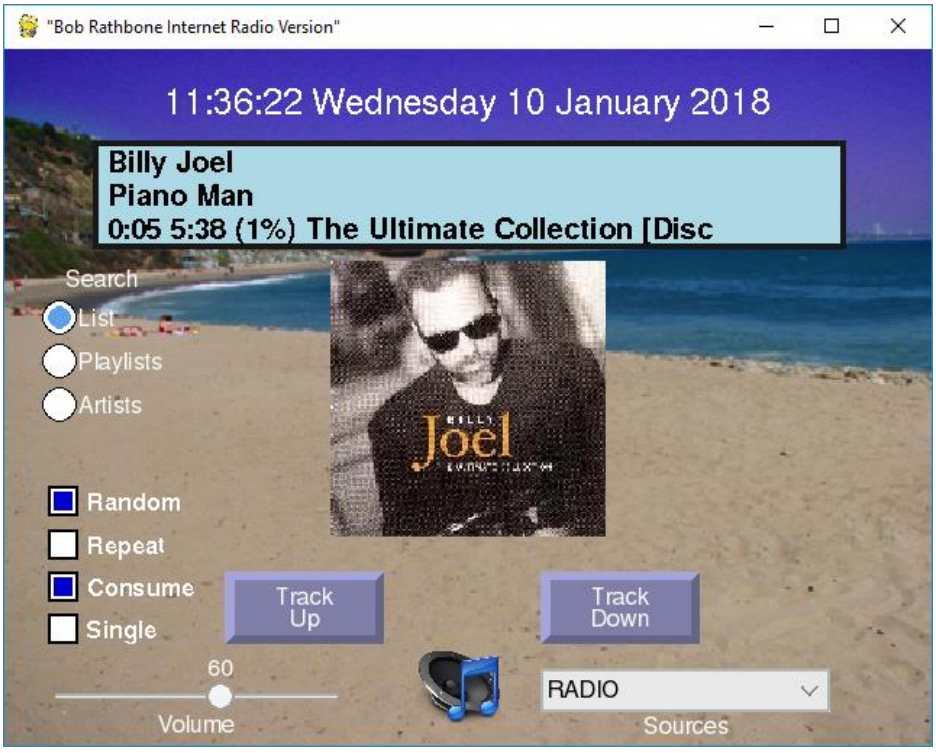
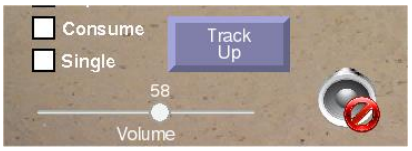


Figure 168 Track artwork display

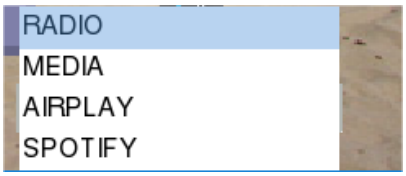
Note that the two grey push buttons now display 'Track Up/Down' instead of 'Station Up/Down'.

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 213 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.

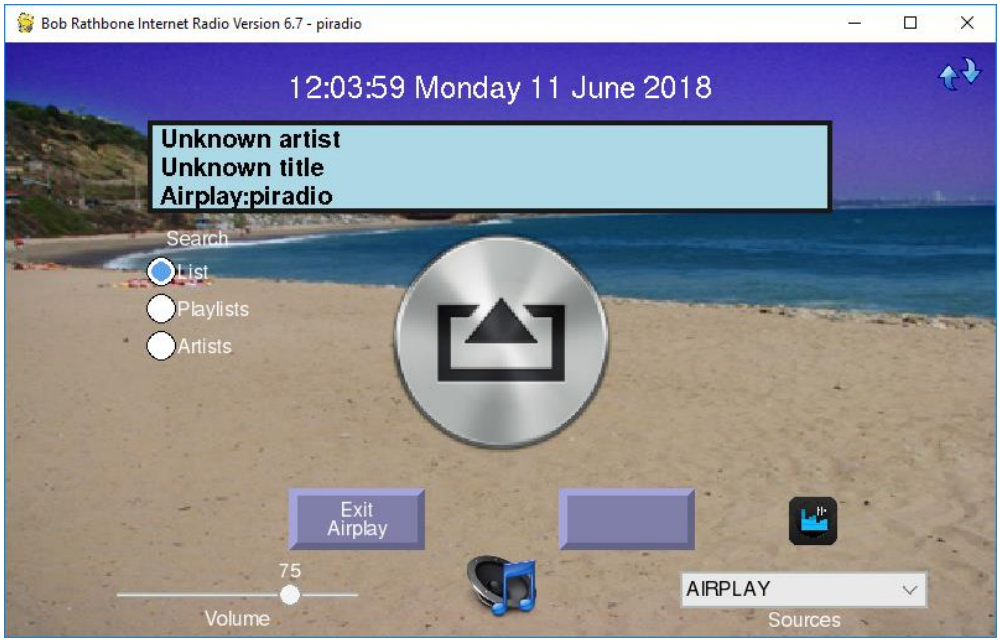


Figure 169 Airplay running on a Graphical screen

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:

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.

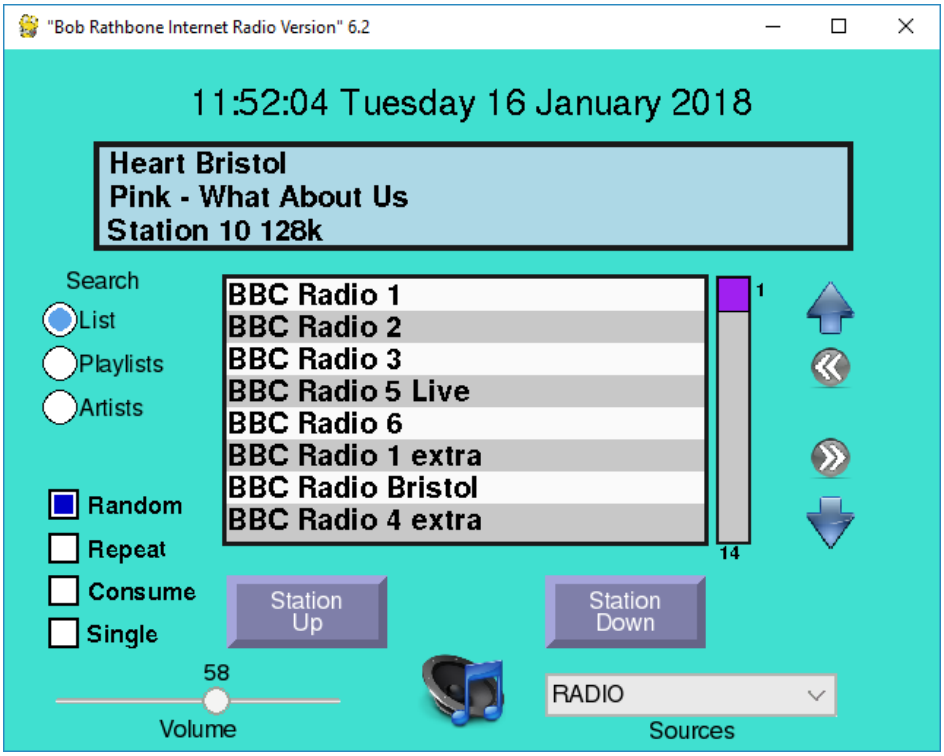


Figure 170 Changing the graphical screen theme

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 16 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 Random	**L**	Select Search List
T	Toggle Repeat	**P**	Select Search Playlists
C	Toggle Consume	**A**	Select Search Artists
S	Toggle Single	**M**	Toggle Mute on/off
D	Cycle display window	**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.

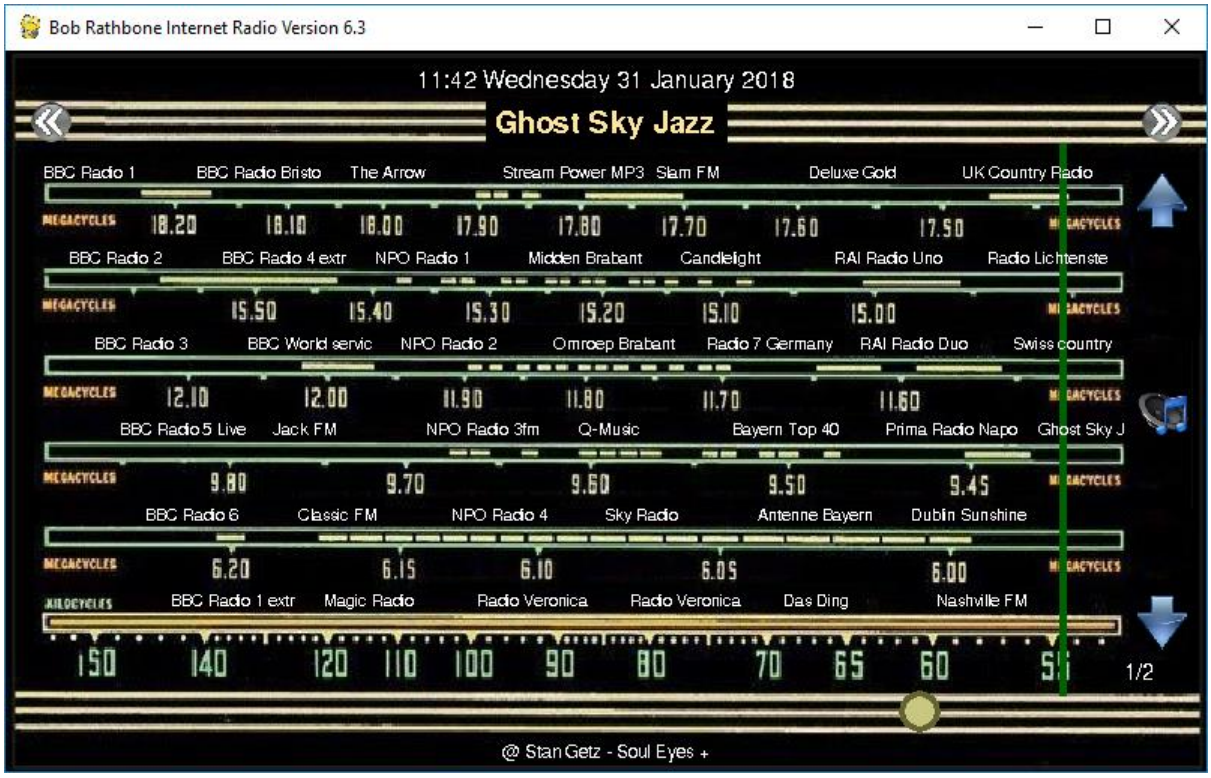


Figure 171 The vintage graphic radio on a touch-screen

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 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 gradio.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

From version 6.4 onwards 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-get 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 160 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 (32GB) SD cards now available music can be stored on in one or more directories on the SD card. It is necessary to first create a directory in **/home/pi** as user **pi** and then link it in the **/var/lib/mpd/music/** directory. Carry out the following instructions as user **pi** to create **mymusic** for example:

```
$ mkdir /home/pi/mymusic
```

Using FTP, copy the music from a PC to the **/home/pi/mymusic** directory and reload the library via the options menu. Now run the **create_playlist.sh** program.Select option 3 (SD card).

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 *172*.

Organising the music files

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

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

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

MPD Logging

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

Radio program logging

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

```
2019-08-05 08:56:20,999 INFO ===== Starting radio =====
2019-08-05 08:56:23,919 INFO Board revision 2
2019-08-05 08:56:23,942 INFO OS release: Raspbian GNU/Linux 10 (buster)
2019-08-05 08:56:23,955 INFO Linux buster2 4.19.63-v7l+ #1249 SMP Thu Aug 1
16:31:35 BST 2019 armv7l GNU/Linux
2019-08-05 08:56:23,957 INFO radio.startMpdDaemon: Starting MPD
2019-08-05 08:56:25,541 INFO radio.startMpdDaemon: MPD started pid=1502
2019-08-05 08:56:25,546 INFO Connected to MPD port 6600
2019-08-05 08:56:25,852 INFO UDP Server listening on localhost port 5100
2019-08-05 08:56:25,860 INFO UDP listen:remote 0.0.0.0 port 5100
2019-08-05 08:56:25,861 INFO IP 192.168.1.152 192.168.1.153
2019-08-05 08:56:34,967 INFO Radio running pid 1383
2019-08-05 08:56:34,969 INFO Radio ['/usr/share/radio/radiod.py',
'nodaemon'] Version 6.12
```

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

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

Configuration and status files

The main configuration file is **/etc/radiod.conf**. See section *A.1 Files added to the system* on page 242. 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 215)
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

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 the it uses the last settings, for example, the volume setting.

Using the Timer and Alarm functions

Note: The Rasbian 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).

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 THERFORE RELY SOLEY ON THIS ALARM FUNCTION IF YOU HAVE AN IMPORTANT APPOINTMENT OR A PLANE OR TRAIN TO CATCH FOR EXAMPLE. ALSO SEE DISCLAIMER ON PAGE 235.

Using the Alarm and Timer functions together

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

Music Player Clients

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

Using the MPC client

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

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

```
$ mpc play
```

To see a list of all available commands, run:

```
$ mpc help
```

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

Table 17 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 Amircan 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 existing play-lists, see the section called *Using old 5.x Radio playlists* on page 161.

Previous 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 18 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. All radio stations begin with an underscore "_". The reason for this is that the radio program has to handle and display Radio and Media playlists differently. The "_" at the beginning of the playlist file name identifies the playlist as a Radio playlist. It also has the added advantage that it puts all the Radio playlists at the beginning of the list of available playlists.

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 167. 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 19 Playlist files and directories

Name	Type	Description
/usr/share/radio/station.urls	File	Initial distribution file containing sample stations
/var/lib/radiod/stationlist	File	The file containing the users list of radio stations
/usr/share/radio/station.urls	File	Initial distribution Radio playlist. This is copied to /var/lib/radiod/stationlist 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** and will contain the title and URLs for each station. Now add or remove radio station definitions in the **stationlist** file. The first statement in the station definition is the name of the playlist in brackets:

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

After modifying the **stationlist** file run the *create_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).

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
```

This 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
```

Finally there is a help parameter:

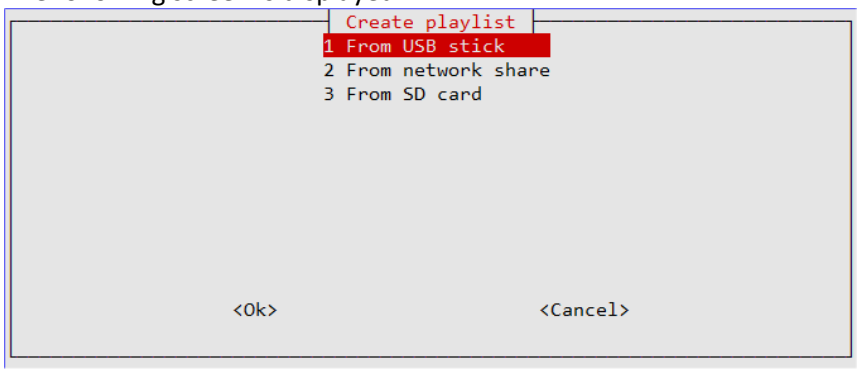
```
$ sudo ./create_stations.py --help
```

Creating Media playlists

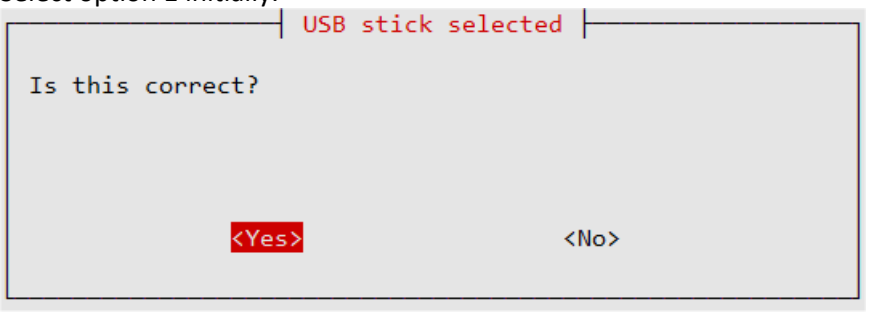
The radio program comes with a program called create_playlist.sh. This creates a single playlist for a USB stick, SD card location or a network share.

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

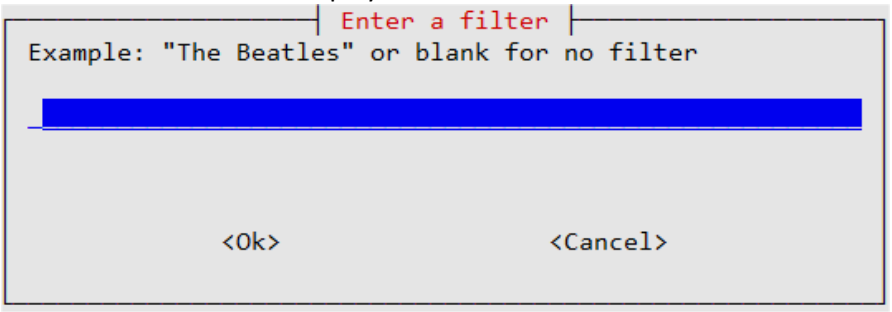
The following screen is displayed.



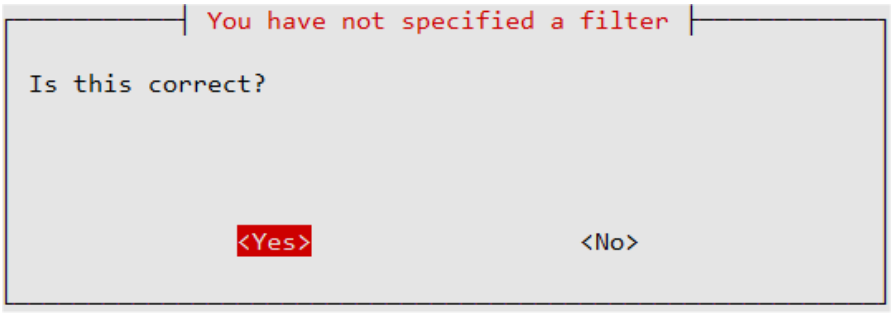
Select option 1 initially.



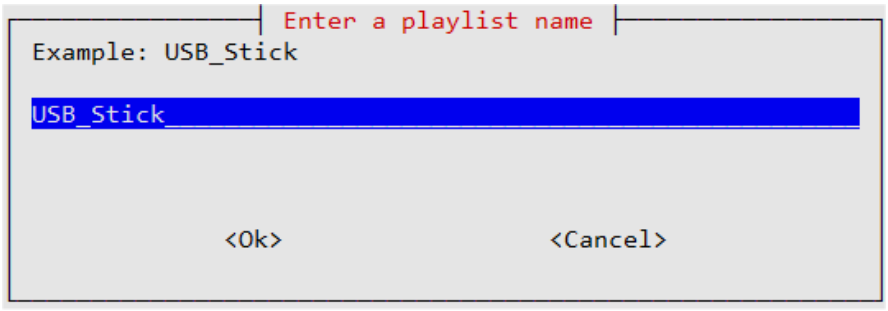
Answer “Yes” to create the playlist from the USB stick.



Enter a filter name or enter for no filter.



Select Yes to continue.



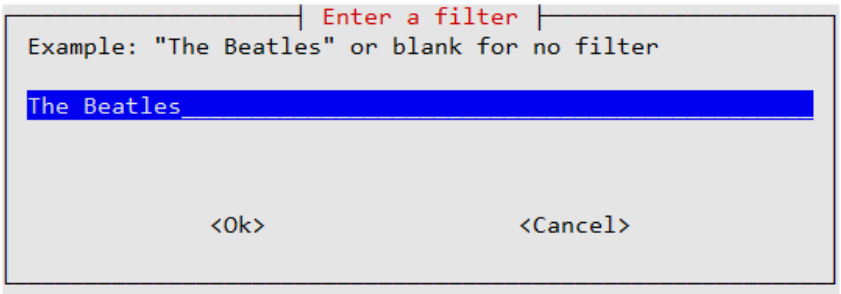
The program will suggest a name for the playlist but you may choose any name (But do not make it too long).

The program continues by creating a playlist called **USB_stick.m3u** in **/var/lib/mpd/playlists**.

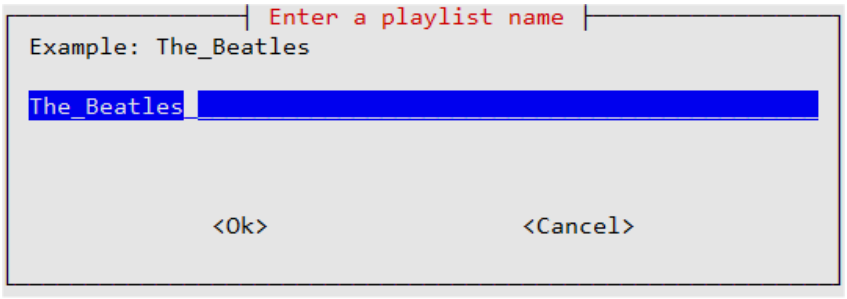
```
sudo service radio stop
sudo service mpd stop
Mounted  /dev/sda1 on /media
cd /var/lib/mpd/music/
/var/lib/mpd/music
sudo find -L media -type f -name *.mp3 -or -name *.ogg -or -name *.flac  >
/tmp/list29073
sudo mv /tmp/list29073 /tmp/USB_Stick

============================================================
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.

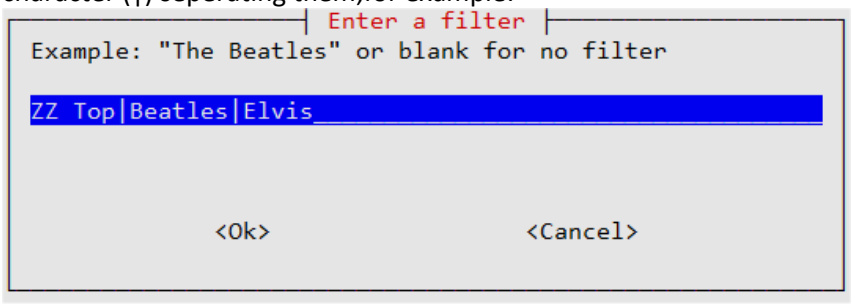


You will note that spaces in the playlist name have been replaced with underscores(_).
This is just for the file name. When the playlist is displayed in the radio program the underscores will be converted back to spaces. The program will now create a playlist with the name **The_Beatles.m3u** (or whatever 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 (|) seperating them,for example:

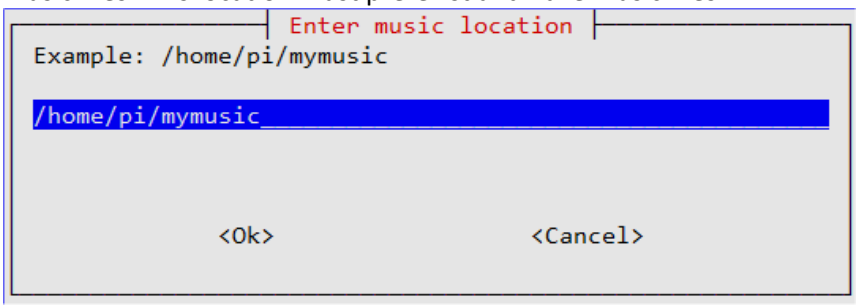


This will filter all songs from ZZ Top, The Beatles and Elvis or any other titles that contain these names. However this may not be want is wanted. Maybe songs by Elvis are wanted and not songs with 'Elvis' in the title. For example *Dire Straits – Calling Elvis*. In such a case use the **/** character 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 you music files. This location must pre-exist and have music files.



Accessing Shoutcast

From version 6.5 onwards 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 disapointed. 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:

```
$ cd /usr/share/radio
$ ./get_shoutcast.py
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 availble genres.
```

If you see the following message then install python-requests as shown in the section called *Install python-requests* on page 72.

```
Traceback (most recent call last):
  File "./get_shoutcast.py", line 19, in <module>
    import requests
ImportError: No module named 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=anCLSEDQODrElkxl&limit=50&genr
e=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=anCLSEDQODrElkxl
```

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.

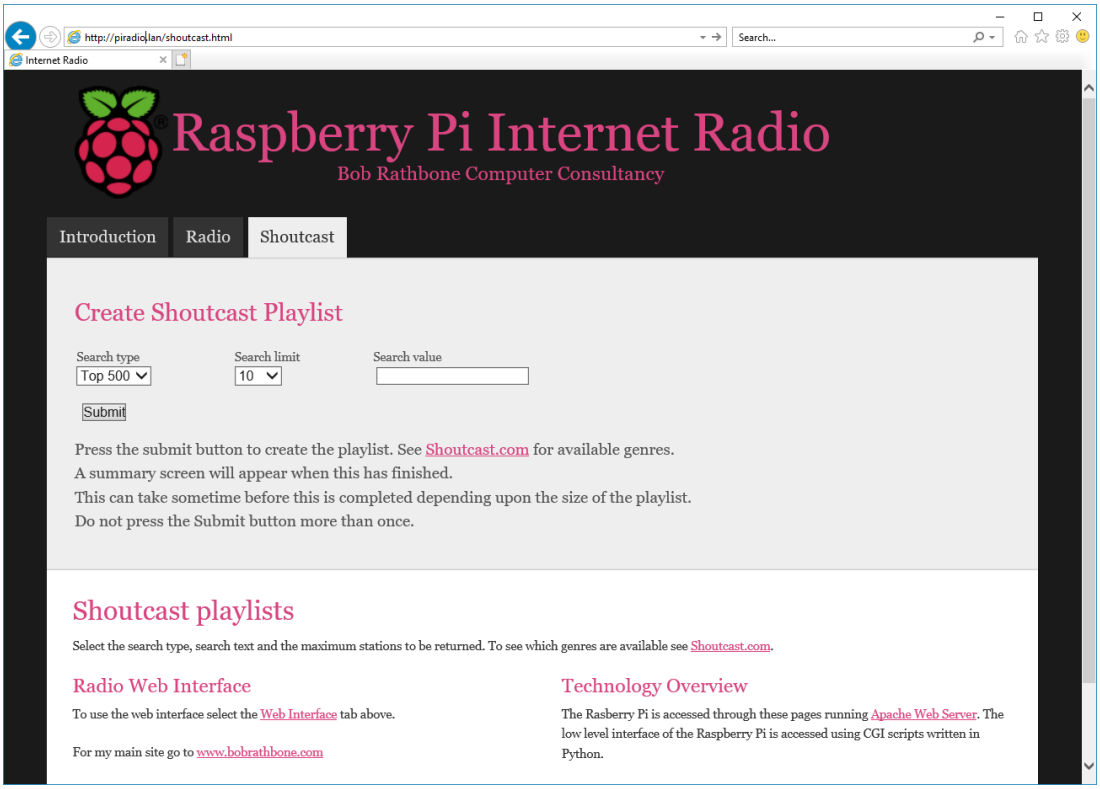


Figure 172 Shoutcast playlist web page

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.

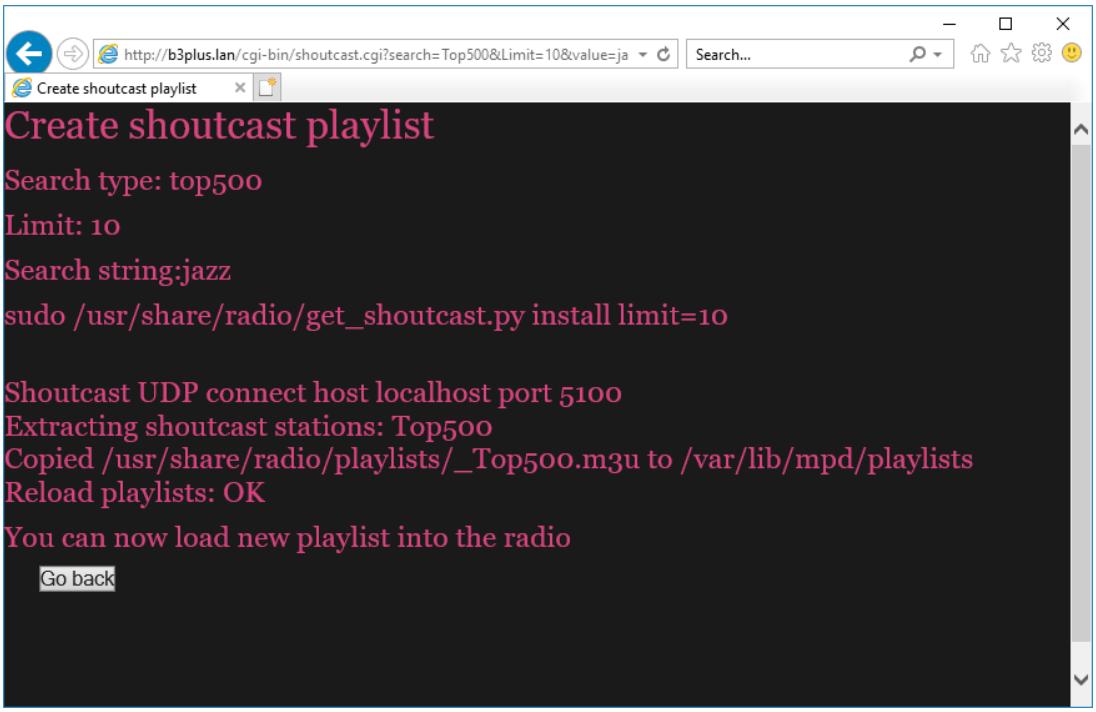


Figure 173 Shoutcast playlist summary

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.

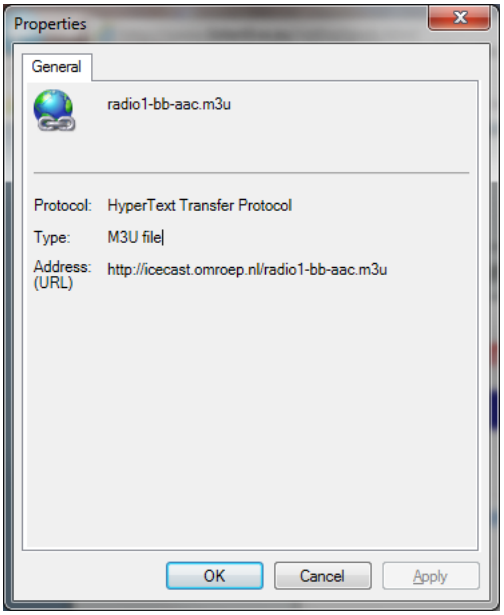
If you find upon running the Radio that you have a lot of radio playlists. Reduce the number by removing title statements in the **stationlist** file as previously mentioned. These are the names in brackets – for example (BBC Radio). Re-run the **create_stations.py** program.

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

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

1, 2 and 3 are so called redirector URLs and point to a playlist file containing one or more URLs to the radio stream(s) itself. The *create_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 Files

M3U stands for MPEG3 URL. This is the format that MPD itself uses. The following Wikipedia article explains the M3U file format:

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

The Music Player Daemon uses m3u files. An example M3U file is shown below:
radio10.m3u playlist file

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

These playlist files have the m3u file extension. i.e. **<filename>.m3u**. The first line is the header and must be #EXTM3U. The second line is #EXTINF: and is information about the radio stream. The -1 means unlimited play length. This is followed by a comma and then the name of the radio station (*Radio 10 Gold* in this case). The third line is the URL (icecast in this case) for the radio stream. More than one radio stream may be defined in the m3u file. 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)

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&amp;SSO
2-UID=" />
        </Entry>
</ASX>
```

Direct stream URLs

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

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

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

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

```
Saving to: `radio1-high.mp3'

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

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

Listening to live Air Traffic Control (ATC)

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

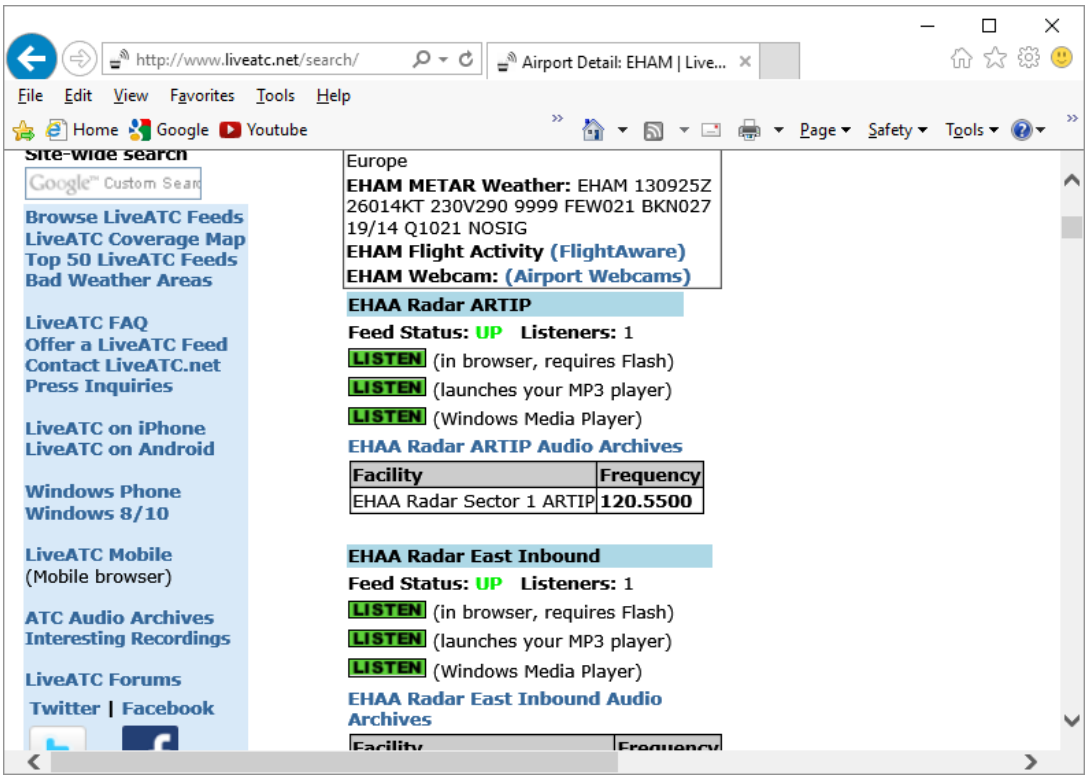


Figure 174 Live ATC web page

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 175 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.

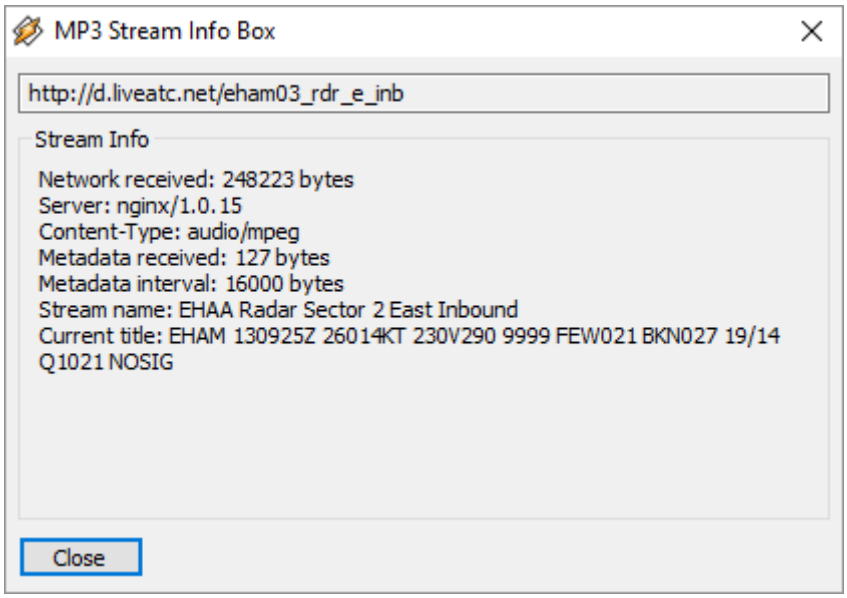


Figure 176 WinAmp station information

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/eheh2_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	13th of the current month. Time 09:55 Zulu (UTC)
25016KT	Wind 250 degrees at 16 Knots
9999	Visibility 10 Kilometres or greater
FEW020	Few clouds at 2000 feet
BKN024	Broken cloud at 2400 feet
19/14	Temperature 19 degrees Celsius. 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 was via a product called SAMBA but has been largely replaced by the mount using the CIFS option in the Linux. The steps to mount the network drive are as follows:

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

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

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

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

The general syntax for the mount command is as follows:

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

Where: <type> is either **nfs** or **cifs**.
-o option1,option2 are the mount options.
<remote IP address and directory> Is the IP address and music directory path
<mount point> This will always be /share for this program

Finding the IP address of the network drive

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

The CIFS mount command

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

```
mount -t cifs -o username=guest,password=guest,uid=pi,gid=pi,vers=1.0
//192.168.2.6/music /share
```

The above command is all on one line. The **uid** and **gid** parameters set the ownership of the music files to user **pi**. The **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 173.

Older NAS drives sec security option

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

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

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

The NFS mount command

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

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

A few things to note here; the NFS mount command uses the volume name (volume1 – can vary), The CIFS mount command doesn't. The second thing is that the IP address and remote directory are separated by a colon (:). If the command was successful you should be able to display the music from the network drive.

Display the share directory

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

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

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

Un-mounting the /share directory

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

```
# umount /share
```

Copy the mount command to the configuration

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

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

The above command is all on one line.

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

Load the music library

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

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

Update the playlists for the new share

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

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

Disabling the share

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

```
# mount -t cifs -o username=guest,password=guest,vers=1.0
//192.168.2.6/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 **sdcard** 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 sdcard -> /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 sdcard
```

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 186.

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.

Keep pressing the back button to exit and then re-start the MPDdroid App. The play screen should be displayed as shown below. Volume, pause, fast forward/back can all be controlled from this screen.

To switch to the play list drag the play screen to the left. The current station list or play list will be displayed. Tap on the desired station or track to play it. Drag the play screen to the right to return to the play screen.

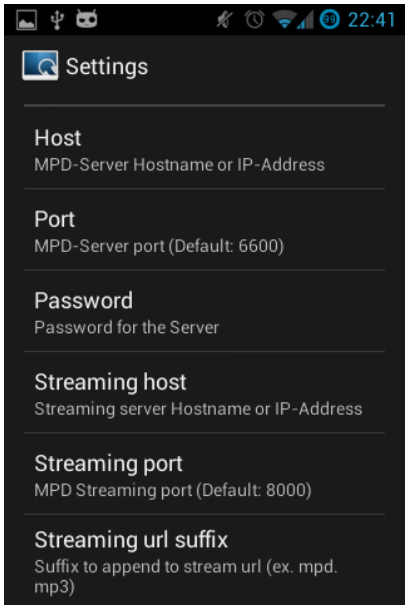


Figure 177 MPDdroid set-up screen

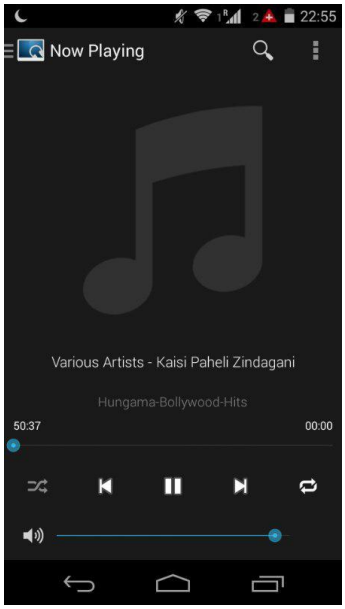


Figure 178 MPDdroid play screen

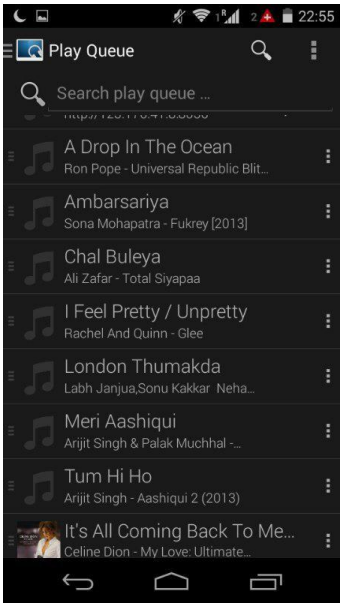


Figure 179 MPDdroid play queue

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 194. If you need to create a log file in DEBUG mode see the procedure for doing this on page 199.

Installation problems

The Raspberry Pi will not boot

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

With Raspbian Busteryou may 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
```

Missing package dependency problems

If an attempt is made to install the radio software without first installing **mpd** and **mpd-python** the following will be seen:

```
dpkg: dependency problems prevent configuration of radiod:
 radiod depends on python-mpd; however:
  Package python-mpd is not installed.
radiod depends on mpc; however:
  Package mpc is not installed.
radiod depends on mpd; however:
  Package mpc is not installed.
```

To correct this first install the missing packages as shown in the section *Music Player Daemon Installation* on page 74. Now run the following:

```
$ sudo apt --fix-broken install
```

Now re-run the radio software package installation.

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 197). Adapt either the configuration or wiring to match each other. The configuration in **/etc/radiod.conf** uses GPIO numbers and not physical pin numbers.

Unexpected message during an upgrade

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

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

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

Network problems

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 WiFi 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 WiFi and Ethernet interfaces and the IP address of the Router (192.168.1.254).

If the problem is with WiFi 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
}
```

A very handy tool for checking devices connected to your network is the **Fing** App. This runs on both Android and Apple devices. See https://www.fing.com/products/fing-app

HDMI/Touchscreen problems

HDMI/Touchscreen does not start

Is it already running but not displaying? Stop it with the following command:

```
$ sudo /usr/share/radio/gradio.py stop
```

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 75. Select the option to start the program at boot time.

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
```

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 waring.

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 73.

The graphic version of the radio does not start automatically

This only applies is due to a missing **autostart** file in the Raspbian desktop released in November 2018 and only applies to version 6.9. Either upgrade to version 6.12 or carry out the following:

1) Install patch **radiod-patch-6.9-3.tar.gz** or later as shown in *Apply patches to the radio software* on page 91.
2) Re-run the **configure_radio.sh** program as shown in *Configuring the radio* on page 75.
3) Re-boot the Raspberry Pi.

Trouble shooting problems with MPD

Most problems are due to the following:

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

Re-run the **configure_audio.sh** program as shown in the section *Configuring the audio output* on page 84. 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 *Music Player Daemon Installation* on page 74.

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 92.

MPD fails to install

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

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

This is due to that an update was not previously carried out as shown in the section called *SD card creation* on page 63. 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 157.

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.
```

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

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.

LCD Problems

LCD screen not showing anything

Check that the wiring conforms to the *wiring list* on page 19. 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 75.

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 197).

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 19. In particular check the data lines to pins 11, 12, 13 and 14 (See LCD wiring on page 26). 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 197).

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 55.

LCD backlight not working

Check that pins 15 and 16 of the LCD display have +5V and 0V (GND) respectively.
See on LCD Module Wiring on page 27.

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 23. 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 19). **configure_radio.sh** script to select the correct radio daemon. Run the **wiring.py** program (See page 197).

The LCD displays the message "No playlists"

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

```
$ ls  /var/lib/mpd/playlists/
_Air_traffic.m3u  Network.m3u  _Radio.m3u  USB_Stick.m3u
```

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

```
$ sudo ./create_stations.py -delete_old
:
:
```

```
New radio playlist files will be found in /var/lib/mpd/playlists/
```

Restart the radio.

Constant alternate display of Station Name and Volume

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

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

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
```

I2C and SMBUS problems

Import errors

The following is seen:

```
$ sudo ./radiod.py nodaemon
Radio running pid 825
:
ImportError: No module named smbus
```

The cause is that the **python-smbus** package has not been installed as shown in the section *Installing the Python I2C libraries* on page 85.

The following is seen:

```
$ sudo ./radiod.py nodaemon
Radio running pid 825
:
ImportError: No module named PIL
```

The **python-pil** package has not been installed as shown in the section *Install libraries for the Olimex OLED* on page 72.

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 87. Also update the RPi firmware by running **rpi-update**.

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 199.

```
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 72.

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
```

Rotary encoder problems

Run the test programs shown in the section called Testing rotary encoders on page 196.
These programs display the configuration found in **/etc/radiod.conf**. 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).

Button problems

This section applies to push button radios only.

Buttons seem to be pressing themselves

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

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 157.

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 nolock' to keep locks local, or start statd.
mount.nfs: an incorrect mount option was specified
Cause:
You need to include the "–o noclock" option.
If the error isn't in the above list then search the web for suggestions.

Sound problems

Noisy interference on the radio

If there is noise interference when playing the radio and this is still present even when the radio is muted this can be for several reasons. This can happen with a wired Ethernet connection and a Wi-Fi dongle are connected to the Raspberry Pi and the Ethernet activity is being picked up by the Wi-Fi dongle. This can be cured by using either a wireless adapter or the Ethernet connection and not both. Another common cause can be an inadequate power supply. See on Power supply considerations on page 29. 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 56.

Music is first heard at boot time then stops and restarts

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

```
$ sudo systemctl disable mpd
```

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

USB sound device won't play

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

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

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

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

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

HiFBerry or other types of DAC no sound

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 84.

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 21 Sound card Device Tree overlays* on page 255. 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 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:

device "bluealsa:DEV=<Your Bluetooth device ID>,PROFILE=a2dp"

```
audio_output {
        type            "alsa"
```

```
        name            "SP-AD70-B"
        device          "bluealsa:DEV=00:75:58:41:B1:25,PROFILE=a2dp"
        format          "44100:16:2"
        mixer_type      "software"
#       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.

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 --noplugin=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 --noplugin=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 hciuart daemon

```
$ systemctl status hciuart
● hciuart.service - Configure Bluetooth Modems connected by UART
   Loaded: loaded (/lib/systemd/system/hciuart.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 (hciattach)
    Tasks: 1 (limit: 1599)
   Memory: 1000.0K
   CGroup: /system.slice/hciuart.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 97.

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:

```
$ 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>
```

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-get 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.**

```
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 157.

Radio daemon doesn't start or hangs

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

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

```
    inet 192.168.2.22/24 brd 192.168.2.255 scope global eth0
```

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 157.

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.

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 102.

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_contol.py program (Called by service **irradiod**)

```
$ cd /usr/share/radio/
$ sudo ./remote_control.py nodaemon
remote control running pid 13299
```

Make a note of the pid (in this example it is 13299). 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 13299
```

If the **remote_control.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
```

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_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

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**
- **display_config.sh** Display the current configuration.

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. All of those using the GPIO interface also require to be called with **sudo**.

Running the radio program in diagnostic mode

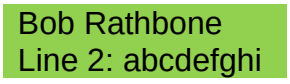
This is one of the first things you should learn to do. Running the **radiod.py** program in **nodaemon** mode will display any errors it encounters. Use Ctrl-C to exit the program.

```
$ sudo ./radiod.py nodaemon
```

Using the LCD test code

```
$ sudo ./lcd_class.py
```

The above program will display the following text on the LCD:



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.

```
$ sudo ./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.

```
$ sudo ./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

```
$ sudo ./rotary_class_alternative.py
```

The remote_control program

The **remote_control.py** program listens on the IR interface for commands from the Remote Control. The **remote_control.py** program is started from the **irradiod** service. It then passes commands to the radio program. The **remote control** program provides complete control of the radio and can change menu options, do searches etc. just as the same as the knobs or buttons. It normally communicates with the radio program using UDP port 5100 on the local network interface.

The display_model program

This program displays the Raspberry PI model details.

```
$ ./display_model.py
000e: Model B, Revision 2.0, RAM: 512 MB, Maker: Sony
```

In this example 000e=Manufacturers revision, B=Model, 2.0=Revision, 512MB RAM, Maker=Sony. If you are unsure of the model or revision of the Raspberry PI use this program to find this out.

The display_current program

This is a useful diagnostic that prints out the raw information available from the MPD daemon. The radio daemon uses the same libraries as this test program.

```
$ ./display_current.py

id: 32
pos: 7
name: Blues Radio UK
file: http://206.217.213.16:8430
title: 04 Billy Jones Blues - I'm A Bluesman
current_id 8
```

```
Status
songid: 32
playlistlength: 25
playlist: 31
repeat: 0
consume: 0
mixrampdb: 0.000000
random: 0
state: play
xfade: 0
volume: 75
single: 0
mixrampdelay: nan
nextsong: 8
time: 22:0
song: 7
elapsed: 22.400
bitrate: 96
nextsongid: 33
audio: 32000:24:2

uptime: 28
db_update: 1400354144
artists: 228
playtime: 23
albums: 132
db_playtime: 302046
songs: 1297
```

To find out the exact meaning of all these fields please refer to the standard **python-mpd** documentation at https://pypi.python.org/pypi/python-mpd/ or http://pythonhosted.org/python-mpd2/topics/getting-started.html.

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 5 on page 24) 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 display configuration program

The **display_config.sh** has been added in version 6.10 onwards. When run it 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 buster2 Mon  5 Aug 09:13:31 BST 2019

OS Configuration
----------------
PRETTY_NAME="Raspbian GNU/Linux 10 (buster)"
NAME="Raspbian GNU/Linux":
:

============= End of run ====================
This configuration has been recored 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 service radiod stop
$ sudo service irradiod stop
```

Edit the **/etc/radiod.conf** file and switch on DEBUG mode as shown below.

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

```
loglevel=DEBUG
```

Remove the old log file:

```
$ sudo rm /var/log/radio.log
```

Start the radio and remote control if required:

```
$ sudo service radiod start
$ sudo service irradiod start
```

Operate the radio including the operation you are having with.
Send the **/var/log/radio.log** file to bob@bobrathbone.com

Switch off DEBUG mode by editing the **/etc/radiod.conf** file and as shown below.

```
loglevel=INFO
```

Displaying information about the Raspberry Pi

There are a number of standard facilities to provide information about the Raspberry Pi which may be useful when diagnosing a problem.

Displaying information about the Operating system

Display **/etc/os.release** using the **cat** command

```
$ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 10 (buster)"
NAME="Raspbian GNU/Linux"
VERSION_ID="10"
VERSION="10 (buster)"
VERSION_CODENAME=buster
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"
```

Display the kernel details

Display the kernel version using **uname**.

```
$ uname -a
Linux raspberrypi 4.19.63-v7l+ #1249 SMP Thu Aug 1 16:31:35 BST 2019 armv7l
GNU/Linux
```

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  | CE0     | 10 | 8  |
 |     |     |      0v |      |   | 25 || 26 | 1 | OUT  | CE1     | 11 | 7  |
 +-----+-----+---------+------+---+----++----+---+------+---------+----+----+
 |  28 |  17 | GPIO.17 | ALT2 | 0 | 51 || 52 | 0 | ALT2 | GPIO.18 | 18 | 29 |
 |  30 |  19 | GPIO.19 | ALT2 | 0 | 53 || 54 | 0 | ALT2 | GPIO.20 | 20 | 31 |
 +-----+-----+---------+------+---+----++----+---+------+---------+----+----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+-Model B2-+---+------+---------+-----+-----+
```

* Alternative down switch for HiFiBerry DAC+ compatibility.

The physical pins are shown in the center numbered 1 to 26 and 51 through 54 (for a model B2) in this example. The above output is for a radio using a directly wired LCD and push buttons. For example:
Physical pin 8 is BCM GPIO 14 and mode is configured as an input for the left switch and is currently low (Column V is 0).

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 **gmpc** on the PC. It cannot be easily accessed from a web browser. If you wish to use the inbuilt streamer see the following URL:
http://mpd.wikia.com/wiki/Built-in_HTTP_streaming_part_2

Introduction to Icecast

You may wish to play the output of the Radio through your PC speakers or a mobile device such as a tablet or telephone. This is possible to do this using **Icecast**. For more information on **Icecast** see the following Wikipedia article: http://en.wikipedia.org/wiki/Icecast .

Please also refer to *Intellectual Property, Copyright, and Streaming Media* on page 234.

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 progra 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:

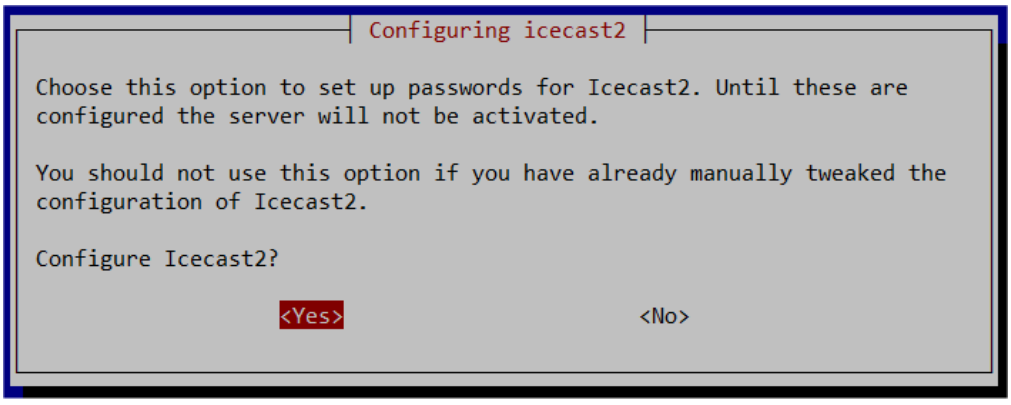


Figure 180 Configuring Icecast2

Answer '**yes**' to this. Configure Icecast as follows:

Icecast2 hostname: **localhost**
Icecast2 source password: **mympd**
Icecast2 relay password: **mympd**
Icecast2 administration password: **mympd**

It is important that you replace the default password 'hackme' with 'mympd' and that you leave the Icecast2 hostname as 'localhost'. The installation program continues configuration. The icecast2 server will be started:

```
Done Configuring icecast2..
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
```

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
```

This completes the installation of Icecast2 however you may need to configure the clock speed.

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:

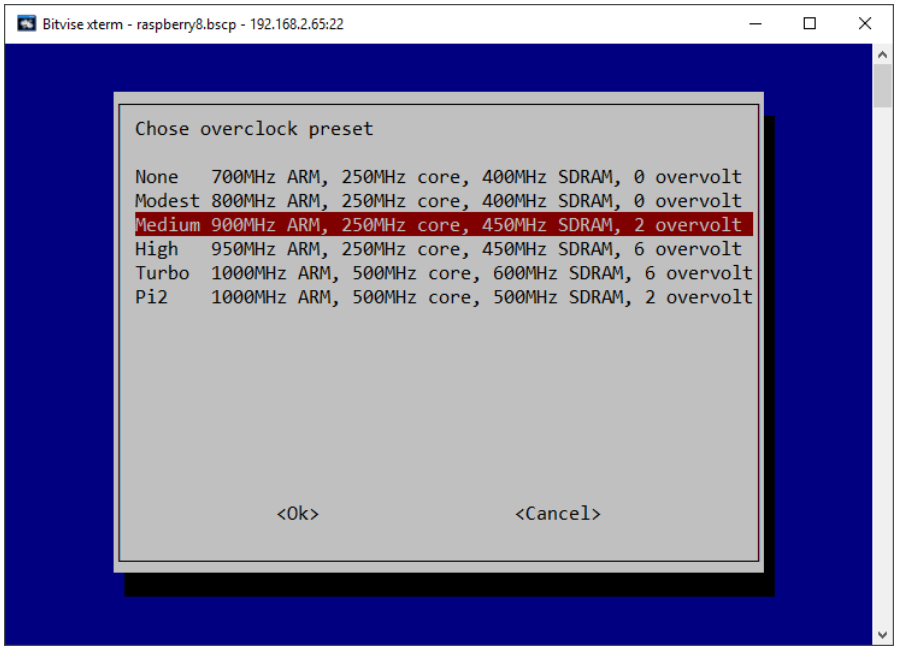


Figure 181 Over-clocking the Raspberry PI

Select 'Medium' to start with. Reboot the Raspberry PI when prompted. Re-test the radio with streaming switched on.

Icecast2 Operation

The radiod daemon has full control over the Icecast2 service and stops and starts it as required. When the radio is first switched on the Icecast2 streaming service will not normally be enabled unless it was enabled as shown below by an earlier run of the radio software.

Switching on streaming

Before you can listen to the streaming on the PC or mobile device it is necessary to start the Icecast2 streaming daemon. It must be switched on first.

Use the options menu (Press menu button three times). Step through the menu option using the Channel up/down buttons until "**Streaming off**" is displayed in the LCD display (assuming Icecast is installed). Press either Volume button and after a short delay the text should change to "Streaming on" in the LCD display. Press the menu button again to exit the options menu.

This starts the Icecast2 service. It also writes the word "on" or "off" to a file called **/var/lib/radiod/streaming**. This is file is used to enable or disable the Icecast streaming function at boot time.

Starting Icecast2 manually

Use the following command:

```
$ sudo service icecast2 start
```

To stop it again:

```
$ sudo service icecast2 stop
```

Enabling Icecast2 at reboot time

It isn't necessary to enable the Icecast2 service at boot time as the radio program will start it depending on the contents of the **/var/lib/radiod/streaming** file. Should you wish to enable streaming at boot time then enable it with the following command:

```
$ sudo update-rc.d icecast2 enable
```

Or

```
$ 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.

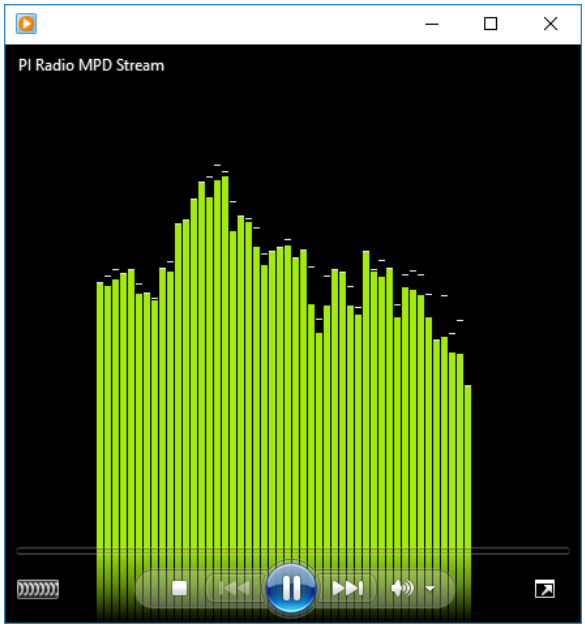


Figure 182 Windows media player

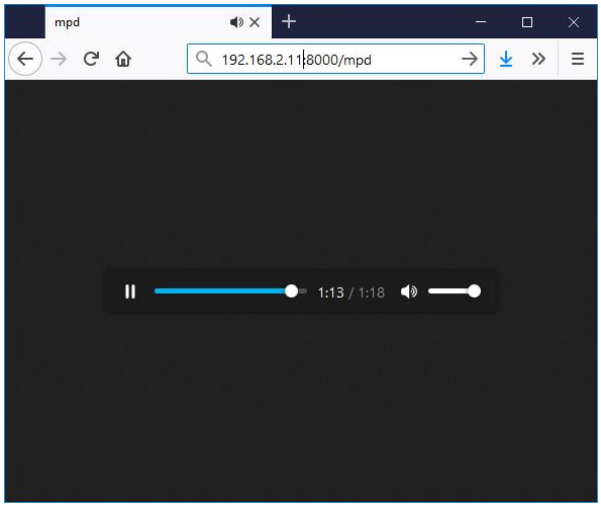


Figure 183 Firefox embedded media player

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 Adminstration 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:

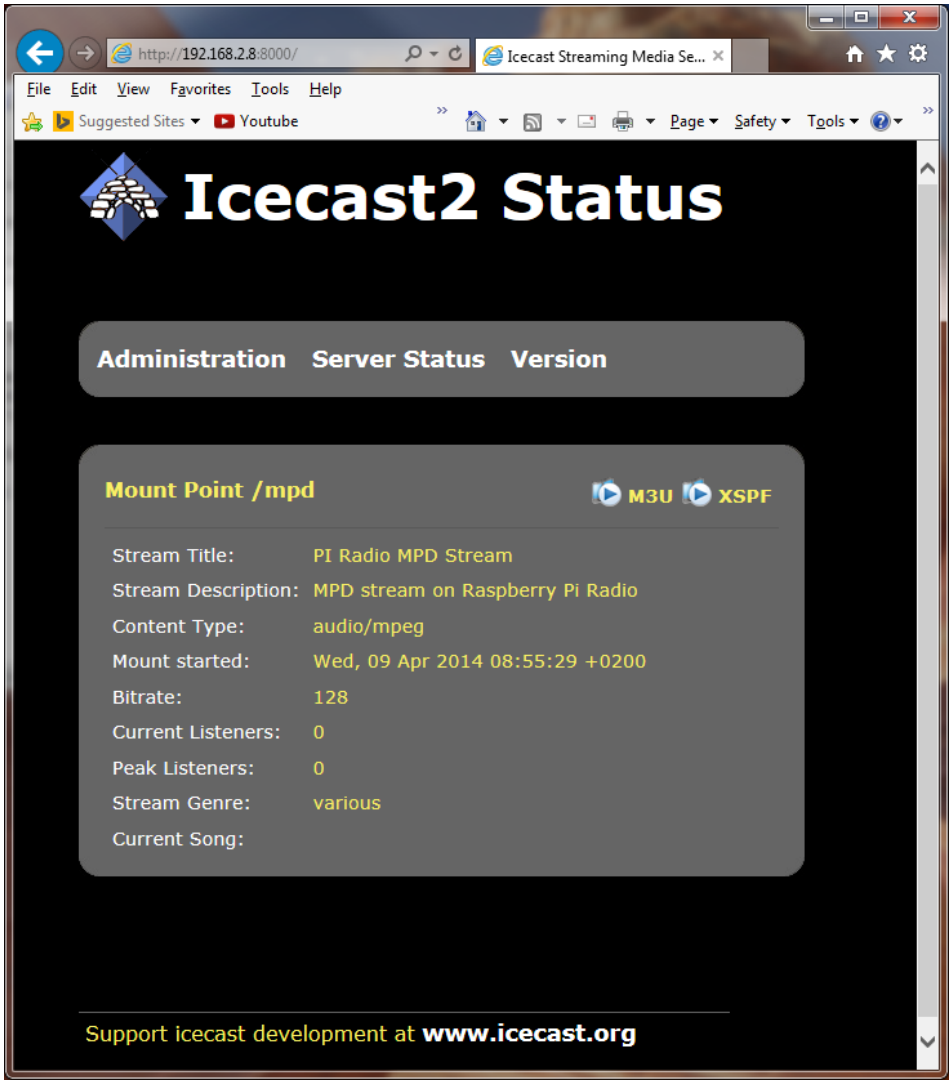
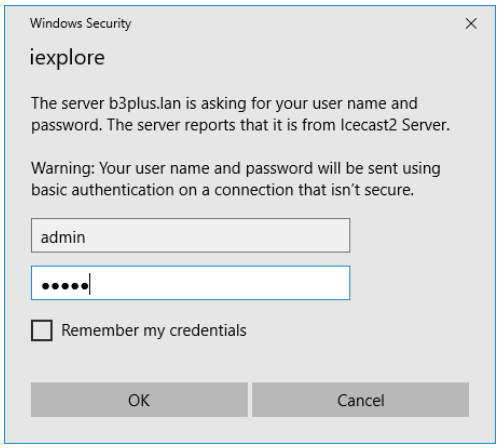


Figure 184 Icecast2 Status

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 *

Troubleshooting Icecast2

Help for general problems with icecast2 can be found on the forums at http://www.icecast.org/

Icecast2 has two log files in the **/var/log/icecast2** directory namely **access.log** and **error.log**. The error log may give a clue as to the problem.

Below is a simulated error caused by mis-configuring the shoutcast entry in /etc/mpd.conf file. Here the hostname 'piradio' has been configured in the **/etc/mpd.conf** shoutcast entry instead of 'localhost'.

```
$ tail -f /var/log/mpd/mpd.log
Apr 07 10:43 : output: Failed to open "PI Radio MPD Stream" [shout]: problem
opening connection to shout server piradio:8000: Couldn't connect
```

Problem - Icecast streaming page says it can't be displayed.

Possible causes:

- The icecast service is not running on the radio.
  - Start it either from the Radio options menu (Streaming on) or run **sudo service icecast2 start** on the Rasperry PI and retry.
- Incorrect IP address or missing port number in the URL.
  - See Icecast2 Operation on page 204

Problem – No Mount Point displayed

Possible causes:

- This is mostly due to a mis-match in the MPD configuration and the Icecast2 configuration.
  - The icecast configuration is file is **/etc/icecast2/icecast.xml** . Make sure that all of the passwords are set to 'mympd'. The password 'hackme' will not work.
- There is no /mpd directory or the permissions are incorrect.
  - Check that the **/mpd** directory exists and that the permissions are set to 777. See Installing *Icecast* on page 202.

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 207.

Problem – Music keeps stopping or is intermittent

This is difficult to give a definitive answer to this problem. It must be remembered that running MPD and Icecast2 together on a Raspberry PI is pushing the Raspberry PI to its limits. It can also depend on your network or the PC you are using. Personal experience showed no problem playing a stream on PC with a wired network connection however a Laptop connected over a wireless network did not work well. Trying to play two or more devices on the MPD/Icast2 stream is also likely to result in poor results.

Try over-clocking the Raspberry PI using the **raspi-config** program. Medium over-clocking seems to be sufficient. See Overclocking older Raspberry PI on page 203. 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 installation

As from version 6.7 onwards you can install **Raspotify** from Dave Cooper.
First carry a system update and upgrade as shown in *Update to the latest the packages* on page 65
Now download the **Raspotify** software with the **curl** command.

```
$ 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>"
```

Edit **/lib/systemd/system/raspotify.service** and disable the restart options.

```
#Restart=always
#RestartSec=10
```

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 with **systemctl**.

```
$ sudo systemctl status raspotify
● raspotify.service - Raspotify
   Loaded: loaded (/lib/systemd/system/raspotify.service; enabled; vendor
preset: enabled)
   Active: active (running) since Wed 2018-05-30 11:41:20 CEST; 1h 0min ago
  Process: 4072 ExecStartPre=/bin/chown raspotify:raspotify
:
```

Disable Raspotify from starting at boot time. Starting and stopping Raspotify is done by the radio.

```
$ sudo systemctl disable raspotify
```

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:

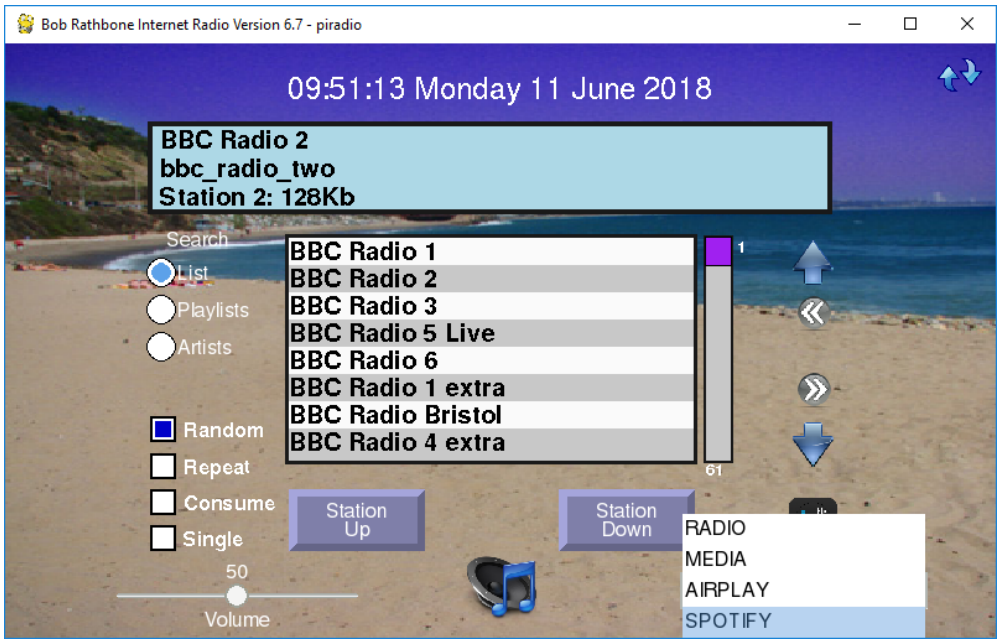


Figure 185 Starting the Spotify Receiver

The following window will appear however a different message may appear on the second line of the display window:

Figure 186 The radio in Spotify mode

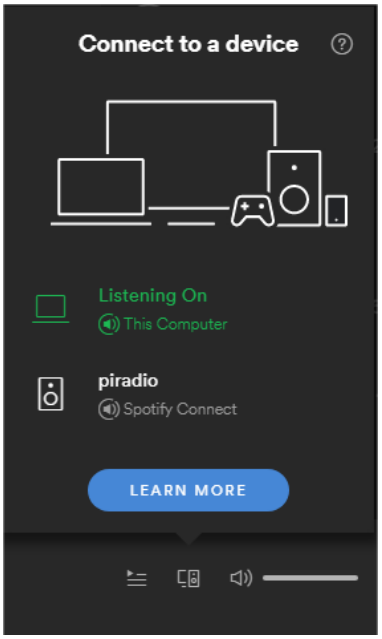


Figure 187 Spotify connecting to the radio

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.

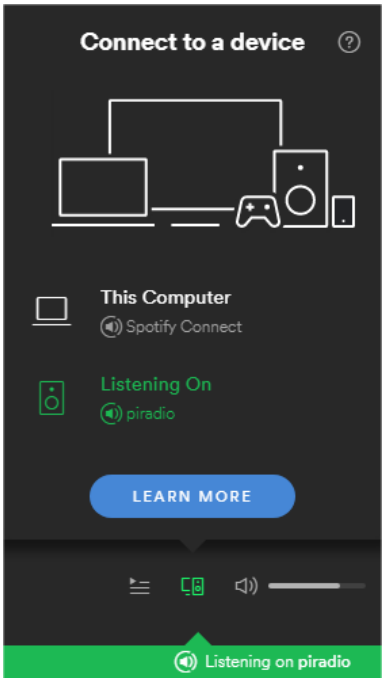


Figure 188 Listening to Spotify on the radio

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.

In the case of the touchscreen version of the program the following screen will be displayed:

Figure 189 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 65.

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.

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

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 65. **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 script

From version 6.5 onwards there is 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-get -f install
```

Re-run **install_airplay.sh.**
Only if this script fails try the manual procedure below. Otherwise skip the next section.

Manual procedure

Install the necessary build library modules and the git program:

```
$ sudo apt-get install build-essential git xmltoman
```

Install other build libraries:

```
$ sudo apt-get install autoconf automake libtool libdaemon-dev libasound2-
dev libpopt-dev libconfig-dev libssl-dev
```

The above instruction is all on one line.

If running on Buster it will be necessary to install the following libraries:

```
$ sudo apt-get install libdaemon-dev libpopt-dev pkgconf libconfig-dev
```

Install the Avahi daemon required for Airplay network discovery.

```
$ sudo apt-get install avahi-daemon libavahi-client-dev
```

Edit the **/usr/share/alsa/alsa.conf** file (use **sudo nano**). Scroll down a few pages until you see the following line:

```
pcm.front cards.pcm.front
```

Change this line to:

```
pcm.front cards.pcm.default
```

Create a development directory and change to it:

```
$ mkdir -p projects/airplay
$ cd projects/airplay
```

Build and install shairport-sync:

Get the source from github and configure the build.

```
$ git clone https://github.com/mikebrady/shairport-sync.git
$ cd shairport-sync
$ autoreconf -i -f
```

If **autoreconf** is missing for any reason install it with the following command:

```
$ sudo apt-get install dh-autoreconf
```

There is a problem with **autoreconf** which causes it to fail unpredictably. At the moment, it seems very unreliable. Below are some suggestions if it exits with an error.

If **autoreconf** does fail with the following message then make sure that the latest packages are installed. The **autoreconf** program can take several minutes.

```
autoreconf: 'configure.ac' or 'configure.in' is required
```

Carry out the procedure in section *Update to the latest the packages* on page 65 and retry.

If autoreconf still fails run the **following** command and then retry the above **autoreconf** command:

```
$ autoreconf -m
```

Note that the above command will fail but somehow it seems to (sometimes) cure the problem. Re-run the **autoreconf -vif** command.

Now configure the package build files (It is the configure script which is created by autoreconf)

```
$ ./configure --sysconfdir=/etc --with-alsa --with-avahi --with-ssl=openssl
--with-metadata  --with-systemd
```

Now build and install shairport-sync:

```
$ make
$ sudo make install
```

The last command installs the **shairport-sync-metadata** program in the **/usr/local/bin** directory and the distribution configuration file in **/etc/shairport-sync.conf**.

Change back up one directory

```
$ cd ..
```

Test Airplay (shairport-sync) – Stop the Music Player Daemon first:

```
$ mpc stop
$ shairport-sync
Successful Startup
```

Now run Airplay on an Airplay mobile phone or tablet. Search for connected devices which should include the raspberry pi. Play a music track which should be heard on the radio. Use **control-C** to exit. Use **Control-C** to exit **shairport-sync**.

Install the shairport metadata reader

To display the track metadata (Artist, Title and Album) on the display screen, it is necessary to install the **shairport-sync-metadata-reader** from Mike Brady.

```
$ git clone https://github.com/mikebrady/shairport-sync-metadata-reader
$ cd shairport-sync-metadata-reader
$ autoreconf -vif
$ ./configure
$ make
$ sudo make install
```

The last command installs the **shairport-sync-metadata-**reader program in the **/usr/local/bin** directory.

Configuring the Airplay feature

Below are the configuration parameters found 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 neccessary
mixer_volume=0

# Mixer volume ID (Airplay) Use command 'amixer controls | grep -i volume'
#                           to identify mixer volume control ID
mixer_volume_id=1
```

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.

Enable Airplay in **/etc/radiod.conf**

```
airplay=yes
```

Set the mixer output volume ID (Volume control for Airplay). Run

```
$ amixer controls | grep -i volume
```

Identify the ID (numid) for the playback volume for your device.
For sound cards or HDMI using digital volume control this is likely to be similar to the following.

```
numid=1,iface=MIXER,name='Digital Playback Volume'
```

For onboard jack sound output

```
numid=6,iface=MIXER,name='Speaker Playback Volume'cd
```

Modify or add the **mixer_volume_id** for your device in **/etc/radiod.conf. For example:**

```
mixer_volume_id=6
```

The **mixer_volume** parameter has a very special use and is used to preset the mixer volume (alsamixer) if using a DAC sound card or HDMI output. It is not relevant if using the onboard audio jack as output and must be set to 0. The reason it is needed is that Airplay can only be controlled by the mixer level unlike the radio which uses Music Player Daemon volume commands.

For on-board audio changing the mixer volume is not relevant as it is controlled by MPD. In this case set it to 0:

```
mixer_volume=0
```

For sound cards or HDMI set it to somewhere between 80 and 100.

```
mixer_volume=90
```

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-get install systemd-sysv
```

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 148. LCD versions of the radio are described here.

Figure 190 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 191 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. See: https://play.google.com/store/apps/details?id=com.nav.aoaplayer

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 hack 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 68). 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!

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 un-encrypted 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.

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. How to enable this is shown in *Using SSH to log into the Raspberry PI* on page 63. 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 software. 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: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 i**d_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
AAAAB3NzaC1yc2EAAAABJQAAAQEAuX+NEQoQECPN2d+Lu+qL2exMT/ICYbrNax6DVWBtKGzTxFOb
:
LeiaFbI3tWyi+ZPXg8Swhr1OaPN6l2E/fnAQPbGl2S+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-get 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-get 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 219.

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.asx
[BBC Radio 2] http://bbc.co.uk/radio/listen/live/r2.asx
[BBC Radio 3] http://bbc.co.uk/radio/listen/live/r3.asx
:
:
[RAIradio3] http://www.listenlive.eu/rai3.m3u
```

This will produce a single playlist called **myplaylist.m3u** with the stations loading in the order that they have been defined in the **/var/lib/radiod/stationlist** file. Make sure there are no blank lines between station definitions otherwise this terminates the playlist. All remaining stations will end up in there own single playlist file.

Why are some station names not being displayed in the web interface?

The reason for this is that some stations don't send the name with the stream. If you run the **mpc playlists** command you will see that some radio stations shows only the station URL and not the name:

```
$ mpc playlist
RAIradio2
:
BBC Radio 4 extra
http://icestreaming.rai.it/1.mp3
BBC Radio 3
BBC Radio 6
BBC Radio 5 live
```

The only way around this is to complain directly to the radio station to ask them to amend their stream to include the station name and title details. The only way reason that the station name is seen with the radio program is that it picks up the names out of the station list file.
The snoopy web interface can't do this however.

Why doesn't the web interface display URLs until a station is selected?

When the Snoopy web interface is loaded it loads the playlists found in the **/var/lib/mpd/playlists/** directory. Snoopy displays the URLs but doesn't appear to use any titles defined in the playlists. It only displays the radio station information (if present) once it starts streaming from a particular radio station. Snoopy is third party software over which this author has no control.

Why are music tracks played randomly when loaded?

This is the default behaviour when the music library is loaded in version 5.2 or earlier. This has changed in version 5.3 onwards. Random always defaults to "off" when the radio is selected. However when the music library is selected the value stored in the **/var/lib/radiod/random** file is used. This value can be changed in the selection menu by selecting "Random off" after loading the music which will store the new value in the **/var/lib/radiod/random** file. 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 129

Why do I see a station number on LCD line 3?

For version 3.3 onwards if no song information is available then the station playlist number followed by the stream speed. In the following example Radio 1 is not transmitting any song information. It is number 37 in the play list. The speed from the stream is 96 Kilobit. The displayed stream speed can also continuously change for some radio stations where the stream speed is variable.

```
12:01 23/08/2015
Radio1
Station 37 96K
Volume 75
```

Is it possible to change the date format?

Yes. Please see the section called *Changing the date format* on page 127.

Is there a pause & resume function?

Yes, but it is called mute and un-mute. The mute function also stops or pauses the MPD player. If playing a radio station, a "stop" command is carried out. If playing a media track a "pause" is carried out. The reason these are different is that media may be safely paused but in the case of a radio station pausing causes buffering and jumping to the next station when the radio resumes normal playing. See the

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 on page 142.

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 42.

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?

The rotary encoders illustrated in this guide are COM-09117 12-step rotary encoders from sparkfun.com. The radio uses so called “Incremental Rotary Encoder”. An incremental rotary encoder provides cyclical outputs (only) when the encoder is rotated. The other type is an absolute rotary encoder and maintains position information even when switched off (See Wikipedia article and my tutorial on rotary encoders).

The cheaper smaller rotary encoders are usually incremental encoders. Absolute rotary encoders are usually bigger and more expensive as they house more electronics. If unfortunately, the seller doesn't provide a specification then there is a small risk that they may not run with this software.

Note: Not all manufacturers’ rotary encoders will work with this project. If they work then fine if not regrettably you will need to purchase the recommended rotary encoders. You can also try the alternative rotary class which may just work with your encoders.

Can this code or documentation be re-used in other projects?

Yes, it can. You can even use it commercially, provided that you do so under the terms of the licence distributed with this package. The software and documentation for this project is released under the GNU General Public Licence and may be re-used it other projects. See Licences on page 234.

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.

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.

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 232.

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* program is only used by the LCD (including PiFace CAD) and OLED versions of the radio 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 LCD display software. Depending upon the actual device configured it will load one of the following:

Table 20 Display classes

Display class file	Description	display_type
lcd_class.py	LCDs with a directly connected HD44870 interface	LCD
lcd_i2c_adafruit.py	LCDs with an Adafruit I2C backpack	LCD_I2C_ADAFRUIT
lcd_i2c_pcf8574.py	LCDs with a PCF8574 I2C backpack	LCD_I2C_PCF8574
lcd_adafruit_class.py	LCDs with an Adafruit RGB plate	LCD_I2C_ADAFRUIT
lcd_pifacecad_class.py	Piface CAD 2x16 LCD with push buttons	PIFACE_CAD
oled_class.py	Solomon Systech SSD1306 OLED Display (IQaudIO)	OLED_128x64
no_display.py	No display attached (vintage radio/Pirate Radio)	NO_DISPLAY

The above settings are performed by the **configure_radio.sh** program.
The Adafruit RGB plate also uses an i2c class courtesy of Adafruit Industries (renamed to i2c_class.py).

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 (*remote_control.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/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 may are 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** 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 157.

The display_current program

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

The display_model script

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

The 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_ir_remote script

The *configure_ir_remote.sh* script sets up and partially configures the LIRC (Linux Remote Control) components for an IR remote control.

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 remote control daemon

The remote control daemon consists of the *remote_control.py* and the *rc_daemon.py* program files. There is a service start stop script called */etc/init.d/irradiod*. This is configured for the correct program by the *configure_radio.sh* program during installation. The server_class.py program is used for communication between the remote-control daemon and the radio program.

The UDP network communications class

The remote control daemon uses the *udp_server_class.py* program which communicates over the local TCP/IP network using UDP port 5100 as the default; however, the port is configurable in **/etc/radiod.conf**. When a button is pressed on the remote control this program sends the button identity (See Table 12 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.

The init file

The *__init__.py* file contains a couple of global definitions plus the package version number.

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 75. 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.

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

Install **git** with the following command:

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

Make a development directory and change to it:

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

Now clone the github piradio repository:

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

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

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

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

Building your own package

If you do modify the code it may well be that you wish to create your own Rasbian package. There are several files and scripts required to build the **radiod** package.

- **build.sh** – Run this to actually build the package.
- **piradio** – The package definition file which define the executables 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.

Install the build environment packages first:

```
$ sudo apt-get -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.

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 235.

Disclaimer

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BELIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE OR DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Technical support

Technical support is on a voluntary basis by e-mail only at bob@bobrathbone.com. 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 177. 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 154, run the program and include the **/var/log/radio.log** file. Also run **sudo service radiod info**.
- Did you vary from the procedure in the manual or add any other software?
- Please do not answer my questions with a question. Please supply the information requested.

Please note that support for general Raspberry PI problems is not provided. Only issues relating to the Radio software will be investigated.

For general Raspberry PI support see the following site:
http://www.raspberrypi.org/forums/

For support on Music Player Daemon issues see the help pages at the following link:
http://www.musicpd.org/

For issues relating to Icecast2 streaming see:
http://www.icecast.org

For those of you who want to amend the code to suit your own requirements please note: I am very happy to help people with their projects but my time is limited so I ask that you respect that. Please also appreciate that I cannot engage in long email conversations with every constructor to debug their code or to teach 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-mpd-and-icecast2-on-raspberry-pi/

To contributors such as Alan Broad who supplied photos of the Lego example of the radio plus code contribution.

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 69 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

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 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.

Glossary

AC	Alternating Current - In this context 110V or 220V
AP	Application processor (Also see CPU)
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
ASX	Advanced Stream Redirector
ATC	Air Traffic Control
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	
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.)
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.)
FLIRC	A USB device and software which maps an IR remote control to the keyboard
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

IIC Alternative name used by some manufacturers for I2C

I2S Inter-IC Sound (Used in DAC interface) from Philips (Now NXP)

IP Internet Protocol

IPv4 Internet Protocol Version 4

IPv6 Internet Protocol Version 6

IATA International Air Transport Association

ICAO International Civil Aviation Organization

IR Infra-Red (sensor) for use with infra-red devices such as remote controls

LCD Liquid Crystal Display

LE Low Energy – In this context Bluetooth LE (Bluetooth 4.0 core specification)

LIRC Linux Remote Control software

M3U MPEG3 URL

MAC Media Access Control (address)

MicroHDMI Miniaturized version of the High Definition Multimedia Interface specification

MMS Microsoft Media Server Internet protocol

MPC Command line client for MPD

MPEG Moving Picture Experts Group

MPEG3 Music encoding standard from MPEG

NAS Network Attached Storage

NFS Network File System

NTP Network Time Protocol

MPD Music Player Daemon

OLED Organic Light Emitting diode

PA Personal Amplifier (Input to audio stage of a vintage radio)

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

PHP A server-side scripting language designed primarily for web development

PID Process ID

PLS MPEG Playlist File (as used by Winamp)

RSS Really Simple Syndication – Web feed usually containing news items

SD San Disk Memory Card commonly found in cameras and Smartphone's

SPI Serial Peripheral Interface (Motorola) used by PiFace CAD

S/P-DIF Sony/Philips Digital Interface for short distance audio transmissions.

SCO Synchronous Connection Oriented link - Bluetooth

SSH Secure Shell – Encrypted terminal

SSID Service Set Identifier. An SSID is the public name of a wireless network.

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.

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)

VDD Voltage Drain Supply

VEE Voltage Emitter

VLC Media player used by Pimoroni software (Not used by this radio software)

VSS Voltage Source Supply

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
WIFI	Wireless Network 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 1.69 2019/11/22 09:43:59 bob Exp $

# Configuration file for version 6.0 onwards
# 40 pin version to support IQ Audio and other sound cards
# Also 26 pin version radios use this file
#
# 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=INFO
# 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

# 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=20

# Volume display text or blocks
volume_display=blocks

# MPD port number (Do not change unless MPD reconfigured)
mpdport=6600

# MPD client timeout from 2 to 15 seconds default 5
client_timeout=5

# Remote control communication  host and port Default localhost 5100
remote_control_host=localhost
remote_control_port=5100

# Remote control UDP server listen host either 0.0.0.0 (All interfaces) or
localhost
remote_listen_host=localhost

# Output LED for remote control, default GPIO 11 (pin 23) or
# GPIO 13 (pin 33) for AdaFruit plate or PiFace CAD (40 pin RPi needed)
# 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 overides the default i2c address. 0x00 = use default
# Some backpacks use other addresses such as 0x2F, then set i2c_address=0x2F
i2c_address=0x00

# 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

# Speech for visually impaired or blind listeners, yes or no
# Needs espeak package - sudo apt-get install espeak
speech=no
# Speech volume as a percentage of the normal MPD volume
speech_volume=40
# Verbose - yes = each station change is spoken
verbose=no
# Speak hostname and IP address
speak_info=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=4
up_switch=24
down_switch=23
left_switch=14
right_switch=15

# Pull GPIO up/down internal resistors (Applies to button interface only).
# Default:down
pull_up_down=down

# 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

display_type=LCD

# Display width, 0 use program default. Usual settings 16 or 20
display_width=20
display_lines=4

# 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

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

# 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

# OLED parameters
# Flip display vertically (yes or no) OLED only at present
flip_display_vertically=no

# Splash screen
splash=bitmaps/raspberry-pi-logo.bmp

# 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


[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/radio.log** to be rotated so that it doesn't continue to grow and fill the disk.

```
/var/log/radio.log {
      weekly
      missingok
      rotate 4
      compress
      notifempty
      maxsize 150000
      copytruncate
      create 600
}
```

Old log files are compressed and renamed, for example **/var/log/radio.log.1.gz**.

/etc/init.d/radiod

This is the start stop script for the radio daemon. The NAME parameter must be changed to point to the correct radio program. This is done by the **configure_radio.sh** shell script which is called during the installation procedure.

```
# Change NAME parameter this next line to the version of the daemon you are
using
# Choices are radiod.py, radio4.py, rradiod.py, rradio4.py, ada_radio.py,
# rradiobp4.py or rradiobp.py
```

```
# No spaces around the = character
NAME=radio4.py
```

This script is linked to the start stop run levels in the **/etc** directory.
For example:

```
ls -la /etc/rc2.d/S03radiod
lrwxrwxrwx 1 root root 16 Nov  8 14:28 /etc/rc3.d/S03radiod ->
../init.d/radiod
```

/lib/systemd/system/radiod.service

This file is part of the new **systemd** startup services and has been added from version 5.8 onwards to increase compatibility with the new **systemd** start-up and shutdown routines.

```
# 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
```

/etc/init.d/asound.conf

This file is only added if a DAC or USB sound device is added and is needed for **espeak** and **aplay**

```
# 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
      }
}
```

/etc/init.d/irradiod

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
```

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;
```

The cron.weekly/radiod script

```
#!/bin/sh

DIR=/usr/share/radio
LOGDIR=${DIR}/logs
LOG=${LOGDIR}/stations.log

mkdir -p ${LOGDIR}
chown pi:pi ${LOGDIR}
${DIR}/create_stations.py --delete_old 2>&1 >${LOG}
chown pi:pi ${LOG}
```

Note: This file requires the **anacron** package to be installed to run.

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 lirc-rpi device definition. For example:

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

It may also be modified to support **HiFiBerry DAC** and **DAC+**. For example:

```
dtoverlay=hifiberry-dacplus
```

For **IQaudIO** devices the relevant overlay will be specified.

```
dtoverlay=iqaudio-dacplus,unmute_amp
```

From version 5.8 onwards the **configure_audio.sh** program disables the onboard sound devices when configuring a sound card by changing the following line in **/boot/config.txt** from:

```
dtparam=audio=on
```

To:

```
dtparam=audio=off
```

The lxsession autostart file for desktop radio

The **/home/pi/.config/lxsession/LXDE-pi/autostart** file is modified to start **gradio.py** or **vgradio.py** if so configured.

```
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
@xscreensaver -no-splash
@point-rpi
@sudo /usr/share/radio/gradio.py
```

Appendix B – Cheat sheets

The following cheat sheet is a list of the basic commands to install MPD and the radio software.

B.1 Operating system and configuration

Update OS

```
$ sudo apt-get update
:
$ sudo apt-get upgrade
$ sudo reboot
```

Run raspi-config to set up hostname, pi user password and timezone.

```
$ sudo raspi-config
```

B.2 Music Player Daemon and Radio software

Install MPD and MPC

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

Download radio package

```
$ wget https://www.bobrathbone.com/raspberrypi/packages/radiod_6.12_armhf.deb
```

Install radio package

```
$ sudo dpkg -i radiod_6.12_armhf.deb
```

If using I2C components install python-smbus

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

If using PiFace CAD (SPI interface) install python-pifacecad

```
$ sudo apt-get install python-pifacecad
```

To install sound cards

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

IQaudIO Cosmic controller and OLED

```
$ sudo apt-get install libffi-dev
$ sudo apt-get install build-essential libi2c-dev i2c-tools python-dev
```

B.3 Installing the Pimoroni Pirate Radio software

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

B.4 Installing Web Interface

Install Apache and the PHP libraries for Apache.

```
$ sudo apt-get install apache2 php libapache2-mod-php
```

Download the web interface package

```
$ wget http://www.bobrathbone.com/raspberrypi/packages/radiodweb_1.7_armhf.deb
```

Install the web interface package
For Raspbian Stretch or Buster:

```
$ sudo dpkg -i radiodweb_1.7_armhf.deb
```

B.5 Installing remote IR software

Install **lirc** and **lirc-python** with the following commands:

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

Configure **/boot/config.txt** for **lirc-rpi** overlay to match the wiring for the IR sensor.

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

Copy the button definitions to **/etc/lircrc** and reboot the Raspberry Pi.

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

Generate the **lircd.conf** file using the remote control.

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

Enable the **irradiod** service to start up at boot time.

```
$ sudo systemctl enable irradiod
```

Configure the remote activity LED in **/etc/radiod.conf** to match the wiring for the LED.

```
remote_led=16
```

Test the LED

```
$ sudo service irradiod flash
```

B.6 Enabling speech facility

Install the **espeak** package:

```
$ sudo apt-get install espeak
```

Enable the speech facility in **/etc/radiod.conf** and restart the radio.

```
# Speech for visually impaired or blind listeners, yes or no
speech=yes
```

B.7 Installing Spotify

Download the Raspotify software with the curl command.

```
$ curl -sL https://dtcooper.github.io/raspotify/install.sh | sh
```

This will both download and install Raspotify.

Edit **/lib/systemd/system/raspotify.service** and disable the restart options.

```
#Restart=always
#RestartSec=10
```

Appendix C – Technical specification and other notes

C.1 – Technical specification

The specification of the Raspberry PI internet radio is:

- The software runs on **Raspbian Buster** (Debian 10) on all Raspberry Pi's except version 1
- Uses the standard Music Player Daemon (MPD)
- The following displays are supported:
  - Raspberry Pi 7-inch touch screen or HDMI screen
  - Most 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
  - Pimoroni Pirate Radio
- The LCD can be directly interfaced via GPIO pins or using an I2C backpack interface
- Optional IR sensor and remote control using LIRC or
- 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)
- 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
- Fully integrated with mobile apps such as Android **MPDdroid** or Apple **mPod**
- Speech for visually impaired and blind persons using **espeak**
- Support for European character sets (Limited by LCD capabilities)

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 235.

C.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

C.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 21 Sound card Device Tree overlays

Manufacturer	Sound Card	DT Overlay	Pulseaudio
Adafruit	3W Stereo Amplifier Bonnet	dtoverlay=hifiberry-dac	Yes
HiFiBerry	DAC+ Light/DAC Zero/MiniAmp	dtoverlay=hifiberry-dac	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	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	dtoverlay=hifiberry-dac	Yes

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
```

C.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 22 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
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
RELOAD_PLAYLISTS	shoutcast.cgi script	Reload (new) playlists

Appendix D – 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 75. It is also necessary to set the **pull_up_down** parameter in **/etc/radiod.conf** depending on wiring.

D1 Push Button and Rotary Encoder 40-pin wiring

The following table shows the wiring for the 40-pin push-buttons or rotary encoders.

Table 23 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

D.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 24 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.

D.3 IQaudIO Cosmic Controller wiring

The following table shows the wiring for the IQaudIO Cosmic controller.

Table 25 IQaudIO Cosmic Controller Wiring

Physical control	Pin	GPIO	Configuration	Radio function
Left hand push button	7	4	down_switch	Channel down
Middle push button	29	5	menu_switch	Menu
Right hand push button	31	6	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

D.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 26 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**

D.5 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 27 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	19	10	left_switch	Volume Down
Volume up button/Rotary switch B	11	17	right _switch	Volume Up

Table 28 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 29 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

D.6 Raspberry Pi Rotary Encoder version with backlight dimmer

The following wiring diagram was provided by Joaquin Perez, Broadcast Engineer, Leeds. He also shows the circuitry to dim the backlight using a BS170 Mosfet transistor (Software to support the LED dimmer to follow in a later release).

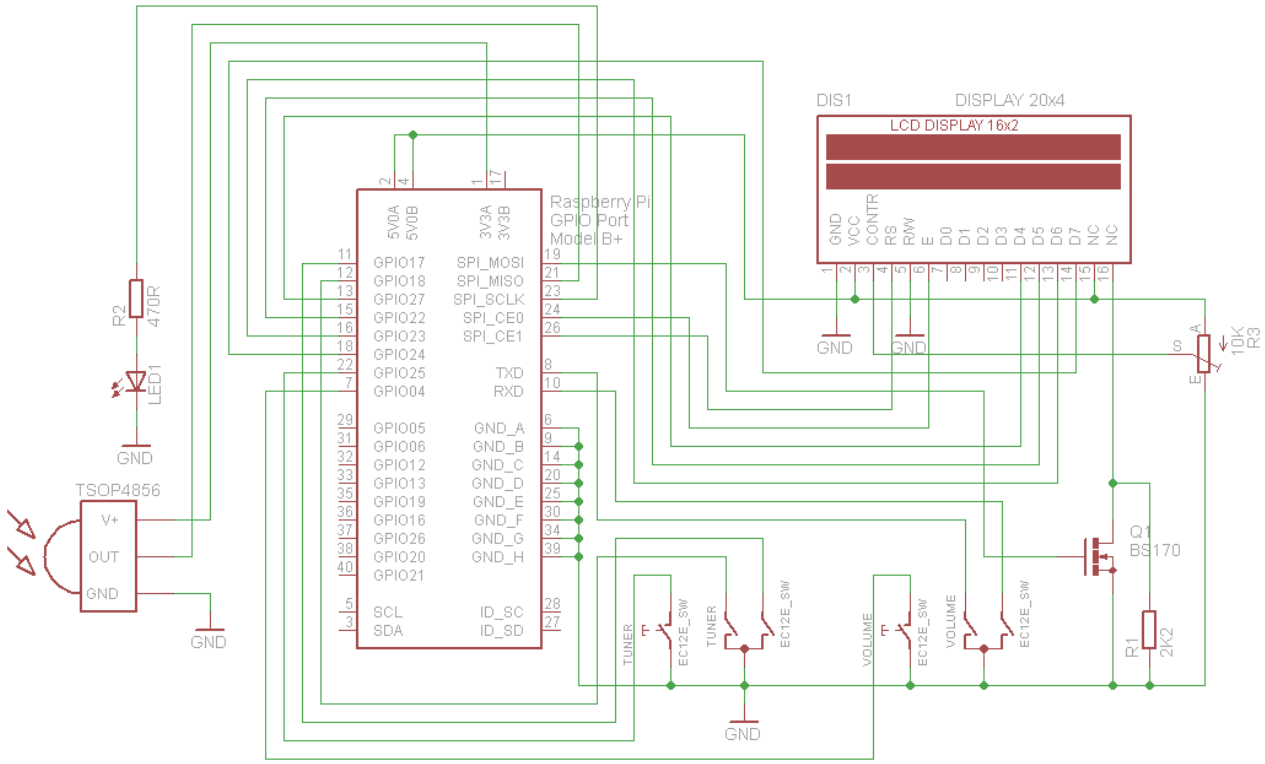


Figure 192 Wiring Raspberry Pi Radio Rotary Encoder version

Index