Part 01

Build a Raspberry Radio

Create your own virtual radio station, with a DJ that reads out the news and weather and announces your songs before they play

MAKER

Sean McManus

Author of *Mission Python*, *Scratch Programming in Easy Steps*, and *Raspberry Pi For Dummies* (with Mike Cook). Get free chapters at Sean's website.

sean.co.uk

We love radio, but don't you ever wish you had more control over the playlist? With Raspberry Radio, every song is your request. You give it a collection MP3 files, and the virtual DJ plays them at random, telling you something about each track. After a few songs, it's time for the news and weather. In this tutorial, you'll see how to download the news headlines using an RSS feed and access your local weather through an API. Next issue, we'll get the DJ talking and cue the music. You can download the code at **magpi.cc/newsreader**.

01 Install Python modules

We'll be using two new Python modules for this project. It's quick and easy to install them. The requests module downloads content from the web using HTTP requests. The feedparser module is used to process RSS feeds, which many news websites use to share headlines, summaries, and links to their stories. Click the Terminal icon on your taskbar to open a terminal window. Then enter the following command at the prompt to download and install the modules:

```
pip install requests feedparser
```

02 Get your API key

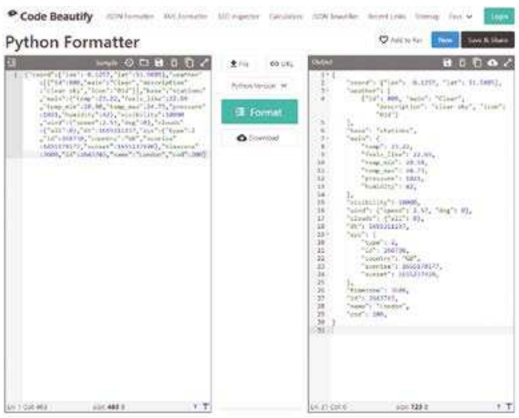
An API (application programming interface) enables applications to talk to each other. In our case, we want our Python program to talk to OpenWeather. The service gives you up to a million requests per month for free, but you need to register for an API key so they can monitor your usage. Visit **openweathermap.org/price** and click 'Get API key' in the free tier. You'll need to register a username, email address, and password, and accept the terms. When you sign into your account, go to your API keys and copy your key.

03 View your weather feed

You'll use a specially formatted URL to fetch the weather. There are lots of different weather forecast options, but we'll keep it simple by looking up the current weather. Let's preview the data feed before we try to use it from Python. In a browser window, enter the following URL.

http://api.openweathermap.org/data/2.5/weather?q=London,uk&units=metric&APPID=YOUR_API_KEY

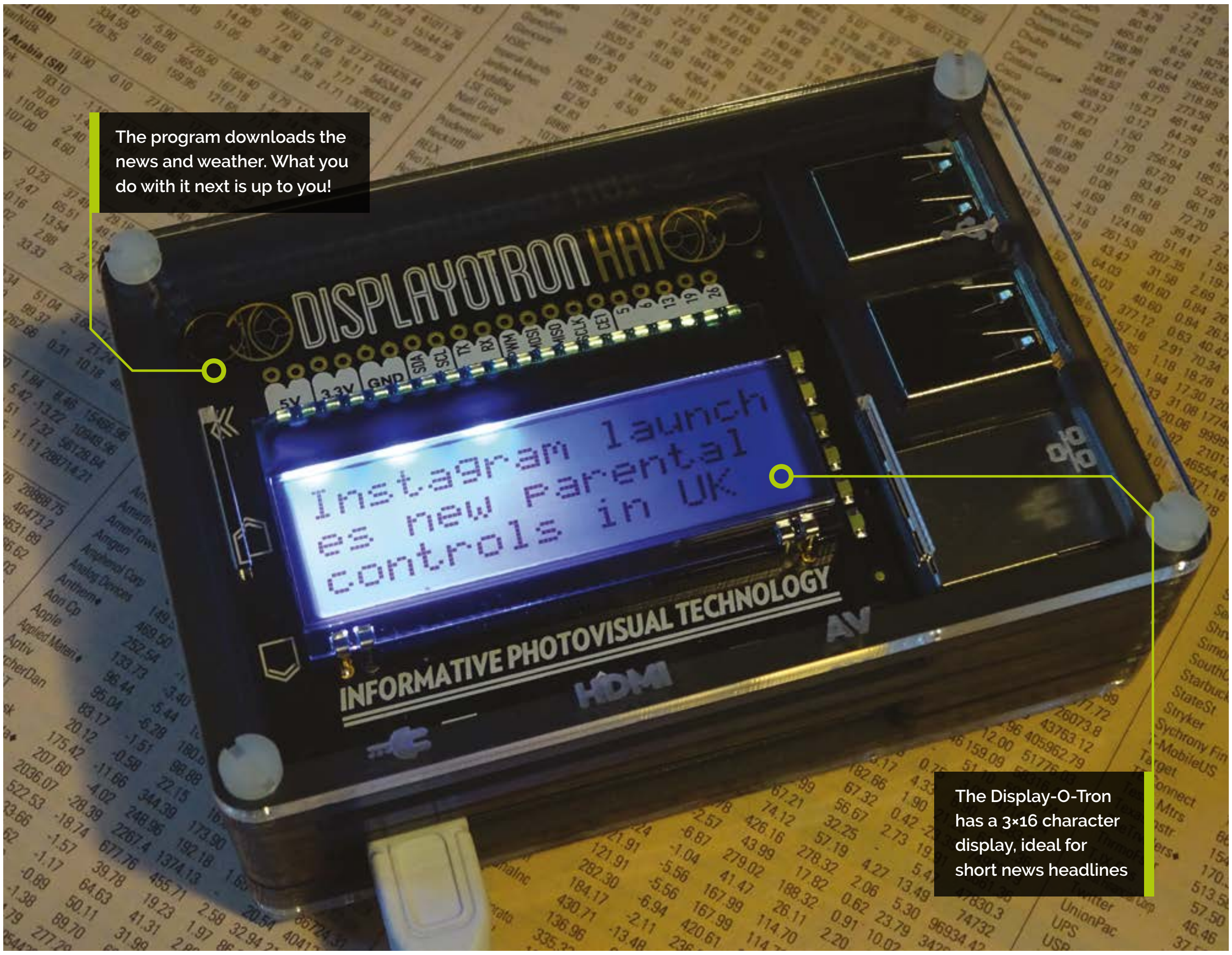
Change the city name to your own, and add your API key after the equals sign at the end. (If you're not near a city, consult the documentation at **openweathermap.org/current** for tips on using longitude and latitude instead.)



▲ This Python Formatter makes it much easier to understand the JSON structure of the weather data we're using

You'll Need

- Raspberry Pi
- Raspberry Pi OS
- Internet connection
- Display-O-Tron HAT (optional) **magpi.cc/displayotron**
- Pibow case (optional)


DISPLAYOTRON HAT
Instagram launches new parental controls in UK
INFORMATIVE PHOTOVISUAL TECHNOLOGY

The program downloads the news and weather. What you do with it next is up to you!

The Display-O-Tron has a 3×16 character display, ideal for short news headlines

04 Beautify the output

When you view the feed in your browser, the code isn't formatted for easy reading. It's much easier to understand the incoming data if we reformat it. Copy the code, visit **magpi.cc/beautifier**, and paste the code in the box on the left. Click Format, and the box on the right will show a nicely formatted version of your code. The code is in JSON format, which behaves a bit like a Python dictionary. You use keywords to access the data associated with them. For example, the keyword 'temp' returns a number (measured in Celsius). The keyword 'description' gives me 'clear sky' today.

05 Build the weather report

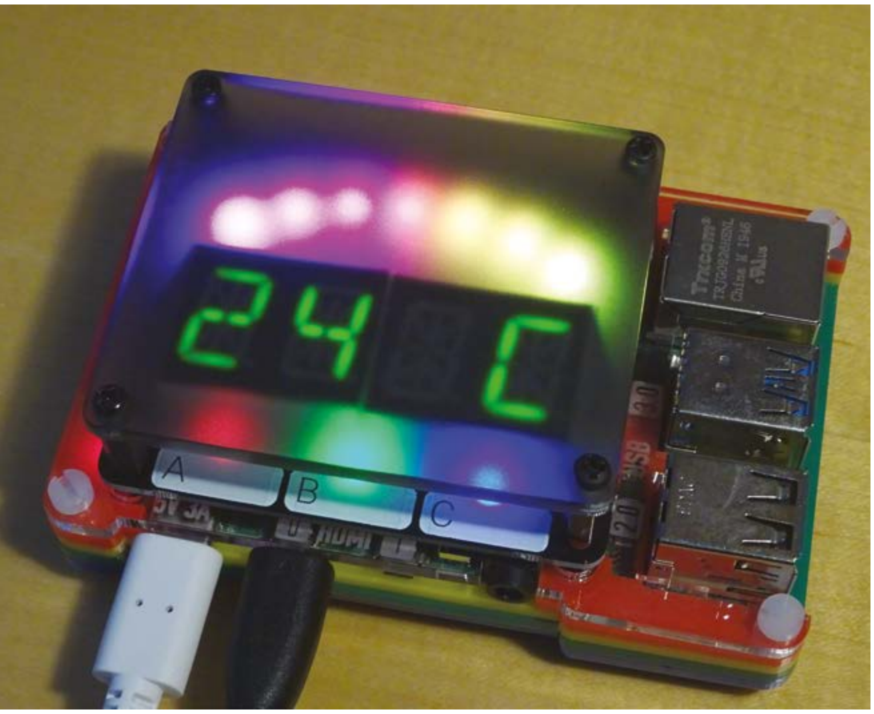
The `get_weather()` function in **rr_newsreader.py** builds a weather report, which is returned as a string. The function also returns the temperature as a number, so you can use it on numeric displays. First, the function uses the requests module to download your weather report. Then it uses the built-in JSON processor in the requests module to extract useful bits of it. Ultimately, Raspberry Radio will read out the weather report, so the report starts with a random phrase to add variety. It's easiest to understand the code if you look at the JSON file (see Step 4) at the same time. The `data` variable stores our full report. We use `data.json().get('weather')` to access the weather attribute of it. The weather section contains an array with one item in it, which we access using `[0]` on line 19. Inside that array, we can find the description attribute. Similarly, we use

> "It's easiest to understand the code if you look at the JSON file"

Top Tip

Extend the weather report

As Step 4 shows, there's more weather data available. Modify the program to report on humidity, wind speed, and what the temperature feels like.



You can use rr_newsreader.py to show the outside temperature on a Rainbow HAT, here shown through a diffuser layer

the attribute ‘main’ to access a subset of data that includes the ‘temp’.

06 Use f-strings for formatting

Line 23 uses a Python f-string to format the weather report. It’s an easy way to insert a variable’s value into a string. You put an f before the opening quote of the string, and then put the variable name in curly braces inside the string. F-strings are more readable than alternatives, such as percentage formatting and the `format()` function. They both put a placeholder in the string and the associated variable after the string.

news_output.py

Language: **Python**

```python
# News display for Displayotron HAT
# From Raspberry Radio project in The MagPi by Sean McManus

import rr_newsreader
weather_report, temperature = rr_newsreader.get_weather()
print(weather_report)
date_report = rr_newsreader.get_date()
print(date_report)
news_report = rr_newsreader.get_news()
for line in news_report:
    print(line)
```

07 Get an RSS link

Many publications publish RSS feeds, but they’re not as prominent on websites as they used to be. You can often uncover them by Google searching for your favourite publication’s name plus ‘RSS’. The BBC publishes news feeds (**magpi.cc/bbcfeeds**) for topics including technology, health, and entertainment, as well as news feeds dedicated to the different geographies. The Guardian (**magpi.cc/guardianfeeds**) has feeds for a huge number of topics, as diverse as Agatha Christie, the Vietnam War, and Pink Floyd. Raspberry Radio works best if you pick a newsy topic that is fast-moving. We’re looking for headlines that make sense in isolation, so avoid feeds promoting feature articles.

08 Get the news

In **rr_newsreader.py**, the `get_news()` function uses the feedparser module to process the RSS feed. Paste the web address of your chosen RSS feed into line 30, or leave the code unchanged for The Guardian’s news headlines. The function builds a list called `news_list`. The first list item introduces the news, before the loop extracts the titles from the first three stories in the RSS feed, and adds them (appends them) to the list. The title contains the headline, and it’s all we need for our purposes. Each story also has a description (a summary, sometimes quite long), and a link to the main article. To find the URL for the article at index 2, for example, you’d use `rss_feed.entries[2].link`.

09 Get the date

Newsreaders start their broadcast with the date, so the `get_date()` function creates a human-readable date. It uses the datetime module, and its `strftime()` function, which creates a string by extracting parts of the date. You use codes to specify the format you want to use. `%A` gets you the day name (e.g. Sunday), %d extracts the day number, and %B gives you the month in full. We're using an f-string again here to build a string that combines those three date elements. There's a list of codes at **strftime.org**. If you used %I, %M, and %p, you would get the time in the format '10 25 AM'.

10 Prepare to import

You might have noticed that **rr_newsreader.py** does not have any output. It defines three functions, but they're not called at any time, and the information they gather isn't displayed. That's because we want to make this code reusable across different projects. This issue, you'll see how to display the news headlines on an LCD screen, but next issue your device will read them aloud. We can import **rr_newsreader.py** into another Python program, as long as both programs are in the same folder. We used the name **rr_newsreader.py** to reduce the risk of confusion with other newsreader modules.

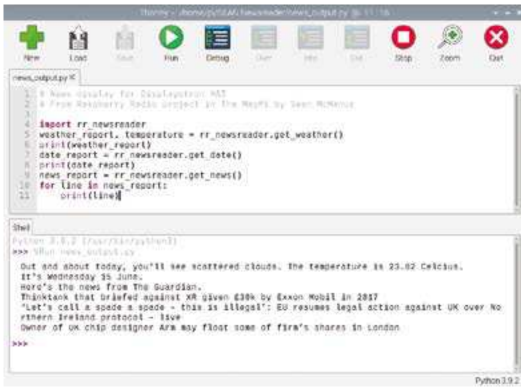
" We want to make this code reusable across different projects "

11 Test the newsreader

The **news_output.py** listing shows how to import **rr_newsreader.py** and access the current date, weather, and news headlines using it. The weather and date are returned as strings. The news is a list, so a loop is used to print each line in turn. You can use this program as a model for collecting the data so you can output it using your favourite HAT.

12 Create your newsreader gadget

The **displayotron_news.py** listing outputs the headlines on a Pimoroni Display-O-Tron. This is just a simple demo: headlines that are too long are shortened to fit the display. You could extend the program so it shows the full headlines and enables you to page through them using the buttons on the HAT. The program uses basic animation, displaying each headline one character at a time, and brightening the LEDs when the headline is complete. It makes it feel like the news is coming in right now, and is more visually interesting than just having it pop up on the screen.



```
# News display for Displayotron HAT
# From Raspberry Radio project in The MagPi by Sean McManus

import rr_newsreader
weather_report, temperature = rr_newsreader.get_weather()
print(weather_report)
date_report = rr_newsreader.get_date()
print(date_report)
news_report = rr_newsreader.get_news()
for line in news_report:
    print(line)
```

```
Python 3.9.2 (/usr/bin/python3)
>>> %Run news_output.py
 Out and about today, you'll see scattered clouds. The temperature is 23.82 Celcius.
 It's Wednesday 15 June.
 Here's the news from The Guardian.
 Thinktank that briefed against XR given £30k by Exxon Mobil in 2017
 'Let's call a spade a spade - this is illegal': EU resumes legal action against UK over No
rthern Ireland protocol - live
 Owner of UK chip designer Arm may float some of firm's shares in London
>>>
```

You can display the news in the Python shell, as shown here, or use rr_newsreader.py to download it and show it on your favourite HAT

Top Tip

Humanise the news

You could add a randomly chosen introduction to the news to increase variety, as we have for the weather.

displayotron_news.py

Language: **Python**

```python
# News display for Displayotron HAT
# From Raspberry Radio project in The MagPi by Sean McManus

import rr_newsreader
import time
import dot3k.backlight as backlight
import dot3k.lcd as lcd

def display_text(text):
    backlight.rgb(200, 200, 255)
    for i in range(min(len(text), 48)):
        substring = text[:i+1]
        lcd.clear()
        lcd.write(substring)
        time.sleep(0.1)
    backlight.rgb(255, 255, 255)
    time.sleep(3)

date_report = rr_newsreader.get_date()
display_text(date_report)

news_report = rr_newsreader.get_news()
for line in news_report:
    display_text(line)
```

rr_newsreader.py

Language: **Python**

DOWNLOAD THE FULL CODE:

magpi.cc/newsreader

```python
# rr_newsreader generates news, weather, and date reports
# From Raspberry Radio project in The MagPi by Sean McManus

import feedparser
import requests
import random
from datetime import datetime

def get_weather():
    data = requests.get(
        "http://api.openweathermap.org/data/2.5/weather?q=London,uk&units=metric&APPID=YOUR_API_KEY")
    if data.status_code == 200:
        report = random.choice(["The weather today is ",
                                "We're looking at ",
                                "Today, expect ",
                                "There's going to be ",
                                "Out and about today, you'll see "])
        weather_forecast = data.json().get('weather')
        description = weather_forecast[0].get('description')
        report += description
        main = data.json().get('main')
        temperature = main.get('temp')
        report += f". The temperature is {temperature} Celcius."
        return report, temperature
    else:
        return "There is no weather report today.", False

def get_news():
    news_list = []
    rss_feed = feedparser.parse('https://www.theguardian.com/uk-news/rss')
    news_list.append("Here's the news from The Guardian.")
    for i in range(3):
        news_list.append(rss_feed.entries[i].title)
    return news_list

def get_date():
    date = datetime.today()
    date_text = f"It's {date.strftime('%A')} {date.strftime('%d')} {date.strftime('%B')}."
    return date_text
```

Part 02

Raspberry Pi Radio: Add a DJ and jingles

On air in 3, 2, 1... You're listening to Raspberry Radio, the only station that plays your own music, all day and all night

MAKER

Sean McManus

Author of *Mission Python*, *Scratch Programming in Easy Steps*, and *Raspberry Pi For Dummies* (with Mike Cook). Get free chapters at Sean's website.

sean.co.uk

With Raspberry Radio, you're guaranteed to hear your favourite songs. The program creates a virtual DJ, who plays your MP3s at random, but introduces each one with some information about it. Every eight songs, there's a break for the news and weather. For an extra touch, you can add a Display-O-Tron HAT to show the artist and track name while it plays, like a DAB radio does. This project shows you how to make your Raspberry Pi speak, how to access the metadata in an MP3 file, and how to play music from Python.

01 Prepare your files

Raspberry Radio makes use of the **rr_newsreader.py** program from Part 1 in this series (see *The MagPi* issue #121, **magpi.cc/121**). Put **rr_newsreader.py** in the same folder as the **raspberry_radio.py** program from this issue. That folder should also have two subfolders: one called **music**, and another called **jingles**. You can download all the code for this project at **magpi.cc/raspradio**.

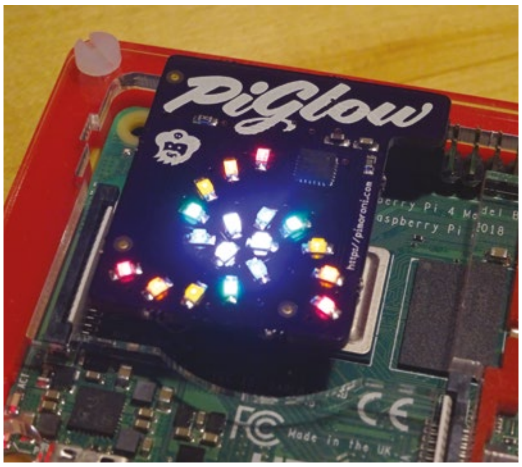
02 Install Python modules

For text-to-speech, we're using pyttsx3. Playsound will play our MP3s. Meanwhile, tinytag will read the metadata of the music files. By pulling out the artist name, song title, album, and year, we can get the virtual DJ to say something smart about each track before it plays. You also need to install the modules that the **rr_newsreader.py** program requires, if you didn't do that last issue. Open a Terminal window and enter the following commands at the prompt to download and install the modules, along with text-to-speech and media player software:

```
 pip install pyttsx3 tinytag playsound
requests feedparser
 sudo apt install espeak mopidy
```

03 Gather music and jingles

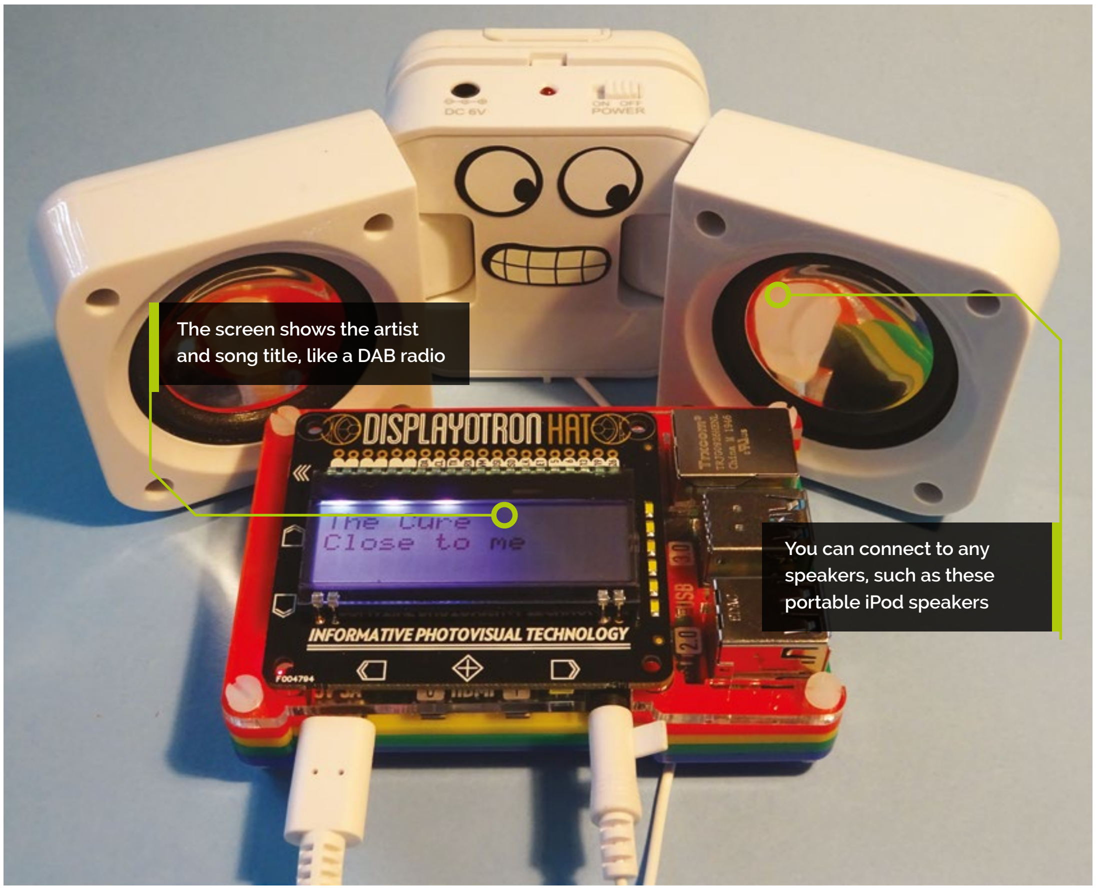
Raspberry Radio uses your own MP3s, which you should copy to the **music** subfolder. If you don't have an MP3 collection, lots of artists make their music available for free download on Bandcamp (find the author's at **magpi.cc/artificial**). To make it feel like real radio, we will top up the **jingles** folder. We recommend Music Radio Creative (**magpi.cc/freejingles**), which has lots of themed collections for free download. It's OK if your **music** and **jingles** folders have subfolders. Find a news jingle and store it beside the **raspberry_radio.py** program, not in the **jingles** folder. Call it **news_jingle.mp3**. We'll play it when the news is read out.



Using Pygame (instead of Playsound) and a PiGlow (instead of the Display-O-Tron), you can flash the lights while music plays. See the code at **magpi.cc/discolights**

You'll Need

- Some MP3 music files
- Some MP3 jingles
- Internet connection
- Display-O-Tron HAT (optional) **magpi.cc/displayotron**
- PiGlow (optional) **magpi.cc/piglow**


The screen shows the artist and song title, like a DAB radio
You can connect to any speakers, such as these portable iPod speakers
DISPLAYOTRON HAT
The Cure
Close to me
INFORMATIVE PHOTOVISUAL TECHNOLOGY

04 Edit the jingles

Radio and podcast jingles often have a slow fade out on them, because the (human) DJ speaks over the end of them. You can edit out silence or cut an excessive fade using Audacity. Install it from the Terminal using `sudo apt install audacity`. You'll find it in the Sound & Video category of your desktop menu. Open a jingle, and click the Play button in the top left to see where the fade becomes inaudible. Click and drag from that point to the end of the sound, and then use **CTRL+X** to cut the selected audio. Select File > Export to save your trimmed MP3 file.

05 Make Raspberry Pi speak

The instructions that make the computer talk are spread out in the program, so here's a simple demo that brings them all together. Enter these instructions in the Python Shell:

```
import pyttsx3
voice = pyttsx3.init()
voice.say("hello")
voice.runAndWait()
```

The `voice.say()` function queues up speech, but it isn't spoken until the `voice.runAndWait()` function runs. You can optionally set the rate (or speed) of the speech. In line 62, we chose 170. It's a bit slower than natural speech, so it's easier to understand. Lower numbers are slower still, and higher numbers

> "You can edit out silence or cut an excessive fade"

are faster. We packaged up the speech instructions in the `output()` function at the start of the program. It also prints the messages to the screen.

06 Choose your DJ

There are several English accents you can choose from, including en-scottish, english-north, english_rp, english_wmids, english-us, en-westindies. As you can see, some use hyphens and some use underscores in their names. RP is short for received pronunciation and is the accent you hear on old BBC news reels. You can change your DJ's voice by adding an instruction like this:

Top Tip

Be selective

Indexing the files takes some time, so choose your favourite tracks and albums, rather than pointing the program at your entire iTunes library.

raspberry_radio.py

DOWNLOAD THE FULL CODE:

magpi.cc/raspradio

Language: **Python 3**

```python
# Raspberry Radio from The MagPi by Sean McManus
import rr_newsreader, random, pyttsx3, os, sys
from playsound import playsound
from tinytag import TinyTag
import dot3k.lcd as lcd # Remove if not using Display-O-Tron

def output(text):
    print(text)
    voice.say(text)
    voice.runAndWait()

def broadcast_news_and_weather():
    playsound('news_jingle.mp3')
    date = rr_newsreader.get_date()
    output(date)
    news_headlines = rr_newsreader.get_news()
    for line in news_headlines:
        output(line)
    weather_report, temperature = rr_newsreader.get_weather()
    output(weather_report)

def index_directory(path, songs, perform_checks):
    print("Processing directory:", path)
    for entry in os.listdir(path):
        path_plus_entry = os.path.join(path, entry)
        if os.path.isdir(path_plus_entry):
            index_directory(path_plus_entry, songs, perform_checks)
        elif entry.endswith('.mp3'):
            tag = TinyTag.get(path_plus_entry)
            if perform_checks == False or \
                  (tag.title is not None and \
                   tag.genre not in ["Books & Spoken", "Christmas"] and \
                   tag.duration < 6000 and \
                   "live" not in tag.album and \
                   "live" not in tag.title):
                songs.append(path_plus_entry)
                print("Track added:", tag.title, "by", tag.artist, "from", tag.album)
    return songs

def play_songs(number_of_songs):
    for _ in range(number_of_songs):
        if random.random() > 0.4:
            jingle_to_play = random.choice(jingles)
            playsound(jingle_to_play)
        song_to_play = random.choice(songs)
        tag = TinyTag.get(song_to_play)
        dj_says = random.choice(
            [   f"What were you doing in {tag.year}? Here's what {tag.artist} was up to.",
                f"Here's a {tag.year} track from the album {tag.album}.",
                f"Fancy some {tag.genre} music? Here's {tag.artist}."
            ])
        output(dj_says)
        DAB_display = (tag.artist + ' ' * 16)[:16] \
                      + tag.title[:32]
        lcd.clear()
        lcd.write(DAB_display)
        playsound(song_to_play)

songs = index_directory("music", [], True)  # folder for music
jingles = index_directory("jingles", [], False) # folder for jingles
voice = pyttsx3.init()
voice.setProperty('rate', 170)
while True:
    broadcast_news_and_weather()
    play_songs(8)
```

```
voice.setProperty('voice', 'english_wmids')
```

If you want to change the voice, we suggest you add the instruction near the end of the program, after you set its speed. You can change the voice back by setting it to `'default'`.

07 Playing MP3s

There are several different ways you can play music tracks from Python on Raspberry Pi. We're using the playsound module because it's easy to use, and the code is concise. You can play an MP3 using just two lines of code:

```
from playsound import playsound
playsound('filename.mp3')
```

By default, playsound pauses the program until the sound has finished. That stops our DJ's speech and the audio files clashing. You can pass an additional value of False to start the sound playing without pausing the program.

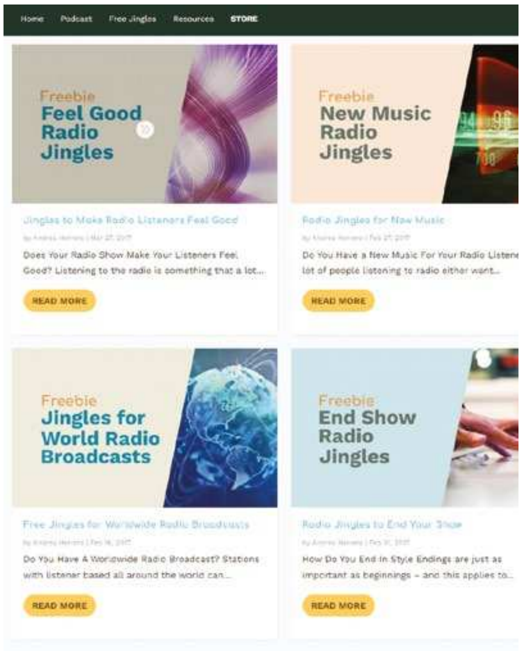
08 Indexing the music files

The `index_directory()` function creates a list containing all the music files in a directory and its subfolders. It might look familiar: we used a similar function to index images for ArtEvolver in issue 119. This time, we pass an additional True or False value to the function to say whether we want to perform quality checks. The quality checks ensure that songs are added that will work well on the radio. First, it checks they have a song title. Then, it excludes tracks that are in the Christmas or Books & Spoken genres, songs longer than 6000 seconds, and songs with 'live' in the album or track name. It's distracting when bursts of applause break through, although we do sacrifice some songs along the way (such as

Top Tip

Speed up indexing

You can speed up the indexing by deleting or commenting out the `print` statements in the `index_directory()` function.



Jingles to Make Radio Listeners Feel Good

Does Your Radio Show Make Your Listeners Feel Good? Listening to the radio is something that a lot...

Radio Jingles for New Music

Do You Have a New Music For Your Radio Listene... lot of people listening to radio either want...

Free Jingles for Worldwide Radio Broadcasts

Do You Have A Worldwide Radio Broadcast? Stations with listener based all around the world can...

Radio Jingles to End Your Show

How Do You End in Style Endings are just as important as beginnings – and this applies to...

Music Radio Creative provides lots of free jingle packs, made with professional voiceover artists and high-energy sound effects

Live to Tell by Madonna). When indexing the music folder, we want to make sure the songs follow those rules. The jingles don't need metadata (and probably don't have it), so we turn off the quality checks for them.

09 Extracting the music metadata

The tinytag module is used to extract metadata from the MP3 file in line 46. We can discover the song title, artist, album, year, genre, and duration. We've assumed that if the song title is present, then other metadata will be too. You can also find the composer, which is well-supported for classical music, but less so for pop and rock. The more accurate and the more complete your metadata is, the more authentic the DJ will sound.

10 Creating the DJ banter

The `dj_says` variable contains a randomly chosen phrase for the DJ to say before the song plays. We use f-strings to insert one or more tags into the phrase. Only a few examples are included here. The more you add, the less repetitive your DJ will sound. Have fun with it: it's easy to make robotic announcements like "This is Prince. Here's *Purple Rain*." It sounds more like a real DJ to say something like "What were you doing in 1984? Here's what Prince was up to!"

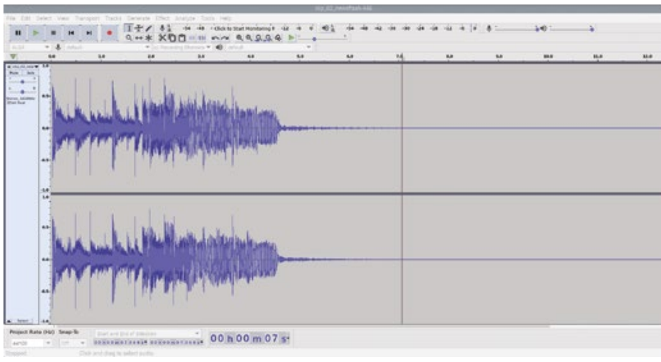
11 Making the DAB display

The best feature of our DAB radio is that it shows us the track and artist that's playing, so if you miss the DJ's introduction, we can still find out what it is. We've used a Display-O-Tron HAT to show the artist name and song name. If you don't have one, you can delete lines 5, and 53 to 56. The display is 16 characters wide and has three rows. The code takes the artist name, adds 16 spaces, and then keeps only the first 16 characters using `[:16]`. That ensures the artist name fills the first line and doesn't spill over. The song name is cut to its first 32 characters so it doesn't wrap from the bottom line to the top.

> "There's lots you can do to extend this project"

12 Build on it!

There's lots you can do to extend this project. You could remove songs from the list when they're played, so they don't get played twice. If you use Pygame to play the music instead of the playsound module, you can display animations while the music plays. The downside is that Pygame doesn't support MP3 files, so you'll need to convert your files to the OGG format. You can find some example code to cycle through the lights on a PiGlow add-on board at **magpi.cc/discolights**



This twelve-second jingle only has between six and seven seconds of audible sound. Using Audacity, you can trim it