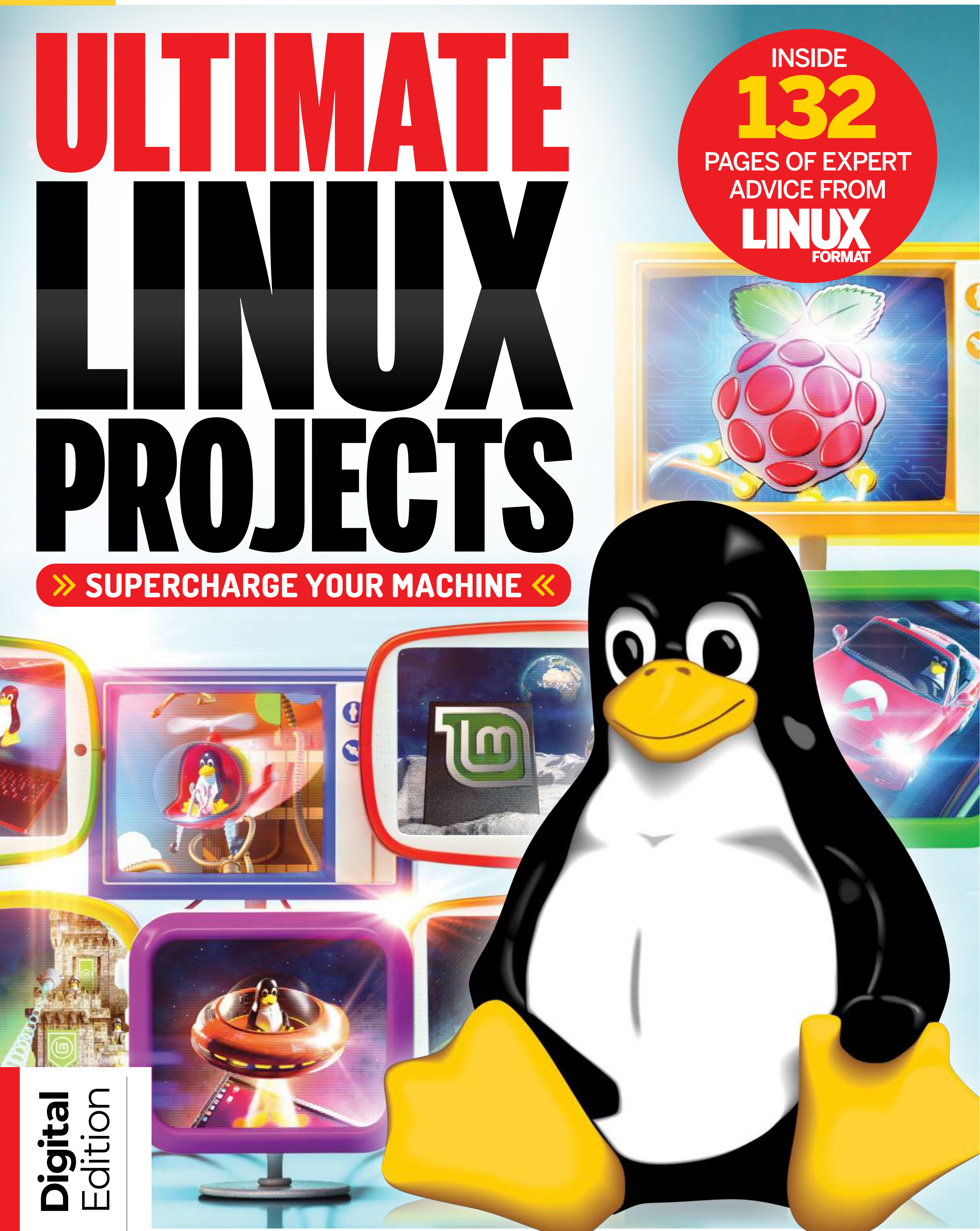
NEW FROM THE MAKERS OF THE NUMBER ONE LINUX MAGAZINE

ULTIMATE LINUX PROJECTS

» SUPERCHARGE YOUR MACHINE «

INSIDE 132 PAGES OF EXPERT ADVICE FROM LINUX FORMAT



Digital Edition

FUTURE

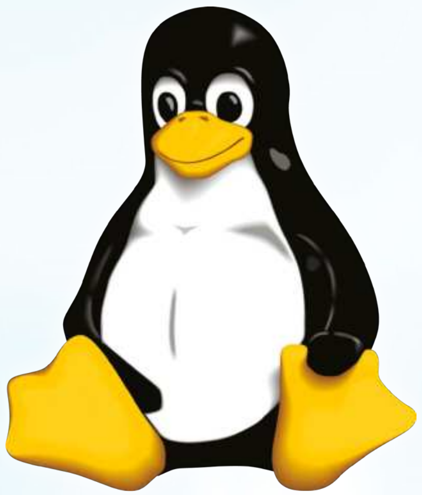
» Mint 20 » Smart home office » Retro games » Livestreaming
» Robotic OS » Safeguard data » Escape Google Photos

ULTIMATE LINUX PROJECTS

Linux is the gateway to exciting open-source software, stimulating coding projects and, of course, impressive Raspberry Pi machinations. In *Ultimate Linux Projects*, we throw those doors wide open and explore the opportunities that Linus Torvalds' creation can offer us.

Inside, the minds behind *Linux Format* magazine provide their expertise to bring you specialist tutorials, guides and advice to help you get some extra joy from your machine. From coding masterclasses that will tell you how to tap into weather satellites and more, to plug-in-and-play Raspberry Pi guides, and open-source alternatives to popular services like Google Photos and Plex – we're sure that if you're a Linux fan, you'll find something to obsess over here.

So, pop that kettle on, boot up your machine and get ready to explore the possibilities that Linux has to offer.



FUTURE

ULTIMATE LINUX PROJECTS

Future PLC Quay House, The Ambury, Bath, BA1 1UA

Ultimate Linux Projects Editorial
Editor **Drew Sleep**
Art Editor **Efrain Hernandez-Mendoza**
Senior Art Editor **Andy Downes**
Head of Art & Design **Greg Whitaker**
Editorial Director **Jon White**

Linux Format Editorial
Editor **Neil Mohr**
Technical Editor **Jonni Bidwell**
Art Editor **Efrain Hernandez-Mendoza**
Group editor in chief **Graham Barlow**
Group art director **Jo Gulliver**

Cover images
magictorch.com
Raspberry Pi is a trademark of the Raspberry Pi Foundation.
Tux credit: Larry Ewing (lewing@isc.tamu.edu) and *The GIMP*.

Photography
All copyrights and trademarks are recognised and respected

Advertising
Media packs are available on request
Commercial Director **Clare Dove**

International
Head of Print Licensing **Rachel Shaw**
licensing@futurenet.com
www.futurecontenthub.com

Circulation
Head of Newstrade **Tim Mathers**

Production
Head of Production **Mark Constance**
Production Project Manager **Matthew Eglinton**
Advertising Production Manager **Joanne Crosby**
Digital Editions Controller **Jason Hudson**
Production Managers **Keely Miller, Nola Cokely, Vivienne Calvert, Fran Twentyman**

Printed by William Gibbons, 26 Planetary Road, Willenhall, West Midlands, WV13 3XT

Distributed by Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU
www.marketforce.co.uk Tel: 0203 787 9001

Ultimate Linux Projects (TCB4123)
© 2022 Future Publishing Limited

FUTURE

Connectors.
Creators.
Experience
Makers.

Future plc is a public company quoted on the London Stock Exchange (symbol: FUTR)
www.futureplc.com

Chief executive **Zillah Byng-Thorne**
Non-executive chairman **Richard Huntingford**
Chief financial officer **Penny Ladkin-Brand**

Tel +44 (0)1225 442 244

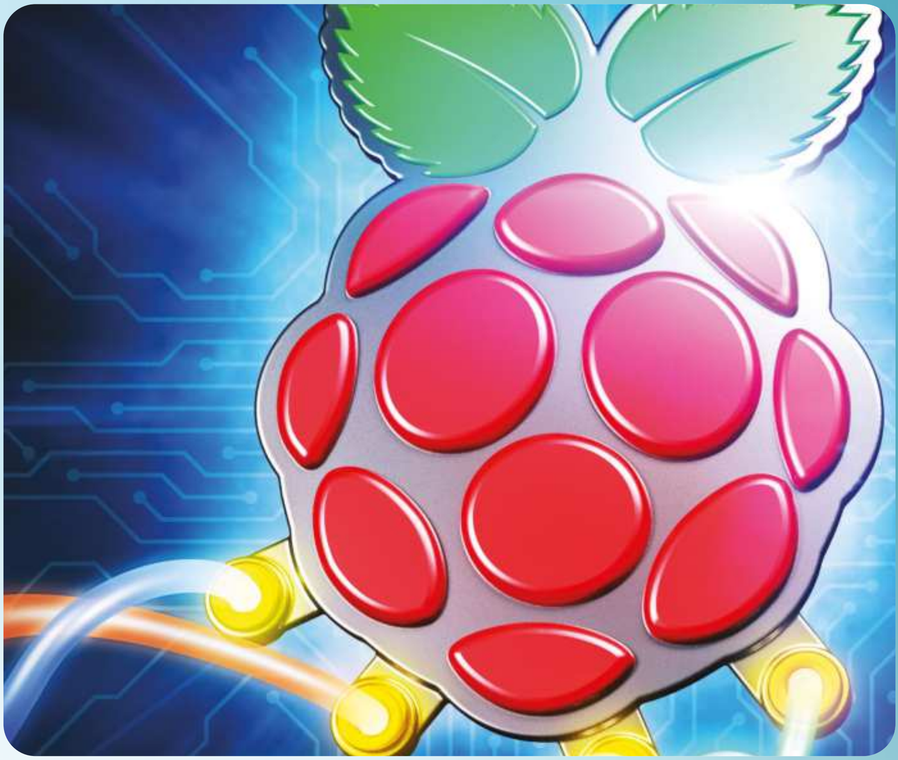
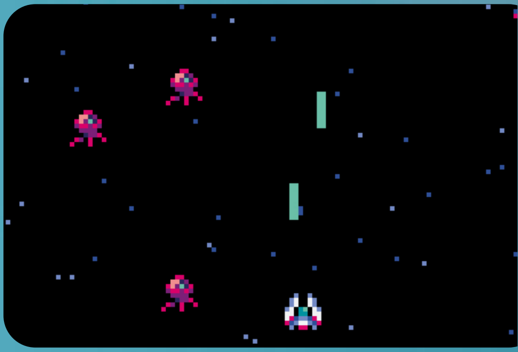
Part of the
LINUX FORMAT
The #1 open source mag
bookazine series

Widely Recycled

ipso. For press freedom with responsibility

Contents

ULTIMATE LINUX PROJECTS

CUSTOMISE MINT 20!

Linux Mint is fantastic, but it's also flexible, malleable and tweakable. **Jonni Bidwell** shows you how to truly make it your own.

Linux Mint continues to go from strength to strength, as you'll know if you've already had a play with the latest 20.2 release. If not, what are you waiting for? Head to linuxmint.com and witness the, er, Mint-ness forthwith. See, now you want to install it, don't you? And that is just the beginning of the adventure. One of the things that makes Linux Mint so cool is its potential for configurability. It's often said (by the brains behind *Linux Format*) that Mint is an ideal beginner's distro, and it turns out it's also ideal (we say) for beginners to tinker around with.

The flagship Cinnamon desktop can be transformed not just with swishy effects and colourful themes, but with all kinds of extensions, applets and desklets (collectively known as 'spices' in Cinnamon parlance). And MATE and Xfce, the desktops featured in other editions of Mint, are equally seasonable. But we can do better than that: why not mix it up and install a whole new desktop environment? We'll show you how to install the outstanding KDE Plasma, we'll even show you how to make it work with the state-of-the-art Wayland display protocol.

If you like things slimline, we'll show you how to go minimal with the featherweight Sway desktop, again powered by Wayland. Sway is based on the i3 window manager, popular with power users and those who cannot abide desktop bloat. We'll have you doing everything in the terminal and tiling like a pro in no time.

And just so no one gets hurt, we'll start with a handy reminder about how you can use Timeshift to easily undo any desktop-related mishaps.

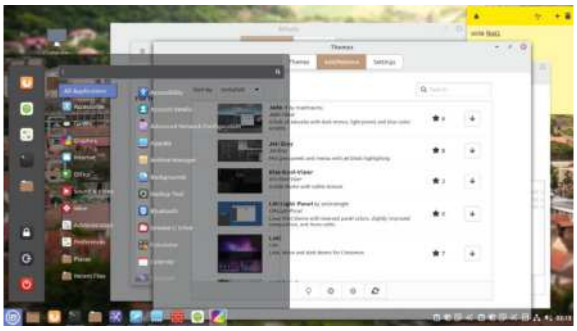
The joy of tinkering

Sort out rollbacks so you can customise Linux Mint with impunity and immunity (to problems).

Purveyors of historic issues of *Linux Format* magazine may be able to correct this, but as far as our research can tell, the first mention of Linux Mint in the core magazine came in the Distrowatch column of LXF094, when Mint 2.2 was released. Even back then, Mint was notable for its out-of-the-box experience, bundling codecs, Java and Flash plug-ins and wireless firmware, saving users from having to shoehorn those on there using fragile instructions from a random forum post.

That experience remains central to Mint, and though wireless hardware is well-supported on most distros (and no one needs Flash any more), it still shines. Right from the Welcome screen in fact, which will invite you to set up backups using *Timeshift*, switch keyboard layouts, or send and receive files from another machine using *Warpinator*. Oh, and there are minimise buttons on windows in Cinnamon – a trend that feels like it is quickly disappearing on other desktops, but one which makes many a user feel at home.

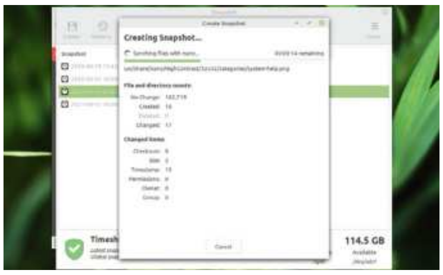
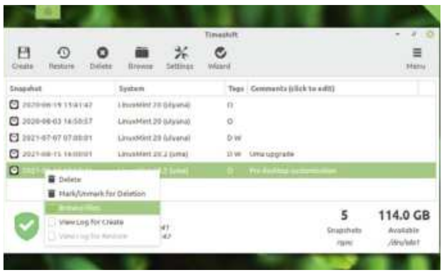
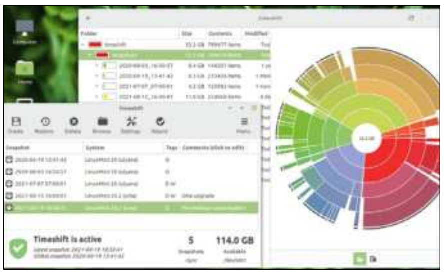
Other desktops are going full steam ahead with Client Side Decorations (CSD, which allows applications to draw their own titlebars). This might allow programs to make best use of space and provide a coherent interface. Or it might make them look inconsistent, clumsy or other pejorative terms – it depends your own personal preference. At any rate, Mint's X-apps are refreshing in their avoidance of the CSD wave, and Mint's substantial fanbase suggests that the distro's creators are still doing all they can to keep their users happy.


Making the menu transparent and pasting Sticky Notes are but one way of customising Mint.

Sooner or later, though, you're going to want to change things up. It generally starts with changing your desktop background and Cinnamon theme. These are important, but also quite easy – easy enough that you don't need us to tell you how to do them. What we'll be doing is a little more earth-moving. Tectonic stuff like installing whole new desktop environments, swish display managers, maybe even switching to the Wayland display protocol. And while these aren't without risk, Mint's *Timeshift* program allows you to back up your system files (much like Apple's *Time Machine* or Windows *Restore Points*), affording an easy way to undo any desktop mishaps.

Even if nothing goes wrong, it's handy to be able to roll back to a cleaner system rather than unpick changes manually: see the walkthrough below. If you've already got *Timeshift* set up, take a manual snapshot now before pouring in all the packages over the page. Go on, you know it makes sense!

EASY ROLLBACKS WITH TIMESHIFT

1 Start Timeshift

Fire up *Timeshift* and set it up to take a couple of daily snapshots to a local drive with plenty of space (at least 1GB more than the current filesystem size). Timeshift will only back-up system files by default, so files in your home directory aren't included. There are better tools for backing these up.

2 Take a Snapshot

It might spring into action immediately if the clocks align. But don't worry if it doesn't, just hit the Back Up Now button to take an on-demand snapshot. Timeshift backs up incrementally so only changed files are stored. Once the snapshot completes, add a helpful description to help future you keep track.

3 Restore a snapshot

If something goes wrong, you can now easily restore a Snapshot by clicking the button. You might want to examine the files within first, which you can do by right-clicking. Even if things go really wrong, and Mint no longer starts, you can use Timeshift from a live medium. Just point it to the **/timeshift** directory.

»

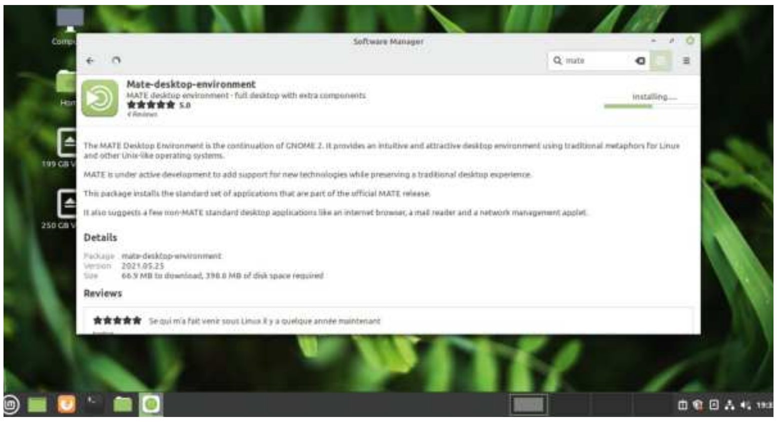
Tweaking Cinnamon

See how easy it is to make your mark on Linux Mint's flagship desktop environment and beyond!

If you haven't had a nosey around Cinnamon's many settings, you might be pleasantly surprised at how configurable it is. When Mint 20 was released, much ado was made about the Mint-Y theme now having fifty shades of colour variations (okay, it was 32), but we haven't found our favourite hue yet. Check out the palette by opening up the main menu and going to Preferences then Themes.

Dark themes are so common these days, even Windows has them, but Mint has a corresponding dark theme for each variation. You can download whole new themes from the web, too; just don't expect them to all be in line with your design preferences. Hidden away in the Settings section of the Themes dialog are some oft-overlooked options for scrollbars, including the option to disable overlays.

To customise the main panel, go to Preferences and then select Applets. Now you can add all kinds of shortcuts and widgets. For example, select the Expo applet and click the + at the bottom to add a shortcut (via a smooth animation) to an expo-style overview of your workspaces. For even more efficient workspace shifting (at the cost of some panel estate) add the Workspace Switcher applet. If you want to disable Expo, or any other applet, just click the – button. Like themes, third-party applets can be downloaded by visiting the appropriate tab. Downloaded themes come with no guarantees, so they come with an uninstall option in case they annoy you.



Mate-desktop-environment
MATE desktop environment - full desktop with extra components
★★★★★ 5.0
4 Reviews

The MATE Desktop Environment is the continuation of GNOME 2. It provides an intuitive and attractive desktop environment using traditional metaphors for Linux and other Unix-like operating systems.

MATE is under active development to add support for new technologies while preserving a traditional desktop experience.

This package installs the standard set of applications that are part of the official MATE release.

It also suggests a few non-MATE standard desktop applications like an internet browser, a mail reader and a network management applet.

Details

Package mate-desktop-environment
Version 2021.05.25
Size 66.9 MB to download, 398.0 MB of disk space required

Reviews

★★★★★ Se qui m'a fait venir sous Linux il y a quelque année maintenant

If you want to experiment with Xfce it's easy to install it and all its apps from the Software Manager.

There are some extremely pleasant new wallpapers in Mint 20.2, and we recommend to right-click the desktop and choose Change Background if you haven't already perused them. But before you click there, take a look again at that desktop context menu. In particular, have a gander at the Add Desklets option. There aren't many pre-installed desklets, but if you want a digital clock or photo frame on your desktop then you're in luck. If you delve into the Download tab you'll find plenty more, including an analogue clock as well as more productive tools such as the Google Calendar desklet.

The final flavour in Cinnamon's Spices cabinet is extensions. These change the way the whole desktop behaves. Again, there aren't many installed by default, but head to the Download tab and it won't take long for 'Wobbly Windows' to catch your eye. Hopefully, you have better luck than us with that extension. If you're of a certain pedigree, you'll remember the Desktop Cube extension too, taking your workspace switching to a whole new level. There are other extensions which some may write off as desktop fripperies, such as being able to tweak window decorations, shadows or transparency, but there's no harm trying them out.

By default, desktop effects are enabled in Cinnamon, unless your install has fallen back to software rendering – in which case have a look at the *Driver Manager* for possible remedies. These effects aren't the sort of in-your-face, windows catching fire, stunts of the early 'aughts, but have been designed to help users navigate

» MUTATING MATE AND EXTENDING XFCE

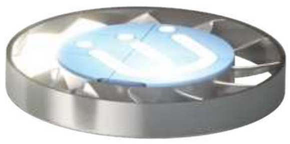
But what about Mint's other flavours, surely they can be customised too? Indeed they can. You'll find a similar arrangement with themes, extensions and effects in both of these. All three of the Mint desktops are ultimately based on GTK3, so the fundamental desktop elements can be themed with a standard GTK theme from the likes of **https://www.gnome-look.org/browse?cat=135**. Extract any themes you like into your **~/.themes** directory, and they should pop up in the theming options. If you're using Mint MATE or Xfce, but desire to try Cinnamon, then that's easy. The standard desktop is available through the **mint-meta-cinnamon** package, or you can get a minimal edition via cinnamon-core. Conversely, if you are using Cinnamon and want to try Xfce or MATE, hit up the **mint-meta-xfce** and **mint-meta-mate** (rolls off the tongue nicely that one) packages. Again, there are minimal packages too if you only want the core applications. These will all add a session to your login screen, so you can choose your desktop from the menu to the right of your username. Speaking of the login screen, you can tweak that too. Just go to Administration > Login Window from the main menu. Such cosmetic tweaking can happen even earlier in the boot process, GRUB can be themed and so too can the Plymouth splash screen (**https://www.gnome-look.org/browse?cat=108**).

the desktop. It's reassuring (sometimes) to see where applications were called into being from, and where they disappear to when they minimise. Be that as it may, you might want to turn these off, and this you can easily do by from the Effects option in the Preferences menu.

Besides eschewing the Gnome desktop, Mint has made a couple of other choices that fly in the face of desktop Ubuntu. One is the absence of the Snap daemon, which prevents installing packages from Canonical's Snapcraft store. Another is that there's no Wayland support (at the time of writing) in either Cinnamon, Xfce or MATE, the three officially supported Mint desktops. You might not care about next-gen, cross-distro packaging systems or banishing old X.org from your machine. Indeed, plenty in the Linux Mint community seem to share this sentiment. But there's good stuff in the Snap store, so let's look now at how we might enable that now. Plus, Wayland is pretty impressive now, so we'll look at that over the page, once we have a desktop that supports it.

When Mint 20 was released sans Snap support, an immediate consequence was that there was no way to install the *Chromium* web browser, since Ubuntu (20.04 and later) now only packages it as a Snap. We think more people should use *Firefox*, and if our user agent tracking on **linuxformat.com** is anything to go by, it seems they are. But it sets a potentially worrying precedent; if *Chromium* was to be abandoned by Ubuntu's DEB packagers, then perhaps other popular applications might go the same way. We don't think you should worry, as so far there's been no sign of that. Team Mint now packages its own *Chromium* DEB package, so if you're craving a hint of Google in your browsing then fetch it from the Mint App Store or with a good ol' fashioned *apt* incantation at the command line.

We've seen that software is also available as Flatpaks, and that this is enables a wealth of software to be installed from outside the Ubuntu (and Mint) repositories. Flatpak is actually enabled out of the box on Linux Mint, and if you look carefully you'll find some Flatpak applications in the *Software Manager*. For example, if you want the latest version of *GIMP*, the Flatpak edition is probably the second one in the search results. You'll also find Flatpaks of *Spotify* and *Steam* so you can queue up your Rush playlist and play *Space Invaders* like it's the olde days.



If you do want Snaps, perhaps to get the latest version of *Blender* or the PyCharm IDE, that's easy to sort out from the Terminal. We first remove the file that prevents the Snap daemon from being installed, and then install it with *apt*:

```
$ sudo rm /etc/apt/prefs.d/nosnap.pref
$ sudo apt install snapd
```

Snaps can now be searched and installed from the command line. You can also browse what's on offer at **https://snapcraft.io**, but for a complete GUI experience you'll want to install the *Snap Store* too. Appropriately, it's available as a Snap and it can be yours with:

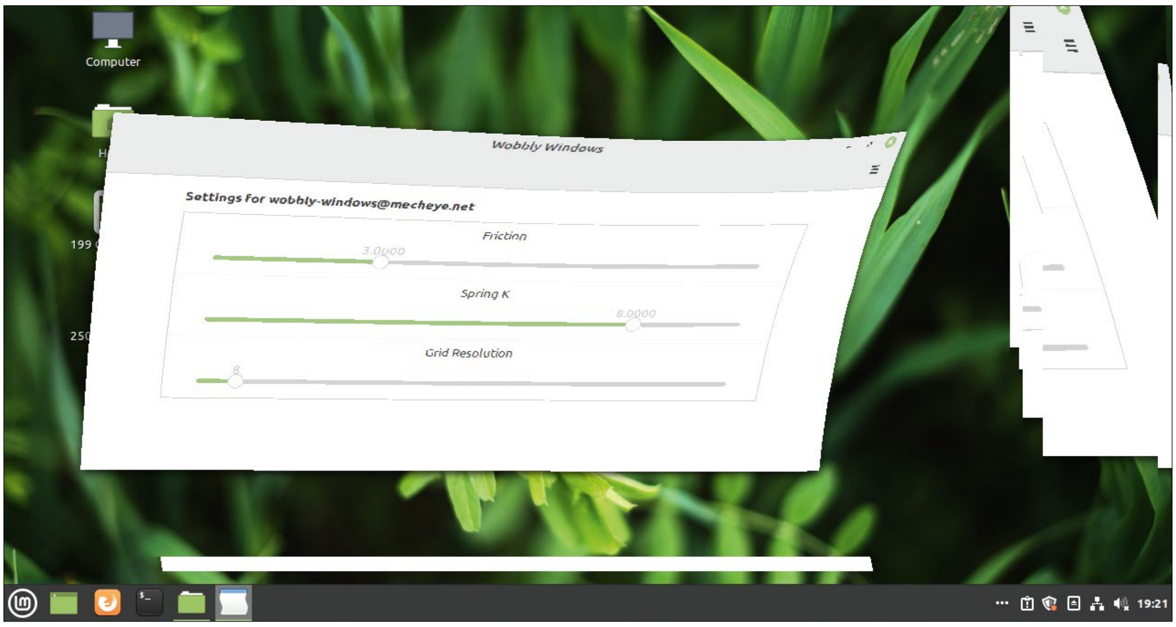
```
$ sudo snapd install snap-store
```

USEFUL DESKTOP EFFECTS

"These effects aren't the sort of in-your-face, windows catching fire, stunts of the early 'aughts."

One easy and surefire way to turn heads (or whatever is the virtual equivalent) is to bling your terminal with a little transparency. Not only does this look pleasant, but if you arrange your windows correctly, and get into the habit of arranging them as such, it can be pretty useful as well. Endowed with X-ray vision, one can make out both the terminal itself and the web browser or whatever substrata lay below this.

To enable the opulent opacity effect in the Terminal, go to Edit > Preferences then select the current profile (it will be named Unnamed Profile if you haven't given it), untick the setting about system theme transparency, then tick the box above it. Play with the slider to find the optimum opacity. Or, if you'd rather more drastic changes look over the page, in which we install the high-fidelity KDE Plasma desktop environment.



Settings for wobbly-windows@mecheye.net
Friction 3.0000
Spring K 8.0000
Grid Resolution 8

We have fond memories of wobbly windows confusing our graphics drivers so its good to see this still lives on.

Installing KDE Plasma

Transform your desktop with the smooth, svelte, sumptuous experience that is KDE Plasma and go complete next-gen with the Wayland too!

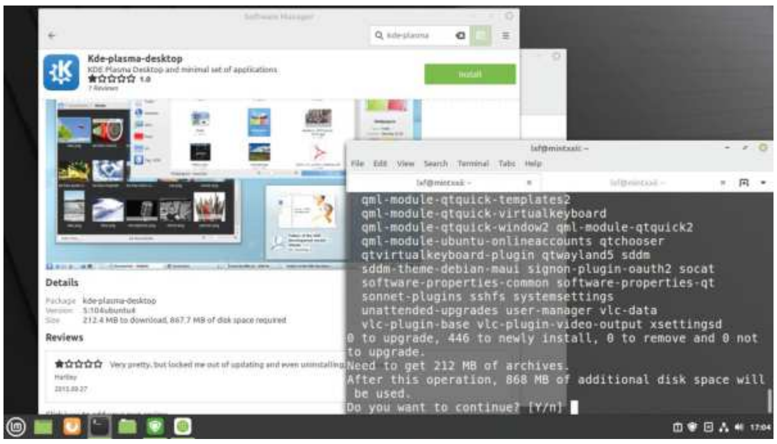
There's no official Linux Mint KDE edition these days, but that doesn't mean Mint users should miss out on the wonderful experience that is KDE Plasma. It's modern, but still has a traditional applications menu. It's incredibly polished, but is nowhere near the resource hog it used to be. Oh, and its *Dolphin* file manager is a joy to work with, especially if you're finding *Nemo* a little too simplistic. Be that as it may, installing a new desktop environment comes with its own consequences, and it's good to be aware of these before you blame us for ruining your system.

Firstly, there's the disk space hit. The smallest KDE Plasma metapackage provides a minimal desktop, but according to the screenshot it pulls in some 850MB of dependencies in 446 packages. If you go for the full-fat edition, with all the applications from the KDE ecosystem, that will cost you close to 3GB. Next is the duplication of core utilities such as text editors, media players and screenshot tools. These all start to crowd your application menus, and if you use, say, KDE's *Dolphin* file manager in Cinnamon, it looks a bit odd.

Finally, it's sometimes hard for *apt* to completely uninstall a desktop. It's likewise hard to repair a broken desktop after you attempt to clear out packages manually. So don't do that; instead, try things out within a virtual machine first, or make use of *Timeshift* to restore things to a known good state (as we demonstrated earlier).

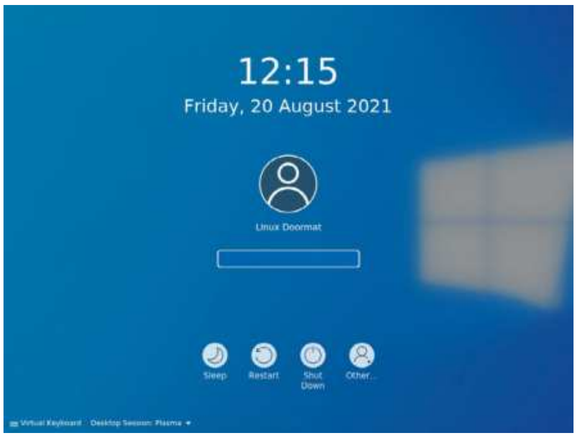
Having installed (at least) the Plasma desktop we can opt to change the display manager from the Mint-themed LightDM to the Qt-powered SDDM (Simple Desktop Display Manager). If you installed from the command line this will be offered to you, and if you didn't you can get to the configuration by running:

```
$ sudo dpkg-reconfigure lightdm
```



```
 qml-module-qtquick-templates2
 qml-module-qtquick-virtualkeyboard
 qml-module-qtquick-window2 qml-module-qtquick2
 qml-module-ubuntu-onlineaccounts qtchooser
 qtvirtualkeyboard-plugin qtwayland5 sddm
 sddm-theme-debian-maui signon-plugin-oauth2 socat
 software-properties-common software-properties-qt
 sonnet-plugins sshfs systemsettings
 unattended-upgrades user-manager vlc-data
 vlc-plugin-base vlc-plugin-video-output xsettingsd
0 to upgrade, 446 to newly install, 0 to remove and 0 not
to upgrade.
Need to get 212 MB of archives.
After this operation, 868 MB of additional disk space will
 be used.
Do you want to continue? [Y/n]
```

The lightest suite of KDE applications weighs in at around 850MB, but it is quite outstanding.



For some reason we found this SDDM theme much more relaxing than the austere default.

Both display managers work with all major Linux desktop environments (and lightweight window managers), so which you choose is a matter of personal preference – or whichever you can find the prettiest themes for. The default SDDM theme is probably not most people's idea of pretty, but once you're logged in it's easy to change this from Administration > Login Screen (SDDM). For some reason, perhaps an attempt at irony, we found ourselves using a Windows-like login theme. Never mind that, you've probably just found yourself immersed in the wonderful world of KDE Plasma. Behold the cool Breeze theme, marvel at the polish and feel at home with the knowledge that all your favourite Mint tooling is just a click away.

KDE 4, now largely retired, received occasional criticism for being too configurable. In part, this was fair. Every widget (and there were a lot of widgets) could be configured, a clumsy edit mode gave them a handle about which they could be rotated or stretched, and one was sometimes left wondering what the point of all this was. Worse, successive iterations of KDE 4 got very good at hiding all kinds of key options just when you thought you'd got a handle on where they ought to be. That version of KDE also faced criticism for being something of a resource hog, and shipping with all kinds of graphical frippery enabled. Modest machines would probably have been fine with this, but as the graphics driver ecosystem of the era was far more fragile back then, hardware acceleration was not something that someone could count upon.

You'll be pleased to hear, then, that KDE 5 (or KDE Plasma 5 as the desktop prefers to be called) is a much sleeker animal. In our tests it did use up a tiny bit more memory than Cinnamon, and slightly more than Xfce and MATE, but what's a hundred or so megabytes between friends? It makes not one iota of difference once you start memory-slayer *Chromium*. Plasma is certainly configurable, but in a way that is not overbearing. Take the default, medium-weight launcher menu (at the bottom-left, as it should be). Right-click it and select Show Alternatives. You will see it can be swapped for a modern, full-screen launcher (sort of like Gnome) or a more classic cascading menu design.

KDE comes with its own graphical application store called *Discover* (one of few KDE apps not to capitalise on any opportunity for an unnecessary letter K). You'll find this already pinned to the favourites menu, and you might also prefer it to Mint's native *Software Manager*. One thing you'll want to do is sort out Flatpak support in Discover. Fire up a terminal (try the *Konsole* application) and run:

```
$ sudo apt install plasma-discover-backend-flatpak
```

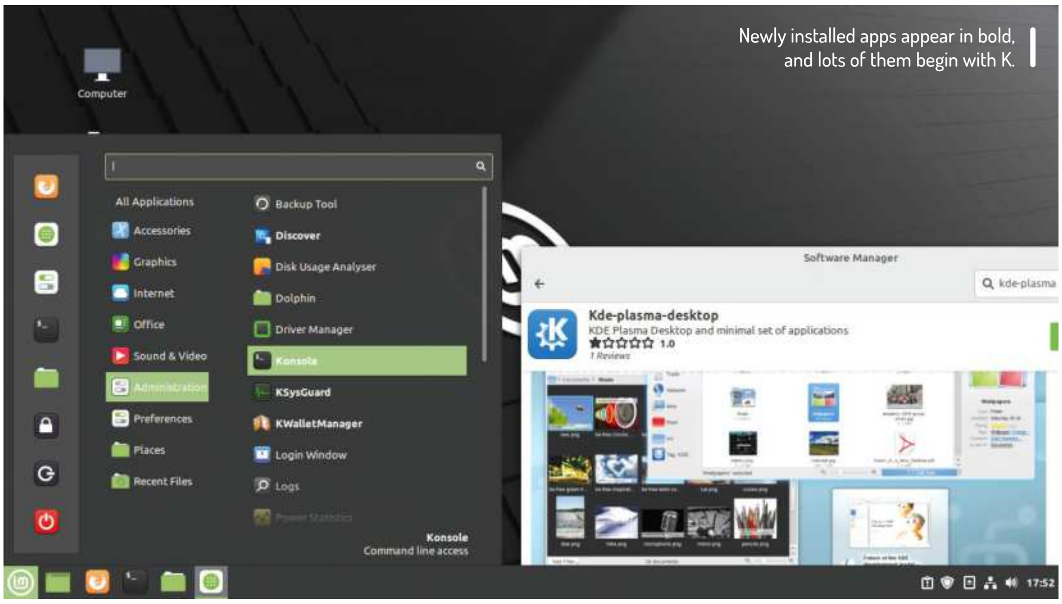
You can now, after restarting *Discover*, browse FlatHub (or any other Flatpak repos) by adding them via the Settings option at the bottom right. Just click Show Contents to the right of the repo name. Flatpak is a much more decentralised idea than Snaps; anyone can set up their own Flatpak repository, but the only Snap outlet in town is Canonical's Snapcraft. Both forms are potentially risky though, since there's little to stop a scoundrel uploading a rogue Flatpak or Snap. And while both have some sandboxing capabilities we have no compunction to endorse the downloading of random binaries. Popular applications are easy to spot on FlatHub and common Snap packages have a reassuring 'Verified' badge.

We mentioned that Wayland isn't explicitly supported by any of the Mint desktops, but that is changing. In the latest MATE release, a great deal of the desktop now works natively with Wayland, so if you switch the Marco window manager for Compton then you're well on your way to display protocol future. Xfce 4.18 plans to introduce support, though that may be a long way off. So it's really desktops, rather than distros, that enable Wayland – and as luck would have it KDE Plasma has support built in to its Kwin window manager. There's just a couple of packages to pull in to bring it to life:

```
$ sudo apt install plasma-workspace-wayland
```

Now if you log out, a new session called Plasma (Wayland) will be available from the menu. The Plasma experience on Wayland has come a long way this year. We're told it even works with the proprietary Nvidia driver now.

One thing that might strike you as jarring about Plasma in general is that your session is automatically saved. If you prefer to start each time without all those stray terminals and whatever else you left open, head over to Settings > Startup and Shutdown > Desktop Session.



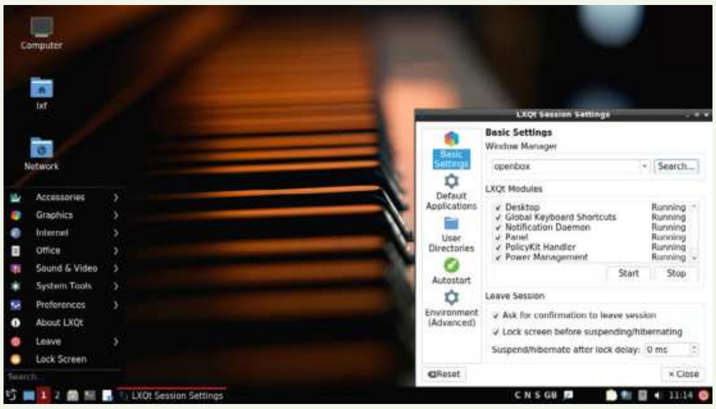
Newly installed apps appear in bold, and lots of them begin with K.

» LXQT

If you're enamoured with the Qt toolkit, but crave a lighter, nimbler desktop, you should look no further than LXQt. That's what powers the current LTS edition of Lubuntu, and it's what could power your new Livingston-seagull-like Mint desktop. Over the page, we'll go pretty much as far as we can go without abandoning the GUI altogether. But if you want something a little more user-friendly and less gymnastic keyboard shortcut-orientated, LXQt may well be for you.

LXQt is the spiritual successor to the GTK2-powered LXDE desktop that used to power Lubuntu and Raspbian. Rather than move to GTK3, which at the time was seen as bloated, LXDE teamed up with the RazorQt effort and the result is LXQt. You can install it with a simple `sudo apt install lxqt openbox`. If you haven't already installed KDE this will pull in around 400MB of dependencies, but if you've already installed Qt et al, the footprint will be much lighter. Don't forget to take a snapshot first, though.

When you start LXQt you'll be prompted to choose a window manager. By default it uses openbox, but it can use Cinnamon's Mutter, Xfce's Xfwm4 or, if you really want to make it pretty, Kwin from Plasma. Openbox is by far the lightest, and for non-scientific comparison purposes a plain LXQt/Openbox session occupied around 500MB of our memories.



LXQt Session Settings

Basic Settings

Window Manager: openbox

LXQt Modules

✓ Desktop	Running
✓ Global Keyboard Shortcuts	Running
✓ Notification Daemon	Running
✓ Panel	Running
✓ PolicyKit Handler	Running
✓ Power Management	Running

Leave Session

✓ Ask for confirmation to leave session

✓ Lock screen before suspending/hibernating

Suspend/hibernate after lock delay: 0 ms

LXQt might be just be the lightest desktop environment that still provides all the friendly GUI crutches we've come to rely on.

Ultralight Mint

Embrace minimality and learn some keyboard gymnastics with the featherweight Sway desktop.

One of our new favourite Ubuntu-derivatives is Regolith Linux. It's fairly unique in its choice of the ultra-light i3 tiling window manager. Tiling window managers take some getting used to, and also a whole lot of configuration, but Regolith ships with remarkably sane defaults and easy-to-learn keyboard shortcuts (i3 is very much keyboard- driven, but converts say they never looked back). Also, it still has all of the GNOME infrastructure and applications for managing sessions and settings, so all of your system administration can be carried out with familiar GUI apps. We're big fans of Pop!_OS too, and in particular its COSMIC (Computer Operating System Main Interface Components) desktop. This features a tiling mode that, while not having the diminutive resource footprint of i3, offers users a gentle introduction to the joy and efficiency of keyboard shortcuts and mouse gestures in harmony.

WINDOW TILING IS AN ART

"To make the most of window tiling, much like an Etch A Sketch you'll want to use a combination of horizontal and vertical arrangements."

There's no reason we shouldn't have these sorts of things in Mint too; the i3 window manager is in the Ubuntu repositories. But we're going to try something else. Sway is a lightweight window manager inspired heavily by i3, except that it is for Wayland. If you are familiar with i3, you will quickly get the hang of Sway; most of the default keys are the same, and you can even use your own i3status scripts. In fact, you should be able to use your **i3config** file without modifications. Sway is in the Ubuntu repos, but it's an old version from January 2020. It would take some work of the compiling variety to get the latest version working, so let's just install the repo version to dip our toes in:

```
$ sudo apt install sway
```

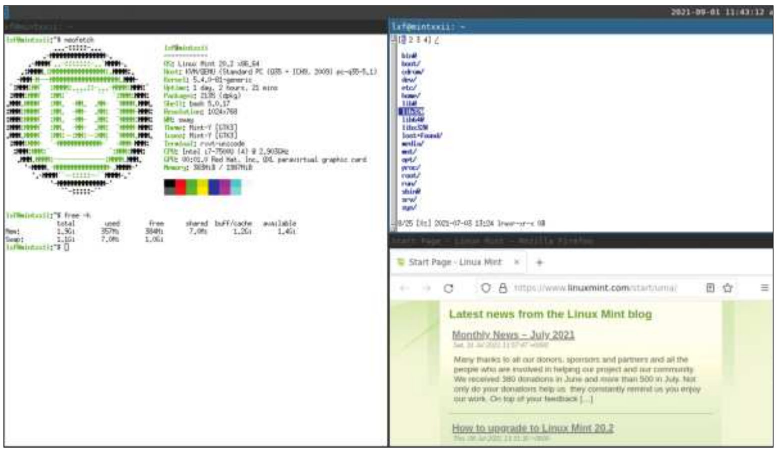
As before, the login screen should now have a Sway session. Dive into it and you should see the rather fetching Sway logo and top bar. Try anything with the mouse (besides moving the pointer) and you'll realise that you're not in Cinnamon any more – nothing reacts to being clicked, double-clicked, dragged or any such thing. Sway is all about keyboard commands: try pressing Super (the brand-independent name for the Windows key) and Enter. A terminal should spring into life, so now you can at least practise your *Bash* scripting for a while.

Now try pressing Super+2; the terminal will disappear, but not really – cast your eye to the top-left and you'll see we've just moved to a new virtual desktop. If you try to open another one with Super+3, you'll see that this doesn't happen. We didn't need a third desktop because we hadn't opened anything on our second, so Sway quietly renamed the previous workspace to 3.

Go back to workspace 1 and hit Super+Enter to open another terminal. Now you can see what tiling window managers are all about. The first terminal, that was occupying the whole desktop, obsequiously squishes over to the left, making room for a new terminal to the right. If you like, you can start any program you want from either of these terminals (notice the focus follows the mouse so you don't need to click in either one).

But Sway also has its own application launcher, after a fashion. Hit Super+D and you'll see some commands in lexicographical order (some beginning with numbers, and several beginning with the letter A, probably). Start typing the first few characters of `firefox` and you'll see this list get rapidly smaller. Press Enter when `firefox` is highlighted in blue to start it.

If you already had a couple of windows open, things by now will probably be getting a little cramped. You could close one of them (either with a Ctrl+D or traditionally with the Close button). But now is also a good time to introduce window resizing. With at least two windows open, hit Super+R to enter resize mode. You'll see this indicated in the top-left of the status bar. Now you can use the H and L keys (like *Vi*) to make the active window (and remember you can change this by



```
2021-09-01 11:43:12
OS: Linux Mint 20.2 x86_64
Host: KVM/QEMU (Standard PC (Q35 + ICH9, 2009) pc-q35-5.1)
Kernel: 5.4.0-81-generic
Uptime: 1 day, 2 hours, 21 mins
Packages: 2135 (dpkg)
Shell: bash 5.0.17
Resolution: 1024x768
WM: sway
Theme: Mint-Y [GTK3]
Icons: Mint-Y [GTK3]
Terminal: rxvt-unicode
CPU: Intel i7-7500U (4) @ 2.903GHz
GPU: 00:02.0 Red Hat, Inc. QXL paravirtual graphic card
Memory: 383MiB / 1987MiB

$ free -h
       total    used    free    shared  buff/cache   available
Mem:   1.9Gi    357Mi   384Mi   7.0Mi   1.2Gi        1.4Gi
Swap:  1.1Gi    7.0Mi   1.0Gi
```

Start Page - Linux Mint

https://www.linuxmint.com/start/uma/

Latest news from the Linux Mint blog

Monthly News – July 2021

Many thanks to all our donors, sponsors and partners and all the people who are involved in helping our project and our community. We received 380 donations in June and more than 500 in July. Not only do your donations help us, they constantly remind us you enjoy our work. On top of your feedback [...]

How to upgrade to Linux Mint 20.2

The rxvt-unicode terminal isn't that pretty in its default state, but we're sure you can make it so.

hovering the mouse over a new window) wider or narrower. That covers one dimension. Press Esc to exit resize mode and the hit Super+E while focused on the leftmost window. Bam! Vertical windows. Be careful of case sensitivity here, because Super+Shift+E (often abbreviated to Super+E) is the shortcut to exit Sway.

In order to make the most of window tiling, much like an Etch A Sketch you'll want to get the hang of using a combination of horizontal and vertical arrangements. This seems straightforward at first, but there are a few subtleties that are best experienced for yourself. Before, the Super+E shortcut operated on several windows at once and tiled them uniformly. They were grouped together along a common dimension. But if you hit Super+V on one of them, then hit Super+Enter to open another terminal (or open any other application for that matter), it will open in the other orientation. In this way you'll see that Super+E and Super+V act as orientation triggers, and you'll notice that the former highlights the right edge of window, and the latter highlights the lower edge, giving a hint as to how the next window will spawn.

Sway's default configuration file can be found at **/etc/sway/config**, but rather than edit this directly, copy it to **~/.config/sway/config** to make personal changes. For example, the line:

```
output * bg /usr/share/…
```

sets the background on all displays. You can change this, or indeed set a custom background for each display, by modifying this. The `output` directive actually controls all sorts of things fundamentally related to the display. For example, if you're running Sway on a virtual machine, putting the final touches to the overdue cover feature, you'll probably need to add a line of the form:

```
output Virtual_1 resolution 1280x720
```

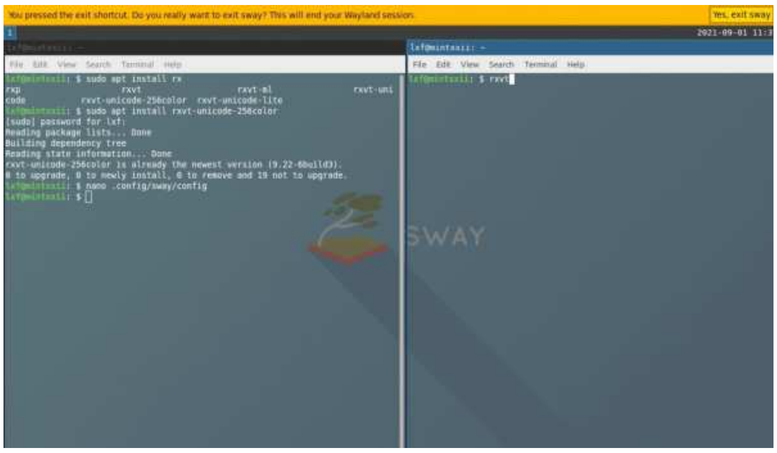
in order to make your screen grabs have the correct aspect ratio. You can get a list of display names and modes by running:

```
$ swaymsg -t get_outputs
```

Sway and i3 are famed for their low memory footprints. But in order to keep these low, and maintain a minimal desktop aesthetic, one has to use lighter applications too. This isn't quite the place to show how much stuff you can run from the terminal, but you should check out the *nnn* file manager, the *w3m* web browser, *mpv* the video player and the *ncmpcpp* front-end to the *mpd* music player. What we will demonstrate is how to swap the default Gnome terminal for something a little lighter.

You might have noticed when we installed Sway that it pulled in a package called **Suckless Tools**. Suckless (**https://suckless.org**) is as much a state of mind as a software suite, and encompasses a range of ultra low-resource utilities and daemons. These include the dwm window manager, the *suckless* terminal and a few more. You can read more about them in **LXF254** and they're actually of limited use for us here because most of them don't cater to Wayland.

These get installed primarily for the sake of the *dmenu* program which (through Xwayland) provides the handy Super+D launcher we met earlier. Have fun tweaking! LXF


You pressed the exit shortcut. Do you really want to exit sway? This will end your Wayland session.
Yes, exit sway
1
2021-09-01 11:3
File Edit View Search Terminal Help
SWAY

```
lxf@mintxxii: $ sudo apt install rx
rxp                 rxvt                    rxvt-ml                   rxvt-uni
code          rxvt-unicode-256color  rxvt-unicode-lite
lxf@mintxxii: $ sudo apt install rxvt-unicode-256color
[sudo] password for lxf:
Reading package lists... Done
Building dependency tree
Reading state information... Done
rxvt-unicode-256color is already the newest version (9.22-6build3).
0 to upgrade, 0 to newly install, 0 to remove and 19 not to upgrade.
lxf@mintxxii: $ nano .config/sway/config
lxf@mintxxii: $
```

```
lxf@mintxxii: $ rxvt
```

You'll need to use shutdown from the login screen with Sway; a nimble Super+Shift+E will get you there.

» KERNEL UPDATES

Being a derivative of the Long Term Support release of Ubuntu, Mint by default uses the same 5.4 series kernel featured there. Don't be put off by the fact that the current branch of the kernel is numbered 5.13, since Canonical backports all manner of features and fixes to the Ubuntu kernel. Also the 5.4 kernel is itself a longterm branch (as you'll see from **kernel.org**). It has always been possible, but not recommended, to use mainline kernels in Ubuntu, but a better way is to activate the hardware enablement HWE stack.

This will give you a newer kernel (which recently bumped from 5.8 to 5.11) which has undergone some Canonical patching and testing, as well as a refreshed graphics stack (new versions of X.org and libdrm and what have you). There is a low-visibility Edge edition of Mint (nothing to do with the web browser) but it's definitely not worth reinstalling just to get a newer kernel. Likewise, you probably shouldn't install the HWE stack unless something is broken, but it's a safer bet than being seduced by a stock kernel with a bigger number.

Desktop Ubuntu now gets the HWE kernels by default, and so it would seem does Pop!_OS, but not Mint. That's okay, because it's easy to install:

```
$ sudo apt install --install-recommends linux-generic-hwe-20.04
```

If you are using the Nvidia proprietary driver this is not enough, since you'll need corresponding proprietary modules too. These you can get with:

```
$ sudo apt install --install-recommends linux-modules-nvidia-NNN-generic-hwe-20.04
```

where `NNN` is the version of Nvidia drivers required by your card. You can find this out by running `ubuntu-drivers list`. Be aware that lots of users have encountered problems with the 5.8 HWE stack (sound, graphics, virtualisation) and the same is likely to be true for this one.

JELLYFIN

Credit: https://jellyfin.org

Get a next-gen media server up and running

Nick Peers reveals how to install and set up a completely open source – but brilliant – alternative to Plex, the streaming media platform.

OUR EXPERT

Nick Peers is shifting his allegiance from Plex to Jellyfin, and not just because it's free!

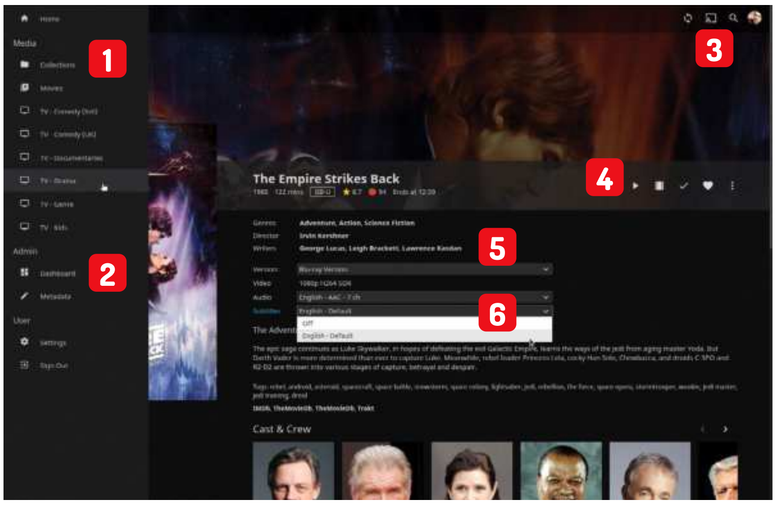
Want the best features of *Plex* and *Emby* without the cost (or closed-source components)? Then take a look at *Jellyfin* (**https://jellyfin.org**), which has come on leaps and bounds from a simple fork of the last open-source *Emby* release (3.5). While *Jellyfin* still has a similar feel to *Emby*, it's carving out its own unique niche in the media server marketplace.

It may be free, but that doesn't mean you'll miss out on features. On the contrary, *Jellyfin* includes features found only behind *Plex* and *Emby's* respective paywalls: hardware GPU transcoding for one, full support for live TV (including DVR facilities) for another.

It's even keeping up with some of the newer features added to its rivals. Early in lockdown, *Plex* and *Emby* introduced features for watching the same server content in real-time with far-flung friends and family. *Jellyfin* quickly followed suit with SyncPlay, available to anyone who's logged into your server.

While it's true that *Jellyfin* isn't quite as slick in terms of clients as its better-known siblings, it still covers most bases (see **https://jellyfin.org/clients** for a full list). What it lacks in polish is more than made up by its customisability, and unlike *Plex* and *Emby*, it doesn't require cloud-based authentication to unlock all its features, making it perfect for those wanting a server exclusively for use over their own network. (You can still open up *Jellyfin* for remote access though – see the box for details, below right.)

NAVIGATING JELLYFIN'S MEDIA LIBRARIES



1 Slide-out menu Click here to quickly jump between libraries – including Jellyfin's Collections, which can be manually curated or created automatically via a plugin.

2 Admin tools The Dashboard is where you go to monitor and administer the Jellyfin server.

3 Sharing options Click here to set up SyncPlay to watch with friends and family, cast to another device or search your library.

4 Playback controls Click here to play the movie, watch the trailer, plus other tasks – for example, adding it to a playlist or editing its metadata.

5 Access multiple versions Rip different versions – say extended versus theatrical release – of the same film and choose which one to watch.

6 Choose audio and subtitle tracks If your movie contains multiple audio tracks – as well as subtitles – you can select which ones you want from these menus.

Getting started

You can install *Jellyfin* natively or through Docker – visit **https://jellyfin.org/downloads** for a complete set of instructions. We recommend a native install if you plan to make use of *Jellyfin's* hardware-transcoding features. Click the Stable link to reveal the full commands, or save time if you're running Ubuntu LTS (16.04 or later) by issuing the following commands:

```
$ sudo apt install apt-transport-https
$ wget -O - https://repo.jellyfin.org/jellyfin_team.gpg.
key | sudo apt-key add -
$ sudo tee /etc/apt/sources.list.d/jellyfin.list
$ sudo apt update | sudo apt install jellyfin
```

Once installed, *Jellyfin* will set itself up a service, and will start running in the background. You can configure it from your web browser by going to **http://127.0.0.1:8096** – if you installed it on a server, you can access it remotely via **http://192.168.x.y:8096** (where **192.168.x.y** is *Jellyfin* server's internal IP address).

The first-run wizard will appear. After choosing a username and strong password you'll be prompted to set up your first media library. The process is the same here as it is when adding further libraries via *Jellyfin's* main Dashboard, so is worth covering.

Set up media libraries

First, choose your content type: Movies, Music, TV Shows, Books, Photos, Music Videos and the catch-all Mixed Content are all available. Name your library, then click + to select a folder containing that library's content. While you can add multiple folders to a single library, it's best to start with just one. After it's been fully scanned, move on to your second folder, and so on.

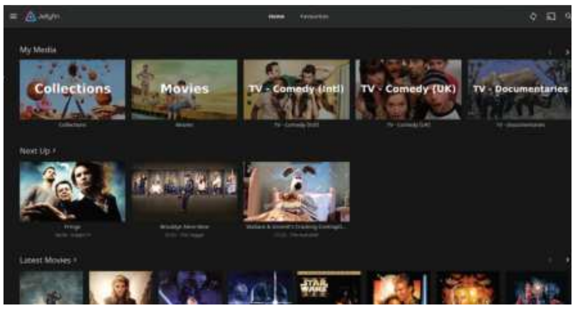
Beneath the folders are your library settings, which vary according to the type of content you've chosen. The key choice here is which 'scrapers' you want *Jellyfin* to use to populate your library with metadata including artwork – **thetvdb.com** is no longer available by default, but you can add it as an option later via the plugins repository (see the final Quick Tip).

A quick note about filenames: *Jellyfin* broadly follows the same naming convention favoured by *Plex* and *Emby*, but there are some quirks with organising movie extras – see the Server>Media section of the online *Jellyfin* documentation (**https://jellyfin.org/docs/general/quick-start.html**) for details.

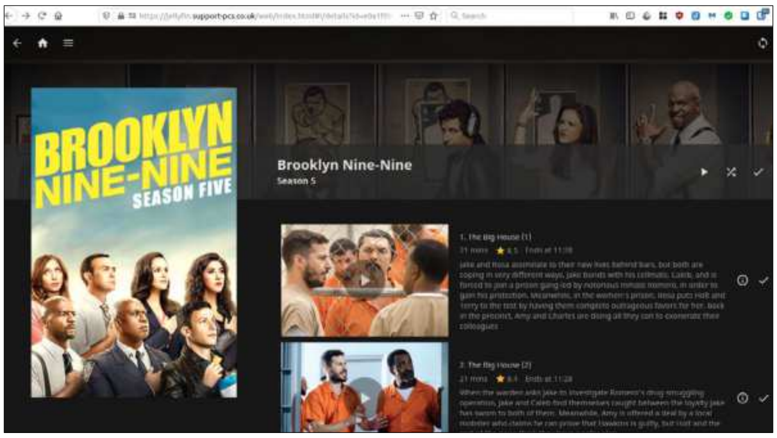
Beneath the scrapers is a section that makes it possible for you to choose how metadata is stored. Beware of ticking 'Nfo' and 'Save artwork into media folders' – this requires write-access to your media folders, which isn't a given. Leaving them unticked stores the information within a central database.

Flick the 'Show advanced settings' switch at the top to reveal even more options. These include pulling in artwork and whether you want to 'download images in advance', which basically means downloading them when media is imported into your library instead of when requested by a *Jellyfin* app.

Once you've configured your library, click OK, then repeat for any other media libraries you'd like to configure now (you can add these later if you're impatient to get going). Click Next when done. Note that, depending on the size of your media folders, setting up each library can take some time. You'll see



Jellyfin's home screen puts all your media within easy reach. Its Next Up feature tracks your progress through favourite TV series.



content start to appear relatively quickly, but it may take several hours before each library is fully populated.

Name your media files correctly and Jellyfin will 'scrape' online databases to provide you with all the necessary metadata.

Complete your setup

You'll need to confirm your default metadata language and country settings, and then click Next. The next step confirms whether you want to open remote access to your server (for streaming content over the internet) – enabled by default – and whether you'd like to attempt UPNP port mapping. If UPNP is disabled on your router, you'll need to manually forward ports to your server's IP address. This form of streaming is insecure – for a better solution, disable the option and follow the guide in the box (below).

Once configured, click Next followed by Finish and *Jellyfin* will start scanning your media folders for content in the background.

Take the tour

You'll find yourself at the main *Jellyfin* screen, which offers you a way to both manage and consume your media from your web browser. The main screen is split into different levels: at the top are shortcuts to your media libraries, followed by 'Continue Watching' (if you're halfway through any movies or shows, they'll appear here, ready to resume where you left off) and 'Next Up', which is *Jellyfin's* equivalent of *Plex's* On Deck view for working your way through series. Beneath this you'll see the latest additions to each library.

If you configure *Jellyfin* with live TV (see the box overleaf), this will also appear as an option on the main

QUICK TIP

Quick Connect speeds up login from selected devices. Log into your account and go to Dashboard> Quick Connect to enable it. Then click your profile photo and choose Quick Connect followed by Activate. Return to your client and tap Use Quick Connect to generate the code you need.

» SET UP SECURE REMOTE ACCESS

Configuring *Jellyfin* to enable friends and family to stream remotely is more complicated than in *Plex* or *Emby*, but the upside is there's no cloud-based authentication involved. Insecure connections are easy to set up, but you should make your connection encrypted and secure for maximum privacy.

We recommend you set up a subdomain on an existing domain you own or sign up for a free dynamic hostname (free ones are available from **www.noip.com**). You'll need to point this to your own public IP address, which is best done using a tool like *ddclient* (**https://ddclient.net/**) or *NoIP DUC* (**www.noip.com/download?page=linux**).

Once these are set up, security is achieved via a reverse proxy working in conjunction with an SSL certificate – you can kill two birds with one stone using *Caddy 2*, following Cognicom's guide (**https://forum.jellyfin.org/t/simpleton-guide-for-remote-access/2707/2**).

After following the guide, open the *Jellyfin* Dashboard and select Networking under Advanced. Tick Enable HTTPS and change the port number to that which you chose when following Cognicom's guide. Click Save. Now try logging on to the server through your web browser using the following syntax, substituting 1337 with your chosen port: **https://subdomain.domain.com:1337**.

If all is well, you should see *Jellyfin* throw up the login screen. Now, not only can friends and family log on to their own user accounts remotely, the SyncPlay feature should work correctly too.

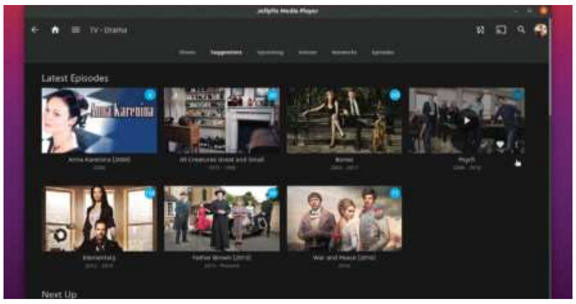
»

QUICK TIP

One area that Jellyfin is still lacking in is, up-to-date documentation. If you run into problems, use its reddit channel (www.reddit.com/r/jellyfin/) to search for answers – or post your own question if needed. Also check out https://forum.jellyfin.org/ – although now closed, you may find additional help here.

screen. All of this is customisable by clicking your profile icon in the top right-hand corner of the screen and choosing Home from the pop-up menu.

Browsing any of these views is self-explanatory, with plenty of on-screen navigation aids. At the top are a range of library views, from 'Suggestions' to 'Genres'. Two views are customisable: Favourites will appear as an option on the home screen for any favourite items you roll over and click the heart icon to highlight. *Jellyfin* also supports Collections, which makes it possible to manually group related items together, whether it's all the *James Bond* or *Star Trek* movies, or films featuring a favourite actor like Tom Hanks.



You can access your Jellyfin server from a range of clients – Jellyfin Media Player works on Linux and is similar to Plex's desktop client.

When browsing a library you'll see three buttons – use these to change the way content is displayed from a choice of six, including banner, list and the default poster view. The A-Z button enables you to sort your library differently – for example, by IMDb rating or release date. The final button is for filtering the view, with options including genres, parental ratings and whether or not the show has been watched or not.

Select a specific movie, television episode or other piece of media to go to its own page. The annotation (see the start of this article) reveals what controls you'll find on these pages.

Fix metadata

It's a good idea to browse your libraries after adding them, because *Jellyfin* is more prone than its rivals to incorrectly identifying shows and movies – particularly those whose name is shared with other releases. When you spot an error, roll your mouse over the incorrectly labelled item, click the vertical ellipsis button that appears and select Identify.

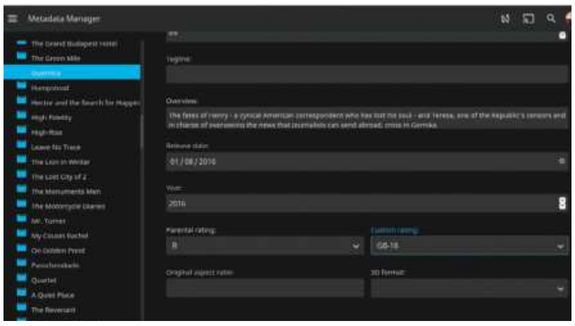
This will bring up a search tool – you can search by title and/or year (which should resolve most issues where an identically titled show or movie from a different year has been selected), but if this doesn't yield a result, go to **www.imdb.com** or **www.thetvdb.com** to manually locate your show and grab its ID. Enter this into the relevant search field and you're guaranteed to get the correct match when you click Search.

Jellyfin takes a little while to update the metadata – so don't be surprised if the correct artwork isn't appear immediately. If you want to perform more drastic surgery to your metadata, use *Jellyfin's* dedicated Metadata Manager instead: access this by clicking the hamburger (≡) button in the top left corner of the screen and then choosing Metadata under Admin.

The Metadata Manager reveals an Explorer-like view of your media by folder. Drill down to a movie title, or parent folder for a TV show, and you'll see its metadata appear in the right-hand pane, ready for editing.

Admin dashboard

The Dashboard is where you can both monitor and administer your media server. From here, you can see what tweaks you can perform to make things run even more smoothly. Your options are Server, Devices, LiveTV and Advanced, with Dashboard at the very top. This provides a handy overview of recent changes, which devices are currently connected and what's streaming and where.



Jellyfin's Metadata Manager is a prime example of how it enables you to fine-tune your media collection to your personal tastes.

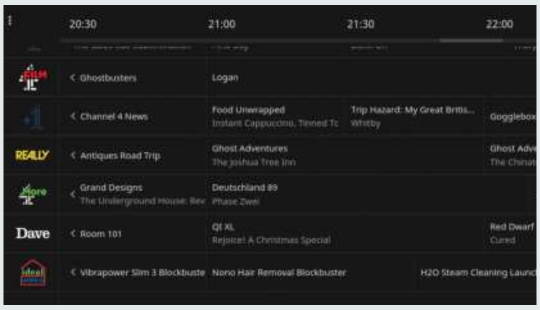
» SET UP LIVE TV

If you have a HDHomeRun network TV tuner device, then adding live TV and DVR capabilities to *Jellyfin* – accessible through any client, and shareable on a per-user basis – is easy. Just click Live TV on the main dashboard followed by + and your HDHomeRun should appear. If you have a USB TV tuner you'd like you use, then you'll need to link *Jellyfin* to a compatible backend like TVHeadend via the appropriate plugin.

You'll then need to assign an appropriate Electronic Programme Guide: Schedules Direct costs US$25/year, or $6 for two months after a six-day free trial, but offers a 21-day guide and is non-profit. XMLTV is free, but fiddly.

Once set up, switch to the DVR section to specify where to save your recordings – separate folders for TV and Movies, and they'll be created as separate libraries in *Jellyfin*, so don't place them inside existing library folders. Streams are recorded uncompressed in MPEG2 format, but if you're running *ffmpeg*, then Bill Thornton's `post-process.sh` script (**https://gist.github.com/thornbill**) will convert it into MP4/H.264 automatically for you. Just invoke it via the 'Recording post processing' option under DVR settings.

Once configured, Live TV will appear on your home screen, with its own shortcut and an 'On Now' section providing convenient shortcuts to shows currently on-air. Click a show and you'll be given the opportunity to record it as well as watch it.



Once set up, you'll have access to a program guide and can watch live TV and schedule recordings to add to your library.

The Server section enables you to change your server name and preferred language (General), plus administer your own and add other users (Users) – see the walkthrough for a more detailed guide on setting up *Jellyfin* for sharing. The options revealed there can also be applied to your own user account, enabling you to personalise *Jellyfin's* look and feel.

Server also contains tools for adding libraries and managing existing ones (Libraries) and configuring playback options (Playback). Playback has three tabs: Transcoding is where you enable hardware-accelerated transcoding using a supported GPU. Transcoding occurs when a client doesn't support your media, and the best approach is to avoid – or at least minimise – it. Find out more in our guide to x264 and x265 video encoding in **LXF272**. If you must transcode, hardware-accelerated transcoding transfers the burden on to the GPU, and newer chips, such as Intel 600-series or later GPUs, produce better-quality results than older ones.

If you do go down the transcoding route, you'll also find advanced conversion settings for transcoding to x264/x265 – familiar to *Handbrake* users – which enable you to fine-tune the quality of your transcodes.

The Streaming tab is where you can set an 'Internet streaming bitrate (Mbps)' limit. You can use **www.speedtest.net** to benchmark your home internet connection, making a note of the upload speed. Unless you have cable or a gigabit broadband connection, you'll need to set a reasonable limit here – 3Mbps or 4Mbps should enable users to watch 720p high-definition content without forcing the server to transcode the stream to reduce its bitrate.

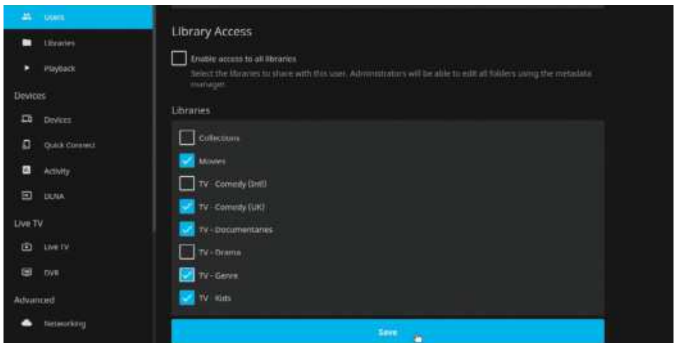
Complete the tour

The other sections of the software are largely self-explanatory – see the box (opposite) for details of configuring live TV. Under Advanced you can access system logs and notifications, tweak the networking setup and install plug-ins. But there's no need to spend lots of time here right now – get your first *Jellyfin* client installed, log into your server and account, and start streaming your media! LXF

QUICK TIP

Select Dashboard> Advanced> Plugins to extend Jellyfin's functionality using plugins. These cover Authentication, Channels, General, LiveTV, Metadata, and Notifications. Click one for a description and options to install it.

CONFIGURE JELLYFIN FOR MULTIPLE USERS



Library Access

[ ] Enable access to all libraries

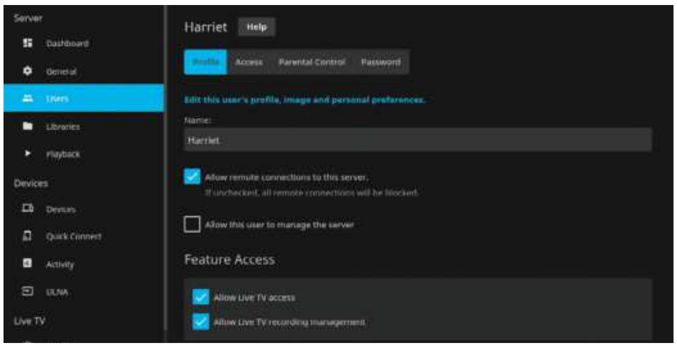
Select the libraries to share with this user. Administrators will be able to edit all folders using the metadata manager.

Libraries

- [ ] Collections
- [✓] Movies
- [ ] TV - Comedy (Intl)
- [✓] TV - Comedy (UK)
- [✓] TV - Documentaries
- [ ] TV - Drama
- [✓] TV - Genre
- [✓] TV - Kids

1 Create new user

Navigate to the *Jellyfin* dashboard and choose Users under Server where you'll see your own account is already in place. Click + to set up your first friend or family member. Give them a suitable username and assign them a password, which you'll need to pass on to them securely. Complete the initial setup by choosing which libraries they're allowed to access. Click Save.



Harriet

Edit this user's profile, image and personal preferences.

Name: Harriet

[✓] Allow remote connections to this server.
If unchecked, all remote connections will be blocked.

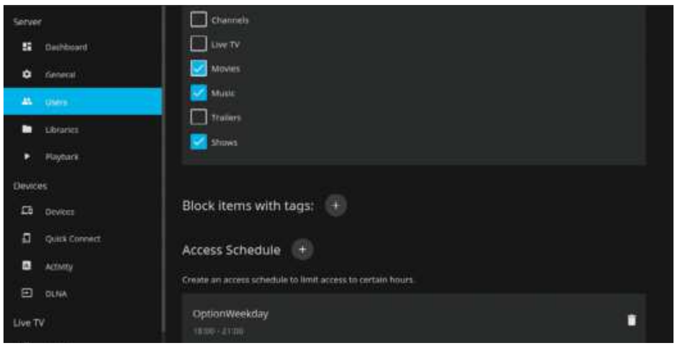
[ ] Allow this user to manage the server

Feature Access

- [✓] Allow Live TV access
- [✓] Allow Live TV recording management

2 Configure settings

Work your way through the four tabs to determine what level of access the user has. The default settings on the Profile tab are a sensible starting point and largely self-explanatory, but we recommend setting limits on the Internet streaming bitrate and maximum number of simultaneous user sessions (five is a sensible limit to cover multiple devices).



- [ ] Channels
- [ ] Live TV
- [✓] Movies
- [✓] Music
- [ ] Trailers
- [✓] Shows

Block items with tags: +

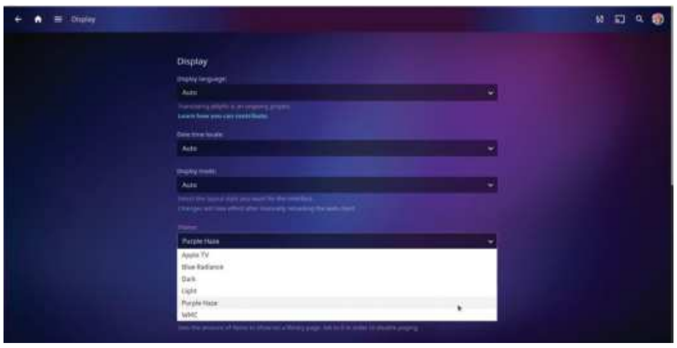
Access Schedule +

Create an access schedule to limit access to certain hours.

OptionWeekday
18:00 - 21:00

3 Access limits and parental controls

The Access tab is where you can redefine which libraries a user has access to, plus restrict their access to specific devices only. If setting up a child account, use Parental Control to set a maximum rating (from U to R18) for content, block unrated content (or content you've add specific tags to), and set an access schedule to restrict overall usage.



Display

Display language: Auto

Date time locale: Auto

Display mode: Auto

Theme: Purple Haze

4 Personalise user

Once the user logs in, they can click their username to customise their own setup. Profile is where they can set a quick-access PIN code for easy access over the local network and add a photo, while Display enables them to change how the *Jellyfin* UI appears to them. They can also customise playback options and subtitles, and set their own home screen.

S-TUI

Stress-test your CPU

While **Shashank Sharma** isn't fond of stress-testing the weighing scale, he's fine with tools like S-TUI, which help him monitor CPU performance.

OUR EXPERT

Shashank Sharma is a trial lawyer in Delhi and an avid Arch Linux user. He's always on the hunt for geeky memorabilia.

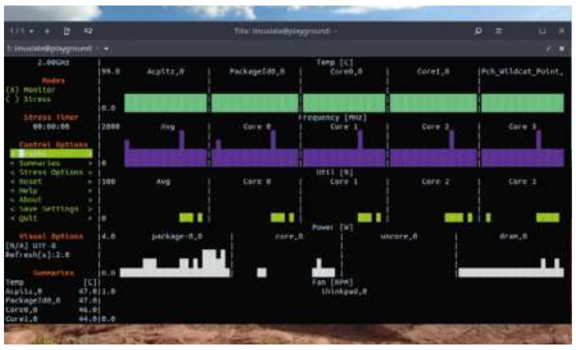
Over the past few years in *Linux Format* magazine, we've introduced a blend of powerful and robust as well as nifty and nimble command utilities that help you perform a variety of tasks. Everything from everyday text editing, connecting with remote machines, performing backups and other administration tasks can be done from the terminal.

A key matrix for determining the vitality of a Linux distro, or the hardware that it runs on, is system performance. Depending on your purpose, you can choose from a variety of dedicated tools to monitor the different components such as CPU frequency, temperature and memory utilisation. But if you favour the CLI, like us, you'll rather enjoy working with *S-TUI*.

With *S-TUI*, which is an acronym for Stress-Terminal UI, you can simultaneously monitor CPU temperature, frequency, power and utilisation. The utility presents all the information graphically and can even be used to export the data into CSV files. Better still, you can configure *S-TUI* to automatically launch scripts when the values of any of the components being monitored breaches the defined threshold values. When coupled with *stress*, another command-line utility, *S-TUI* can also be used to stress-test your system.

Don't stress over installation

Although *S-TUI* isn't available in the software repositories of most popular desktop distributions,



Some sensors, such as Power, are only available when you run S-TUI as sudo, and not as a regular user.

installing the tool is fairly straightforward, and the project's GitHub page describes various installation techniques. If your Linux distribution is already configured to use `pip`, you can install *S-TUI* with the `sudo pip install s-tui` command.

For other installation methodologies, such as installing it from Git, you'll have to ensure all the dependencies are installed. These include tools such as *urwid* and *psutil*. Although it isn't a dependency, if you decide to use *S-TUI* to also stress-test your CPU, you should also install *stress* or *stress-ng*. Thankfully, these are offered in the software repositories of almost all desktop distributions.

» MAKING USE OF THRESHOLD SCRIPTS

In addition to the **s-tui.conf** file, the **~/.config/s-tui** directory also contains a directory called **hooks-d**. You can place custom scripts you wish to execute when a certain threshold, such as CPU temperature or frequency is exceeded, within this directory. The name of the script must be **<name>source.sh**, where **<name>** is the name of the supported source. While you can use *S-TUI* to track various sources, for now it only supports running scripts based on CPU temperature threshold setting.

The default threshold value for the CPU temperature is 80. To change this value, you must invoke *S-TUI* with the `s-tui --t_thresh <value>` command. You can then save the new threshold value into the **~/.config/s-tui/s-tui.conf** file by saving the settings from within *S-TUI* itself.

If you make any changes to the settings from within the *S-TUI* interface, and wish to save them for future use, you'll need to navigate to the Save Settings option on the sidebar and then press Enter. Depending on how you installed *S-TUI*, you may have to amend the file permissions on the **s-tui.conf** file for it to be writeable, otherwise the tool will exit with a curt 'Permission Denied' error.

When you now open the **s-tui.conf** file in a text editor, you'll find a new entry showing the newly defined temperature threshold value:

```
[GraphControll]
refresh = 2.0
utf8 = True
tthresh = 70
```

While the stress testing feature is of little use to home Linux users, there's no denying the benefit of graphical representation of the collected data for administrators and programmers.

Unlike many other command-line utilities, *S-TUI* requires no configuration and you can begin using it immediately after installation. Run the `s-tui` command to launch the utility.

At the top of the sidebar on the left are the details about your CPU. Our Lenovo test machine was correctly identified as running an Intel Core i3-5005U CPU @ 2.00GHz processor. This is because *S-TUI* utilises various other native tools and utilities to gather the relevant information. For instance, the same information and far more details can be ascertained from running the `cat /proc/cpuinfo` command.

When you first launch *S-TUI*, it displays all four parameters – Frequency, Utilization, Temperature and Power – and refreshes the data for each every second. You can change the refresh rate by changing the value of Refresh[s]:1.0 on the sidebar.

Depending on your terminal application, and the colour scheme, you may not be able to clearly see the different elements on the *S-TUI* interface. Should this happen, exit *S-TUI*, then switch to a basic White-on-Black colour scheme on your terminal emulator, and run *S-TUI* again. You should now be able to see all the different colours used by *S-TUI* to display all the collected information, and you can edit your terminal profile accordingly.

You can use the up and down arrow keys to navigate the sidebar, but *S-TUI* also supports the use of H and J to scroll, much like you would in *Vim*.

Add some stress

By default, *S-TUI* is configured to only monitor your system. This is evident from the (X) next to Monitor on the sidebar. If you installed the *stress* utility as well, you can enable the stress feature by navigating to the Stress entry on the sidebar and pressing the spacebar. You'll notice the empty brackets would be replaced with (X) now. You can similarly enable or disable monitoring of the different components. When you disable a component, the corresponding graph will automatically disappear from the interface.

Unlike most other command-line utilities, *S-TUI* can also be controlled with the mouse. You can left-click on an entry/option in the sidebar to select it. You must still hit Escape to return to the main screen and use the arrow keys to scroll through the sidebar as the scroll-wheel on your mouse doesn't work with *S-TUI*. To disable the mouse, you must invoke *S-TUI* with the `s-tui -nm` command.

You can also switch to a different temperature sensor from within *S-TUI* itself, if you believe the graph is inaccurate. Navigate to the 'Temp Sensors >' entry on the sidebar, and press Enter. This opens the Available Temperature Sensors dialog, and you can then select one from the list. As before, after navigating to an entry in the list, press Spacebar to select it. Remember to select Apply for the changes to take effect.

You must hit Escape to return back to the main screen of the application, such as from the About> Help or Temp Sensors section.

While the graphical interface for *S-TUI* also has a Help section in the sidebar, it provides little more than a quick introduction. For a complete list of all the supported command options, you must run the `s-tui –help` command.

The collected data is lost as soon as you exit *S-TUI*, because the tool doesn't save it by default. If you want all the collected information to be automatically saved to a CSV file when you exit *S-TUI*, you must invoke the utility with the `s-tui -c` command. The collected data will be stored in the users' home directory. You can then view the **s-tui_log_<TIME>.csv** file in your preferred text editor. You can also provide a custom name for the CSV log file with the `s-tui -c <path_to_CSV_file>` command.

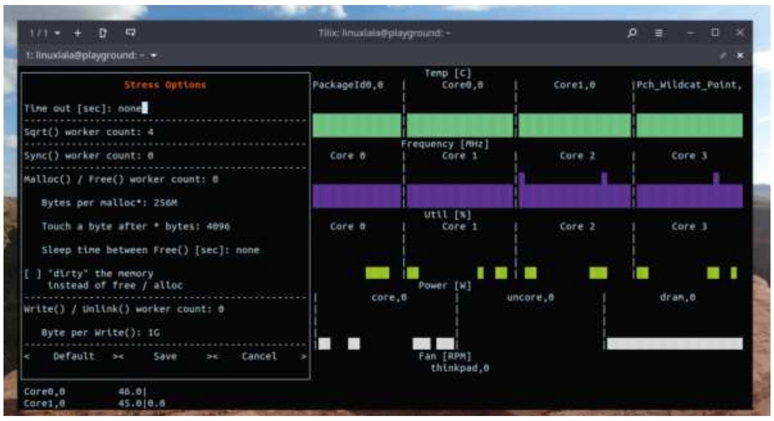
Another useful command option is `-j`, which can be used to print on the screen the current status of the machine in JSON format. The output generated by the `s-tui -j` command is a record of the current status of the machine, and not a running log:

```
$ s-tui -j
{
"Frequency": {
    "Avg": "1282.1",
    "Core 0": "1630.6",
    "Core 1": "1295.6",
    "Core 2": "1175.6",
    "Core 3": "1026.4"
  },
  "Temp": {
    "Acpitz,0": "48.0",
    "PackageId0,0": "49.0",
    "Core0,0": "47.0",
    "Core1,0": "47.0",
    "Pch_Wildcat_Point,0": "40.5"
  },
}
```

S-TUI supports a number of different temperature sensors and you can choose a different one by navigating to the Temp Sensors screen. This is only needed if the default configuration fails to accurately graph the CPU temperature on your machine.

All the basic settings, such as the refresh rate (in seconds) and which components to graph, by default are stored in **~/.config/s-tui/s-tui.conf**. Unfortunately, you must set the Stress options and the temperature sensors every time you run *S-TUI* as the tool doesn't permit saving these settings in the configuration file. LXF

QUICK TIP
The stress utility performs a loop that calculates the square root of a random number in order to stress the CPU. You can customise the stress options by navigating to the Stress Options screen from the S-TUI sidebar. The graphical interface enables you to define all the major settings such as the worker count for the sqrt(), sync(), malloc() functions, as well as choose whether to dirty the memory.


```
Stress Options

Time out [sec]: none
------------------------------------
Sqrt() worker count: 4
------------------------------------
Sync() worker count: 0
------------------------------------
Malloc() / Free() worker count: 0

   Bytes per malloc*: 256M

   Touch a byte after * bytes: 4096

   Sleep time between Free() [sec]: none

[ ] "dirty" the memory
    instead of free / alloc
------------------------------------
Write() / Unlink() worker count: 0

   Byte per Write(): 1G
------------------------------------
<   Default   ><   Save   ><   Cancel   >

Core0,0          46.0|
Core1,0          45.0|0.0
```

You must refer to the man page and documentation on the stress utility to understand the different supported functions.

Credit: https://cozy.geigi.de

COZY

Build the ultimate audiobook collection

Nick Peers reveals how to put together your own audiobook library using CDs, free downloads and Audible purchases.

OUR EXPERT

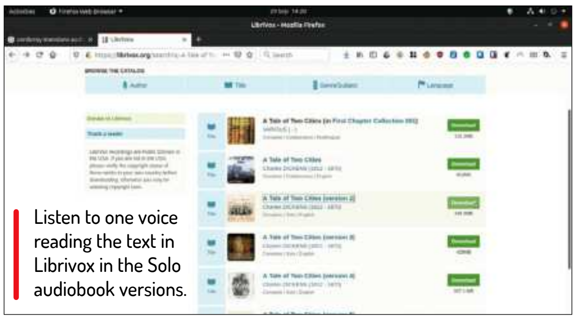
Nick Peers has been busy building a huge audiobook library from his massive CD collection, all powered by the lovely Booksonic.

We all love a good story – after all, telling stories around the fire is as old as human language itself. Audiobooks – and radio dramatisations – continue that tradition into the modern age, and thanks to your phone and PC, give you the opportunity to listen to your favourite books wherever you are.

In this guide, we'll show you how to bring all of your audiobooks – whether on CD or in the cloud – into one central library. You'll add tags to make them easy to identify, plus discover how to listen to them from your PC or phone. Let's get storytelling…

Set up your Audiobook library

There are three principal places to source audiobooks: on CD, with unabridged editions spanning dozens of discs; from an online vendor (the obvious example being Audible); and finally, from free websites where classic books – namely, those now out of copyright – have been narrated by volunteers can be found.


Listen to one voice reading the text in Librivox in the Solo audiobook versions.

One of the best sources of free audiobooks is *LibriVox* (**www.librivox.org**). When you find a title you like, choose the 'Whole book (zip file)' option to download the book as a series of MP3 files, one per chapter. Each file is tagged with author, title and other key details, and you can also download cover art from here to illustrate the book in your library – more on tagging and artwork later.

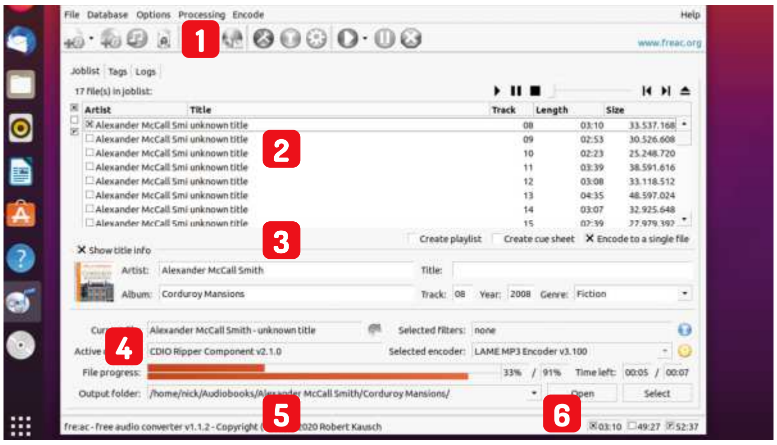
Where will you store your audiobooks? The simplest option is to create a dedicated folder – Audiobooks – which can reside inside your Home folder. Books can then be stored within the following folder structure: **Audiobooks\Author\Title** – for example, **Audiobooks\Herman Melville\Moby Dick**. If your collection threatens to get out of hand, or you'd like to subdivide your library (say separating children's titles from your wife's collection of thrillers), add another layer – Genre – between Audiobooks and Author.

Audiobook file breakdown

Another important consideration is how you intend to store individual audiobooks. Audible supplies entire books as a single file, and whatever chapter markers are embedded in its native file format (AAX) will be lost on conversion. Conversely, audiobook CDs split each CD into short tracks – three- to five-minutes long – to aid navigation through CD players.

Neither option is particularly desirable. Your audiobook player should be able to pick up from where you left off, but it's still handy to be able to jump to a specific chapter or section of a book quickly. The

Rip audiobooks with fre:ac



```
17 file(s) in joblist:
Artist                          Title        Track  Length  Size
Alexander McCall Smi unknown title           08     03:10   33.537.168
Alexander McCall Smi unknown title           09     02:53   30.526.608
Alexander McCall Smi unknown title           10     02:23   25.248.720
Alexander McCall Smi unknown title           11     03:39   38.591.616
Alexander McCall Smi unknown title           12     03:08   33.118.512
Alexander McCall Smi unknown title           13     04:35   48.597.024
Alexander McCall Smi unknown title           14     03:07   32.925.648
Alexander McCall Smi unknown title           15     07:39   77.979.392
Create playlist   Create cue sheet   X Encode to a single file
X Show title info
Artist: Alexander McCall Smith          Title:
Album: Corduroy Mansions                Track: 08  Year: 2008  Genre: Fiction
Cur[...]: Alexander McCall Smith - unknown title     Selected filters: none
Active [...]: CDIO Ripper Component v2.1.0           Selected encoder: LAME MP3 Encoder v3.100
File progress:   33% / 91%   Time left: 00:05 / 00:07
Output folder: /home/nick/Audiobooks/Alexander McCall Smith/Corduroy Mansions/   Open   Select
fre:ac - free audio converter v1.1.2 - Copyright (...)2020 Robert Kausch   03:10   49:27   52:37
```

1 fre:ac controls
The buttons provide shortcuts to frequently accessed tools and settings, including the all-important rip button.

2 Joblist
All tracks requiring processing are listed here. Use the controls on the left to quickly select all, deselect all or invert selection.

3 Metadata
Although fre:ac enables you to edit the currently selected track's metadata, it's quicker and easier to use Kid3.

4 Monitor progress
Progress bars show you how far the ripping process has got – both for the current track and the overall selection.

5 Output folder
You'll need to set this manually for each audiobook CD you rip owing to issues with fre:ac's automated tools.

6 Monitor track length
When dividing audiobooks into roughly equal 'parts', the figure next to 'X' helpfully reveals the combined length of your selection.

approach adopted by *LibriVox* books is best – one file per chapter, and this is something that *OpenAudible* can do with your Audible library by splitting books into separate files – one for each chapter – during the AAX-to-MP3 conversion process.

It's more complicated with CDs where you need to combine multiple tracks into a single file, and choose where to break them: by CD, so each file spans 60-70 minutes; chapter – can be fiddly to locate where each chapter begins, and not always an option with abridged audiobooks; or arbitrary break – say splitting each CD into two 35-minute or three 25-minute 'parts'. Thankfully, our ripping tool of choice – *fre:ac* – can handle any of these options with aplomb.

Convert your Audible library

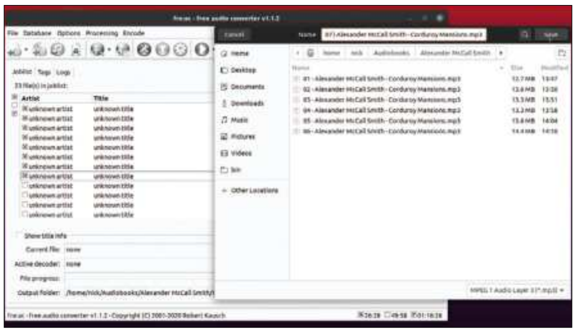
Let's start with revealing what do with your Audible collection. You're probably already listening to them using an app – either on your phone or via the open-source *Audible for Linux* tool (**https://github.com/cross-platform/audible-for-linux**). This ties you into Audible's DRM system, meaning you won't be able to combine your Audible titles with those sourced elsewhere.

Until recently, there was a free solution to this in the form of *OpenAudible* – it's now shareware, but if you're desperate to produce non-DRM versions of your Audible books, then the £12 registration fee isn't too prohibitive. By way of compensation, it supports up to three separate Audible accounts and comes with a year's worth of updates.

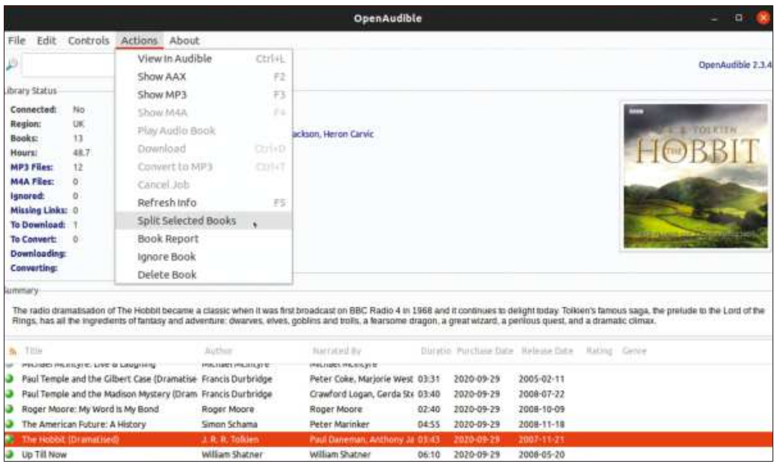
To get started, head over to **https://openaudible.org** to download a DEB package or AppImage – the latter is the simplest to use: simply place it in your Home folder and don't forget to right-click it and choose Properties>Permissions to tick the all-important 'Allow executing file as program' box before launching it.

Once up and running, press Ctrl+K to open a browser window that will redirect to **Audible.co.uk** – look for the Sign In button in the top-right corner to log into your Audible account. Once logged in, close the browser window to return to the main OpenAudible interface. Next, choose Controls>Quick Library Sync and after a short pause you should see your Audible purchases appear.

Next, select a title – you'll see a summary and artwork appear. Its State should read 'Ready to download'. Choose Actions>Download and wait for the audiobook to download in its native AAX format. All of this can be done for free, but this is the point where you'll need that licence. If About>Purchase doesn't work, head to **https://openaudible.org/purchase** to buy


fre:ac - free audio converter v1.1.2
File Database Options Processing Encode
Joblist Tags Log
Artist
Title
unknown artist
unknown title
Show title info
Current file: none
Active decoder: none
File progress:
Output folder: /home/nick/Audiobooks/Alexander McCall Smith/
Cancel
Name
Save
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
bin
Other Locations
01-Alexander McCall Smith - Corduroy Mansions.mp3
02-Alexander McCall Smith - Corduroy Mansions.mp3
03-Alexander McCall Smith - Corduroy Mansions.mp3
04-Alexander McCall Smith - Corduroy Mansions.mp3
05-Alexander McCall Smith - Corduroy Mansions.mp3
06-Alexander McCall Smith - Corduroy Mansions.mp3
MPEG 1 Audio Layer 3 (*.mp3)

Ignore tagging your files when ripping in fre:ac – your focus should be on naming them in the correct sequential order.



Connected:	No
Region:	UK
Books:	13
Hours:	48.7
MP3 Files:	12
M4A Files:	0
Ignored:	0
Missing Links:	0
To Download:	1
To Convert:	0
Downloading:	
Converting:	

The radio dramatisation of The Hobbit became a classic when it was first broadcast on BBC Radio 4 in 1968 and it continues to delight today. Tolkien's famous saga, the prelude to the Lord of the Rings, has all the ingredients of fantasy and adventure: dwarves, elves, goblins and trolls, a fearsome dragon, a great wizard, a perilous quest, and a dramatic climax.

Title	Author	Narrated By	Duratio	Purchase Date	Release Date	Rating	Genre
Paul Temple and the Gilbert Case (Dramatise	Francis Durbridge	Peter Coke, Marjorie West	03:31	2020-09-29	2005-02-11		
Paul Temple and the Madison Mystery (Dram	Francis Durbridge	Crawford Logan, Gerda Sti	03:40	2020-09-29	2008-07-22		
Roger Moore: My Word Is My Bond	Roger Moore	Roger Moore	02:40	2020-09-29	2008-10-09		
The American Future: A History	Simon Schama	Peter Marinker	04:55	2020-09-29	2008-11-18		
The Hobbit (Dramatised)	J. R. R. Tolkien	Paul Daneman, Anthony Ja	03:43	2020-09-29	2007-11-21		
Up Till Now	William Shatner	William Shatner	06:10	2020-09-29	2008-05-20		

OpenAudible converts your Audible purchases into DRM-free audiobooks for your own consumption on your choice of platform.

through FastSpring. Once you've received your licence code, choose 'About > Activate' to enter it, then press Ctrl + T to convert the file to MP3 format.

One final step: the file is converted into a single file by default, so once done you'll need to choose 'Actions > Split Selected Books' and wait while a progress bar reveals how many chapters or files will be created, plus keep you updated with how it's getting on.

Once all this is complete, a folder window will open to reveal your converted audiobook inside a sub-folder with the title as its name. You can now copy or move this into your Audiobooks library (you'll need to create an Author folder if you've not done so already to place the folder in).

For those with large Audible libraries, there's some good news: once you've purchased a licence, any book you subsequently download will be automatically converted to MP3 (see Edit>Preferences where you can also adjust the encoding quality of the resulting MP3 file if you wish to change the balance of file size versus audio quality). Select multiple books at once and choose Actions>Download to do so – you'll be

QUICK TIP

When it comes to assigning genres to your audiobooks, avoid the generic 'Audiobook' to come up with specific book genres like Mystery or Sci-Fi. Search Goodreads.com to see how other readers have categorised a book, if necessary.

» TRACK DOWN COVER ART

To hunt down artwork, head to **https://images.google.co.uk** and search for the title of your book plus the word 'audiobook'. You'll almost certainly get a match, but wait – is it suitably hi-res? Our preferred size is 600x600 pixels, although 500x500 is the size of artwork embedded into your Audible collection by *OpenAudible*.

To locate higher res artwork more easily, click Tools under the Google Search button and change Any size to Large, which will filter out smaller images. Hopefully at least one will remain: click it to view a preview. Roll your mouse over this to reveal its size – if large enough, right-click it and choose 'Save image as' to save a copy to your hard drive. Use your image editor to reduce both its physical size to 600x600 if necessary, plus its file size by saving it as a JPEG.

If you're struggling to find a suitably sized image via a keyword search, save the largest version of the artwork you can find, then click the camera icon inside the Google Image Search box and choose 'Upload an image' to search the internet using the actual image itself. If you're lucky, you'll see an 'All sizes' button appear next to the image thumbnail: click this and hopefully it'll reveal a higher resolution image that you can use.

»

QUICK TIP

Fancy turning your audiobook collection into a streaming server accessible from anywhere over the internet? Dig out your copy of **LXF259** and turn to page 64 for our two-page guide to building and maintaining the perfect audiobook server using Booksonic.

prompted about automatic conversion for the first book, but thereafter it'll all proceed in the background. And once that process completes, choose Actions>Split Selected Books to complete the process.

Rip audiobook CDs

When it comes to ripping audiobook CDs, you'll need a ripping tool that can combine multiple tracks into a single file – we recommend *fre:ac*, which you can install through the *Software Centre*. To enable this feature in the program, choose Options>General settings. Verify the *LAME* MP3 encoder has been selected, then check 'Encode to a single file'. Click Select under Output folder to select your Audiobooks folder and untick 'Use input file folder if possible'. Ignore the 'Filename pattern' field for automatically naming your output files – it doesn't work when ripping multiple tracks to a single file.

Spoken audio recordings take up less space than regular music due to their relative simplicity, but you can squeeze them even smaller by switching from stereo to mono recordings: click Configure encoder next to the *LAME* MP3 encoder to do so. Change the Use preset to Custom settings, then switch to the Misc tab and choose Mono under Stereo mode before clicking OK.

Next, insert the first disc of your first audiobook and click the 'Add audio CD contents to the joblist' button. You'll see all the CD's tracks appear, accompanied by an error message stating that the **freedb.org** database has no record of the CD. This will be true for most – if not all – audiobook CDs, so choose Database>Automatic CDDB queries to remove the automatic check.

At this point, you could tag your files using *fre:ac's* Tags>Albums tab section where you can manually input the author (Artist) and book title (Album), plus add artwork, a year and genre. While this works perfectly, it's very fiddly – you need to add this information manually each time you insert a new CD, and it's much quicker to handle the process later on using *Kid3*, so for now leave everything set as 'untitled'.

Select tracks to rip

Now, you need to select all the tracks you plan to rip as your first file. If you're ripping an entire disc, just make sure everything's selected; otherwise you need to locate the first and last tracks that make up your selection – either aim for 25 (or 35) minutes for an arbitrary break, or use the playback controls above the track list to work out where each chapter begins and ends.

Finally, click Select next to Output folder to open your Audiobooks folder, then manually create a new author folder (or select an existing one if applicable) followed by a folder named after the book title. With this selected, you can now click the green 'Start the encoding process' button from the main toolbar.

You'll be prompted to save the file – automatically named Artist – Album (or Author – Book Title).mp3. Be sure to insert a track number at the beginning corresponding to the chapter, part, or disc number – so 01-Author-Book Title.mp3, for example. You might even prefer to name the file 01-Chapter One.mp3 or whatever naming convention you like – just make sure the track number is at the beginning.

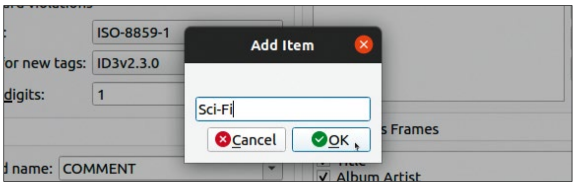
Once you click Save, the rip will begin. A progress indicator will reveal how your rip is progressing and completed tracks will disappear from the queue. When finished, a single MP3 file will have been created. Now repeat the process for the next chapter or part, and when you've finished with a disc, move on to the next.

Tag your media

Once you've ripped your audiobook CD, it's time to tag it using a tagging tool. We've opted for *Kid3*, which is regularly updated and has evolved into just the tool for audiobook tagging. It's best installed through the Terminal to ensure you get access to the latest version:

```
$ sudo add-apt-repository ppa:ufleisch/kid3
$ sudo apt-get update && sudo apt install kid3-qt
```

Once installed, launch *Kid3* and choose Settings>Configure Kid3. Up the 'Track number digits' to 2 (so 01, 02 rather than 1, 2, etc), and tick Album Artist and Composer under Quick Access Frames before dragging and dropping them in alphabetical order. Like Artist, Album Artist will record the author's name, while


ISO-8859-1
or new tags: ID3v2.3.0
digits: 1
Add Item
Sci-Fi
d name: COMMENT

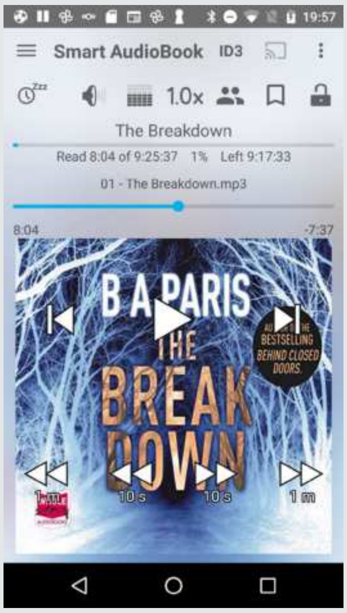
You can create your own custom genres in Kid3, enabling you to add another layer of organisation to your audiobooks.

» ANDROID AUDIOBOOK HEAVEN

Want to listen to audiobooks on your Android phone? If you've ever used *Booksonic* (covered in **LXF259**), then the obvious solution is to install the official *Booksonic* app (£2.49), which not only streams books from your server to save space on your phone, but enables you to mark individual titles for listening to offline. This means that they're temporarily downloaded to your phone's storage, which makes it possible for you to listen to them anywhere, then easily removed when you're done.

If all you want is to listen to audiobooks you've transferred to your phone's storage – after all, you can fit a heck of a lot of books on to a 32GB microSD card – then try *Smart AudioBook Player* instead. When you first launch it, simply point the app towards your *Audiobooks* folder, then refresh the library to view a complete list of books, sorted by author and then title. Tap one to access its own playback screen where you'll find all the controls you need – and then some, including Characters (to help you remember who's who) and Bookmarks (for jumping to favourite passages quickly).

After 30 days, the app reverts to a basic cut-down version – it has all the core functionality you need, but the full version costs just £1.89. Select Help> Features to compare versions followed by Help>Version>Buy to purchase.



Smart AudioBook Player has all the playback controls you could ever need (or want, frankly).

the Composer field is for recording the audiobook's narrator. You'd expect things to be the other way round, but go figure…

Use the Custom Genres section to input custom audiobook genres – click Add for each one, from Horror and Crime to Sci-Fi and generic Fiction. When you're done, tick 'Show only custom genres' if you want to remove all other auto-suggestions as you fill the Genre tag. Once you've tweaked *Kid3's* settings to your satisfaction, click OK. You're now ready to tag your files – the step-by-step guide (*below*) reveals how to apply consistent tags across all your audiobook's files, and add album art.

How to enjoy your audiobooks

Once your audiobooks have been ripped, organised and tagged, it's time to enjoy the fruits of your labours. While you can listen to an audiobook through any music player in Linux, it'll be a messy process trying to combine music and audiobooks in the same program, never mind library. Instead, use a dedicated audiobooks player called *Cozy*, installed via Flatpak:

```
$ sudo apt install flatpak
$ flatpak install flathub com.github.geigi.cozy
$ flatpak run com.github.geigi.cozy
```

The program will launch – going forward, it can be opened through the Launcher. Step one is to import your audiobooks – simply select your Audiobooks folder to proceed and *Cozy* will import them into its library. Once complete, choose your view: Author displays titles by Composer field (meaning the narrator), whole Reader displays them by Album Artist (so the author).

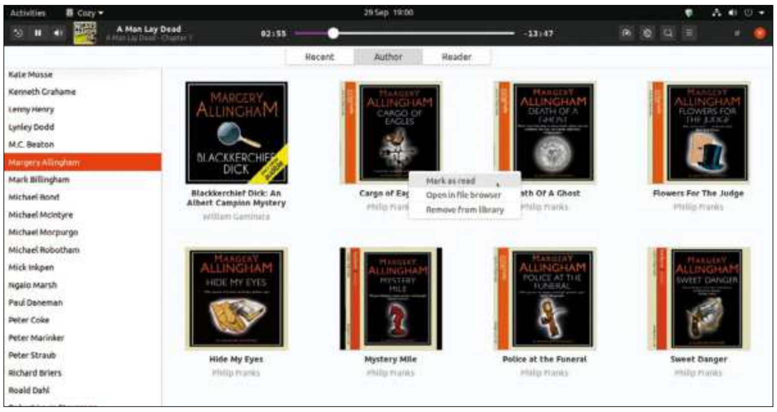
Hang on – haven't we just organised everything the other way around? Yes, because other audiobook programs and services organise them that way, as do regular music players. Don't worry, though, because *Cozy* can be configured to swap them round to avoid confusion: click its menu button and choose Preferences>Behaviour tab, then flick the 'Swap author and reader' switch to instantly correct the view.

Now return to your chosen view, where you'll be greeted by a two-pane window. On the left is a list of authors or narrators (depending on the view that you're using), while the right-hand pane displays a list of titles, again in alphabetical order, complete with artwork. Use the left-hand pane to filter the view by author/narrator if necessary, then roll your mouse over its image cover to reveal a play button to click; alternatively, click elsewhere on the artwork to view the title up close.

Cozy's user interface is nice and straightforward to navigate – you'll see a series of buttons across the top, including a handy 30-second rewind button next to the play/pause button, plus a progress bar to navigate quickly through the current track. You'll also find playback speed, sleep timer, and search buttons to the right.

Click the menu button for more options – you can manually refresh the library after ripping new audiobooks, tick the box to hide unavailable books (such as those stored on currently disconnected external or network storage), plus revisit the Preferences section, where you can add additional folders to *Cozy's* library if you wish via the Storage tab.

And what about consuming your audiobook collection on other devices? The box (opposite) reveals all you need to know about listening to your audiobook collection on your Android phone. LXF

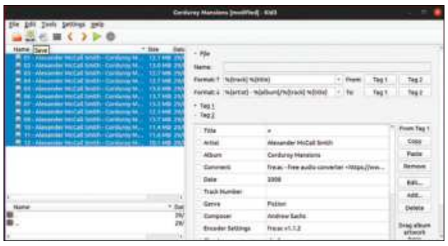


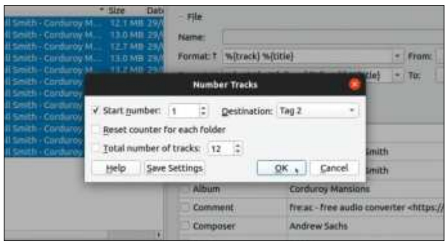
Dedicated audiobook player Cozy works beautifully with your audiobook library – no need to play them in a music utility.

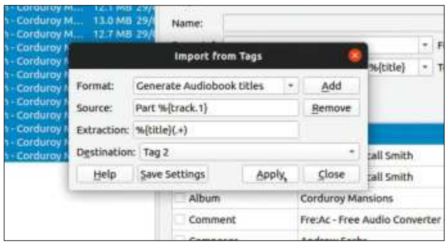
QUICK TIP

We've chosen MP3 as the file format for our audiobook library because spoken word recordings don't require audiophile levels of quality. When you factor in the length of most audiobooks – many hours per title – the smaller your files the better.

TAG YOUR MEDIA WITH KID3







1 Set common tags

Choose File>Open to select all your audiobook's files and click Open – they should appear highlighted in blue. Tick all the tags that will apply to all tracks: Album Artist, Artist, Album, Date, Genre and Composer, and fill in each field. Add your artwork by dragging the file to where it says 'Drag album artwork here'. Click Save to save your changes.

2 Add track number

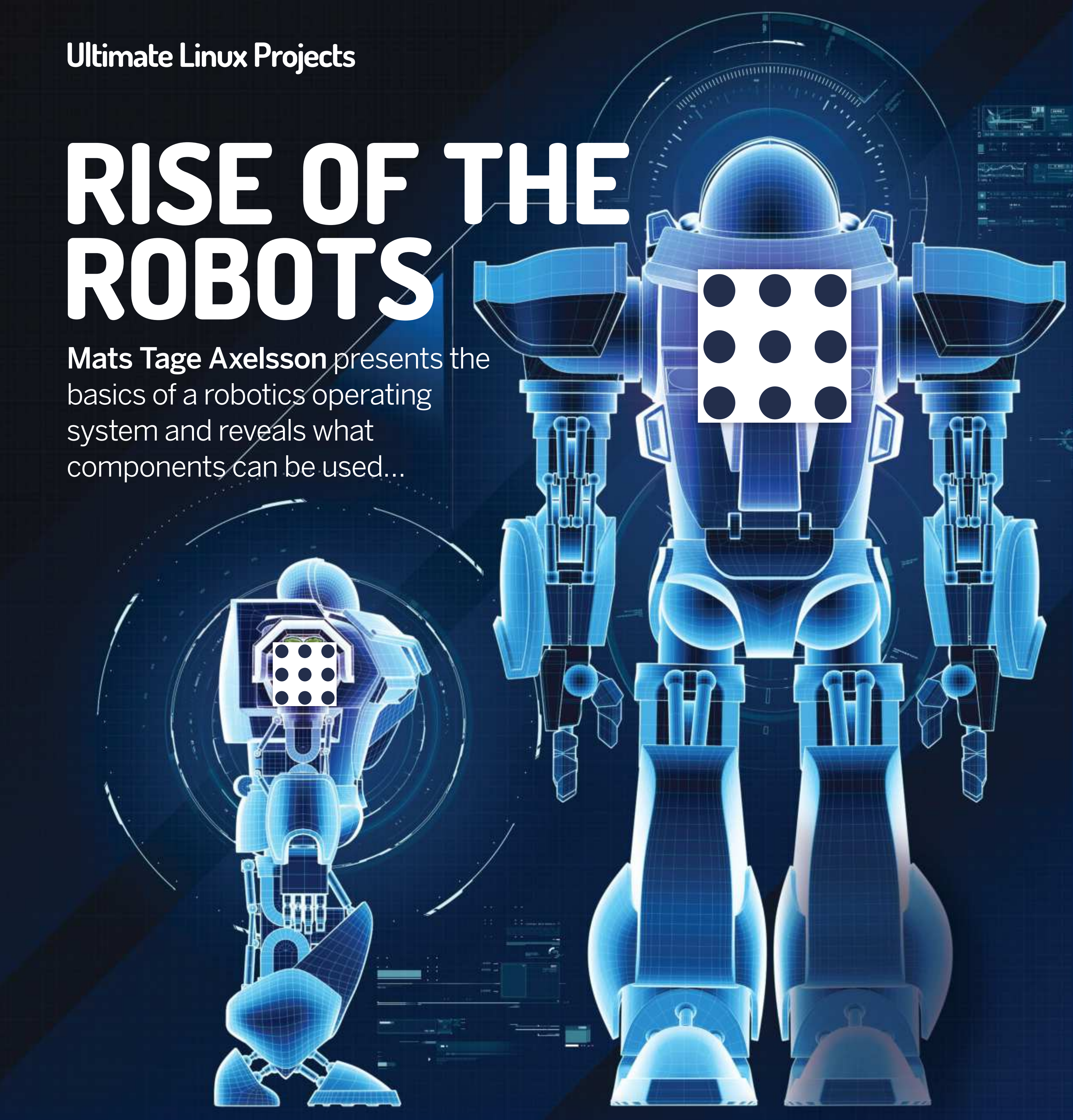
Next, you need to tag each file with its specific track number. To do this, verify all files are still highlighted, then choose Tools>Number Tracks. Make sure the Destination is set to Tag 2, click Save Settings to make this default going forward, and click OK. Select each file in turn to verify the correct track number has been assigned. Click Save again.

3 Add title information

Again, with all the tracks highlighted, choose File>Import from Tags. Click Add, name it Generate Audiobook Titles, and type the following string into the Source box: `Part %{track.1}` – replace Part with Chapter or Disc as required. Type `%{title}(.+)` into the Extraction box, and select Tag 2 under Destination. Click Save Settings> Apply, then click Save for the last time.

RISE OF THE ROBOTS

Mats Tage Axelsson presents the basics of a robotics operating system and reveals what components can be used…



Assembling your own robot and getting it to walk around will entertain kids of all ages for a good few minutes. However, for robotics to be long-term fun it needs to be a challenge.

A common misconception about robotics is that you need a degree to make anything of your own. Having said that, a robot does require many subsystems to operate. There are legs to move, arms to stretch, things to see and react to. Basically, everything we do as humans – except you must design the entire system that can handle all those tasks.

This isn't something you whip up in your coding cave on a rainy afternoon. Being free and open source enthusiasts, we know better than to believe we have to cover every detail ourselves. There have been many projects to make robotics an area you can thrive in and help expand, the most successful organisation to date is the Open Source Robotics Foundation (**www.openrobotics.org**). Out of this has developed the Robotics Operating System (ROS). It's not an operating system as such, but rather a collection of tools that help you create your version of what a robot should be and do.

The ROS is better described as a robotics framework, but what do you want from the ROS and what do you need to do to get a robot up and running? We've already mentioned that many subsystems control different parts of the robot. Your robot will need to navigate, move and interact with its environment. That may include interacting with you and other humans, but that's for another time. The simplest robot will run around the house exploring – a more advanced version will go to the kitchen and bring you a new beverage. For all this you need sensors, micro controllers and a lot of mechanics…

It's important to bear in mind that all these parts must communicate with each other. This is where things can become complex and you need an overall control system. Using ROS, you can declare and define all the nodes (the distinct parts of the system) and how they communicate. Other things included in the ROS framework are simulators and visualising software to help you spot problems and solve them.

Knowing what the different pieces of software do requires a fundamental understanding of the architecture. Yet defining a robot as the sum of its components will quickly become complex, so there needs to be a standard method of communication – either within the robot or from outside. This is, in essence, what ROS is for: it establishes the standard communications protocols. Within this standard are certain components that define the overall system: nodes, services and topics.

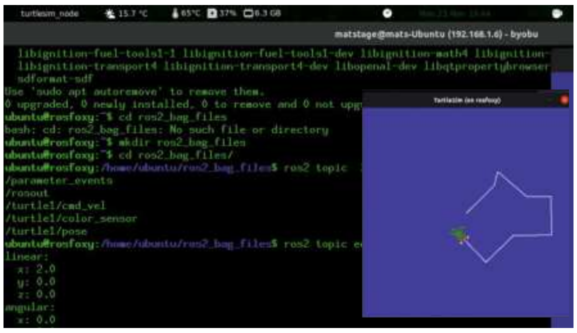
A node is one collection of functions that communicate with other nodes using the ROS client library. This makes it possible to choose between many different solutions for each task, so long as they use ROS to communicate with the other nodes. Nodes will use the other parts of the architecture – topics and services – in different ways, depending on what the information is used for.

Topics are places that contain data, which many nodes will need to access. One node will send data to the topic and other nodes can subscribe to the topic. In short, this is a broadcast-type message – information that many nodes may need and one can create. In contrast, services are created so that one node can request information from another node. The first node is the client that requests information from the server.

Component core

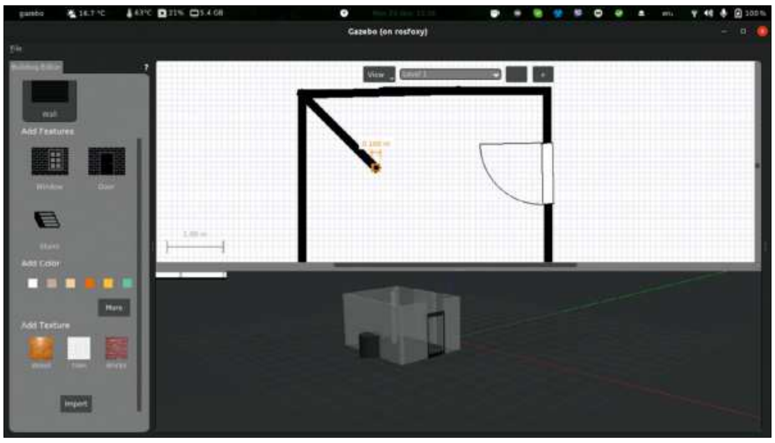
To run ROS, you need a number of packages. In the core version, you're provided with everything you need to run the actual robot. These packages implement the ROS interfaces, the middleware interface and some specifications for the messages that you need to use for development.

The two that you'll come in contact with the most are the rclcpp and rclpy libraries. They both implement the client libraries; as you may guess, one is for C++ and the other for Python programmers. The client libraries cover the messaging we mentioned earlier. This is where you decide whether your client needs to broadcast to topics or set up a service for other nodes. This is also where you find the message formats so you can add your information for your applications. The documentation



```
turtlesim_node   15.7 °C   65°C  37%  6.3 GB
matstage@mats-Ubuntu (192.168.1.6) - byobu
  libignition-fuel-tools1-1 libignition-fuel-tools1-dev libignition-math4 libignition-
  libignition-transport4 libignition-transport4-dev libopenal-dev libqtpropertybrowser
  sdformat-sdf
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upg
ubuntu@rosfoxy:~$ cd ros2_bag_files
bash: cd: ros2_bag_files: No such file or directory
ubuntu@rosfoxy:~$ mkdir ros2_bag_files
ubuntu@rosfoxy:~$ cd ros2_bag_files/
ubuntu@rosfoxy:/home/ubuntu/ros2_bag_files$ ros2 topic
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
ubuntu@rosfoxy:/home/ubuntu/ros2_bag_files$ ros2 topic e
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
```

Controlling a turtle across the screen is ideal for learning the fundamentals of robot design.



You can create an entire environment in Gazebo. This includes walls, stairs and windows.

shows many useful examples of a service, client, publisher and others. Everything else uses the client libraries, so make sure you understand them.

While you run ROS on your machine, you can find all the different message formats through using the various commands. The other parts of the ROS system are specific to different functions. The *turtlesim* executable demo displays some of the available functions. It shows how to move the robot around.

As well as snaps, you can also use virtual machines for your ROS installations. The snap route to a ROS installation is an easy way to see the first examples before you start creating your own code:

```
$ sudo snap install ros-beginner-tutorials
```

With these snaps, you can run all the demos on the **ros.org** website. The commands are even named according to the activities in the documentation. A ROS installation contains a massive amount of software. All packages are small, but dependencies can be troublesome on your regular machine that you use for

» YARP: A SIMILAR FRAMEWORK

YARP is the framework for communications within robotics. It can replace the ROS master as a name server. You can also do this the other way around using the YARP as a name server. The name server will support the nodes and protocols across your system. You can also use both in a single system.

One reason why you may want to use this is if you have a single unit that requires a separate underlying protocol. Another reason might be because your project is in transition between the two frameworks. These protocols can be used in other fields. For example, if you have a number of Kubernets nodes you can have them share information using either YARP or ROS.

The project is open source and available on **https://github.com/robotology/yarp/releases/latest**. The project provides ways to install using Debian packages, or they have their own repository at **www.icub.org/ubuntu**. Naturally, you can always compile from source using their git repository. It requires *CMake*, *cmake-curses-gui* and *git*.

This framework also has its own visualisation system and plotting functions. The focus with YARP is to use as much of the standard GNU tools as possible. There is browser port access, so you can use a browser on one machine to check the stat, along with many other aspects. Discover more about the project at: **www.yarp.it/git-master/index.html**. YARP has been tested and have language-specific packages for Python, Java, Lua, MathLab and others.

»

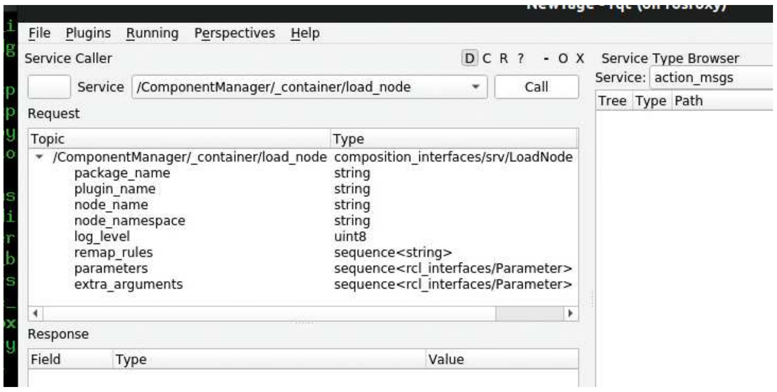
other tasks. You may also want to use many nodes, possible across different types of hardware. For these reasons we would advise setting up a separate environment for ROS whenever possible, rather than use your main system.

You might as well start running it in different machines, even if they are virtual. The communication will be between machines; this makes simulation more accurate. You can find a Linux container solution on the ROS website. The basic idea is that you install Ubuntu and add ROS to the image when it's running.

When you use a container you won't experience problems with dependencies on other parts of your system. See the NixOS boxout (*below*) for more on this.

Building blocks

The full install includes many planning tools and the emulators/simulators for testing without hardware. As we mentioned earlier, the main function of ROS is to establish the communication protocols between nodes.



```
File  Plugins  Running  Perspectives  Help
Service Caller                                     D C R ?  - O X   Service Type Browser
      Service  /ComponentManager/_container/load_node        Call   Service: action_msgs
Request                                                             Tree  Type  Path
Topic                                     Type
▾ /ComponentManager/_container/load_node  composition_interfaces/srv/LoadNode
      package_name                        string
      plugin_name                         string
      node_name                           string
      node_namespace                      string
      log_level                           uint8
      remap_rules                         sequence<string>
      parameters                          sequence<rcl_interfaces/Parameter>
      extra_arguments                     sequence<rcl_interfaces/Parameter>
Response
Field      Type                    Value
```

The RQT displays the format of all the messages that you can create, along with all available services states.

» RUNNING WITH NIXOS

For fans of the NixOS and its package manager, there's also a way to set the whole ROS system up using NIX. Most of the problems with revisions, particularly those involving libraries, are solved with NixOS so it enables you to keep a clean system while using ROS. There's a solution available at **https://aira.life** and **http://robonomics.network**. You can clone this repository and pick anything from the whole ROS project. Even if you are using other distributions, you can use this approach to create a development environment. Learn more at **https://bit.ly/lxf272aira**.

If you want to develop with Python, you can always add that in the call to the shell that you'll eventually run. It's as simple as using the `add` option to the `nix-channel` command. You can also make this work in a directory where the `nix-shell` command runs. In this case, you can have different libraries in different directories. It's equivalent to having a virtual environment when using Python in other projects.

If you want to contribute your own code, you should fork the code and clone your own fork. If you want to contribute any changes, you can raise a Pull Request to the project. Under NixOS, this works from any git repository. You can find the current status and progress of the project at **https://hydra.aira.life/project/aira/channel/latest**. The name is an acronym that spells out: AUTONOMOUS INTELLIGENT ROBOT AGENT. Much of the technology revolves around block chain and the Etherium block chain. This gives you the ability to have more secure communications.

You also need nodes to handle things like the wheels, moving arms and drivers for hardware.

Advanced tasks that you'll need to put into a node include: navigation, image processing and machine-learning tasks. Some machine learning will already be on board, while other such tasks must run on separate servers, or even in the cloud – it's cheaper and more doable than you think! Your main tools for running a ROS system are `roslaunch`, `rosrun` and `roscore`.

In reality, you'll start the system with the `roscore` command and then run a package at a time with `rosrun`. The `roslaunch` command will run all your nodes using an XML-based file with the launch extension. The API for the files isn't stable though, so be careful with upgrades if you decide to use this tool. These are the tools you'll run while exploring the ROS system. They'll run many other tools that are specific for your current task. *Turtlesim* is one, and you run that with `ros2 run turtlesim turtlesim_node`. If you look for more things to run, just type the same thing but use tab after `...run` and you'll see what exists.

Once the nodes are running, you need to control elements of the robot and manage communications. You saw a little bit about the communication system earlier – here it is in more detail. Each node will run one or all of the following:

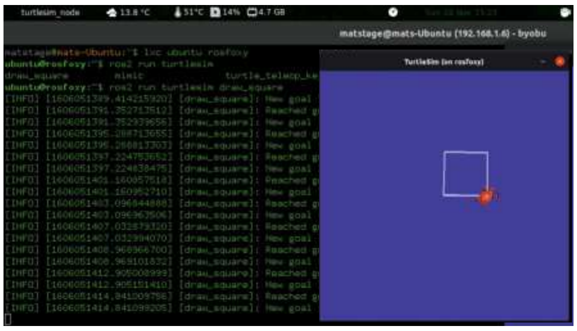
> Services

A service sits ready to answer a request from another node. The content of the messages cover values such as a goal, diagnostic values and the state of movements. You can also pick up information from different sensors, both the function of the robot and position using GPS and the other environment sensors that help your robot complete its task.

If your node needs this information, you need to set up a service client that asks for this information. The node will then process it and send other messages. When there's data that other parts of the system need to know about, you create a publisher in your node that updates a topic. Any node that needs the information subscribes to the topic. Those nodes then receive notifications all at once when the publisher sends it. Most of these will serve streaming data. The ROS implementation isn't designed to know what data is sent – that's for the implementation to take care of.

> Nodes

Each node has a specific task. In your average robot, you'll need to control wheels, view the surroundings and interpret the data to take appropriate action. Each part



```
matstage@mats-Ubuntu:~$ lxc ubuntu rosfoxy
ubuntu@rosfoxy:~$ ros2 run turtlesim
draw_square          mimic                turtle_teleop_ke
ubuntu@rosfoxy:~$ ros2 run turtlesim draw_square
[INFO] [1606051389.414215920] [draw_square]: New goal
[INFO] [1606051391.352713512] [draw_square]: Reached g
[INFO] [1606051391.352939656] [draw_square]: New goal
[INFO] [1606051395.288713655] [draw_square]: Reached g
[INFO] [1606051395.288813303] [draw_square]: New goal
[INFO] [1606051397.224753652] [draw_square]: Reached g
[INFO] [1606051397.224838475] [draw_square]: New goal
[INFO] [1606051401.160057518] [draw_square]: Reached g
[INFO] [1606051401.160952710] [draw_square]: New goal
[INFO] [1606051403.096844888] [draw_square]: Reached g
[INFO] [1606051403.096963506] [draw_square]: New goal
[INFO] [1606051407.032879320] [draw_square]: Reached g
[INFO] [1606051407.032994070] [draw_square]: New goal
[INFO] [1606051408.968966700] [draw_square]: Reached g
[INFO] [1606051408.969101832] [draw_square]: New goal
[INFO] [1606051412.905008999] [draw_square]: Reached g
[INFO] [1606051412.905151410] [draw_square]: New goal
[INFO] [1606051414.841009758] [draw_square]: Reached g
[INFO] [1606051414.841099205] [draw_square]: New goal
```

The turtlesim project stems from the Turtlebot company. It uses ROS to design its line of robots.

will have its own node. Note that you define what a node contains; there are no set rules. You must make a node that makes sense to your specific purposes.

The most expected examples are the ones previously mentioned. For more advanced projects, you'll be separating the hardware layer of a camera from the interpretation. All these things are already solved for you, although you may have better solutions. All nodes communicate using topics and services, and to control the entire system you have the ROS Master, which manages all the communication methods. Your node creates the service by informing the ROS Master about it. Using this method, you'll write all the code to send and receive to topics and services, and never to a specific IP address. In ROS2, you create components and the code that contains a string, which registers a node. The node code contains the protocols that the node supports and data that it'll transmit or receive.

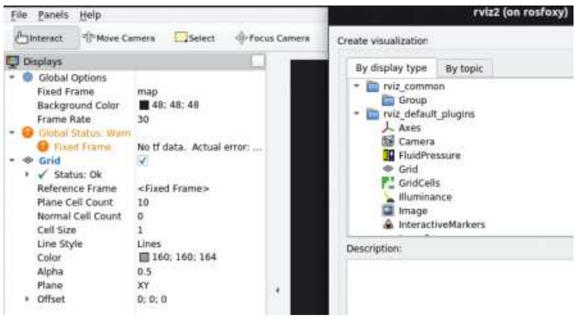
> Visualisation

You can visualise the robot for several purposes. Ensure safe and efficient communication between all nodes. Show the robot in its environment and plan the movements of the body itself. There are different tools for each of these jobs, and they come both as command line and graphical forms. The command line tools are useful for the basics inside the robot. For the most part, they're for the communication that ROS defines, but when you have made enough progress that your robot is able to move, you need more advanced kits.

When you use the *turtlesim* program to run the turtle around and you want to log what you're doing, you can access many commands and have three open terminals to do it. You have many debugging tools available, beyond those that *cmake* provides out of the box. A graphical way to do things is preferable in most cases. The tools that are already available in ROS are *rqt*, *rviz* and *Gazebo*. The first gives you drop-down lists to activate all the parts of the robot and measuring messages. The second shows the robot while navigating. Combining those two approaches is vital to reach of your robotics goals. Make sure you practise these techniques as you become acquainted with ROS.

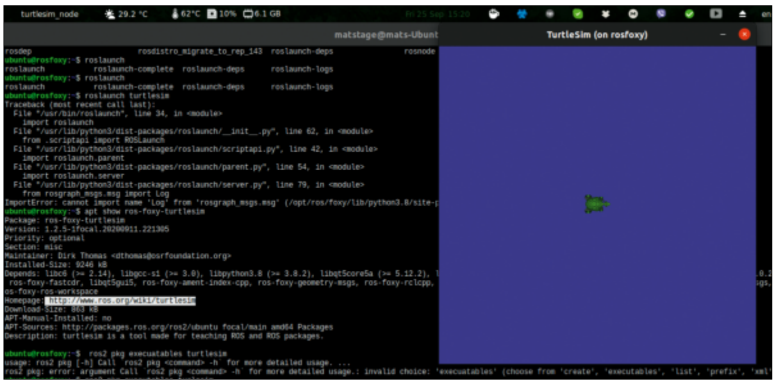
> Rviz, Rqt & Gazebo

These default packages will be installed with the full desktop install. They have different purposes and should be used only when needed. *Rqt* shows the messages that go between all the services you have in your system. You can choose a node at a time and analyse what it does. All messages can be broken down and



```
Global Options
  Fixed Frame            map
  Background Color       48; 48; 48
  Frame Rate             30
Global Status: Warn
  Fixed Frame            No tf data.  Actual error: ...
Grid                     ✓
  Status: Ok
  Reference Frame        <Fixed Frame>
  Plane Cell Count       10
  Normal Cell Count      0
  Cell Size              1
  Line Style             Lines
  Color                  160; 160; 164
  Alpha                  0.5
  Plane                  XY
  Offset                 0; 0; 0
```

You can use rviz to view environments and create obstacles for your robots to overcome.



```
rosdep                     rosdistro_migrate_to_rep_143  roslaunch-deps        rosnode
ubuntu@rosfoxy:~$ roslaunch
roslaunch          roslaunch-complete  roslaunch-deps      roslaunch-logs
ubuntu@rosfoxy:~$ roslaunch
roslaunch          roslaunch-complete  roslaunch-deps      roslaunch-logs
ubuntu@rosfoxy:~$ roslaunch turtlesim
Traceback (most recent call last):
  File "/usr/bin/roslaunch", line 34, in <module>
    import roslaunch
  File "/usr/lib/python3/dist-packages/roslaunch/__init__.py", line 62, in <module>
    from .scriptapi import ROSLaunch
  File "/usr/lib/python3/dist-packages/roslaunch/scriptapi.py", line 42, in <module>
    import roslaunch.parent
  File "/usr/lib/python3/dist-packages/roslaunch/parent.py", line 54, in <module>
    import roslaunch.server
  File "/usr/lib/python3/dist-packages/roslaunch/server.py", line 79, in <module>
    from rosgraph_msgs.msg import Log
ImportError: cannot import name 'Log' from 'rosgraph_msgs.msg' (/opt/ros/foxy/lib/python3.8/site-p
ubuntu@rosfoxy:~$ apt show ros-foxy-turtlesim
Package: ros-foxy-turtlesim
Version: 1.2.5-1focal.20200911.221305
Priority: optional
Section: misc
Maintainer: Dirk Thomas <dthomas@osrfoundation.org>
Installed-Size: 9246 kB
Depends: libc6 (>= 2.14), libgcc-s1 (>= 3.0), libpython3.8 (>= 3.8.2), libqt5core5a (>= 5.12.2), l
 ros-foxy-fastcdr, libqt5gui5, ros-foxy-ament-index-cpp, ros-foxy-geometry-msgs, ros-foxy-rclcpp,
os-foxy-ros-workspace
Homepage: http://www.ros.org/wiki/turtlesim
Download-Size: 863 kB
APT-Manual-Installed: no
APT-Sources: http://packages.ros.org/ros2/ubuntu focal/main amd64 Packages
Description: turtlesim is a tool made for teaching ROS and ROS packages.

ubuntu@rosfoxy:~$  ros2 pkg executables turtlesim
usage: ros2 pkg [-h] Call `ros2 pkg <command> -h` for more detailed usage. ...
ros2 pkg: error: argument Call `ros2 pkg <command> -h` for more detailed usage.: invalid choice: 'execuatables' (choose from 'create', 'executables', 'list', 'prefix', 'xml'
```

To start the simulation you only have to run one command. All other commands need to be run from other terminals.

even sent to test all iterations of the system. You can also view plots of the messages and images from any cameras. You use this system to ensure your nodes communicate correctly. It doesn't display the environment around the robot.

Rviz shows the robot in action in a simulated environment, including obstacles, forces and even weather. Even more powerful and included in ROS but developed independently is *Gazebo*, which enables you to see environments *Gazebo* is a separate project but is included in ROS2. You can use this software to simulate the robot both visually as it moves around and moves itself. You can also film an environment and superimpose the robot on the live feed.

The ROS package is powerful, yet can also be used by keen amateurs thanks to its many available tools and examples. To get used to doing all this work is a challenge worth pursuing but watch out – you'll spend a lot of time before you can take new strides in robotics. As you work with it, you'll learn many techniques and technologies. Remember that this will involve a lot of programming. Running your robot around the room will be just a small part of your work! LXF

» MICRO-CONTROLLERS

If you want to use JavaScript and interface directly with Arduino and other system-on-chip boards, consider these simpler systems. They're designed to interface directly to a large amount of boards, including the Arduino, Raspberry Pi and other small boards.

Johnny-Five is one popular such framework, which interfaces to these boards using JavaScript. The team behind it, Bocoup, has partnered with SparkFun to create a kit that you can start playing with on delivery. At the website (**www.johnny-five.io**) is a long list of examples that demonstrate how you can control and read a long list of servos and sensors.

Cylon is another solution – it's a single NPM module. If you already have some JavaScript projects going, this takes minutes to set up and get started with. It was built using extensions for new hardware and already supports an impressive list of hardware. These include boards, OpenCV and even drones. The libraries are full of code samples, so for a simple way to get started with any of the included platforms, this is an excellent option. Find out more at **www.cylonjs.com**.

If you're more into the Go language, look no further than Gobot (**www.gobot.io**). Installation is via the usual Go package system. Use it with the `go get` command. As all the others, the first example is a flashing LED on an Arduino. The Gobot supports many other platforms though, so you have plenty of options as with the others micro-controllers mentioned here.

XIBO

Credit: https://xibo.org.

Build a custom digital signage system

Discover how to create a digital signage display for an open-source conference using Xibo digital signage and the help of **Matt Holder**.

OUR EXPERT

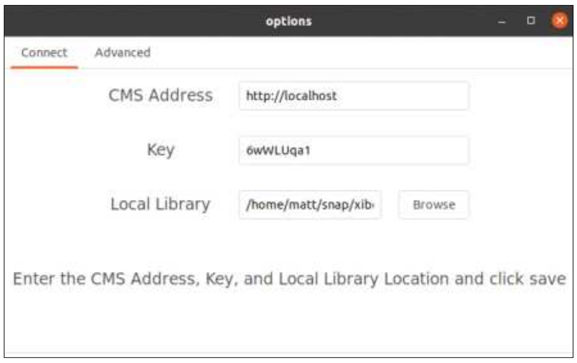
Matt Holder has worked in IT support for over a decade and has always tried to utilise Linux alongside the other installed systems.

Most Linux aficionados will have noticed an increase in the number of large-format screens in public places, displaying a range of information. This is called digital signage. In this tutorial we'll be looking at *Xibo*, a flexible system that can be used to show images, videos, RSS feeds, weather data, mapping data, tables of information, websites, embedded HTML and more. We'll demonstrate how to set up a server and client, before finally designing a layout and scheduling this content.

The system comprises a server component, which is open source, as well as a number of clients for different display types. The Windows and Linux clients are both free and open source, whereas the other platforms are paid-for, licenced clients. The server can be self-hosted, or hosting can be provided by the team themselves.

Before going any further, let's talk about some of the terminology utilised by the system. Displays is the name of the device which runs the player software and displays the content. This could be a Windows/Linux PC, Android device or large-screen TV running embedded software. Display groups can be used to group multiple displays and content can be scheduled to multiple devices at once. Schedules are then used to define when content should appear on Displays.

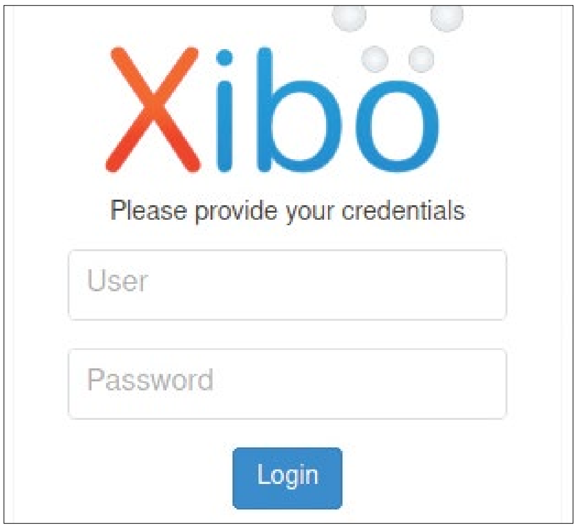
Layouts are designed by the end user to define what content should be displayed on specific areas of the display. Layouts are then split into Regions and each Region contains a playlist. See the diagram (top right), which explains the basic concepts of the *Xibo* digital signage system.



CMS Address: http://localhost
Key: 6wWLUqa1
Local Library: /home/matt/snap/xib

Enter the CMS Address, Key, and Local Library Location and click save

This information is required to configure the client.



Enter your username and password at the login screen.

Each Region can then display a number of pieces of content within its timeline. These concepts will become clearer as we work our way through this tutorial.

Installing the server

The *Xibo* server utilises a LAMP stack (which refers to the Linux operating system, Apache web server, MySQL database server and PHP programming language) as well as other tools to provide messaging between the clients and server. In recent years the entire setup has been wrapped inside *Docker* containers to make installation and updates as simple as possible. The server we'll use to install this on is Ubuntu 20.04, but container systems make the base OS fairly unimportant, so feel free to work with what you're most familiar with.

The software will work equally as well on bare metal or in a virtual machine (VM). When used in production, however, a VM wouldn't necessarily be the most suitable choice, especially due to possible issues with video playback using the VM's drivers.

Once the base machine has been installed, open a terminal, run updates and then install *Docker* and *docker-compose*. Commands should be run as root or prefixed with sudo:

```
apt update && apt upgrade
```

QUICK TIP

Using PHP functions, more advanced projects can be carried out. For example, if information is held in a dataset about meetings booked into a meeting room, then the current date and time can be used by the CMS to serve relevant content to the player. See https://community.xibo.org.uk/t/getting-started-guide-datasets/14149.

```
apt install apt-transport-https ca-certificates curl
software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/
gpg | apt-key add -
add-apt-repository "deb [arch=amd64] https://
download.docker.com/linux/ubuntu $(lsb_release -cs)
stable"
apt install docker-ce
curl -L https://github.com/docker/compose/releases/
download/1.24.1/docker-compose-`uname -s`-`uname
-m` -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

With the prerequisites installed, there are a few steps required to configure the *docker-compose* files to enable the *Docker* containers to run and communicate with one another.

```
mkdir /opt/xibo
cd /opt/xibo
wget -O xibo-docker.tar.gz https://xibo.org.uk/api/
downloads/cms
tar --strip-components=1 -zxvf xibo-docker.tar.gz
cp config.env.template config.env
nano config.env
```

On the relevant line, enter a strong password for the MySQL database and configure any other required options. Find out more about the *docker-compose* file here: **https://xibo.org.uk/docs/setup/xibo-cms-with-docker-on-ubuntu-18-04#create_config.env_file**.

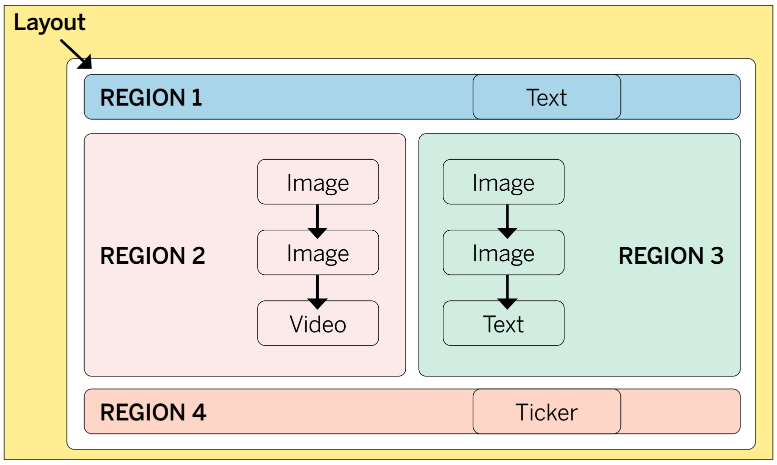
Next, bring up the containers using `docker-compose up -d` and allow *Docker* to complete its work. The server has now been installed and the server can be accessed from your favourite browser, by entering **http://IP_ADDRESS_OF_XIBO_SERVER** – enter your username and password at the login screen.

The player

There are open source players for Windows and Linux as well as paid-for options for other platforms as well, such as Android, Tizen and Web OS. In this article we'll install the player on Ubuntu 20.10. The *Xibo* team uses Canonical's Snap packaging system, which makes the installation of the client straightforward and updates are applied automatically. Again, back in the terminal the client can be installed by entering:

```
$ snap install xibo-player --channel=stable
```

To enable the client to communicate with the server, there are a couple of steps that need to be followed.



This diagram explains the concepts of the Xibo digital signage system.

First, in a web browser, login to the CMS using the default credentials of `xibo_admin` and `password`. Navigate to the Settings tab and copy the key from the CMS Secret Key field. The player can be opened from the Desktop or from a terminal, by entering `xibo-player`. The first time this is opened, the configuration screen will load. Enter the URL of the CMS (server) component, paste the value from the CMS for the Key and a location that the client can use to save files to (see screenshot, left). The information shown in the screenshot is required to configure the client.

When selecting the Save option, the client will contact the server, which will report back that the Display isn't currently registered. Now, back in the web browser, navigate to the Displays tab and one entry will be shown. On the right-hand side of the screen, select the Options menu and then click Authorise. Finally, back at the client click Save and the client will report that the Display is active and ready to start. The options screen can then be closed and when the client is opened again the default layout will be displayed (see screenshot, previous page). Once configured, the client will download the files required for the default layout. These will then be displayed.

Conference time

The example that we'll be walking through are displays that could be set up around a conference venue. The layout (see diagram above) will contain a title in Region 1, text in Region 2 which will relate to videos and

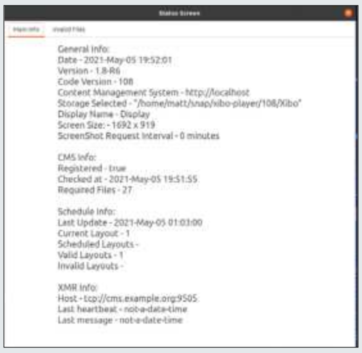
QUICK TIP

The new third version of Xibo, released earlier in February 2021, contains lots of exciting features, including interactivity, folders to store media, reworking of permissions and support for sending commands via RS232.

» LINUX PLAYER OPTIONS

Once the initial configuration of the client has been carried out, further options are set from the CMS. To access these options, navigate to the Display Settings and either change the relevant profile or create a new one, which can be assigned to a particular Display. Within the Display Settings a range of features can be changed, such as the interval between the client collecting information from the server, the size and position of the player window (the default is full screen) and whether shell commands can be executed by the player. The latter option is useful because a display can be set to shut down at a certain time of day.

When the client is loaded, press the I key and then the information screen will load. This provides access to diagnostic information such as media that's being downloaded, if there are any invalid files and gives the option to exit the player without it being reloaded automatically. Because clients are devices that need to run for large proportions of the day, should an error occur then the default behaviour is for the player to reload itself automatically.



```
General Info:
Date - 2021-May-05 19:52:01
Version - 1.8-R6
Code Version - 108
Content Management System - http://localhost
Storage Selected - "/home/matt/snap/xibo-player/108/Xibo"
Display Name - Display
Screen Size: - 1692 x 919
ScreenShot Request Interval - 0 minutes

CMS Info:
Registered - true
Checked at - 2021-May-05 19:51:55
Required Files - 27

Schedule Info:
Last Update - 2021-May-05 01:03:00
Current Layout - 1
Scheduled Layouts -
Valid Layouts - 1
Invalid Layouts -

XMR Info:
Host - tcp://cms.example.org:9505
Last heartbeat - not-a-date-time
Last message - not-a-date-time
```

Pressing I when in the client brings up a status screen.

»

images in Region 3. Region 4 will contain a scrolling ticker of an open-source RSS feed.

Before creating the example, we should investigate the GUI (see below). 1) Layouts are scheduled at a date and time. 2) Layouts are created here. 3) Media is managed here. 4) Displays are managed here.

Back in the web browser, login to the *Xibo* digital signage CMS, navigate to the Layouts section and create a new one. Once a name has been entered, the defaults should suffice. When created, the Layout editor will open. The first step is to resize the default Region, which is created by default. Select the option to edit Regions, which looks like an image of a pencil and notebook, and use the drag handles to resize the orange rectangle. Add further Regions to the layout, so that the image in the diagram has been replicated and four Regions exist within.

Regions can be set to loop, which is useful when they contain certain types of content. To do this, select the Region and then read the guidance before deciding whether to enable the loop option or not. Before saving the changes that have been made to the layout, set a light background colour or upload a background image by using the options in the right-hand panel.

Before carrying on with the setup of the digital signage, it's a good idea to take stock of what has been accomplished. So far, the server/CMS has been installed, the client has been installed and configured to communicate with the server, and the basic Layout has been set up with the correct number of Regions. In the next step, the Regions will be populated with information and finally the Layout can be scheduled and then viewed on the player.

Establish your Region

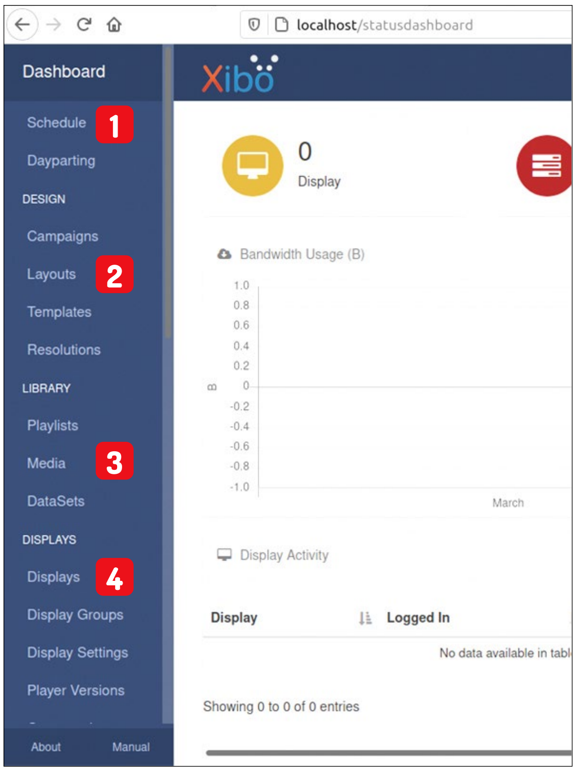
To start with, the top Region will be populated with a title. Add the title into Region 1 by selecting the icon that shows four squares, click the plus on the Text widget and then select Region 1 in the playlist editor. Select the Edit icon on the text item and enter a name for the conference as well as formatting the text. When completed, select Save (see screenshot, below right). When editing Regions, options appear on the right-hand side of the preview. This is an example of the options associated with the text widget.

The next step is to add an RSS feed of Linux news to the bottom Region. This is very similar to entering text, but requires the usage of the Ticker widget. This widget is formatted from the configuration panel in the top-right of the Layout editor. The defaults should suffice on the General tab. On the Configuration tab select the Feed URL field and add the link to a suitable RSS feed. On the Templates tab, select Title Only and finally, on the appearance tab, select Marquee Left. When these changes have been made, save the changes.

One of the many powerful things about *Xibo* is the ability to schedule content in two different Regions that can change at the same time. This will be useful for the final part of this tutorial because we're going to populate the final two Regions with textual information on the left-hand side and a video and image on the right-hand side. The textual information will relate to the video being shown. When the video changes to a second piece of content, the text should also change at the same time.

The content that we'll be adding is a video of Big Buck Bunny as well as some information about it. The second piece of content will be a photograph of Linus Torvalds, as well as some information about him and the

There are four key areas in the CMS' interface (see below for details).



1 Layouts are scheduled at a date and time to show on a display.

2 Layouts are created here.

3 Media is managed here.

4 Displays are managed here.

» DO MORE WITH XIBO

As well as the media formats that have been discussed, *Xibo* can utilise a large range of resources to create effective digital signage displays. Audio files can be loaded as background music, calendar items can be read from, for example, a Google Calendar, countdown times can be displayed and clocks can be added to alert the user to the current time.

Looking at more data-oriented items now, currency trackers can be added as well as displaying data from the tabular dataset objects. These data can be displayed by means of a ticker, table view or chart view. Information from other websites can also be embedded using the Embedded HTML widget. When using this though, make sure you have permission to display information from another source.

While not strictly a media type, playlists are an incredibly powerful feature as these enable a previously configured timeline of items to be added to multiple layouts. When multiple layouts are scheduled at the same time they can either play sequentially or one can be configured to have priority. In this way an urgent layout can be displayed for a short period of time before normal scheduling is resumed.

Campaigns are used to group together a number of layouts and these can then be scheduled in one operation, rather than having to spend time scheduling multiple layouts separately.

Linux kernel. First of all, download a Creative Commons licenced photo of Linus Torvalds as well as the Big Buck Bunny film from **https://download.blender.org/demo/movies/BBB**.

To add the video, select the Region you wish to display it from, and from the toolbox at the bottom, hover over the video button and then use the Grab option to drag it to the Region you wish it to appear in. At this stage, it's important to know the length of the video file so that you can set the timing for the text in the adjacent Region to be the same.

Add text to your display

To add some text, grab some from a suitable source and store it temporarily in a text file. To add this to *Xibo*, select the Region from the Region editor and then navigate to the Text button. Again, use the Grab button to drag this to the relevant Region. When the Text widget has been added, set the duration to be the same as that of the video file and click the pencil icon on the left of the Options panel. Paste the content into this Region and then format appropriately, before saving the Region settings on the right-hand side. Add a marquee effect to the Region if you want the text to scroll.

These previous steps can now essentially be repeated. With the Layout design tool, select the same Region that contains the video of Big Buck Bunny. From the Widget selection tool at the bottom of the screen, select Image and use the grab handle to drag this to the selected Region. When asked, select Add Files to select the photograph to be uploaded. When a thumbnail appears in the Upload media window, click Start Upload. Now click Done and you'll see that in the Region in question there are now two items: a video and an image (you may need to scroll to the right). Select the image, click 'Set a duration' and set this to 60 before saving the Region.

Now select the adjacent Region and use the grab handle on the Text widget to drag it to the Region. This Region will then contain two text widgets. Make sure you check that the text widgets appear in the correct order and therefore match the order of the content in the other Region. If necessary this can be changed by dragging the order. With the final text widget selected, set a duration of 60 again and then use the pencil icon on the Region preview to enable Edit mode, where you can paste in your content and format. As before, set a marquee effect if the text should move in your Region.

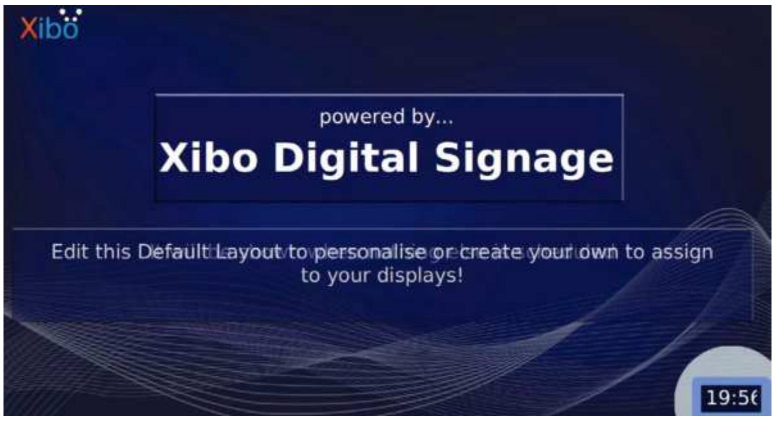
Now that the content has all been added to the layout, this can be previewed within the web browser. To do so, click the back arrow button, which is underneath the Region editing preview. This will then display an overview of the entire layout. Simply selecting the Play icon will show a preview in the browser. Please note that not all content can be previewed in this way.

Once the layout has been completed, the final stage is to use the Actions option at the top of the screen to publish the Layout. This means that the changes will be saved and ready to be displayed. When making further changes to the layout, these can be saved part-way through and then published at a later time. Also, publishing changes can be made on a schedule, which can come in very handy.

The final stage in the process is to schedule the Layout to the display. To complete this, on the left-hand side select Schedule and add a new entry. When the pop-up loads add the Layout, select the Display, Layout and select the date and time options for the Display. Use the option to save the schedule addition and this can be seen on the calendar (make sure to select the Display in the option at the top of the screen if the calendar appears empty).

Now, moving back to the Desktop, open the *Xibo* player and within a couple of minutes the content will be downloaded and displayed on the player.

So far we've merely scratched the surface of what this digital signage system is capable of. For some more advanced things to try, look at using datasets to display table data (datasets can also be used in a ticker) and the use of PHP functions to display certain parts of the dataset data, based on the date and time. For further information, the *Xibo* team has spent a considerable amount of time crafting documentation to support all users with the software. This can be found at **https://xibo.org.uk/manual/en/index.html**. LXF

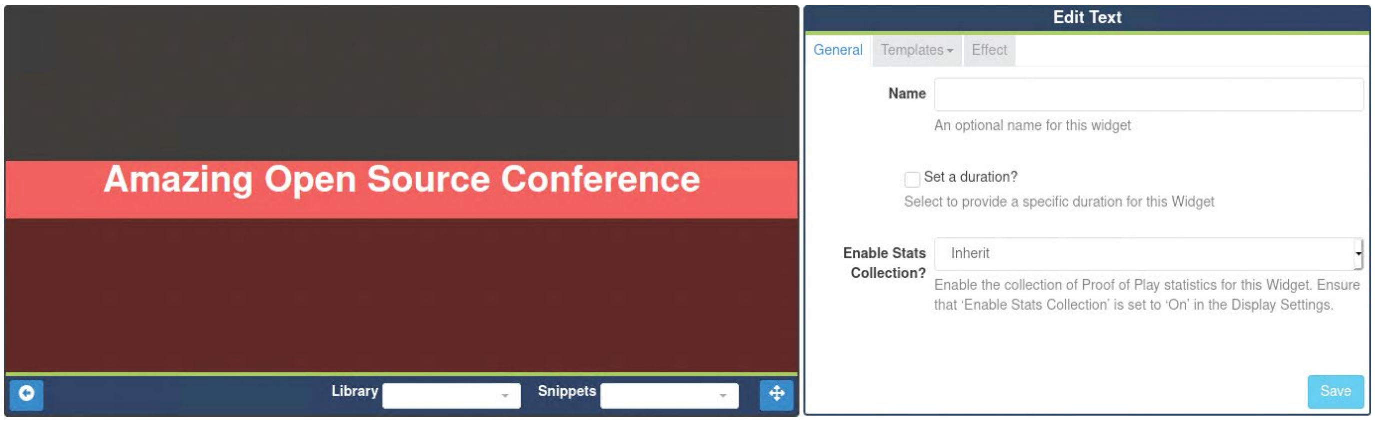

Xibo
powered by...
Xibo Digital Signage
Edit this Default Layout to personalise or create your own to assign to your displays!
19:56

Once configured, the client will download the files required for the default layout. These will then be displayed.

QUICK TIP

The default credentials for the server are xibo_admin and password. The password should be changed after first login and can be done so by selecting the avatar in the top right-hand corner and selecting Edit Profile.

When editing Regions, options appear on the right side of the preview. This is an example of the options associated with the text widget.



Name

An optional name for this widget

Set a duration?

Select to provide a specific duration for this Widget

Enable Stats Collection? Inherit

Enable the collection of Proof of Play statistics for this Widget. Ensure that 'Enable Stats Collection' is set to 'On' in the Display Settings.

Credit: https://noaa-apt.mbernardi.com.ar

SOFTWARE DEFINED RADIO

Decode signals from weather satellites

Weather satellites transmit signals that you can receive and decode to generate images of the Earth. **Mike Bedford** shows you how.

OUR EXPERT
Mike Bedford is fascinated by receiving signals from weather satellites and processing the data to generate images.

Satellite-based images of the Earth showing cloud cover have been a regular sight in TV weather broadcasts since the TIROS 1 satellite launched in 1960. Yet while the end result is familiar, the technology that brings us those images is much less well-appreciated. If you want to learn about the technology involved, though, it's not too difficult to get a basic understanding.

However, you can gain some hands-on experience, as we demonstrate here, and a practical approach is often one of the best ways of learning. So, we'll show you how to receive the radio signals transmitted by these satellites and how to decode that data to generate pictures. In addition, we'll also investigate some ways of image processing to improve that image and perhaps introduce some false colour into the inherently monochrome images transmitted by the satellites.

NOAA satellites

Here we're going to be looking at the NOAA polar orbiting satellites – NOAA-15, NOAA-18 and NOAA-19 – that are operated by America's National Oceanic and Atmospheric Administration. We've chosen these satellites because they orbit the Earth at an altitude of just 850km. Since they aren't too far above our heads, their radio signals are quite strong and easy to receive.

If you're wondering why the satellite numbers aren't contiguous, all satellites from NOAA-6 have existed but most are no longer operational, including NOAA-16 and NOAA-17 even though they were launched after NOAA-15. However, NOAA-15 has been experiencing operational problems since 2019 so there are no guarantees that you'll be able to receive and decode its transmissions, even though we had no difficulties. There's also an NOAA-20, but this is the first of a new generation of polar orbiting satellites. It transmits on a much higher frequency which makes it significantly more difficult to receive. For that reason, decoding software probably won't be too plentiful, although one offering is reportedly available at an alpha testing stage.

The laws of gravity dictate that any Earth satellite has a velocity and hence also an orbital period, which depends on its altitude. This means that the launch and fine-tuning of its altitude is critical to ensuring that it orbits the Earth in the time that was intended by its designers. Specifically, the closer a satellite is to the surface of the Earth, the faster it orbits.

Let's take the example of a satellite at an altitude of 35,786km. It'll orbit the Earth every 24 hours so, if it's in an equatorial orbit, travelling in the same direction as the Earth's rotation, it'll appear to be stationary above a particular point on the planet. This is referred to as a geostationary orbit and there are geostationary weather satellites, for example NOAA's GEOS-15, GEOS-16 and GEOS-17. Turning to our main theme of the polar orbiting NOAA satellites, their 850km altitude means that they orbit the Earth just over 14 times every 24 hours. As the name suggests, the orbits of these particular satellites take them over both the poles. However, most importantly, because the Earth is rotating below them, every orbit is shifted in such a way that it passes over a different swathe of the Earth's surface compared to the previous orbit. This arrangement enables each satellite to obtain images from any point on the Earth's surface twice each day: once during daylight and once at night.

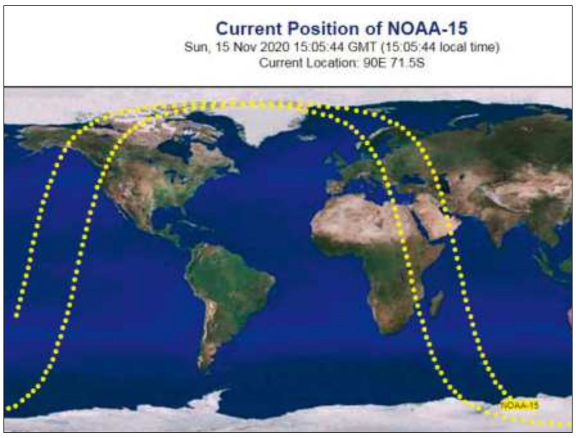
Is it still raining?

That's enough background for now, so let's take a look at how to receive signals from these satellites. Not too long ago you'd have needed to buy a radio receiver that covered the frequencies used by the NOAA polar orbiting satellites, but now you can use so-called Web SDRs, which are radio receivers hosted on the Web. Take a look

Launched in 1998, NOAA-15 is still mostly operational, despite it having had a design lifetime of just two years. CREDIT: NASA

QUICK TIP

Since the NOAA polar orbiting satellites orbit at only 850km and you can easily receive their radio signals, you might wonder if you can see the NOAA polar orbiting satellites by looking at the sky. According to the NOAA, they are considered naked eye objects but only just. In practice, therefore, you're only going to see them in dark rural areas.


Current Position of NOAA-15
Sun, 15 Nov 2020 15:05:44 GMT (15:05:44 local time)
Current Location: 90E 71.5S
NOAA-15

Using a satellite prediction utility means you'll know when each of the three NOAA polar orbiting satellites is due to fly close to any Web SDR.

at **www.websdr.org**, which acts as a directory to these online receivers. For each there's information on the receiver's location, a list of the frequency bands covered and a link that takes you to that receiver. The ones of interest are those that cover the 137MHz satellite band, which is in the VHF part of the radio spectrum, but these are the exception rather than the rule since most concentrate on the amateur radio bands.

You'd probably prefer to use a receiver close to your own location so you'll be receiving images of your locale. If you're based in the UK the closest we found is the one at **http://erc-websdr.esa.int**, which is based at the European Space Research and Technology Centre at Noordwijkin in the Netherlands. We only found one in the US – **http://websdr2.K3FEF.com:8902** – which is in Milford, Pennsylvania. There are also Web SDRs in Germany and Russia that cover the 137MHz band.

Your first task is to become familiar with your Web SDR of choice (although most share the same user interface) and try receiving some satellite signals. The main part of the user interface is a waterfall display that shows signals graphically with the frequency plotted horizontally and time vertically. The most recent signals appear at the bottom of the display and, since the data scrolls up continuously, you can always see the previous few seconds of signals. You should also be able to hear any signals on the selected frequency audibly, although with some Web SDRs you have to click a button to enable audio. Note that if you're using *Chrome* you might find you hear no sound, in which case you will have to enable sound for the website in Settings.

To receive the NOAA satellites you should select FM as the mode and, because these signals have quite a high bandwidth, you should increase the receiver bandwidth to around 40kHz. Do this by clicking "Wider" repeatedly, but it's quicker to drag the sides of the yellow "filter" icon just below the waterfall display. You'll need to know that NOAA-15, NOAA-18 and NOAA-19 transmit on 137.6200MHz, 137.9125MHz and 137.1000MHz, respectively, and some of the Web SDRs, albeit not the one in the Netherlands, help by indicating the frequencies for the various satellites just below the waterfall display.

However, for best results you might need to fine-tune the frequency because the received frequencies will differ from the transmitted frequencies due to Doppler shift. At any one time it's likely that none of the three satellites will be receivable at your selected Web SDR so it's a good idea to use an orbit predictor to find out when one of the satellites will be flying close to the receiver – see **www.amsat.org/track**. Satellites don't have to be flying above the receiver – they'll be receivable on orbits that are displaced east or west by a thousand kilometres or more. You'll soon learn to recognise a satellite signal from its shape on the waterfall display and from its pulsing sound.

Now it's time to acquire some data from a satellite, so make sure you're on the correct frequency before you expect the satellite to come into range. Although the signal will be strongest at its closest approach, you'll be able to receive a reasonable signal for several minutes before and after that point. You'll probably also prefer to record signals when it's daylight at the receiver location, although this isn't essential because the satellites are able to acquire images in spectral bands other than visible light.

Once you can see and hear the signal, click Start next to Audio recording and, after the signal has been lost, click Stop. We got an error message telling us that our browser didn't support downloading of recordings but this wasn't true. Having stopped the recording, click the adjacent download link to grab a copy of the recording.

Decoding signals

We're about to look at how to decode the signals you've recorded from NOAA satellites, but first a bit more theory. It might seem surprising – but perhaps less so when we bear in mind that the current NOAA polar orbiting satellites are the last of a series that first entered operation in 1979, well before broadcast TV migrated to digital – but our satellites of interest transmit analogue signals. To clarify what we mean by that, like old analogue TV, the image is discrete vertically, but each horizontal line is represented by a continuously varying signal. In the case of the NOAA polar orbiting satellites, the signal is encoded in a format referred to as APT, which stands for Automatic Picture Transmission. In this format, the data for each

QUICK TIP

Although it's easy to receive signals from the low-altitude NOAA satellites, you might like to see photos from the much more distant geostationary weather satellites. Near real-time images are plentiful. See, for example, www.star.nesdis.noaa.gov/GOES, which hosts images from the NOAA's geostationary weather satellites.

» USE A LOCAL RECEIVER

If using a remotely hosted Web SDR seems like cheating, it's possible to use your own hardware to receive satellite signals. You could put together all the kit you need for as little as £10 plus the cost of some bits and pieces to make the antenna. The secret is to use an SDR: a software-defined radio, just like the ones that form the basis of the Web SDR sites. Unlike conventional radio receivers, which need a lot of analogue hardware, an SDR is little more than an analogue-to-digital converter that relies on software to provide most of the functionality. The SDR of choice is a USB dongle called RTL-SDR, that you can read about at **www.rtl-sdr.com**. The hardware design is open so you'll be able to find lots of sources but do shop around. The software you'll need is also open source.

The other thing you'll require is an antenna and you can build one yourself. The RTL-SDR website has a tutorial on receiving NOAA polar orbiting satellites and included here are links to a few antenna designs. Alternatively, lots of other antenna designs are available. The designs can be as simple as a basic framework made from PVC electrical conduit, which supports the active elements that are made from coaxial cable.

»

scan line is an audio tone at 2.4kHz, which is amplitude modulated with an eight-bit (256-level) signal representing the visible brightness. In other words, the brighter the point along the scan line, the louder the audible tone.

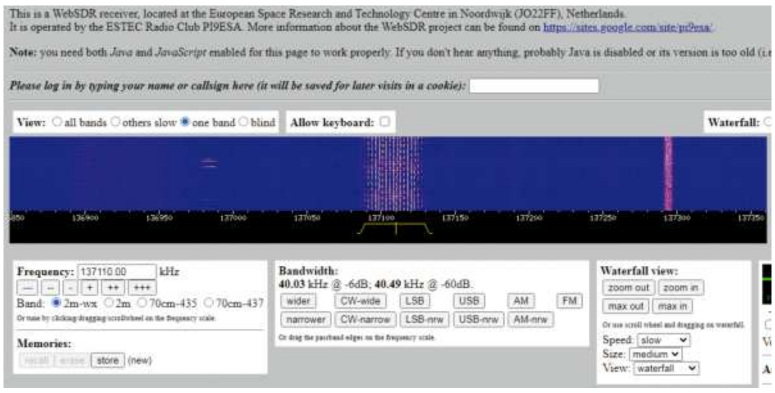
Two such lines are transmitted each second. There are also periodic synchronisation pulses and telemetry data. Taken together, this explains why the signal is audible and can be recorded as a WAV file (2.4kHz is approximately the fourth D above middle C on a piano keyboard) and why it has a pulsing sound. This audible ATTP data is superimposed onto the 137MHz radio signal by frequency modulation.

The purpose of the decoding software is to translate the data in the APT signal into an image. One of the best respected tools to do this job is called *WXtoIMG*. It's referred to as abandonware, which means that its author no longer supports it and apparently has no interest in it. However, it isn't open source and, while there was a free version, the full version was licenced commercially and requires a licence key.

Today, enthusiasts are attempting to keep the software alive by making it available for download, together with the licence key. However, given its questionable legal status, we're recommending a different software package which is called *noaa-apt*. It's free and open source, and you can download it from **https://noaa-apt.mbernardi.com.ar**.

You'd work it out eventually for yourself, but since it took us a while to manage to display an image, we'll provide some brief instructions on how to use *noaa-apt*. If you want to try it out before you've recorded any satellite signals, you can download a sample recording from **https://project.markroland.com/weather-satellite-imaging/N18_4827.zip**. Use the larger of the two WAV files and note that this zip file also has some images that show what you ought to see when you've decoded the file.

Here, on the Noordwijkin WebSDR, the signal from NOAA-15 is clearly visible in the waterfall display, on a frequency close to 137.1000MHz.


This is a WebSDR receiver, located at the European Space Research and Technology Centre in Noordwijk (JO22FF), Netherlands.
It is operated by the ESTEC Radio Club PI9ESA. More information about the WebSDR project can be found on https://sites.google.com/site/pi9esa/

Note: you need both *Java* and *JavaScript* enabled for this page to work properly. If you don't hear anything, probably Java is disabled or its version is too old (i.

Please log in by typing your name or callsign here (it will be saved for later visits in a cookie):

View: ○ all bands ○ others slow ● one band ○ blind **Allow keyboard:** ☐ **Waterfall:**

Frequency: 137110.00 kHz
Band: ● 2m-wx ○ 2m ○ 70cm-435 ○ 70cm-437
Or tune by clicking/dragging/scrollwheel on the frequency scale.
Memories: recall erase store (new)

Bandwidth:
40.03 kHz @ -6dB; **40.49 kHz** @ -60dB.
Or drag the passband edges on the frequency scale.

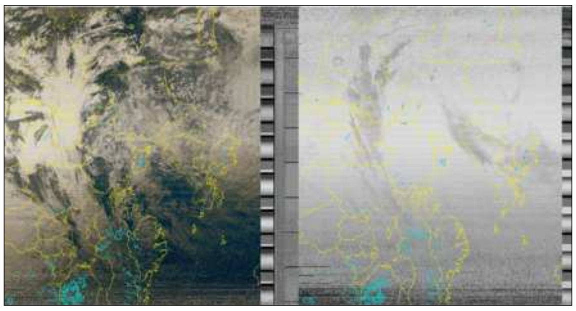
Waterfall view:
Or use scroll wheel and dragging on waterfall.
Speed: slow
Size: medium
View: waterfall

Start decoding

The software will start up on the Decoding tab and here you need to select the audio file you recorded (or downloaded) before clicking Decode. The bar at the bottom will show progress (it won't take long) and, on completion, it'll indicate "Decoded". Note, though, that this doesn't necessarily mean that the file contained valid data, and if it didn't you'll only find out in the next step. This is because, even if a valid signal has been decoded, no picture is displayed at this point. So, select the Processing tab and click Process. The image should appear almost immediately and, in the case of the sample recording file, it should be fairly easy to recognise the Great Lakes and the Florida peninsula.

We'll describe the characteristics of a displayed image in more detail later, but first you might like to take a look at the information shown in the panel to the left of the image. It shows the name of the satellite and either the start or end time and date of the recording. These are *noaa-apt's* best guesses, based on the file's time stamp and its filename. For example, the name of a file we downloaded from the Dutch Web SDR was **websdr_recording_2020-10-31T16_31_24Z_137109.0kHz.wav** that shows how, at least in this case, the necessary data can be deduced. This information is used to calculate the orbit and so add coastlines and national borders to the image, this being an option you can select a bit lower down in the panel.

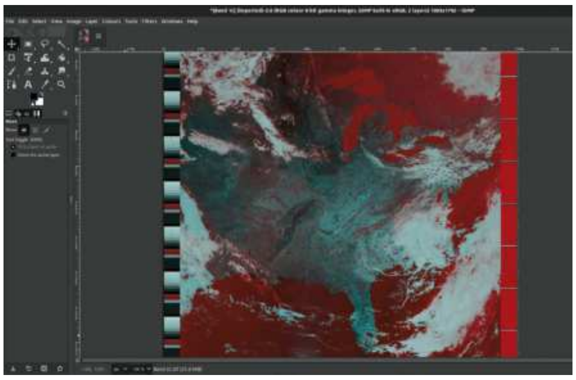
We found that *noaa-apt* will often guess wrongly. If this happens you can alter the information manually. If you change anything anywhere in this panel, though, the image won't change until you click Process again. You'll probably notice that the map overlay won't correspond to the recorded data with the sample audio file, but it should work correctly with data you record yourself subject, perhaps, to altering the identity of the satellite.


noaa-apt output of data from the Noordwijkin Web SDR with coastlines added. The map is upside down because it was a north-to-south pass.

» THE AVHRR SENSOR

The NOAA polar orbiting satellites carry an on-board instrument called the Advanced Very High Resolution Radiometer (AVHRR). You can think of this as a camera, although it's substantially different from regular cameras. In particular, the sensor isn't a two-dimensional array that captures a 2D scene in a single instant. Instead, because the satellite is continually moving, it's necessary only to capture a row of pixels at right angles to the flight path. These rows are acquired continually and, because the satellite will have moved between any two adjacent rows, they'll represent different strips of the Earth's surface. Stacking these strips, therefore, creates a two-dimensional image. The 2D image is infinite in length, even though the image that can be captured from a receiver at any particular location represents a limited area due to the range of the radio signal.

Despite it only being necessary to capture a strip of pixels, the AVHR doesn't have a sensor containing a row of 2,600 elements, this being the effective horizontal resolution. More recent sensors do operate this way, and are referred to as push broom sensors, but the AVHR, being an older design, utilises a whisk broom sensor. In a reference to the way a whisk broom or "witch's broom" is used, the device scans mechanically back and forth at right angles to the direction of movement. Using a mirror, it directs the light from successive points along the horizontal strip to a single sensor for each spectral band.


By combining images in two different spectral bands a false colour image can be generated, making some features more recognisable.

Finally, assuming the end result is something you're satisfied with, don't forget to save the image to file which you can do in the Save tab.

Image processing

Although *noaa-apt* offers a degree of contrast enhancement, depending on the lighting at the time the satellite acquired its image, and on the strength of the signal when you received it, the image might not be perfect. In that case you could try improving it in photo-editing software. Improving the brightness and contrast is easy, and can make a significant difference. However, the Curves feature that's available in many image-editing packages can make an even more dramatic difference very easily. If you've never used this feature before it would be worthwhile reading up on it.

One thing you'll notice the first time you decode an NOAA polar orbiting satellite image is that you don't end up with one map but two, which are displayed side by side. Both are monochrome, but they're not identical, even though they both represent the same area of the Earth's surface. This is because the satellites are able to acquire images in five spectral bands covering the red/green portion of the visible spectral, near infrared, mid-infrared, and two wavelengths of thermal infrared. However, the bandwidth available at 137MHz is insufficient to transmit them all, so only two channels are transmitted at any one time, the selection being made by ground controllers. Normally, during daylight hours, one of the photos will be the visible light image.

Because only one visible light band is captured – and you probably won't get this at night – it isn't possible to generate a true full colour image. However, *noaa-apt* does offer a false colour option on the Processing tab that you can fine-tune using the three threshold sliders. In reality, it's not false colour but artificially added colour that aims to approximate actual colours: blue for water, white for clouds and green for vegetation.

It's not clear how this works and, since it's a fairly recent addition, information isn't currently provided. All we can say is that it uses a form of image analysis to attempt to identify clouds or the type of ground cover. Apparently it's not as good as the false colouring feature that was provided by *WXtoIMG* so there might be an opportunity to contribute to this open source project by helping to enhance this feature.

Another possible way to create a false colour effect – and here we genuinely do mean false since the colours will generally bear no resemblance to actual colours – is to combine the two images externally to *noaa-apt* using image manipulation software such as *Gimp*. Needless to say this will only give an interesting result if the two images appear to be substantially different. With satellites that transmit images in more than two spectral bands (for example, not the NOAA APT signal), it's common to choose combinations of three of the spectral bands and reproduce them in red, green and blue. Although the end result is most definitely not realistic, is does have the potential to produce all possible colours. With just two spectral bands, though, things are different.

You could choose two primary colours so, for example, using red and blue would give you those colours plus purples, using red and green would give you those colours plus yellows and oranges, and using green and blue will give you those colours plus shades of turquoise or cyan. None of these options enables white to be represented so you might prefer to go for one primary colour together with the complementary secondary colour. Those possible combinations are red and cyan, green and magenta, or blue and yellow.

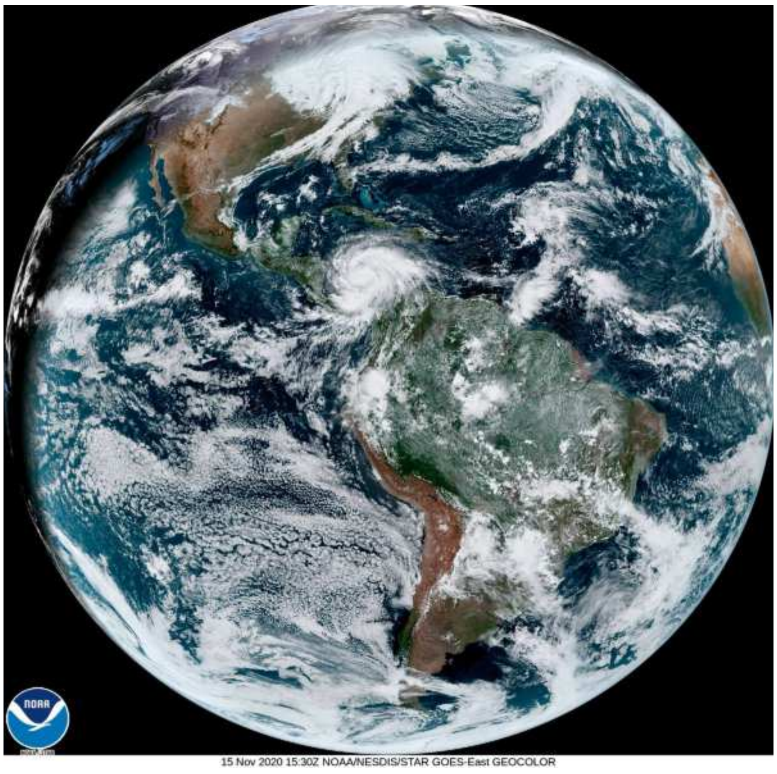
We're not going to give specific instructions on how to combine the images in *Gimp*, or any other package, but here's the gist of it. First, you need to cut out the two side-by-side images from the composite photo generated by *noaa-apt* and save them as separate files. Now, convert those files from monochrome to RGB images and colourise them in your selected colours (in *Gimp* you'd do that by adding a separate layer filled with your selected colour as the top layer and use Darken only as the mode). Finally, load the two colourised images as different layers and adjust the transparency of the top layer.

Initially you should try a transparency of 50 per cent, but if one of the two colours appears to dominate then you should increase or decrease the transparency of the top layer accordingly. LXF

QUICK TIP

In addition to the APT signals transmitted on 137MHz, the NOAA polar-orbiting satellites also transmit a so-called HRPT (High Resolution Picture Transmission) signal at a higher frequency of around 1.7GHz. The signal has a higher resolution and dynamic range, and all five spectral bands are transmitted at all times. This signal is much harder to receive, though.

Each of NOAA's geostationary satellites produce a true colour image of almost half the planet, but they're much more difficult to receive.



SCRIBUS

Credit: www.scribus.net

Sharpen your desktop publishing skills

Aspiring media mogul **Nick Peers** reveals how to design newsletters, flyers and more with Scribus, the powerful open-source DTP tool.

OUR EXPERT

Nick Peers has been dabbling with DTP software since he kickstarted his writing career in – gulp! – 1995.

Word processors like *LibreOffice Writer* can do a great job of sprucing up documents, but if you're serious about page design, you can't beat a dedicated desktop publishing tool. *Scribus* is capable of producing anything from brochures and flyers to full-blown newsletters, which you can print or share digitally via PDF as you see fit.

It's packed with powerful tools and options, but in this tutorial we'll introduce you to the fundamentals of using *Scribus*: from changing existing content to designing your own documents from scratch. It can be installed or run several different ways: via AppImage, using its own dedicated PPA, or through Flatpak. While the stable version (1.4.x) is listed as 'recommended', it's been effectively abandoned because the current development version (1.5.7) is on the cusp of replacing it, so we'll be focusing on that version in this tutorial.

The PPA version is best for those running the non-LTS version of Ubuntu, currently 21.04:

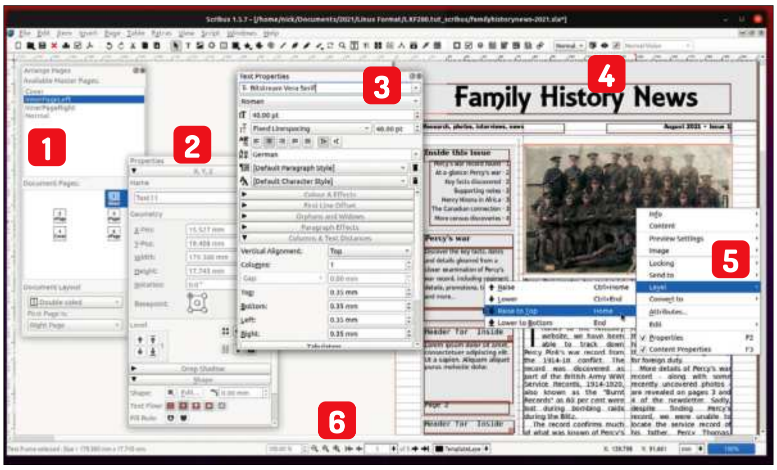
```
$ sudo add-apt-repository ppa:scribus/ppa
$ sudo apt-get update
$ sudo apt install scribus-ng
```

If you're on the LTS release cycle (Ubuntu 20.04) install *Scribus* through Flatpak if you have that installed, or use the portable AppImage if not (**https://sourceforge.net/projects/scribus/files/scribus-devel/1.5.7**). If you're using Flatpak:

```
$ flatpak install flathub net.scribus.Scribus
```

Once installed, *Scribus* can be launched via the Show Applications button. It'll be described as *Scribus* (Beta), but don't worry – it's as close to stable as you can get.

TAKE THE SCRIBUS TOUR



1 Arrange Pages Use this box to manage the structure of your document – view, add and remove pages as needed.

2 Frame Properties This box enables you to define the frame's properties. Use the Shape section to flow other objects around it.

3 Item Properties Press F3 to tweak the properties of a selected frame's contents rather than the frame itself.

4 Page Preview Click the eye icon shown to show the page without any invisible elements, such as grids and frames.

5 Context-sensitive controls Right-click an object for more options, including moving it above or below other elements on the page.

6 Navigation and view controls Use these controls to zoom into and out of the page, as well as move between pages (and layers within a page).

First steps

On first launch, you'll be taken to the New Document window where you have a choice of four options: New Document, New from Template, Open Existing Document, and Open Recent Document. The New Document option enables you to create a single blank page or 'facing pages' (a 'spread', in publishing parlance).

To start using *Scribus*, switch to the 'New From Template' tab where you'll see a selection of templates, split into categories such as books and brochures. Start with a newsletter – pick Newsletter 2 and read through the 'About' description. It's a simple black and white A4 newsletter template where you right-click existing images and text to change it to your chosen content.

Click OK and you'll be warned the file was created in *Scribus 1.3.3* – click OK again. Next, if the Font Substitution box pops up with a warning telling you the document's fonts (Bitstream Vera family) are missing, then cancel opening the document, close *Scribus* and install the missing fonts via the Terminal:

```
$ sudo apt install ttf-bitstream-vera
```

Reopen *Scribus* and create the newsletter template again – it should now work.

Edit texts

The main *Scribus* window will open to show you the first page of your newsletter. It looks rather bare, so let's start by seeing how you edit existing components.

Double-click inside the 'Newsletter Title' text box, where you should now be able to edit the text in the usual manner. Replace this with the title of your newsletter, then select all the text and look to replace the font with something more striking.

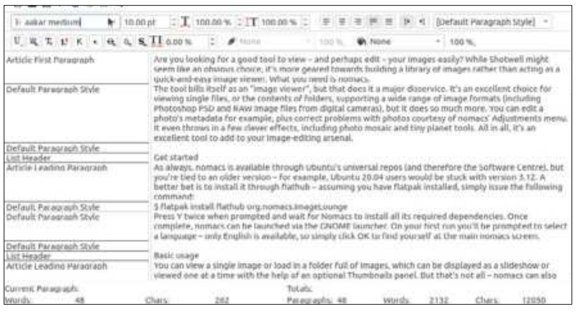
This is done using the Text Properties box, a floating window you can bring up via the Windows menu or by pressing F3. Click the font drop-down at the top to try an alternative font and size – see the box (below) for some hints and tips involving font selection.

The Text Properties box has more options too: line spacing will be relevant when formatting multi-column text to ensure it all lines up correctly. Anyone who's used DTP software should recognise most of the options on offer – including the tracking and word spacing controls under Advanced Settings – but for most people the basics should be sufficient.

Once you've formatted your newsletter text, you can move on to the rest of the text. When it comes to longer-formatted text, directly editing it in the *Scribus* window can be a little fiddly. Instead, click inside the text box and press Ctrl+T. This opens the Story Editor window with the current text on display inside. It's similar to your word processor, with one key difference – on the left you'll see each paragraph is assigned its own formatting style: Article First Paragraph, Article Following Paragraph, and so on.

Formating text

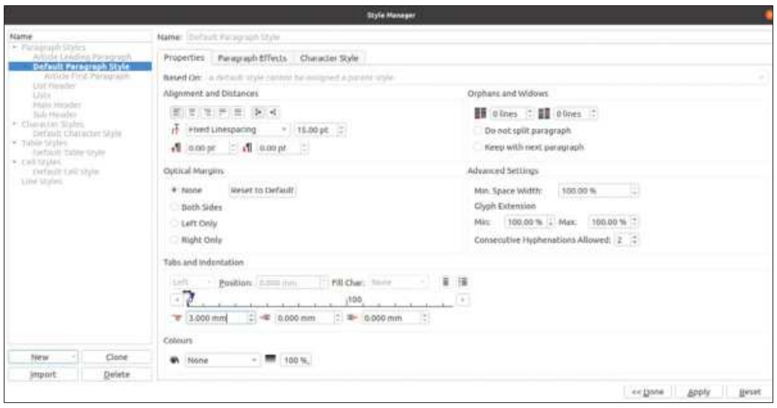
Consistent formatting is key to creating professional-looking documents, and this can be achieved by applying styles to entire paragraphs. You can change existing styles by right-clicking the current style displayed in the left-hand pane and then clicking the drop-down menu that pops up to select an alternative, but how do you edit these styles to fit your own design?



Article First Paragraph	Are you looking for a good tool to view – and perhaps edit – your images easily? While Shotwell might seem like an obvious choice, it's more geared towards building a library of images rather than acting as a quick-and-easy image viewer. What you need is nomacs.
Default Paragraph Style	The tool bills itself as an "image viewer", but that does it a major disservice. It's an excellent choice for viewing single files, or the contents of folders, supporting a wide range of image formats (including Photoshop PSD and RAW image files from digital cameras), but it does so much more. You can edit a photo's metadata for example, plus correct problems with photos courtesy of nomacs' Adjustments menu. It even throws in a few clever effects, including photo mosaic and tiny planet tools. All in all, it's an excellent tool to add to your image-editing arsenal.
Default Paragraph Style	
List Header	Get started
Article Leading Paragraph	As always, nomacs is available through Ubuntu's universal repos (and therefore the Software Centre), but you're tied to an older version – for example, Ubuntu 20.04 users would be stuck with version 3.12. A better bet is to install it through flathub – assuming you have flatpak installed, simply issue the following command:
Default Paragraph Style	$ flatpak install flathub org.nomacs.ImageLounge
Default Paragraph Style	Press Y twice when prompted and wait for Nomacs to install all its required dependencies. Once complete, nomacs can be launched via the GNOME launcher. On your first run you'll be prompted to select a language – only English is available, so simply click OK to find yourself at the main nomacs screen.
Default Paragraph Style	
List Header	Basic usage
Article Leading Paragraph	You can view a single image or load in a folder full of images, which can be displayed as a slideshow or viewed one at a time with the help of an optional Thumbnails panel. But that's not all – nomacs can also

Current Paragraph: Words: 48 Chars: 262
Totals: Paragraphs: 48 Words: 2132 Chars: 12050

Use the Story Editor to edit and style your text without having to worry about the layout, too.



Name: Default Paragraph Style
Based On: a default style cannot be assigned a parent style
Alignment and Distances: Fixed Linespacing 15.00 pt; 0.00 pt; 0.00 pt
Orphans and Widows: 0 lines; 0 lines; Do not split paragraph; Keep with next paragraph
Optical Margins: None (selected); Both Sides; Left Only; Right Only; Reset to Default
Advanced Settings: Min. Space Width: 100.00 %; Glyph Extension Min: 100.00 % Max: 100.00 %; Consecutive Hyphenations Allowed: 2
Tabs and Indentation: Left; Position: 0.000 mm; Fill Char: None; 3.000 mm; 0.000 mm; 0.000 mm
Colours: None; 100 %

Press F4 (or choose Edit>Styles) to open the Style Manager to define both paragraph and character styles. Paragraph styles can be defined from scratch, or you can save time by defining common character styles and applying those to your paragraph styles.

Start by selecting Default Character Style and clicking Edit. This is where you can define the default font that can be used to build your paragraph styles down to the smallest detail. Once done, click Apply – any existing text based on that character style will automatically update, revealing another advantage of styles: you can make wholesale changes without having to reformat everything from scratch.

You can edit existing styles and add new ones of your own. After choosing your default character style, you can then start to build out your paragraph styles. As you'd expect, these add extra elements such as line spacing, margins, tabs and indentation, and text alignment. The Paragraph Effects tab is where you can define bullet and number lists, as well as choose drop caps. The latter is a great tool to draw people's eye to the start of an article.

You'll also see a Character Style tab – you can select your previously defined character style here, then make adjustments (for example, to choose a different style and size for a header or other page element) or – if you're not bothering to use character styles – define this paragraph's character style from scratch.

To speed up the generation of paragraph styles, you can also base new styles on existing ones. They'll then inherit all the characteristics of that previous style, enabling you to concentrate on defining the characteristics unique to that specific style. You could, for example, define a default paragraph style, then use it

Use paragraph and character styles to ensure your text is formatted correctly and consistently.

QUICK TIP

Scribus offers a third type of frame: Render Frames. These make it possible to easily insert specialist typesetting, such as musical notation and scientific formulas, into your document with the aid of third-party programs.Visit https://wiki.scribus.net/canvas/Help:Manual_Renderframes for a guide to using it.

» DESIGN TIPS AND TRICKS

When it comes to designing a document in *Scribus*, one rule of thumb applies: less is more. This means keep things simple for your reader, and prioritising readability and clarity over all else. Start by keeping your page layouts minimal and clean, and ensuring page elements don't crowd each other out. Consider restricting the number of columns on the page to just two or three, and make sure the text flows logically between them.

Readability is key when picking fonts. Again, keep the number to a minimum by using variations from a single font family – bold, black, italics and so on – to style different elements such as headlines or picture captions.

Make sure the font family you choose matches your document's theme – Comic Sans is rarely the best choice for a work-related or professional document, for example. One place where you can look to be a little different is your logo or masthead, although again this should reflect your document's overall theme.

A key choice you'll need to make is deciding between serif (traditional, established) and sans serif (modern, clean) fonts. Which one you choose depends on your content, but if you're still struggling to choose, visit **https://bit.ly/lxf280-font-advice** to read some expert advice on the subject.

»

QUICK TIP

Save time recreating frequently used elements: group, then choose Windows> Scrapbook to open the Scrapbook window. Drag your element on to here to store it securely. Drag the element out of the window to create a copy to reuse.

to generate additional paragraph styles such as the first article in a paragraph, or a crosshead – a sub-heading within the text itself. Anything specifically defined within the new style overrides the style it's based on.

Working with images

Our newsletter template has space for a single image. To fill this with a new image, right-click it and choose Content>Get image (or press Ctrl+I). The image will be placed full-size within the picture frame, so will require adjusting. Right-click the image and expand the Image sub-menu where you'll be able to adjust the frame to the image or – more likely – resize the image to fit the frame (choose Adjust Image to Frame). This will resize the image to fit the frame, while respecting its aspect ratio, so there will likely be space below or to the side of the image depending on its aspect ratio.

Other options you'll see include Image Effects (add up to ten different effects, then tweak each one using the preview to see the effect they'll have), Edit Image (opens the image in the image editor you specify under File> Preferences>External Tools, which is set to *GIMP* by default), and Embed Image. Selecting this will convert the image from a link to an actual object within the file. You'd do this if you wanted to share the document with someone else and not have to worry about including all the images as separate files.

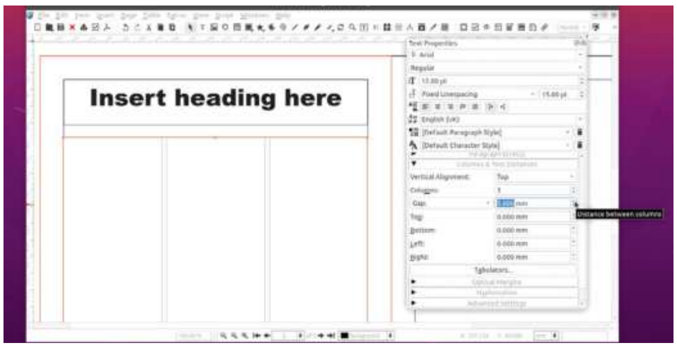
Understanding frames

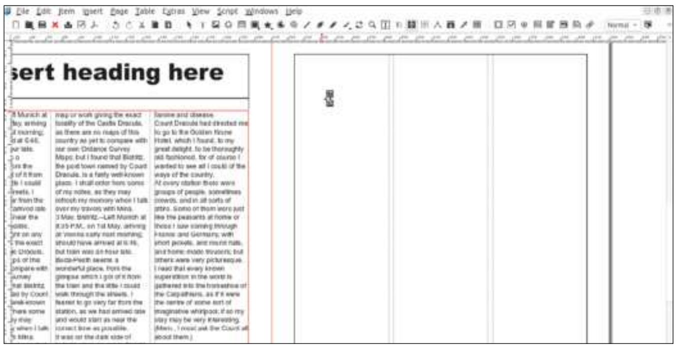
So far we've respected the newsletter's layout and focused on populating it with content. You'll notice that text and images reside in frames, and that these can be moved, resized, rotated and more – press F2 to open the Frame Properties window to access these options.

With images, we've already seen how we can resize them to fit inside their frames, leaving some white space beneath or to the side. Complete the job by right-clicking the image again, but this time choosing Image> Adjust Frame to Image. You'll see the frame now fit around the image, leaving white space beneath it.

The next step is to move the caption frame so it sits underneath the image again. By default, the template

WORKING WITH FRAMES



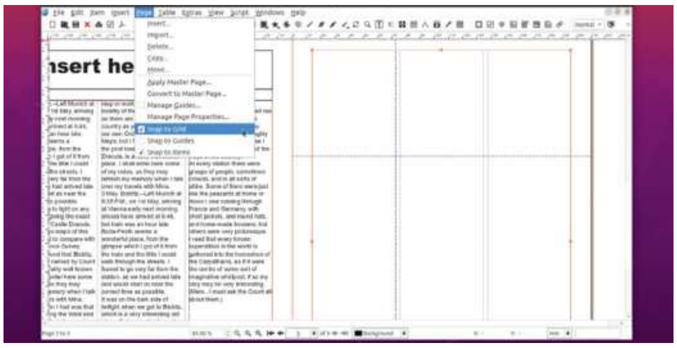


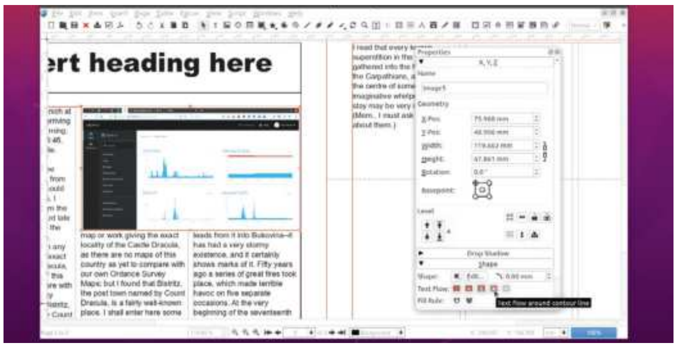
1 Set up a text frame

After opening a blank document, click the text frame button, then click and drag on the page where you'd like the frame to appear. This can then be resized and moved as required. Press F3 to bring up the Text Properties box to set other attributes including text columns (under Columns & Text Distances). Be sure to set a gap between columns as shown.

2 Link text frames

If you want to flow text between different frames and/or pages, you need to create links between them. Select your first frame and press N – you'll see the cursor change. Simply place this over your second text frame and click to link them. Repeat the process for any other frames. You can unlink frames by pressing the U key instead.





3 Make use of guides

To ensure frames line up correctly, create guides to help – click and drag from the vertical or horizontal rulers to the left and top of the page respectively to create them. As you click and drag, you'll see their exact X or Y position is shown to help you position them precisely. Choose Page>Snap to Guides to help align frames more easily to guides.

4 Layer frames

Frames can overlap or be placed one on top of another thanks to *Scribus's* support for layers. Right-click a layer and choose Level to move it up, down or to the top or bottom of the current pile. This affects how content in other frames is displayed – for example, to flow text around an image, place the image at the top, press F2 and set the flow using the Shape tab.

has locked all its frames so they can't be accidentally moved. To fix this, right-click the frame and choose Locking>Is Locked. You'll see the frame change to reveal drag handles – click and drag it to a new location, or use the arrow keys to nudge it around the screen.

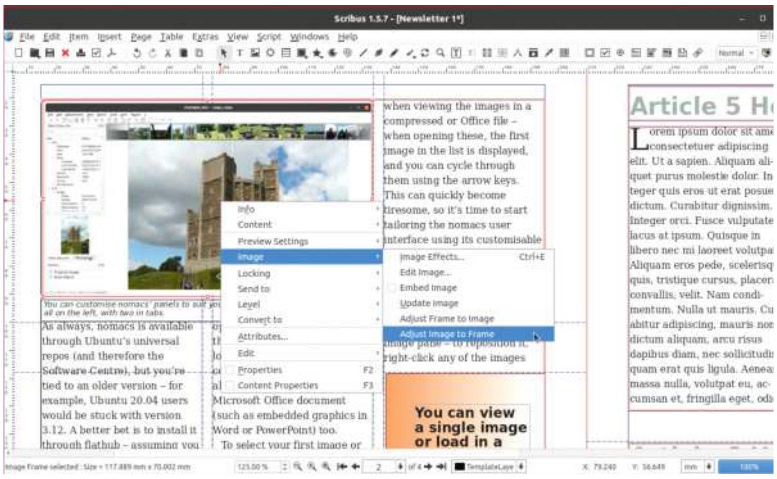
Now's the perfect time to discover how you can group frames together, so they act as a single object. Doing so would enable you to move the image and its caption as one: just hold the Shift key as you click each frame you'd like to group and you'll see the selection box encompasses all selected frames. You could drag this to a new location, or choose Item>Grouping>Group to convert them into a single object (you can undo this from the same sub-menu, choosing Ungroup instead).

Next, drag both the headline and the body text frames beneath so they sit underneath your image (select both, then unlock them and finally move them into position). You now have space left over beneath the text, so select the body text frame to click and drag the bottom of the frame down to fill the space. You'll end up with room with more text to fill too.

Add more pages

Once you've filled your first page, move on to the next. Each template can support multiple page layouts, and our newsletter is no exception. Choose Page>Insert. You can insert one or more pages after the current page and choose different master pages for both left and right-facing pages. For the purposes of producing a four-page newsletter, insert three pages, choosing InnerPageLeft and InnerPageRight as your page templates. Click OK.

Three new pages will be created using the layouts you specified, ready for you to populate them with content. This is all very straightforward, but only if you're happy to work within the confines of the newsletter template itself. In reality, you'll want to make changes to this layout following the frame tweaks we've discussed, or look to make bigger changes by designing pages yourself – the walkthrough (opposite) reveals how to create text and image frames from scratch.



Scribus makes it easy to fit imported images into an image frame. It's also possible to embed the image directly into the document.

Once you've started to master page design, you'll want to incorporate those layouts in your own custom templates. The box (below) reveals how to build a template using a series of master pages, which are used to pre-populate pages with non-editable content.

Share your creation

Your work is complete, and you're ready to print a physical copy or export it as a PDF to share digitally. The best way to do this is via File>Print Preview or File>Output Preview (for PDFs). A Preflight Verifier window will pop up listing all known errors – including those frames where there's too much text to fit. Click an item to jump to it in the document, check over the problem and correct it with some editing if necessary, then click Check again.

Alternatively, click Ignore Errors to see how the document will look – you'll see various additional options depending on whether you're printing or exporting to PDF – and finally click Print or Export... to complete the job. LXF

QUICK TIP

Want to add page numbers to your document? Create a text frame in the usual way, then choose Insert>Character>Page Number or press Ctrl+Shift+Alt+P. The inserted page number will update automatically if the page is moved around.

» BUILD A TEMPLATE

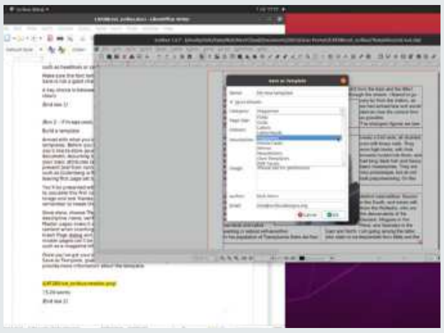
Armed with what you've learned in this tutorial, you can now create your own templates. Before you begin, choose File>Preferences>Paths to specify where you'd like to store your templates. Once done, click OK and create a new blank document.

Assuming it's a multi-page document, select Facing Pages, choose your basic attributes (size and orientation) and use the Margin Guides tab to prevent text from running to the edge of the page (use one of the Preset Layouts such as Gutenberg or Magazine if you're not sure). Choose 3 or 4 for your pages, leaving first page set to Right Page to accommodate a cover. Click OK.

You'll be presented with a blank canvas. Follow the walkthrough (facing page) to discover how to populate this first cover page with your own design using a combination of image and text frames. You can then repeat the process for the inside pages – remember to tweak the design for left and right pages.

Once done, choose Page>Convert to Master Page. Give it a suitably descriptive name, verify the right type (left or right) is selected and click OK. Master pages make it straightforward to pre-populate pages with previously created content when inserting new pages. Just select the chosen master page from the Insert Page dialog and *Scribus* will do the rest. Note that content added to master pages can't be edited or removed, so use them for consistent elements, such as a magazine title or page numbering.

Once you've got your pages and master pages in place, choose File>Save as Template, give it a suitable name and optionally tick More Details to provide more information about the template.



You can record details about your new template – perfect if you plan to share it with others.

ESCAPE GOOGLE PHOTOS

Google's free photo storage is a thing of the past. **Jonni Bidwell** has some free software to get your albums in order.

When invitations to Google's new mail service found themselves in the hands of the great unwashed, the world of free email changed forever.

No longer did users have to make do with a handful of megabytes of storage, obnoxious adverts and the terrible fear that at any point a connectivity failure would dash the missive you were currently working on into oblivion. Now users could enjoy 2GB of storage, and further enjoy watching that quota rise, ever so slowly, in real time. Yes, there were adverts, but they were small and textual. They were also relevant because they were chosen by algorithms that fed off the contents of your email.

Later, Google would add free, unlimited storage for all your photos and later still, free storage for up to 50,000 audio files. A decade later, however, Google discovered that even with their impressive resources, this was untenable – or perhaps a potential avenue for profit. Google Play Music shut down at the end of 2020, and from June 2021, users of Google Photos (née Picassa) saw new photos count towards their 15GB storage quota.

At the time of writing, the relevant support page warns that accounts exceeding their quota for 24 months face potential file deletion. So if the aggregate of your Google Drive and Gmail inbox is already approaching that, then perhaps it's time to take action.

There are all manner of other free storage options, which we'll cover in this exhaustive feature. But there's also no shortage of open-source solutions that you can self-host, either at home (for free) or in the cloud (for cheap). We'll look at two of these: *Nextcloud* and *Lychee*. The former can, of course, do much more than just photos (it can be a full-blown groupware suite) while the latter is a lightweight solution for hosting photos and hosting them well.

Both of these work can be used locally or remotely, whether on a rented VPS in the cloud or a Raspberry Pi under your desk. And if you just want to pay someone to take care of your photos, then we'll look at some of those services, too.

Free your photos

Whether at home or in the cloud, your photos deserve a new host.

Let's not catastrophise – no one's going to lose any photos for a couple of years. But perhaps it's a timely reminder that things freely given can be freely taken away.

We should note that photos uploaded in their original quality have always counted towards the 15GB Google quota (and will continue to do so). The new rules concern images uploaded using High and Express quality compression. So serious photographers are unlikely to be bothered by this. But for millions of users, it's yet another iteration of a Google service becoming less useful. Of course, if you want to pay for more storage then Google One will be happy to take your money (100GB of storage will cost you £15.99 per year).

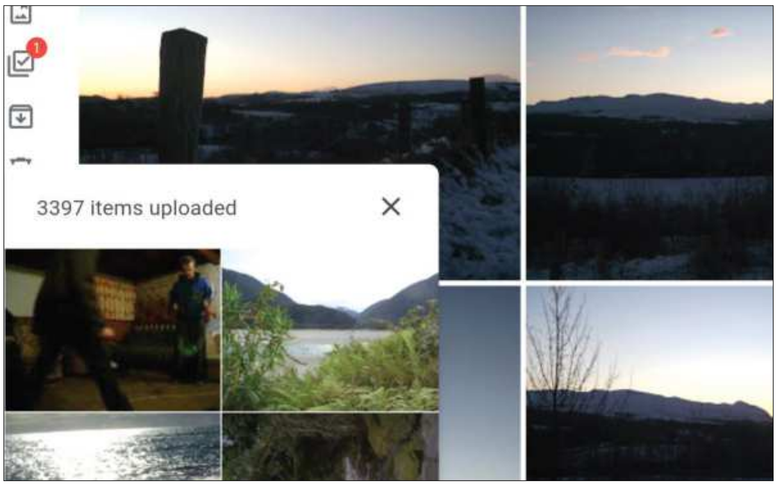
If you're willing to part with cash, then there are much more privacy-conscious, less Google-ey photo storage offerings. However, if you want to keep costs at zero, and don't fancy partitioning your massive photo collection across several different providers (or Google accounts), keeping each collection carefully within the free storage tier, then you may want to consider abandoning cloud storage altogether. Or more precisely, run your own personal cloud at home.

Taking the cloud option

We're all for self-hosted solutions here and it'll come as no surprise that we'll be looking at *Nextcloud* over the page. It's easy to set up on your desktop or home server, and it has superb photo management facilities. If you like, and are prepared to do battle with your router's port forwarding configuration, you can make this remotely accessible and enjoy all the utility of Google Photos without worrying about quotas. Or at least if you run out of space you can always add more.

Self-hosting comes with a pretty strong caveat though: you're responsible for your data. So you should always back up, and if you're serious about backups, then you should back up off-site, too. At which point you might start looking at cloud solutions again. Instead of falling into an infinite loop at this juncture, note that the hybrid approach is a reasonable one. Cloud storage and local storage can be synchronised to mutually mirror one another, and thanks to the *Nextcloud Photos* mobile app, it's easy to ensure new images from your phone are automatically added into the mix.

Virtual private servers (VPSes) with 20GB of storage can be rented for around £5/month. You can even rent a Raspberry Pi from Mythic Beasts at this price. This offers excellent value, since besides hosting your photos, these offerings give you root access, enabling you to burden the machine with whatever duties your imagination allows (subject to resource constraints). We've looked at setting up VPSes before, and we'll cover it again soon. But for now let's get on with the show.



This feature was originally written back when you could upload 20GB of photography to Google for free.

» KEEPING IT SIMPLE

Some people, preferring to avoid unnecessary thumbnailing, cataloguing, configuring and hosting complications (and PHP applications), choose to manage their photos as regular files. Some people feel the same way about their music collections, and keep their OGGs and FLACs in directory hierarchies according to tempo, chronology, genre or whatever else suits their purpose. If this sounds like you and already have your photos as you like them, but are looking for a bloat-free back-up solution to SSH your photos somewhere safe, then *Duplicity* is worth a closer look. It works incrementally, so once the first backup is done, only new data is stored, meaning subsequent runs are much quicker. It can optionally encrypt data as well for added privacy (and we'll look at this with *Nextcloud* too).

Duplicity stores backups as archives though, so you don't have the immediate access to files as you would if they were stored regularly. So you might also consider *SyncThing*, which will happily mirror your photo, video and whatever else files wherever you like. Or you could be Puritan hacker about it and have your Devuan box run a *Cron* job that *rsyncs* your local photo store to a secure server at an undisclosed location.

These things can be brought down by a stray '/' though, so always check your backups are doing what they're meant to be through manual inspection.

Install Nextcloud

Discover how you can tap into the power of Docker Compose to get Nextcloud up and running in a jiffy just like the professionals.

Whenever our perennial "Escape Google (et al)" feature graces pages, we're always quick to point out the benefits of *Nextcloud*. It can handle storage (like Google Drive), chat (like *Zoom*) and, with *Collabora Online*, even be a full-blown online office suite. As you may have guessed it can handle your photos too, and we're going to show you how to get the most out of that feature.

It's entirely possible to set up *Nextcloud* at home and access it remotely, but there are a number of reasons to consider running it in the cloud (not just because it's in the name). First, your ISP might let you down at a critical time (imagine trying to show off your photos at Tarquin's dinner party only to be met with a timeout and the cold stares of your peers). Cloud providers (good ones at least) take the flow of packets very seriously.

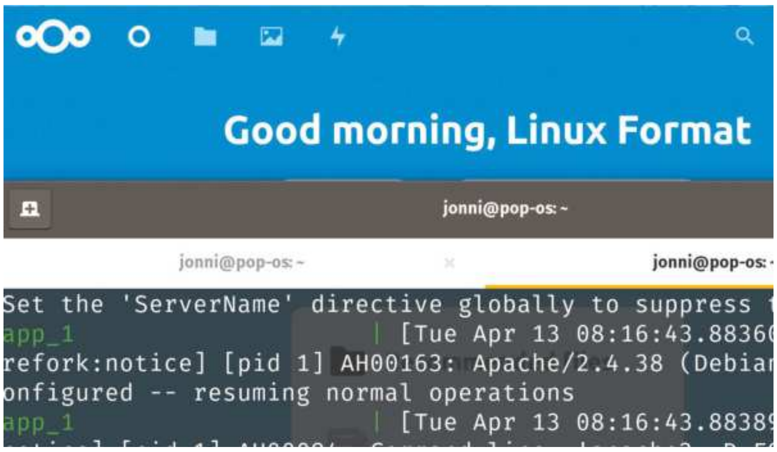
RUN NEXTCLOUD IN THE CLOUD

"It's just nice to have a cloud server sometimes, because you'll never know when they'll come in handy."

Next, you don't have to worry about your hardware breaking, or losing your photos if your house burns down. Many providers offer daily backups for an additional fee (as well as having their own redundancy measures to protect against hardware failure on their side), affording you some peace of mind and covering against your own errors too. Finally, it's just nice to have a cloud server sometimes, because you'll never know when they'll come in handy.

Setting up *Nextcloud* from scratch is a slightly protracted but well-documented affair (we covered doing it on the Pi in **LXF272's** smart home office feature). Most of the effort expended is in laying the groundwork – more precisely, installing a web server, database and PHP and getting them all talking to each other. Once that LAMP stack is in place it's all largely plug and play.

Be that as it may, we don't much care for repetition here so this time we'll look at installing *Nextcloud* using *Docker*. This comes with the advantage that your installation will always be kept up to date and you won't have to worry about maintaining a LAMP stack (or setting it up in an overly permissive manner). If you're running Ubuntu (Desktop, Server or Pi) or otherwise, then make sure you have the Snap daemon installed. You could equally well use the Snap image that brings the same benefits, and means you can skip this next section where we install *Docker*. Otherwise SSH to your server (if your installing remotely) and read on.


If you run Docker Compose without the -d switch, you can see some debugging info as the containers spring into life.

```
Set the 'ServerName' directive globally to suppress
app_1                        | [Tue Apr 13 08:16:43.8836
refork:notice] [pid 1] AH00163: Apache/2.4.38 (Debia
onfigured -- resuming normal operations
app_1                        | [Tue Apr 13 08:16:43.8838
```

Get Docker up and running

The easiest method of installing *Docker* is to use the official convenience script (it works on Debian, Raspberry Pi, Fedora and more):

```
$ wget https://get.docker.com -O get-docker.sh
```

Before running this as root, it would be prudent to cast your eyes over the script so you know what it's doing. Once you're satisfied, run it with:

```
$ sudo sh get-docker.sh
```

This works on most distributions, and adds the appropriate repositories so that it's kept up to date as part of the regular package management process. The *Docker Engine* should start automatically after install on Debian and Ubuntu, but Fedora users will need to start it and enable it manually with:

```
$ sudo systemctl enable –now docker
```

At this point you could run:

```
$ sudo docker run -d -p 8080:80 nextcloud
```

and find *Nextcloud* by pointing your browser at port 8080 on that machine,for example **http://localhost:8080**. Using the *Docker* volume directly like this is fine for testing locally, but since the image doesn't provide HTTPS it's not smart to use this on a remote server, since traffic could be intercepted. In fact, *Nextcloud* will do everything in its power to stop you using it this way.

We'd recommend reading the documentation at **https://github.com/nextcloud/docker/tree/master/.examples** that covers using *Docker Compose* to link the *Nextcloud* container (running Apache) to an Nginx reverse proxy, a Let's Encrypt certificate updater and a MariaDB database (which will cope much better with large photo collections than SQLite).

In order to use HTTPS (or use it without generating warnings) you need a domain name. You can get a free subdomain (of the form **linuxformat.duckdns.org**) from

DuckDNS or any number of other Dynamic DNS providers, as well as a script. This will mean that if you're running *Nextcloud* at home and your IP address changes, your subdomain record is updated.

Check the latest version of *Docker Compose* at the releases page (**https://github.com/docker/compose/releases**) and if necessary change the version number below to grab it:

```
$ sudo wget https://github.com/docker/compose/releases/download/1.29.0/docker-compose-Linux-x86_64 -O /usr/local/bin/docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose
```

The second command makes the script executable, and so long as **/usr/local/bin** is in your **$PATH**, the system will be able to run it. We're going to copy the example Dockerfiles from the official pages, and since they span two directories we'll need to match that.

```
$ mkdir -p ~/docker-nextcloud/proxy
$ cd ~/docker-nextcloud/
$ wget https://github.com/nextcloud/docker/raw/master/.examples/docker-compose/with-nginx-proxy/mariadb/apache/docker-compose.yml
```

We need to modify the reverse proxy container to allow large uploads. The example also disables request buffering, so let's do that too. Create a file **~/docker-nextcloud/proxy/uploadsize.conf** with these contents:

```
client_max_body_size 10G;
proxy_request_buffering off;
```

Then in the same directory create a file named **Dockerfile** to source this:

```
FROM jwilder/nginx-proxy:alpine
COPY uploadsize.conf /etc/nginx/conf.d/uploadsize.conf
```

Before we can compose all these images, we need to edit the YAML file with the hostname of the machine running *Docker* and, optionally, an email address for Let's Encrypt. Open the **docker-compose.yml** file we downloaded earlier, populate the **MYSQL_ROOT_PASSWORD** variable with a strong password and set the **VIRTUAL_HOST** and **LETSENCRYPT_HOST** variables to, say, `linuxformat.duckdns.org`. There's one more final tweak to this file since the Let's Encrypt companion now requires an additional volume for ACME data. So add

```
   - acme:/etc/acme.sh
```

to the Let's Encrypt `volumes:` section and also add the `acme` volume to the list at the end of the file.

Now, in the same directory, create a file **db.env** with the contents, ideally using a stronger password than us (this one is for the user, rather than the root, account on the database).

```
MYSQL_PASSWORD=password1
MYSQL_DATABASE=nextcloud
MYSQL_USER=nextcloud
```

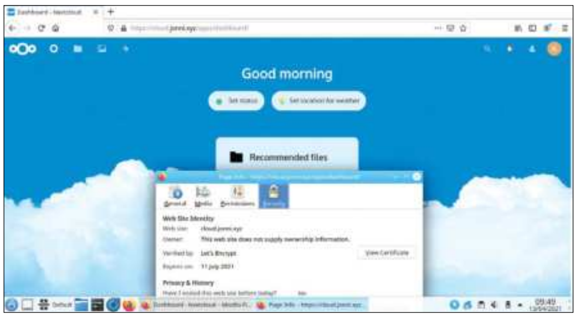
Finally, we should be able to grab the latest images and orchestrate everything into action with

```
$ cd ~/docker-nextcloud/
$ sudo docker-compose build –pull
$ sudo docker-compose up -d
```

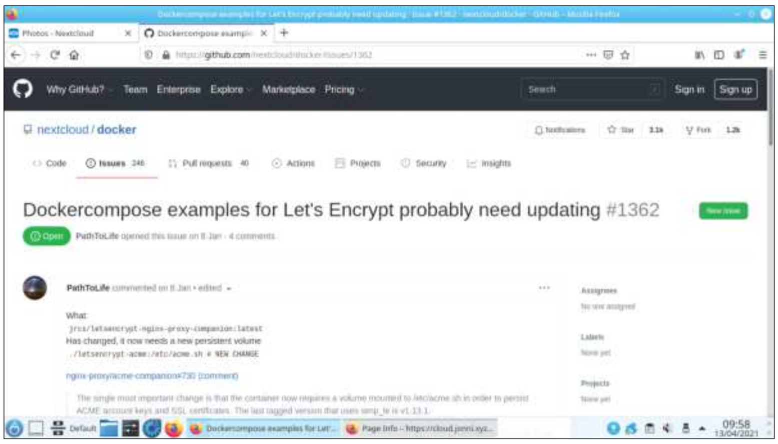
You should find the *Nextcloud* setup screen by pointing your browser to your host, but don't be too quick – it takes a few seconds to set up everything. Follow the instructions there to set an administrator password, and if you're feeling Spartan uncheck the box that installs collaborative tools.

Note that the database and SSL certificate are all automagically preconfigured. As an aside, getting *Docker* working on an IPv6 only network was an altogether hellish experience when we tried it, so if you figure out how that works please do tell us.

Over the page, we'll set up users and finally get to some photo management.



Nextcloud's friendly Welcome screen – note that the SSL certificate was generated automatically.



Dockercompose examples for Let's Encrypt probably need updating #1362

PathToLife commented

What:
```
jrcs/letsencrypt-nginx-proxy-companion:latest
```
Has changed, it now needs a new persistent volume
```
./letsencrypt-acme:/etc/acme.sh # NEW CHANGE
```

This problem with the Compose example almost outfoxed us. If you're reading from the future then it'll probably be fixed.

» RASPBERRY Pi REVISIONS

If you try the above on a Pi you'll find *Docker* works but *Compose* doesn't. We downloaded an x86 docker-compose binary, when the Pi needs an ARM one. And sadly there's no official ARM builds yet. To remedy this, you can compile *Compose* yourself, or (much easier) install the community-composed *Compose* through Pip, the Python module installer:

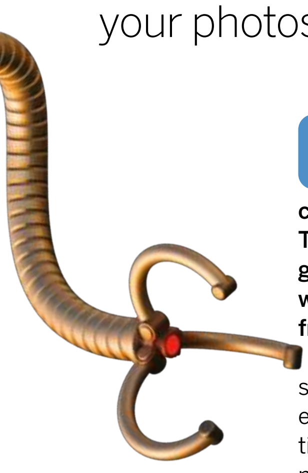
```
$ sudo apt install python3-pip
$ sudo rm  /usr/local/bin/docker-compose
```

You'll also find there's no armhf build of the official MariaDB image, but that's okay because the generous humans at **linuxconfig.org** provide one, which you can avail yourself of by replacing `mariadb` with `linuxconfig.org/mariadb` in the **docker-compose.yml**.

At the time of writing, there's a slightly bizarre *Docker* problem on a few Pi distros (including those based on Debian Buster such as the official Raspberry Pi OS) involving an outdated libseccomp, which the *Docker* Engine depends on. Symptoms of this include dates in containers being set to 1970 (the Unix epoch) and all kinds of other things not working. A good solution is to install from the Buster Backports repository, as described at **https://docs.linuxserver.io/faq**.

Transfer your photos

Now that Nextcloud is installed and operational, it's time to get your photos into it, and out of reaching distance from Google.

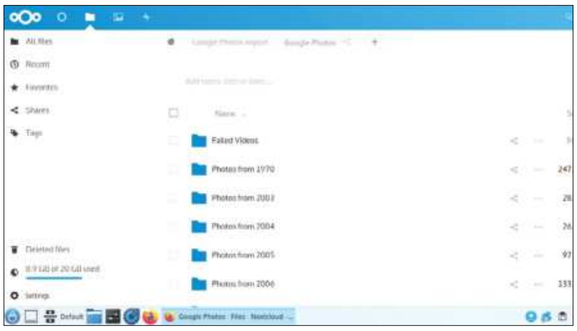


See the three-step guide below to get an archive of your Google Photos. The archive is organised by year and includes metadata corresponding to each image in the form of .json files. These consist of album information, descriptions and geodata that have been added via Google photos, as well as EXIF data from the original images stripped from their downloaded counterparts.

Extract the archive, taking care to ensure you have sufficient space. Note that if it's a huge file and you're extracting it to slow, or network, storage, it'll take a long time to extract. So either make yourself a cup of tea or make enough space on the fastest storage you have to hand. You still have the original archive, so you can safely delete these once they're uploaded.

Alternatively, you can download images in batches (of up to 500) or by album from Google Photos without using the Takeout service. This method seems to retain the original images' metadata. If you click the More Info button at step two below, you'll find the option to transfer photos to Flickr or Onedrive. There are some other commercial alternatives listed in the box overleaf. But why trade one proprietary service for another when our *Nextcloud* container is eager to contain some photos?

Before we do that though, it's good practice to avoid using the administrator account for your data. Instead, create a user account by logging in and finding the Users option from the menu in the top-right. Click the New user in the top-left, fill in at least a username and password, and click the tick. You'll be prompted to confirm with the admin password. Then log out of the admin account and log back in under your newly created persona, or repeat this process if you're planning to allow other users onto your instance.

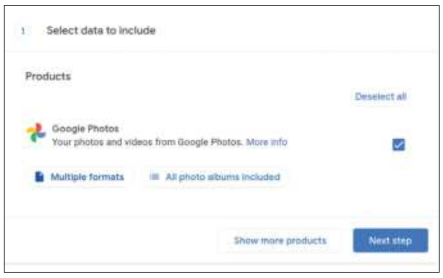


Linux Format filled up its corporate Google Drive with server backups – don't let them fill up your Nextcloud, too.

Time-saving tips

Nextcloud's Photos tool is included in a default installation and doesn't require any further configuration. Any images you upload to your *Nextcloud* instance will automatically be catalogued by date. A *Photos* folder containing some sample images is included in every *Nextcloud* user's account, but you may as well create a new folder to rehouse the images you liberated from Google (or wherever else you're uploading from). Go to

GETTING YOUR PHOTOS OUT OF GOOGLE



1 Select data to include

Products

Deselect all

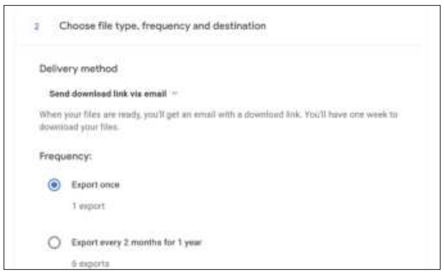
Google Photos
Your photos and videos from Google Photos. More info ✓

Multiple formats | All photo albums included

Show more products | Next step

1 Visit the Takeout

Log in to Google Photos and then go to the Settings page by clicking the cog icon in the top left. Scroll down to the Export your Data section, and then click it to reveal the option to back up an archive of your data. Click Backup to open Google Takeout. It's not an eatery, unfortunately, but rather Google's tool for extracting your data from its services.



2 Choose file type, frequency and destination

Delivery method

Send download link via email

When your files are ready, you'll get an email with a download link. You'll have one week to download your files.

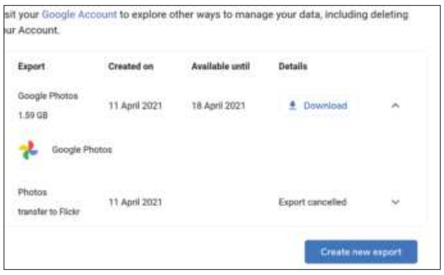
Frequency:

[✓ Export once] 1 export

Export every 2 months for 1 year — 6 exports

2 Customise

You can restrict the archive to certain years and, if you like, download data from other Google Services. Takeout offers a choice of .zip or .tar.gz archives, and if you like to be different you'll probably choose the latter. The archive, which you can choose to split into same-sized chunks, will then be prepared by Google robots.



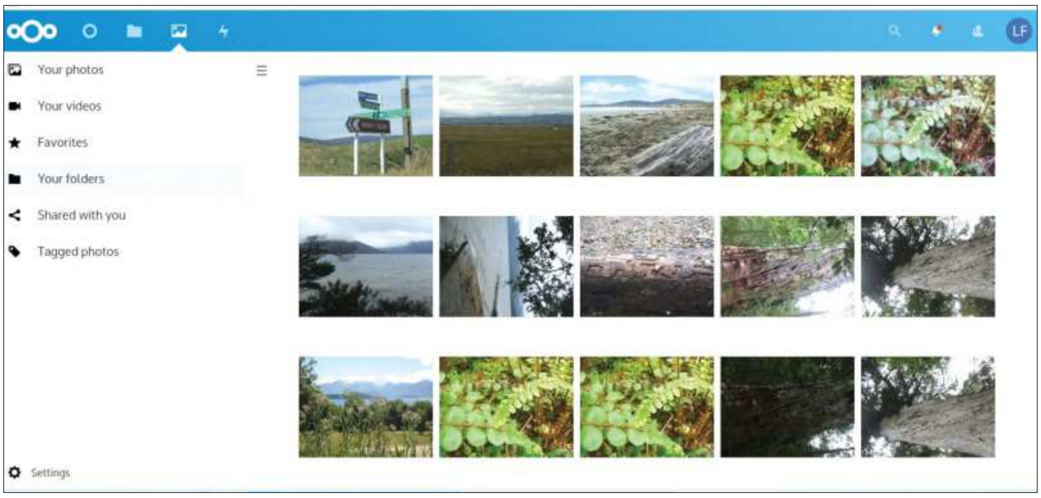
sit your Google Account to explore other ways to manage your data, including deleting ur Account.

Export	Created on	Available until	Details
Google Photos 1.59 GB	11 April 2021	18 April 2021	Download
Photos transfer to Flickr	11 April 2021		Export cancelled

Create new export

3 Download

Google will give you a link that's valid for a week. If you like, you can opt to download over the course of a year, getting a new link every two months. Hit the Export button and be appalled that it will take Google "hours or days" to prepare your export. You'll get an email when it's ready, so in the meantime you might want to make some space.



Once your photos are uploaded, you are free to fall into a nostalgic image-scrolling reverie.

Nextcloud's Files tool and click the + to do this. If you're feeling organised, you might want to do some photo housekeeping before you upload, for example adding directories according to trips or lockdowns, repositioning any misdated images (and correcting their EXIF data if you're really keen). Otherwise we already have them arranged by year so you could at this point try and drag the whole Google Photos folder (on your local machine) into the newly created *Nextcloud* folder (in the browser). *Nextcloud* threw a handful of scary-looking "deadlock" warnings, but despite having over a gigabyte of files dropped on it the progress bar steadfastly made its way to 100 per cent.

Android uploads

The official *Nextcloud Android* tool is available from the F-Droid store, as well as Google's and Apple's own shopfronts. Just point it to your *Nextcloud* instance and enter your credentials to get instant access to all your photos (and other files). You can browse any images uploaded there by date from the Media section, which is easier than squinting at thumbnails.

In the Auto upload section in the Settings page (available from the hamburger (☰) menu) you can opt to synchronise any media folders on your device, including Screenshots, WhatsApp Images and the excruciatingly titled Camera Roll. If you do this then you'll never need to worry about manually transferring photos off your phone again. They'll all be beamed up to your Nextcloud as soon as you're in reach of a network connection.

Well, that's the theory anyway. A few things could go wrong. If your Nextcloud instance runs out of disk space then it'll probably stop working. If things go really wrong, then you might not be able to log in to the host to free up space, since logging in requires (or at least used to require) a tiny quark of disk space. To avoid this you can use the admin account to set space quotas for each user. If you're the only user then set this to just below how much space is actually available, so that there's some wiggle-room if the quota is reached. By default the tool will only upload on unmetered Wi-Fi connections, but you can change this if you want your photos in the cloud sooner.

It's not so easy to automate uploads from regular digital cameras, but seasoned photographers will be used to the copying-from-the-SD-card-and-uploading drill. They might also be interested in the RAW plugin for *Nextcloud* that enables RAW files to be previewed directly in your browser, rather than having to download them and open them with *GIMP*.

There's plenty of things you can do with your photos once they're uploaded. First, you'll enjoy just browsing through them. The *Photos* tool lists them by upload date initially, and since they were all uploaded on the same day and probably not in chronological order, that may not much use. That's okay though, just choose the Your Folders option and at least they'll be organised according to the directory structure Google provided. Thumbnails are generated on the fly, so the first time you browse a new set of images there will be a slight delay in rendering them. You can also add comments and even tag fellow users of your instance using the ubiquitous @ mentions. You can also share images internally and externally (with users that don't have an account on your instance). And of course, there's a button at the top to start a slideshow.

» MANAGE CONTAINERS USING PORTAINER

What's great about this setup, although in hindsight maybe it's easier to use the Snap, is that it's highly extensible. You can host websites in other containers, and have the Let's Encrypt container take care of their certificates and the reverse proxy route traffic to them. Even if you don't do that, you could do a lot worse than installing the wonderful *Portainer* to manage the four containers we've just set up. This way you can make sure they're all in rude health from the comfort of a friendly web interface.

Portainer is easy to install. It ships as a Docker image that you can run locally with the following:

```
$ sudo docker run -d -p 9000:9000
--name=portainer --restart=always -v /
var/run/docker.sock:/var/run/
docker.sock -v portainer_data:/data
portainer/portainer-ce
```

Recent releases can also handle *Compose v3* files (like the ones we made earlier) natively, so you can paste their contents directly in the Stacks area, which also supports Kubernetes and Docker Swarm. From *Portainer* you can examine container logs, inspect properties or even run shell commands directly in running containers.

Portainer also makes it easy to tidy up after oneself. Containers and volumes started in an ad-hoc manner (as you do when you're trying to get things working) continue to hang around long after they've stopped being useful. These can be quite large so it's good to look for unused ones (helpfully labelled as such) and put them out of their misery.

»

Let's look at Lychee

Get simply beautiful photo hosting with a tropical, fruity aftertaste.

L*ychee* is a much simpler program than *Nextcloud*. It only enables you to upload photos into galleries and displays them, but it does so very well. A VPS with 1GB of RAM will occasionally run out of memory when doing complex things with *Nextcloud*. It happened to us over the course of this feature (it's also quite common during the update process). Most of the time, this is easy to recover from by restarting the MariaDB service, but it's still annoying when you refresh the interface only to see an unhelpful HTTP 500 error.

Lychee by comparison is much more lightweight, so you should be able to run it on even the most spartan of VPSes. It's also designed with photographers in mind, with support for viewing EXIF and IPTC metadata. It's good for social types too, with easy sharing to social media platforms and clear indications when you choose to make photos public. *Lychee* also enables you to upload from Dropbox.

Like *Nextcloud*, *Lychee* requires a LAMP stack to run, and since we clearly don't have enough pages to cover setting one of those up we'll cheat and use Docker again. Not only that, but we'll adapt the Docker Compose configuration we used for *Nextcloud*.

If you've got the hang of *Compose* then you might be tempted to add another service for *Lychee* to the existing file to make a conjoined *Lychee/Nextcloud* stack. This almost, but not quite, works. The MariaDB container only allows one additional account to be set up by environment variables.

So you have three choices: manually add an account for *Lychee* to the `db` container (which is bad because this won't persist if the container is destroyed), add a second database container for *Lychee* (which seems inefficient), or just forget about *Nextcloud* for now and create a solo *Lychee* Docker setup.

CONFIGURE ACCESS TO LYCHEE

"If you also want friends to be able to log in then it's best to keep the admin account for administrative functions."

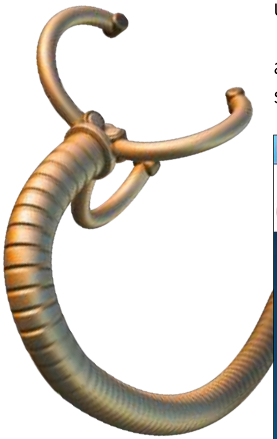
We'll go with the latter

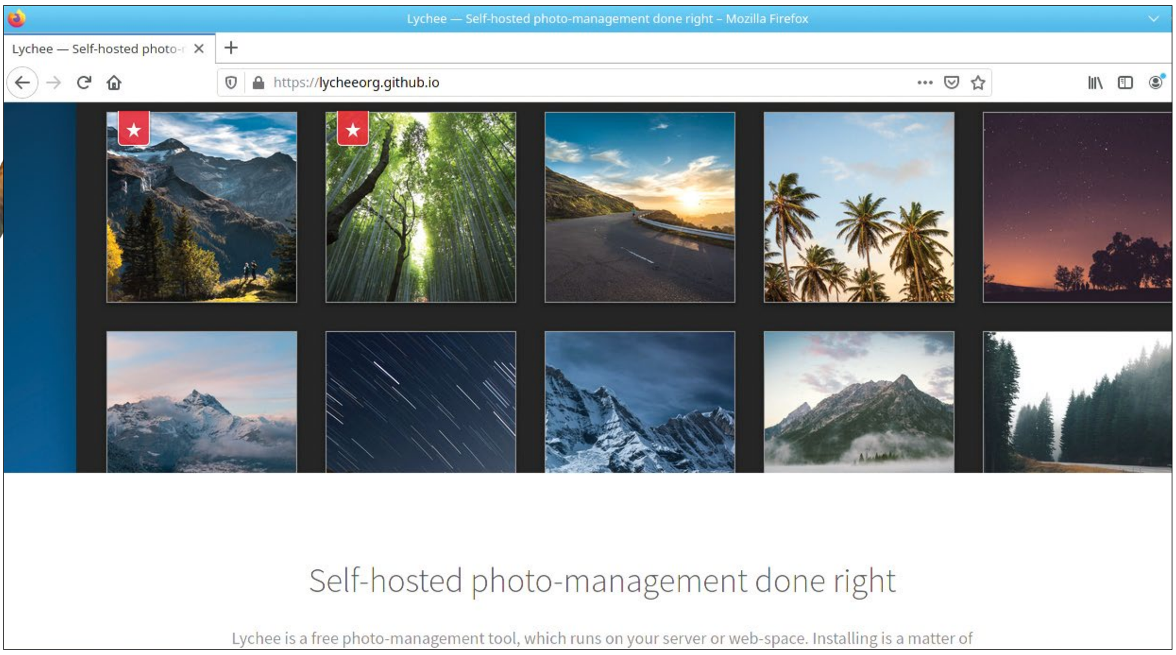
On your Docker host create a new directory for *Lychee*, and copy the files from the *Nextcloud* set up:

```
$ mkdir ~/docker-lychee
$ cp -r ~/docker-nextcloud/* ~/docker-lychee/
```

Now edit **~/docker-lychee** and replace the `app:` service (which defined the *Nextcloud* container) and replace it with this block:

```
 lychee:
  image: lycheeorg/lychee
  restart: unless-stopped
  volumes:
   - lychee_conf:/conf
   - lychee_sym:/sym
   - lychee_upload:/uploads
  environment:
   - VIRTUAL_HOST=lychee.duckdns.org
   - LETSENCRYPT_HOST=lychee.duckdns.org
```




Lychee — Self-hosted photo-management done right – Mozilla Firefox
Lychee — Self-hosted photo-
https://lycheeorg.github.io

Self-hosted photo-management done right

Lychee is a free photo-management tool, which runs on your server or web-space. Installing is a matter of

Lychee's web page says it all really.

```
    - LETSENCRYPT_EMAIL=contact@mymagazine.
com
    - DB_CONNECTION=mysql
    - DB_HOST=db
```

As before, if the hostname does really resolve to your Docker host, then certificates will be automatically generated. Before we get to starting it up, don't forget to edit the **db.env** file, changing the DB_DATABASE variable to something relevant like `lychee`. As before, we could just `docker run` the *Lychee* image, but we'd be stuck with an SQLite database and no encryption, so it wouldn't really be fun. Where the *Compose* file really comes into its own (though we won't see it here) is when you specify multiple webapps on multiple domains and watch them all spring into life with no further configuration required.

And speaking of configurating, let's launch *Lychee* by running the following code

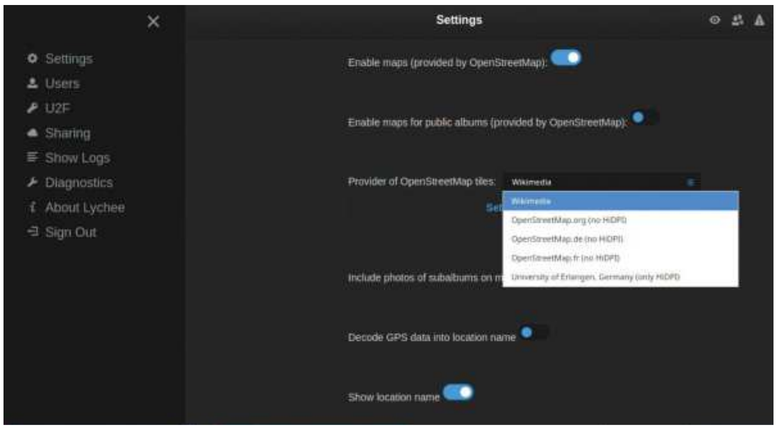
```
$ sudo docker-compose up -d
```

from our **docker-lychee/** directory. After a couple of seconds you should find it running at the hostname you specified, and asking you to set up a username and password for the admin account. Once you're logged in there's not much to see – just the four default albums on a rather empty-looking black background. Before you start filling this void consider, again, creating a user account. If you're planning to be the only *Lychee* user this doesn't matter so much, but if you also want all your friends to be able to log in then it's best to keep the admin account for administrative functions only. You'll find the option to create a new user by clicking the cog icon at the top-left and selecting Users.

Make sure that you check the easy-to-miss Allow Uploads button, otherwise that blank screen isn't going to get less blank any time soon. While you're here have a look at the Diagnostics and Show Logs buttons. You might see a warning about uploading large photos. The default upload limit seems to be set to 20MB, so most users should be fine.

Sign out of the admin account and notice that you'll be presented with an empty, public gallery. Users can add to this if they so desire, or they can keep things private. Sign in to the new account from the top-left, and then use the + in the top-right to create your first album. The Upload Photo dialog enables you to select multiple images, and so long as they're all less than 20MB in size you should be able to upload lots of images this way. Up to 10GB if our configuration is to be trusted. Tempting though it is to upload everything at once, it's much easier to proceed one album at a time.

Browse through your galleries, then share your photos with friends and family. Or create your own graveyard-based stock photo library.



Enable maps (provided by OpenStreetMap): ✓
Enable maps for public albums (provided by OpenStreetMap):
Provider of OpenStreetMap tiles: Wikimedia
Wikimedia
OpenStreetMap.org (no HiDPI)
OpenStreetMap.de (no HiDPI)
OpenStreetMap.fr (no HiDPI)
University of Erlangen, Germany (only HiDPI)
Include photos of subalbums on m
Decode GPS data into location name
Show location name ✓

Lychee has some neat features, such as integration with OpenStreetMap.

You can use the About Album tab (the lowercase i in the toolbar in the top-right) to add metadata, such as a description of the album. Use the Visibility menu (the eye in the toolbar) to make the album Public or mark it as Sensitive. If you make albums public, you can also add a password so that only people you know can see them. Alternatively, you can set your friends up with accounts on your Lychee instance and explicitly grant those accounts access to your albums.

You can also import photos directly from the server *Lychee's* running on. Just place files in the **lychee_upload/** Docker volume defined above and make sure their permissions are set correctly. We last covered *Lychee* back in **LXF188** (unless the venerable *Linux Format* Archive lies to us) so it's nice to see how the project has developed since then.

There are many other options for self-hosting your photos, or see the box (below) if you'd rather pay someone else to do it. There are also a number of companies offering a free 2GB *Nextcloud* hosting, if setting it up looks scary but you want to see what all the fuss is about. One project we're certainly going to be keeping an eye on in the coming months is *LibrePhotos* (**https://librephotos.com**) a fork of *Openphotos*, which itself set out to replace Google Photos. Do let us know how you fare on this mission. LXF

» HASSLE-FREE ALTERNATIVES

If you want to put your photos somewhere that isn't Google, then there are a few options. Obviously there's iCloud and OneDrive, but these won't be to everyone's tastes. Dropbox has excellent photo management facilities, and is easy to set up on mobile such that your photos are automatically uploaded to the cloud. The free tier only allows for 2GB of storage, but if you're willing to pay £5.99/month you can get a thousand times that. Dropbox integrates with Google Photos so migrating your data across couldn't be simpler. There's no shortage of other cloud storage providers, though – for example, Box or I-Drive,. These don't really have photo-specific features, but they have a free tier and are competitively priced beyond that.

One worthy of mention is pCloud, which besides giving you previews of images in RAW formats enables you to pay for lifetime storage (500GB costs $175, with traffic limited to prevent abuse), although we don't really know what happens if the company fails, or is bought by Google. Serious photographers may want to consider Adobe's Creative Cloud Photography plan, but for £10/month, there's much better value out there if you're not beholden to Adobe's tools.

QGIS

Credit: https://qgis.org

Create, edit and use custom mapping data

If you like Google Maps you're going to love QGIS. **Mike Bedford** shows you how to get started and introduces some of its functionality.

OUR EXPERT

Mike Bedford is a big fan of QGIS. However, he admits to still loving good old-fashioned paper OS maps while he's walking on the hills.

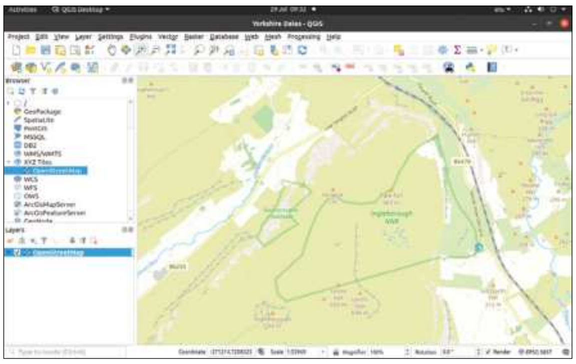
Back in 1962, archaeologists in the Czech Republic discovered a mammoth's tusk that has been engraved with a geometrical design. The pattern was subsequently identified as representing the hills, rivers, valleys and routes of the surrounding area. It was dated around 250,000 BC, making it probably the earliest map ever discovered.

In the library of Hereford Cathedral is the Mappa Mundi. Created around 1300, it's the largest surviving medieval map of the world. The UK's Ordnance Survey was set up in 1791 to help address the perceived risk arising from the Jacobite rising, and of invasion by France, and it went on to become one of the world's foremost national mapping agencies. And 1999 saw California-based ESRI release its *ArcGIS* graphical information system, which would become the world's most used GIS (geographic information system).

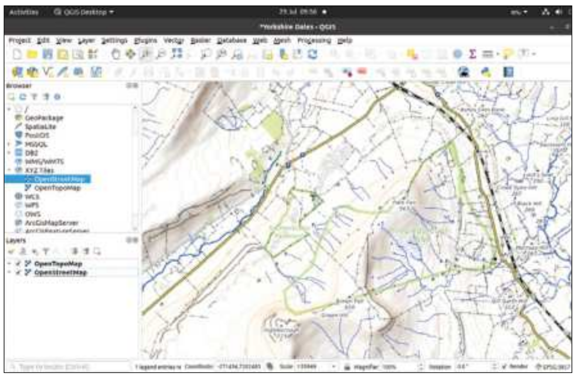
These few notable dates in the history of cartography illustrate 27,000 years of change and it's clear that the rate of change has accelerated significantly in recent years. This isn't surprising since digital technology has had the same effect on so many other areas, but it's been suggested that a GIS is so different to its predecessors that it's a totally new entity. Professionals point to support of multiple layers and an ability to carry out analyses of geographically based data as just some of the things you can do with a GIS but not with an ordinary map. And if you're wondering where Google Maps fits in, while we're not going to get embroiled in the debate over whether or not it's a GIS, we would agree with the sentiment that a fully blown GIS is like Google Maps on steroids.

QUICK TIP

When you work on a project, you'll end up with several files including, for example, files associated with the layers. For this reason, it's good practice to use a new folder for each project and keep all of the files organised together.



Adding OpenStreetMap data as a base map couldn't be much simpler.



From street maps through topographic maps to aerial imagery, any number of options are available as base maps.

ESRI's well-respected *ArcGIS* isn't cheap at £1,192 per year, including VAT, for professional use. This reduces to £139 for non-commercial use, but that's still a lot unless you have a fairly serious application. Our subject here, therefore, is the free open source *QGIS* which is widely used professionally and is considered to be on par with many professional GIS offerings.

We're going to provide a hands-on introduction to using *QGIS* but we're certainly not going to be looking at all its features, in fact we won't come close. After all, if you can think of something you believe you ought to be able to do with mapping data, it's a fair bet that *QGIS* can do it, either natively or via one of its many plug-ins. However, we trust that our tentative first steps will be enough to excite you about the possibilities of *QGIS* and start you off on your own voyage of discovery.

Orientate yourself

If your main experience with digital mapping has involved Google Maps, the first thing you'll notice when you start up *QGIS* for the first time is that you don't see a map at all. Instead, unless you open a project that someone else has produced, you'll need to create a new project but, even having done that, you'll be faced with a blank canvas to which you'll need to add data as layers.

Usually you'll want to start a project by adding a base map so, let's see how to do that, but first create a new project at Project>New. Now, in the Browser panel at the top-left, find and expand the XYZ Tiles entry and double-click OpenStreetMap. This will add the open source OpenStreetMap as a new layer. The map will appear in the main window and you'll also be able to see it listed in the Layers panel at the bottom-left.

Although OpenStreetMap is the only base map shown by default, there are lots of others you can use, and you might like to familiarise yourself with what's on offer. You'll have to search out the URLs yourself, but as an example, right-click XYZ Tiles in the Browser panel and select New connection… In the Connection Details dialog, enter `https://tile.opentopomap.org/%7Bz%7D/%7Bx%7D/%7By%7D.png` as the URL, give it a name such as OpenTopoMap and click OK.

OpenTopoMap will appear below OpenStreetMap under XYZ Tiles and, if you double-click it, OpenTopoMap will appear as a second layer. In fact, it'll appear to have replaced OpenStreetMap in the main window, but that's only because it's the top-most layer: it appears above OpenStreetMap in the list, and it's fully opaque. We'll look at opacity later but, for now, note that you can swap to seeing the OpenStreetMap again by clicking the tick box next to OpenTopoMap so it's not displayed, or by dragging OpenStreetMap to the top of the list.

This method of adding base maps isn't limited to maps in the normal sense of the word. You can also add satellite or aerial imagery using the same method. To add Google imagery, for example, use `https://mt1.google.com/vt/lyrs=s&x=%7Bx%7D&y=%7By%7D&z=%7Bz%7D` as the URL.

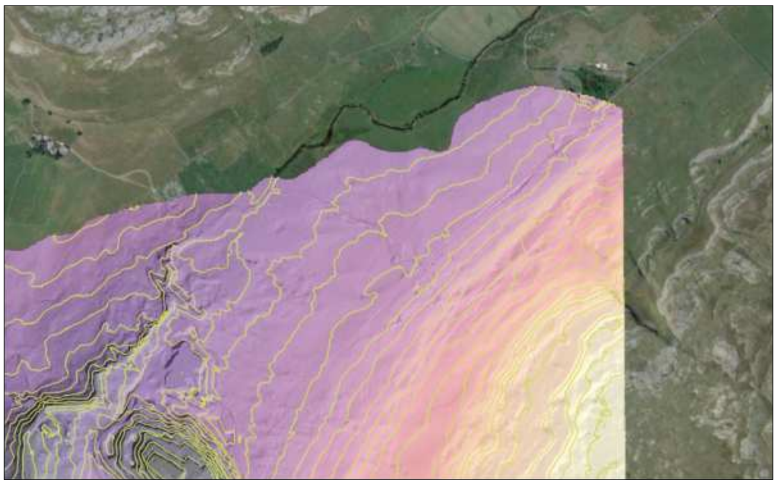
Elevation data

So far, we've seen how to add various layers to a *QGIS* project, although all of those layers were essentially images of one type or another. But since a GIS is concerned with far more than just images, or maps if you prefer, let's take a look at how other types of data can be added.

To start we're going to add some elevation data, but first a few words on types and sources. The types you'll mostly find are DSMs (digital surface models) and DTMs (digital terrain models). DSMs include objects such as trees, buildings and cars, as well as the actual ground, while in a DTM the former types of object are filtered out so it represents only the topography. In the UK, elevation data has been produced by the Environment Agency at a resolution of 1m or 2m, and is distributed freely online by DEFRA. Over the Pond, the USGS makes elevation data available for the US at resolutions ranging from 10m to 30m, but with the aim of migrating to 1m. They also provide Space Shuttle-derived data for most of the world at 30m resolution. Commonly elevation data is provided as a GeoTIFF file, which is a geo-referenced variant of the familiar TIFF image format, although there are other formats, and *QGIS* supports most of them.

Once you've downloaded the elevation data, it's time to import it into a *QGIS* project, ideally one into which you've already added a base map. Select Add Layer from the Layers menu and then choose Add Raster Layer… In the dialog, select the elevation data file by clicking the three dots against Raster Dataset(s) and clicking Add.

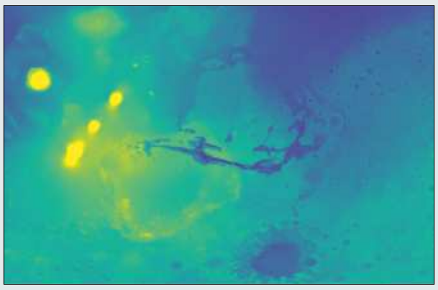
Because the elevation data might cover a smaller area than the base map, it's possible that you won't see the new layer on the map, even though it appears in the Layers panel. If that happens, just right-click the name of the layer containing the elevation data and select Zoom to Layer. Your elevation data will often appear as a rectangle, because you'll have downloaded a so-called


Freely available elevation data can be added to a project and formatted in so many ways.

» QGIS OR MGIS? TIME TO LOOK BEYOND EARTH…

If you were wondering what *QGIS* stands for, the Q is for Quantum – or at least it was until 2013 but we guess now Q just means Q – and the G stands for Geographic which includes the "geo" suffix that relates to the Earth. As we proved in our experiments, though, it might be equally appropriate to call it QMIS, since we used it to display Martian data. Indeed, such systems have been used with data representing a wide range of non-terrestrial astronomical objects.

Interested in the wider Solar System? Then you'll probably want to investigate the various planetary base maps and digital elevation models that are freely available. However, you're going to need to delve into the subjects of CRSs – that's coordinate reference systems – although it's a subject you really need to be familiar with whatever your use of *QGIS*. Needless to say, data representing anywhere other than the Earth won't use any of the CRSs devised for terrestrial use. And we have to admit that we struggled with one particular Martian base map which loaded, but wouldn't display. However, we did do better with elevation data so that might be a good place to start. Search for "Mars_MGS_MOLA_DEM_mosaic_global_463m" and you'll find that it can be downloaded from **https://astrogeology.usgs.gov**. And if the moon is your thing, you might enjoy **https://lunaserv.lroc.asu.edu**, which hosts data from the Lunar Reconnaissance Orbiter Camera and provides some information on how to use it in *QGIS*.


From Io to Titan, maps and elevation data is available for lots of heavenly bodies and visualised in QGIS.

QUICK TIP

With v3, QGIS now offers a 3D visualisation feature. This is commonly seen in online mapping applications, but considered a novelty by some GQIS professionals. The 3D view can be manipulated by zooming, panning, rotating, tilting and so on.

tile, although it's possible it might be less than a complete rectangle because coverage isn't always 100 per cent.

Initially, the elevation data might not look too impressive. It's just in shades of grey with white representing the highest elevation in the tile, and black representing the lowest. In passing, this means that if you have two elevation layers, they'll be scaled differently, but this can be solved by merging them using Raster>Miscellaneous>Build Virtual Raster...

The uninspiring greyscale image is just a start and *QGIS* has plenty of tricks up its sleeve. For a start, let's just change the colours: right-click the layer and select Properties. Now, in the Layer Properties dialog, ensure that Symbology is selected on the left, under Render type choose Singleband pseudocolor instead of Singleband gray, and select one of the options for Color ramp. You might not find that any of the Color ramps are to your liking, and none correspond to the common colouring of altitudes on topographic maps, but you'll notice that you can edit each of the colours.

Moving away from just changing colours, Hillshade is another interesting option for Render type. It'll revert the colours to greyscales in the process, but there's a solution to that. As before, use the Layer properties to select your preferred colour scheme. Next, with the elevation layer selected, select Analysis>Hillshade... from the Raster menu, before accepting all the defaults in the dialog and clicking Run. Unlike last time, the latter won't be converted to a hill-shaded version, but a new hill-shaded layer will be created.

As always, the new layer will be at the top of the list, so drag the coloured elevation layer to the top in the Layers panel and, of course, it will obscure the hill-shaded layer. However, adjusting the transparency of the pseudo colour layer will give the best of both worlds. This option is available in Layer Properties.

And as a final way of processing elevation data, a few words on contours. Depending on your choice of base map, you might already be able to see contours. However, if you're using a base map without contours, you can generate these from elevation data at Raster>Extraction>Contours....

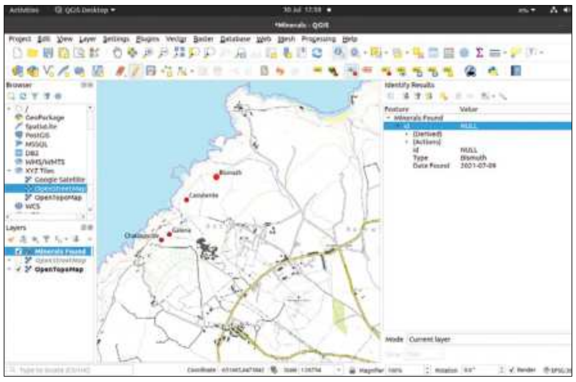
Beyond conventional maps

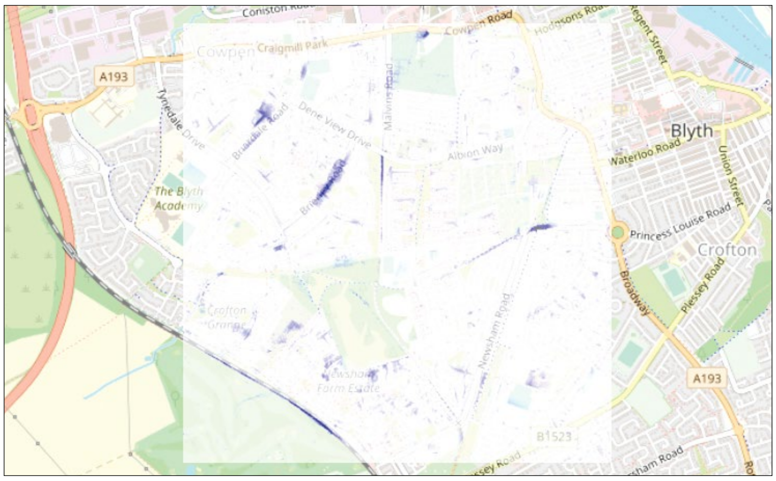
So we've seen how *QGIS* handles data that's basically an image, or in other words a map, and we've seen that it can process elevation data in so many ways. But this is far from the limit of its data-handling capabilities. *QGIS* can work with any type off data so long as it's georeferenced. So, for example, this could enable you to see the variation in rainfall or hours of sunshine across a region, or it could provide a means of visualising the correspondence between the geological classification of rocks and surface features.

A recent project involved showing how *QGIS* can operate with other types of data, and via a brief return to elevation data, how it can carry out calculations on that data. The exercise involved using rainfall and elevation data to simulate areas at risk from flooding. The simulation software could input raw elevation data, so no need for *QGIS* there, except for the fact that elevation data for the area of interest was incomplete. In particular, there were holes in the preferred 1m resolution data, but 2m data was available for the area.

QGIS was used, therefore, to compile a file of elevation data using 1m resolution where present, and filling the gaps with 2m data where it wasn't. This was achieved by loading both sets of elevation data as separate layers and then using *QGIS*' raster calculator (Raster>Raster Calculator) to combine them. This was done using the statement `("1m_dtm@1" = 0) * "2m_dtm@1" + ("1m_dtm@1" != 0) * "2m_dtm@1"` having first ensured that the **no data** value in the 1m elevation data file is 0. It generates a new raster layer with values defined by the expression, and it's fairly clear how it achieves the desired effect.

This is only presented as an aside, but the ability to carry out any arithmetic operation on raster data is a powerful tool. *QGIS* then came into play again when the simulation software had crunched the input data and created a geo-referenced output file of water depths. By importing this to *QGIS*, and viewing it, partially transparent, over a base map, at-risk areas could be clearly identified.


Points can be added to a project and data fields can be defined for subsequent display or analysis.



As this example of water depth during a flood illustrates, how geo-tagged data can be used in QGIS.

Lines, points, photos

Informative pushpin labels are a familiar sight on online maps, and *QGIS* doesn't disappoint here. This sort of information can be created elsewhere and imported as a new layer, or it can be created within *QGIS*, as we're about to see. As always, you need at least a base map in your project before proceeding.

In the Layer menu, select Create Layer>New Shapefile Layer.... In the dialog, give the layer a name and choose a type, which must be Point, Multipoint, Line or Polygon. Now start adding fields, and there can be as many as you want. In our example, we're creating a layer that shows places at which particular minerals were found. Our first field, therefore, is the type of mineral, so we give it the name "type", and specify that the field is defined as text. Having filled in the first field, click Add to Fields List and you'll see it appear in the list below. We also added a second field, namely the date on which the mineral was found, in just the same way. When you've defined all the necessary fields, just click OK and you'll see that the layer appears in the Layers panel, although nothing will appear on the map, because there are no points defined yet.

To add points, make sure the new layer is selected and click the Toggle Editing icon (yellow pencil) in the Data Source Manager toolbar. Once you're in editing mode, click the Add Point Feature icon (three green dots and a crosshair), and the cursor will change to a crosshair. Click it on the map at the place of interest and the Feature Attribute dialog will appear. You might not want to bother with the "id", but fill in the various other fields and click OK. A coloured dot will appear on the map. Now click the Toggle Editing toolbar again to turn off editing and see how a user can interrogate that point. Click the Identify Features icon (the "i" on a blue circle with an arrow cursor) in the Project toolbar. Click a coloured dot and a window will appear providing data on all the fields defined for the layer.

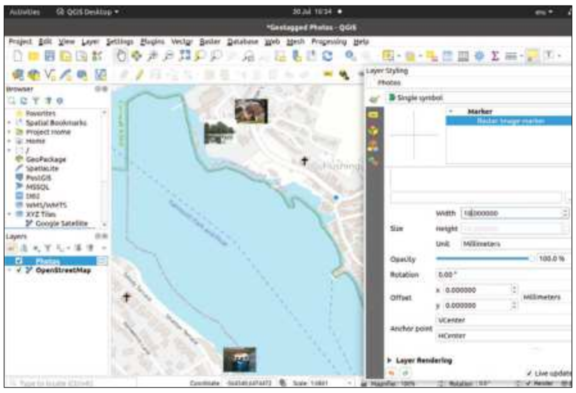
You might find it surprising that some of that data doesn't appear alongside the point without interrogating it, but since there's no limit to how many fields you can assign, it's fairly obvious why that doesn't happen by default. However, you can choose to add labels for all points on a layer, which you'll do by clicking the Layer Labelling Options icon ("abc" on a yellow arrow) on the Data Source Manager toolbar. In the Layer Styling dialog, ensure the Labels tab is selected (the "abc" on a yellow arrow again), choose Single labels in the top box and choose the field you want the point to be labelled with against Value. You'll see the point labelled on the map. You might want to fine-tune the options under "Placement". Also, if the size, shape and colour of the point aren't to your liking, this is something else you can adjust in the Layer Styling dialog, this time on the Symbology tab (the paintbrush).

QGIS also enables you to show photos at the locationçthey were taken, but only if they're geolocated. So, if they were taken on a phone you'll be okay, although if you took them on a camera they might not be. Bizarrely, given that *QGIS* doesn't seem to be short on features, we could find no way to import non-geotagged photos. That's not a show-stopper, though – it just means you've got to add the geotags separately. There are plenty of standalone applications and web-based utilities to do that (for example, **www.tool.geoimgr.com**), although many limit how many photos you can process a day without paying a fee.

So, with a folder full of geotagged photos at the ready, here's how to import them into a *QGIS* project. In the Processing menu, select Toolbox>Vector creation>Import geotagged photos. Against Input folder select the folder containing the photos you want to add and click Run. A new layer will appear in the Layers panel and you'll notice some small markers appear on the map, at their correct locations, just like the ones we added previously.

However much you zoom in, though, they won't appear as photos unless you make some changes. Bring up the Layer Styling dialog and ensure the Symbology tab is selected. Single Symbol will be shown and below that the marker type will be shown (expand it if not already expanded to show Simple Marker). Click Simple marker and then, against Symbol Layer Type, choose Raster Image Marker instead of Simple Marker. Lower down there's an unnamed box where you can define a particular raster image, but instead we want it to pick up the image at that point. So, click the icon at the right, which looks like a filing cabinet and a couple of arrows, to allow the Data Defined Override option to be selected, and against "Field type: string", choose "Photo (string)". Photos will replace the blobs, but they will be very small so specify a new size against Width.

QGIS is a very powerful tool and what we've presented here is only a taster to what it can achieve. Experimentation is one of the best ways of learning, so what are you waiting for? LXF


As an alternative to points, places of interest can be shown on a map using geo-tagged photos.

QUICK TIP

With some actions that create new layers, those layers will be temporary scratch layers. The method we described for creating a layer with photos is one such case. These layers will be lost when you close a project (although QGIS will warn you), but you can prevent this by right-clicking the layer and selecting Make Permanent.

» ANALYSIS OPPORTUNITIES

Although the ability to analyse geographical data is considered a key feature of a GIS, we've largely ignored this feature. As you grow in your confidence with *QGIS*, though, depending on your application you might want to delve into analysis, and here the sky's the limit.

For a start, there's lots that can be done with elevation data. Mobile phone companies use GIS software to visualise the coverage that would be achievable from base stations at given positions. This sort of exercise can be used to decide where to locate base stations and forms the basis of published coverage maps. But it goes much beyond that, a fact that should be obvious when we think about the Shapefile layers that we created and populated with points. There can be any number of such layers, and the points in each of those layers are defined by their geographical location plus data in several fields.

We can, therefore, start to think about a *QGIS* project as a geotagged database and the potential starts to become clear. A common approach is to carry out some sort of analysis and create a new layer from the results that can be viewed on the base map. The *QGIS* documentation shows an example of an exercise that involved analysing the cause of a cholera outbreak in London in 1854. And while that exercised used fairly simple analytical tools, *QGIS* can also be interrogated using SQL queries.

Credit: www.dosbox.com

DOSBOX

How to easily emulate 486 PCs and run DOS

Relive the halcyon 32-bit years of PC computing, play your favourite games and look back at OSes of yore with the ever-dependable **Les Pounder**!

OUR EXPERT
Les Pounder is associate editor at Tom's Hardware and when not writing about Raspberry Pi he is found tinkering with old computers.

This author first got their first PC back in 1994: a Dan 486 DX 33MHz with 4MB of RAM. It was much more powerful than the Amiga 500, and almost three times the price. The author's journey with computers up to this point had been via Commodore machines, but this Dan 486 PC captured his heart. He got it home and launched MS-DOS, but he had no manual or user guide. He taught himself MS-DOS and Windows 3.1 by making lots of mistakes, including formatting his 120MB hard drive and having to reinstall everything, and by installing a CD ROM drive incorrectly and the machine refused to boot.

The mid-1990s was a crazy time for PCs. They were still beige boxes that held mysteries such as IDE, SCSI, lots of wires and dust. This mighty bookazine is devoted to Linux, but for this journey down memory lane we'll take a look at emulating a Windows 3.1 era PC. First via a simple program that will enable us to run DOS-era games in Ubuntu. Then we'll learn a little BASIC with Microsoft's Quick Basic. Finally we'll create our own Windows 3.1 PC in a virtual machine.

Emulating a retro PC

The retro PC scene has users who want to simply run their old games in an emulator, and it has users who want to run real hardware with a subtle 21st century twist. Let's start with emulation because it's the simplest route to enjoying retro content. The champion of retro PC emulation is *DOSBox* which offers a quick hit of nostalgia with very little effort. To install *DOSBox* on a Debian or Ubuntu machine, open a terminal and type the following:

```
$ sudo apt update
$ sudo apt install dosbox
```

With *DOSBox* installed, launch the application from the terminal.

```
$ dosbox
```

DOSBox is an x86 emulator for 486 and early Pentium computers and games. It's dependable and works really well for a quick hit. When it first starts, we see a familiar DOS prompt with drive Z as the active drive. *DOSBox* provides a brief help menu – highlighting that we've a Soundblaster compatible sound card and that we can learn more about using *DOSBox* via the `intro` command. At any time we can obtain general help using `help`, or for more detailed help use

```
help /all
```

What can we do right now? Well, let's remember a few DOS commands. First we have a command to list the contents of the current directly. We know this as `ls` but in DOS it's `dir`.

This will show all of the files in **Z:**. Right now there isn't much of interest, but as we move onwards we shall learn a few extra *DOSBox* features.

The 486 DX 33 may have only run at 33MHz, but coupled with 4MB of RAM and a little imagination you could fly an X-wing for the Rebel Alliance.

Writing code

The retro PC era spans decades, and with that we've a fantastic choice of languages and applications with which to write code. In this instance, we're going to focus on BASIC, in particular *Quick BASIC* (*QBasic*) that was bundled with MS-DOS. QBasic is a cutdown IDE and it can be found easily online. We're lucky enough to own MS-DOS 6.22 floppy disks and a hard drive with *QBasic* installed.

Download a copy of *QBasic* and extract the contents of the archive to a directory in your home directory. Remember the location of the extracted files as we'll need that exact location for *DOSBox*. Launch *DOSBox* from the terminal. In the DOS prompt we shall use a command to mount the location of our *QBasic* download as another drive, **C:**. Our *QBasic* directory is called **QB45**, but change this to match your own:

```
mount c: ~/QB45
```

A message will be printed to the screen, advising us that C: drive is mounted to that directory. Change to C: drive and the list the contents of the drive.

```
C:
dir
```

We should now see a list of files, but we're only interested in **qb.exe** which is *QuickBasic*, so let's launch it by typing the file name and pressing Enter:

```
qb
```

QuickBasic's user interface is basic, no pun intended. It has a simple menu at the top of the screen, and a blank section in the centre where we can write our code, and a shell interface, called Immediate, at the bottom where we can test code snippets. Our first quick test of *QBasic* is to write the familiar "Hello World" code, but this time it will be a little different and use a for loop to print the text 20 times. In the blank area of *QBasic* is where we write the code.

We start by first declaring that we're creating a variable, "counter" and that it'll store an integer (numeric) value. Then we clear the screen of any existing data.

```
DIM counter AS INTEGER
CLS
```

We start the for loop, updating the value stored in the `counter` variable each time the loop iterates. It starts at 0 and ends at 20.

```
FOR counter = 0 TO 20
```

The next lines of code don't need to be indented, but we've chosen to do this to show that they are inside the for loop. We use the `PRINT` command to print the "Hello World" message to screen, then we pause the code for one second using `SLEEP`.

```
 PRINT "HELLO WORLD"
 SLEEP 1
```

Finally we iterate the counter variable, each time adding one to the previous value. When the for loop ends, the program ends.

```
NEXT counter
END
```

Save the code by pressing ALT then F and A. Name the file **HELLO.BAS** and save it to C drive. Use TAB to navigate through the sections. Press OK to save. This will save the code to our host machine, our Linux PC. When ready, press F5 to run the code and you should see a DOS prompt appear and run the code.

Let's make something a little more complex, a tool to calculate the circumference of a circle. Create a new file (ALT, F. N) and we start the code by clearing the screen of any data.

```
CLS
```

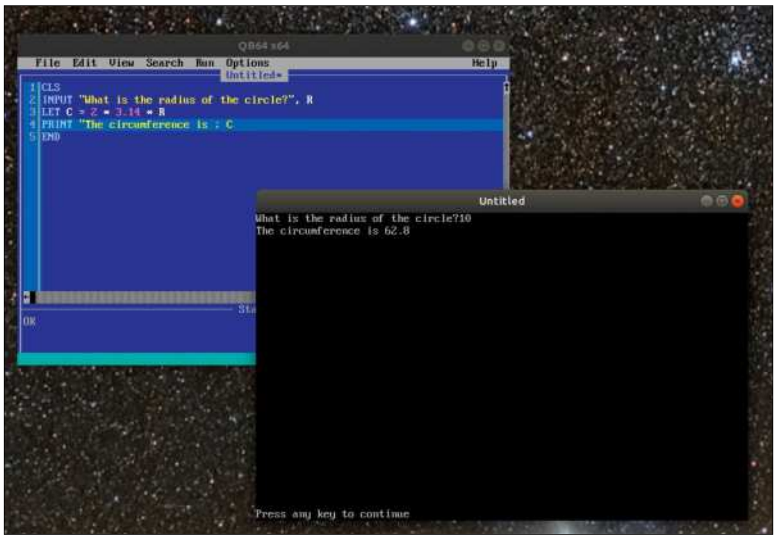
Now we ask the user a question, "What is the radius of the circle?" and we use INPUT to read the raw keyboard input, which is saved to a variable, R.

```
INPUT "What is the radius of the circle?"; R
```

We then create the variable C and in there store the answer to the equation which will work out the circumference of the circle using the value stored in R.

```
LET C = 2 * 3.14 * R
```

We then print the answer to the screen, with a sentence explaining what the returned value, C, refers to. Finally we end the code cleanly.



```
CLS
INPUT "What is the radius of the circle?", R
LET C = 2 * 3.14 * R
PRINT "The circumference is : C
END
```

```
OK
```

```
What is the radius of the circle?10
The circumference is 62.8

Press any key to continue
```

QB64 is a modern-day version of QuickBasic, which runs on Linux, Windows and Mac. It's also compatible with QuickBasic 4.5.

```
PRINT "The circumference is"; C
END
```

Save the code as **CIRCUMF.BAS (ALT, F, A)** to your C: drive. Then press F5 to run the code. Type in a radius (we chose 10) and the code will print the answer. We can also make an EXE executable file from the Run menu. This will launch our code outside of *QBasic*.

Playing a game

DOSBox will run most DOS games, even from CD-ROM. We tested it with a vintage copy of LucasArts' point-and-click adventure game *Day of the Tentacle*. Inserting the disc into our DVD drive, sr0, we opened *DOSBox* and from the prompt entered a command to mount the CD-ROM. Note that we needed to go into the **/media/{username}** directory to get the name of the directory used as a mountpoint:

```
mount D /media/{username}/{directory name}
```

We now have our DVD drive mounted. Change to D: drive and list the contents of the disc.

QUICK TIP

MS-DOS is not case sensitive, unlike Linux. So QBASIC and qbasic are one and the same. DOSBox has a basic form of history, which we can scroll through using the up and down arrows.

» TAKE FREEDOS FOR A SPIN

FreeDOS is an open-source implementation of MS-DOS that works with more modern machines. We installed FreeDOS on an Asus Eee PC 701 and it makes a great retro machine that boots in less than ten seconds. FreeDOS was created to enable anyone to run their old DOS games and productivity software on newer hardware. Downloading a USB image from the FreeDOS website (**www.freedos.org**) and then writing the image to a stick takes no time at all and the installation process on our test machine was quick and painless.

If you'd like to run FreeDOS on real 486 hardware then there are many options available, such as a legacy CD-ROM image or a Boot floppy and CD-ROM download that has the necessary CD-ROM driver for MS-DOS. FreeDOS can be used in a multiboot environment with more recent Microsoft OS. It also comes with tools to manage archives, browse the web, play MP3/OGG and WMV files, CD-ROM support and a select suite of Linux-based tools.

FreeDOS is ideal for virtual and real hardware. It's an open source version of DOS that enables you to reminisce and learn new skills. Take it for a spin on an old machine, you might just enjoy using DOS.

»

QUICK TIP

We can take Windows 3.1 for a spin in our browser. ClassicReload has a web-based version of DOSBox, which means we can skip creating a virtual machine and start enjoying this classic interface: https://classicreload.com/win3x-windows-31.html

```
D:
dir
```

We can now read the drive and launch any executables that we need.

Creating our own Virtual 486

DOSBox is great for a quick hit, but what if we want something a little more permanent? Well, we can create a virtual machine to recreate a top-spec 486 (almost) machine and run Windows 3.1. For this you'll need your own copies of Windows 3.1 and MS-DOS 6.22 saved as image file (.img). Luckily for us we have these in the attic, together with a USB floppy drive.

We start by downloading *Virtualbox* from **www.virtualbox.org/wiki/Downloads** and then install the download. Open *Virtualbox* and create a new virtual machine (VM). We called our VM Windows 3.1 and this automatically triggered the VM to configure itself for Windows 3.1. We next set the memory size to 32MB, a huge amount for this era. Then we created a virtual hard disk (VHD), fixed to 1GB in size. With the machine "built" we need something to boot from, and this is where disk one of MS-DOS 6.22 is needed. We can insert a disk image by going to the Settings option, then Storage>Controller: Floppy and navigating to the floppy image. After inserting the disk, click Start and the VM will power up and load MS-DOS. Follow the standard install process and when prompted to swap disks, press the right Ctrl key to unlock the mouse, click the Devices menu, then Floppy Drives, then Choose a Disk File...

Navigate to the directory containing your MS-DOS disks and select the appropriate disk. Installation will take less than five minutes, much quicker than back in the day. Remove any disks, and reboot when prompted. If at any time you need to reboot, you can select Reset from the Machine menu.

MS-DOS takes seconds to boot, and the prompt is a great place to learn a few commands. We can get help at any time by typing HELP. Our goal is to install Windows 3.1, so now is the time to insert the Windows 3.1 floppy disk images (Devices>Floppy Drives>Choose a Disk File...). Change directory to the first virtual floppy drive and then list the contents of the disk.

```
A:
dir
```

On A: drive there'll be a **setup.exe** file – this is what we need to install Windows.

```
setup.exe
```

Follow the installation process, and swap disks just like we did for MS-DOS. In a few minutes Windows 3.1 will be installed. Reboot the machine (Machine>Reset) and from the prompt type the following code to launch Windows.

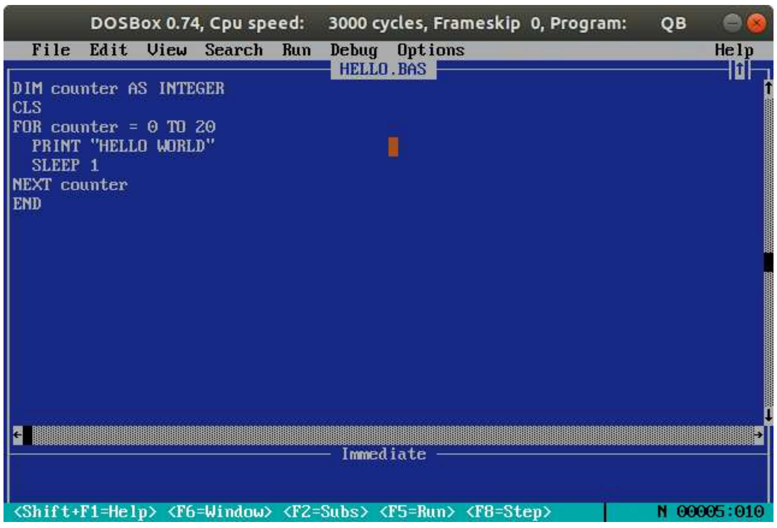
```
win
```

In seconds the familiar Windows 3.1 desktop will appear and a few memories are sure to rise to the surface, too.

What's on the demoscene?

Just like many other machines, the PC enjoys a rich demoscene of artists, musicians and coders all eager to share their skills and show off their talents. After researching on **pouet.net** we found *Memories*, a demo from Desire who released this demo at the Revision 2020 demo party. This demo has no music, but it achieves what should be impossible: it squeezes an entire demo into 256 bytes. That's not a typo by the way! This is a tiny demo that illustrates what great coding can do.

Here at the start of 2021 the PC demoscene is still very much active. New demos that push even the most


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: QB
File Edit View Search Run Debug Options Help
HELLO.BAS
Immediate
<Shift+F1=Help> <F6=Window> <F2=Subs> <F5=Run> <F8=Step>
N 00005:010

```
DIM counter AS INTEGER
CLS
FOR counter = 0 TO 20
  PRINT "HELLO WORLD"
  SLEEP 1
NEXT counter
END
```

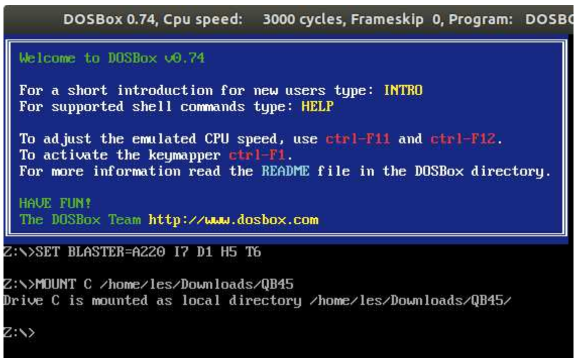
Hello World isn't exciting, but it enables us to grasp enough of the syntax to check that everything is working, including our brain.

» TAKE THINGS FURTHER WITH QB64

QuickBasic was fun but firing up a VM every time we want to write BASIC can be time consuming. Fear not, *QB64* is a modern-day version of *QBasic* that runs on Linux, Windows and Mac. Download and installation is a breeze, and in five minutes we were writing BASIC code in Ubuntu. Code written in *QB64* is compatible with *QBASIC*, but *QB64* automatically compiles our BASIC code into machine code. It can also be enhanced via extensions for OpenGL enabling eager users to build their own games and programs using this powerful tool.

The *QB64* user interface looks just like *QBasic* and it even works in the same manner. We tested our circumference project in *QB64* and were amazed that every line we typed was checked for errors as we typed! This is great for learning and really helps the user to see where their code went wrong, before they run it.

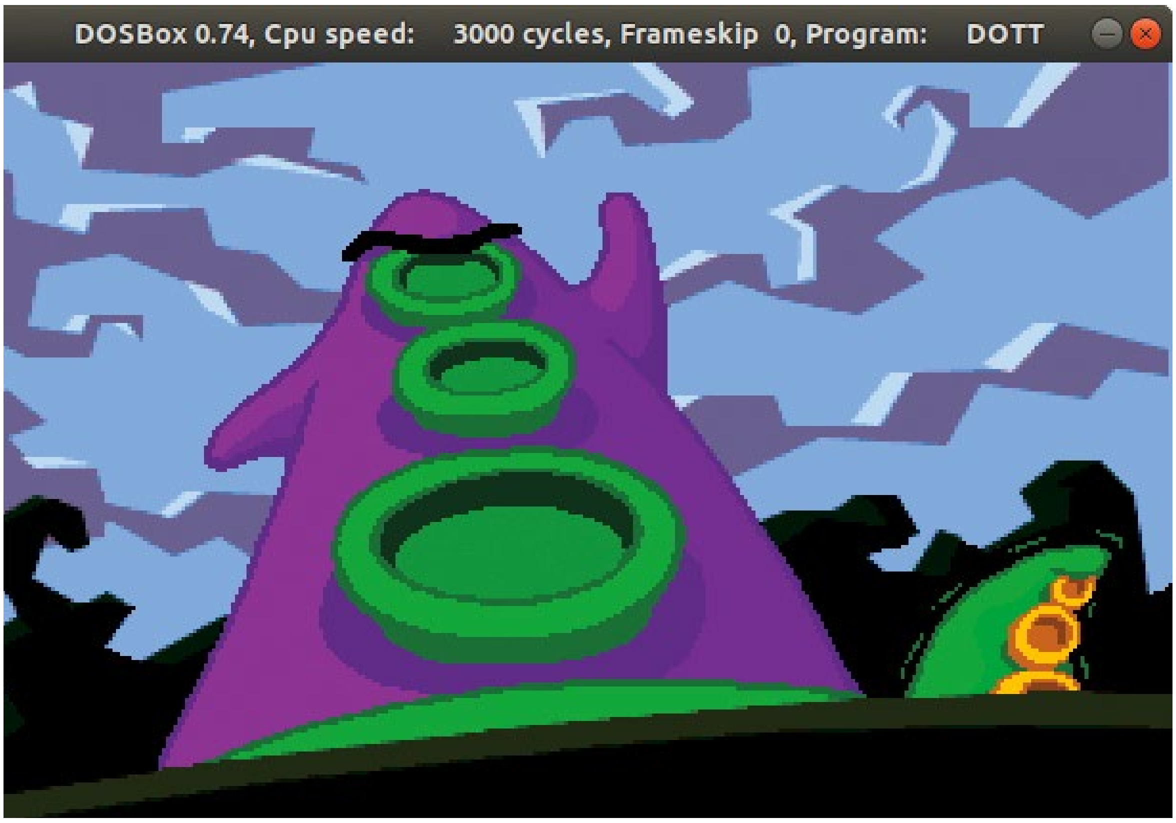
QB64 is a free download from **www.qb64.org/portal** and it's well worth investing an afternoon with this editor and the great wiki at **www.qb64.org/wiki/Main_Page**. This covers all aspects of the application and explains the BASIC language syntax and structure using a series of examples.


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBO

```
Welcome to DOSBox v0.74

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>MOUNT C /home/les/Downloads/QB45
Drive C is mounted as local directory /home/les/Downloads/QB45/

Z:\>
```

In DOSBox we can mount a directory or drive to a specific drive letter. It can link out Linux host machine to DOS – handy for sharing files.

CREDIT: LucasArts


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOTT

Day of the Tentacle was a point-and-click adventure. It was released on floppy disk and a 'talkie' CD-ROM with specially recorded speech.

QUICK TIP

Windows 3.1 just a little too retro for you? You can take Windows 95 for a spin in your browser. See www.pcjs.org/blog/2015/09/21. Windows 95 had a massive marketing campaign, and saw users queuing overnight to get their hands on an OS. Wowsers!

powerful hardware to its limits are being produced by artists who favour the keyboard and the mouse as their tools of expression.

We would think that with newer, more powerful hardware that the retro PC scene would be dead and buried. Oh how wrong we were. The price of 486-era machines online can vary from bargains which require a little work to new old stock machines that are perfect specimens. The sweet spot of the 486 era was a 486 DX2 66Mhz with 8MB of RAM and prices for those machines vary wildly. Research and patience will yield the best price. Don't just jump into the first machine that you find.

Safety first

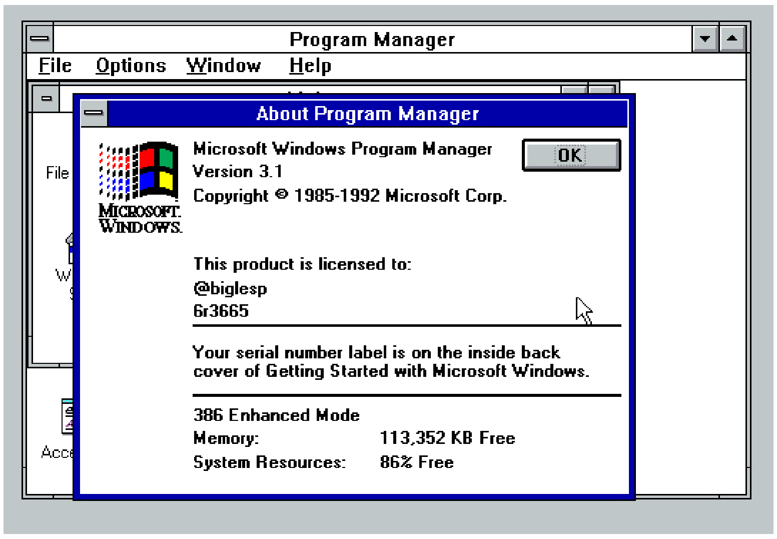
So you have a retro machine – what do you do now? The first thing to do is check that it's safe for use. Are there any leaking capacitors, dodgy wires? Check everything before applying the power. If the machine boots up to a DOS prompt or Windows desktop then we can consider the machine good.

There's one thing to address and that's hard drives. A period perfect drive is now around 20 to 30 years old and it will fail at some point. Fear not as the community has your back, in the form of an IDE to Compact Flash adapter that can be purchased for around £5. We use one of these inside an Amiga 600 along with a 2GB card which holds Workbench and our games.

Using adapters and drive enclosures you can clone your original hard drive to a CF card and replace the drive with a smaller, quieter and more reliable option. Versions exist both for internal and external use, along with SD card versions. External versions are particularly handy because we can swap hard drives without opening the case. They simply fit to a rear ISA/PCI slot and connect directly to the IDE interface.

The retro PC community is a strong and passionate place. We just have to look at Reddit's retrobattlestations subreddit to see that the 386/486 and early Pentium-era machines are cherished by their owners. With new community-designed and produced hardware being created to enhance the experience for all, the future of retro PCs is bright and it lives on in the continued success of the demoscene and the many YouTubers such as 8bit Guy, LGR, RMC and NostalgiaNerd who have created vast amount of content devoted to showing off the best of this era.

Whether you just want to dabble with some retro games, or you have a need for real antique hardware, this era of PCs is an exciting and challenging time where standards were created and legacies created. LXF



Microsoft Windows Program Manager
Version 3.1
Copyright © 1985-1992 Microsoft Corp.

This product is licensed to:
@biglesp
6r3665

Your serial number label is on the inside back cover of Getting Started with Microsoft Windows.

386 Enhanced Mode
Memory: 113,352 KB Free
System Resources: 86% Free

Windows 3.1 sat upon MS-DOS and provided a functional means to work with files and applications. For many computer users was their first taste of a GUI. CREDIT: Microsoft

QEMU

Credit: www.qemu.org

How to run classic distros with QEMU

Where we're going we don't need roads, we need QEMU and a bunch of retro Linux ISOs, reveals **Les Pounder**.

OUR EXPERT

Les Pounder is associate editor at Tom's Hardware and a freelance maker for hire. He blogs about his projects at bigl.es.

When home computing was a fine shade of beige.

Back in the late 1990s, this author chose their PC magazines by what was on the cover disc, something that had started back in their Amiga days. One month, they chose a PC magazine that had something called Red Hat Linux on the second disc which was promptly installed on an AMD K6-2 333MHz PC.

It's fair to say that it wasn't love at first sight. The sheer volume of application choices, the differing commands and file system was enough to scare this author back to Windows 98. A few months later and another coverdisc, and this time it was something different. It was still Linux, but Corel Linux 1.0 which was a little more 'noob' friendly. The install went well and this was this author's distro for quite some time.

In the 1990s, Linux was still in its infancy and the jump from Windows to Linux seemed massive and exciting. How can we experience these days again? In the absence of a giant Tux-shaped time machine, we can use virtual machines to emulate hardware of the era and install Linux on a virtual PC.

We've chosen three Linux distros from the past three decades of Linux and using virtual machines we shall install and use each distro. Our choices span the 1990s, 2000s and 2010s and show the similarities, and differences between Debian, Ubuntu and Linux Mint of these eras.

QUICK TIP

Want to share a USB device with your virtual machine? You can easily do that via the Virtual Machine> Redirect USB Device menu. Handy for using USB flash drives with your virtual machine.

How To Install QEMU

QEMU is a generic and open-source machine emulator and virtualiser. In other words, it can emulate the hardware of many different CPUs and machines, and create virtual machines. To install *QEMU* we need to open a terminal and use the package manager (in our case Ubuntu 21.04 and the *apt* tool) to install *QEMU*, a Kernel Virtual Machine (KVM) and its dependencies.

Why are we installing a KVM? It's because *QEMU* is an emulator, and so by using a KVM and having a CPU that supports Intel VT-x or AMD V we can speed up the virtualisation to near-native speeds. Most modern processors have some form of support for virtualisation – our machine is an i7 3770 from 2012 and it supports VT-X.

To check if your Intel or AMD CPU supports these features, open a terminal and type the following. For Intel VT-X, use

```
grep --color vmx /proc/cpuinfo
```

and for AMD V, type

```
grep --color svm /proc/cpuinfo
```

Now install the *QEMU* application with KVM

```
$ sudo apt update
$ sudo apt install qemu-kvm libvirt-daemon-system
libvirt-clients bridge-utils
```

To make managing our systems a little easier, we'll install a GUI manager for our virtual machines.

```
$ sudo apt install virt-manager
```

We now need to reboot our machines for the changes to take effect. If we skip this step then the GUI manager will be unable to connect to *QEMU*.

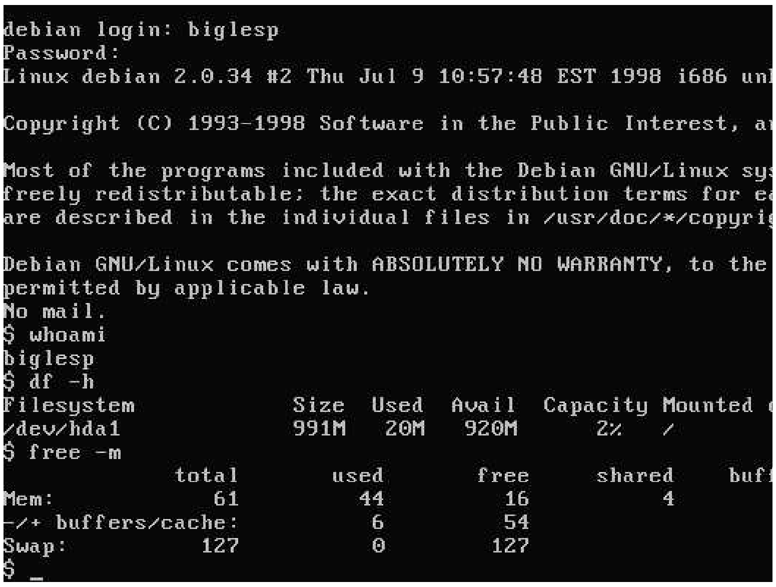
Open the *Virtual Machine Manager* and take a look around. In the top left we have an icon – a monitor with a yellow star – to quickly create a new virtual machine. We shall be using this icon quite a lot in the tutorial.

Virtual machines are listed in the main section of the window and we can select and run machines from here. Selecting a machine and then clicking the Play button will start the machine in the background. Clicking Open will open a new viewer window – our interface with the virtual machine. Launching a machine will launch a player window, where we interact with the virtual machine. In this player window we can administer our virtual machine: redirect host USB devices to the virtual machine, configure the system with more RAM or a better CPU, insert CD/DVDs and configure our networking.

The 1990s – Debian Linux

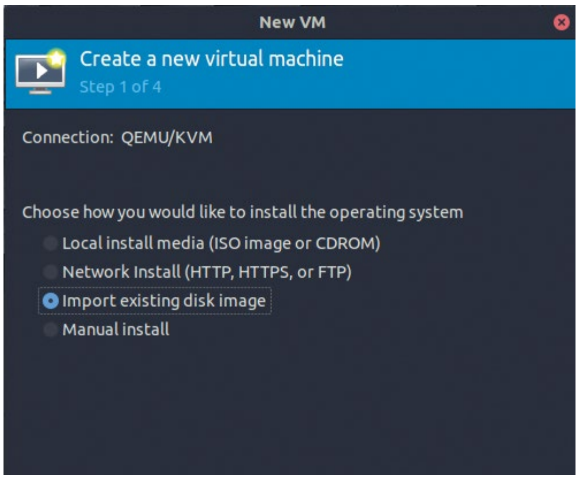
Linux in the 1990s was the new frontier. New distros appeared that took advantage of 486- and Pentium-era machines, while downloading the ISO images took hours, if not days over a dial-up connection. Luckily for most of us in this modern time, we have much faster broadband-enabled internet.

Debian Linux was created by Ian Murdock in 1993 and the first version (0.01) was released that year. But it wasn't until 1996 that we saw the first stable release (1.1). Today, Debian is the backbone for many popular distros including Canonical's Ubuntu and the Raspberry Pi Foundation's Raspberry Pi OS. It's fair to say that without the existence of Debian the Linux landscape would be totally different.



```
debian login: biglesp
Password:
Linux debian 2.0.34 #2 Thu Jul 9 10:57:48 EST 1998 i686 unl

Copyright (C) 1993-1998 Software in the Public Interest, an

Most of the programs included with the Debian GNU/Linux sys
freely redistributable; the exact distribution terms for ea
are described in the individual files in /usr/doc/*/copyrig

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the
permitted by applicable law.
No mail.
$ whoami
biglesp
$ df -h
Filesystem            Size  Used  Avail  Capacity Mounted o
/dev/hda1             991M   20M   920M      2%   /
$ free -m
             total       used       free     shared     buff
Mem:            61         44         16          4
-/+ buffers/cache:          6         54
Swap:          127          0        127
$ _
```

From these humble beginnings, Debian rose to become one of the most important distros in the history of Linux.



New VM

Create a new virtual machine
Step 1 of 4

Connection: QEMU/KVM

Choose how you would like to install the operating system
- Local install media (ISO image or CDROM)
- Network Install (HTTP, HTTPS, or FTP)
- [✓ Import existing disk image]
- Manual install

When creating a new VM we can insert an ISO image or CD/DVD ROM. We can also use a qcow2 or img disk image for a ready-made system.

We could download and install our own version of Debian Linux, or we can download a hard drive image that we can use with *QEMU*. The latter is the most simple, being ready to go with little configuration. That said, the Debian installation process is sublime, more so in recent years. The installation process takes us by the hand through partitioning, user setup and then we can configure the system for our intended use.

We downloaded the Debian 2.0 image from **https://github.com/palmercluff/qemu-images** and then extracted the *qcow2* file from the archive.

Open *Virtual Machine Manager* and click File>New Virtual Machine or click the icon of a monitor with a star. In the first screen Create a New Virtual Machine, select Import Existing Hard Disk Image and click Forward. In the next screen navigate to the location of the extracted *qcow2* file by clicking Browse, and then Browse Local. Once you've selected the volume select Choose Volume and then type in "Generic OS" under Choose the operating system. Click Forward. Set the RAM to 64MB and the number of CPUs to one. This is more than enough for Debian 2.0. Click Forward and in the next screen name the machine "Debian2" and click Finish to set the configuration and start the machine.

QUICK TIP

Need to quickly use a live CD or DVD image? With QEMU we can quickly start the ISO, or a real disc with 1GB of RAM using qemu-system-x86_64 -boot d -cdrom image.iso -m 1024.

» OLD PC HARDWARE

Emulating old PC hardware with *QEMU* is cheap – it only costs us a little hard disk space. But what if we want to use real hardware? This is where things become expensive.

Looking online, we can see incomplete 486-era machines trading hands for hundreds of pounds. So that 486 DX2 66MHz you fancy is now just out of reach. Old PC hardware is expensive to ship and unless it's in the hands of a collector, it often needs lots of work to make them ready for use.

But say you do buy an old PC – what do you need to do? First, make sure you know your stuff before starting the job. Take care with electrics and chemicals for restoring a machine. If in doubt, seek the knowledge of an expert. Older machines used AT power supplies: it's possible they're broken, so make sure you have a known working supply to test.

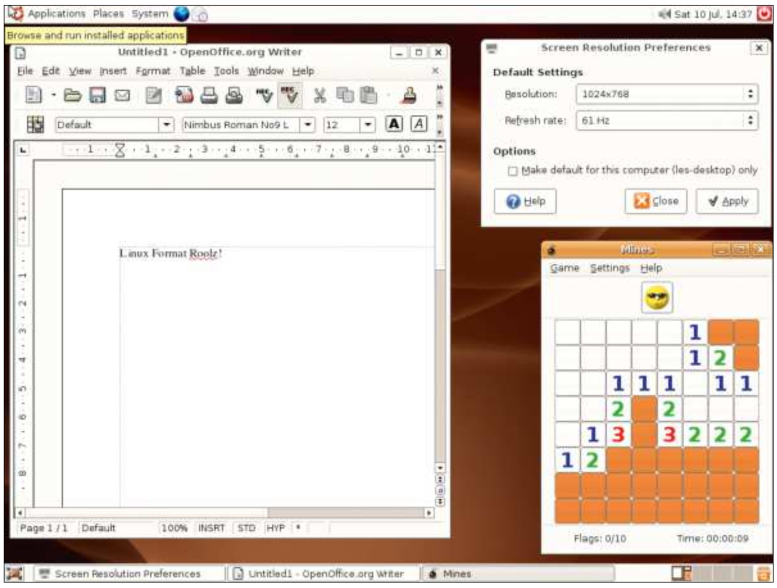
Capacitors may need replacing, and there may be leakage damage to the motherboard, so read up on how to fix this and test on a non-visible area before moving on. The old beige cases may need some Retrobright which requires UV light and hydrogen peroxide in order to bleach the plastic to the desired colour. Older hard drives are also prone to dying, so perhaps consider an IDE to SATA or IDE to compact flash/SD card solution.

On first boot we go through a typical Debian configuration process. And yes it does say LILO: this author remembers having to learn how to configure that to get their system to boot.

The Debian configuration process starts by setting a root password, then we create a new user account for general access. When prompted to set up shadow passwords, you can select yes or no – it makes no difference. Do remove PCMCIA drivers if prompted. Next we're asked if we'd like to set up PPP, used for connecting to the internet via a modem. Answer N.

The next screen asks if we'd like to run the *dselect* program, used to automate installation. Select No and press Enter. If the *dselect* screen appears anyway, press Q and Enter to drop into a Debian login. Enter your username and password and start using a few commands to mess around with the system. Note that our user has no access to the sudo group; rather, we need to switch user to root in order to make any system-wide changes.

Debian was a refreshing distro at this time. It was easy to install and a joy to use. But our virtual time machine is ready to make a leap forward, to a time when the GUI installer was a big deal to sell the distro.


Applications Places System
Sat 10 Jul, 14:37
Browse and run installed applications
Untitled1 - OpenOffice.org Writer
File Edit View Insert Format Table Tools Window Help
Default
Nimbus Roman No9 L
12
Page 1 / 1 Default 100% INSRT STD HYP
Screen Resolution Preferences
Help Close Apply
Mines
Game Settings Help
Flags: 0/10
Time: 00:00:09
Screen Resolution Preferences
Untitled1 - OpenOffice.org Writer
Mines

Linux Format Roolz!

Default Settings
Resolution: 1024x768
Refresh rate: 61 Hz
Options
☐ Make default for this computer (les-desktop) only

Ubuntu 6.06 "Dapper Drake" was the first LTS from Canonical. It changed how Ubuntu was perceived and it set the standard for many future releases.

The 2000s – Ubuntu 6.06 Dapper Drake

The first LTS (Long Term Support) release of Ubuntu was a big deal in 2006. It saw Canonical take a stance as a serious player in Linux, and saw many adopting the Debian-based distro as their main OS. In 2006 this author had just been playing with 5.10 for a few months and was ready to make the switch from SUSE. Dapper Drake came along, and despite being a shade of brown they adopted it for their daily workflow. Let's take a look at it all over again. The Ubuntu installation process in 6.06 is barebones and really effective. Fifteen years after it was released it still feels fresh and new.

Download the 32-bit ISO from **http://old-releases.ubuntu.com/releases/6.06.0** before opening *Virtual Machine Manager* and creating a new machine.

In the first screen Create a New Virtual Machine, select Local Install Media and click Forward. In the next screen navigate to the location of the downloaded ISO image by clicking Browse, and then Browse Local. Once you've selected the volume the operating system name should auto-populate to Ubuntu 6.06. Click Forward. Set the RAM to 1024MB and the number of CPUs to two. This is more than enough for Ubuntu 6.06. Click Forward and in the next screen click Finish to set the configuration and start the machine from the Live CD. Now we follow a typical Ubuntu install.

Ubuntu 6.06 should run fine on the default settings, but should we need to tweak them we can break out of the virtual machine by pressing Ctrl+Alt+G and click the blue information icon in the viewer window to open a configuration menu. We can change the RAM, CPU type and insert virtual/real devices into the virtual machine. Note that every change must be applied before moving to the next item in the configuration. Changes to the configuration take effect the next time the virtual machine is power cycled.

Installing software and getting online is tricky. Most of the software repositories are now unavailable, so we need to install programs by compiling the code. Getting online is possible, but this version of *Firefox* is no longer supported and so we can't access any HTTPS sites. But we can still relive the glory years of Ubuntu's rise to dominance, and that lovely shade of brown.

The 2010s – the rise of Linux Mint

Ubuntu had been the darling of the Linux distros for some time, but in the 2010s some grew upset with the introduction of Unity and an aubergine-based colour scheme. Linux Mint was seen as an alternative. It had a similar back-end to Ubuntu, but a more traditional user interface. Linux Mint also offered pre-installed codecs for popular music and video formats. So via the super simple installer, we can quickly get up and running. We downloaded a suitable ISO from **https://mirrors.evowise.com/linuxmint/stable/17.3**.

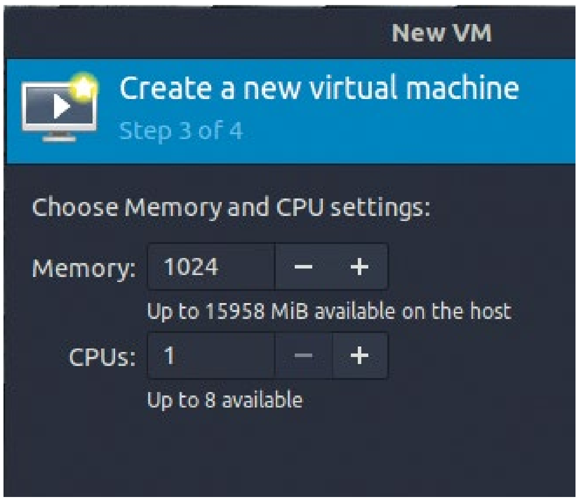
Setting up *QEMU* for Linux Mint is the easiest of them all. In the first screen choose Create a New Virtual

» VIRTUALBOX

QEMU isn't the only way to create virtual machines. *VirtualBox* from Oracle is a tried and trusted means to create virtual machines, but it can't emulate older hardware as effectively as *QEMU*. Installing *VirtualBox* is a simple task of downloading the latest version from **www.virtualbox.org/wiki/Downloads** and then following the installation guidance. We've successfully virtualised many Linux distros, including Linux Mint and Fedora with *VirtualBox*, and we have also virtualised Windows installs for projects.

VirtualBox has a straightforward process in which we create new machines. During the process we can create virtual hard disks that are either a fixed size, for example an 8GB image file, or we can dynamically create a file hard disk file that will grow to the maximum that the host OS expects. Once a virtual machine has been created, we're free to tweak the RAM, number of CPU cores, VT-X or AMD-V acceleration and add ISO images for boot. We can also create shared folders with the host system, enabling easier file transfer. Networking is also handled for us, with no configuration necessary.

VirtualBox is proprietary software and as such it may not be for you. If not, then as we have shown in this feature, *QEMU* is more than up to your virtualisation needs.



New VM

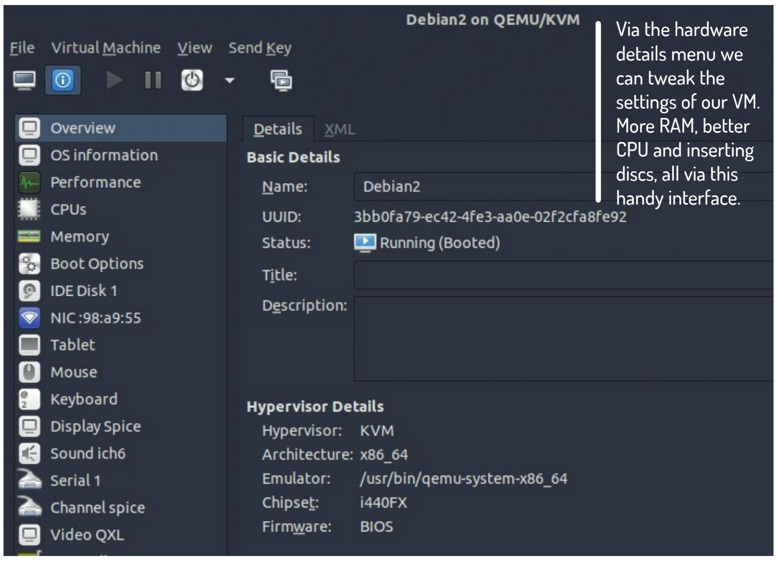
Create a new virtual machine
Step 3 of 4

Choose Memory and CPU settings:

Memory: 1024 − +
Up to 15958 MiB available on the host

CPUs: 1 − +
Up to 8 available

When creating a new virtual machine we can set the RAM and CPU. If the OS is detected during the process, it'll suggest the ideal values.



Debian2 on QEMU/KVM

Basic Details

Name: Debian2
UUID: 3bb0fa79-ec42-4fe3-aa0e-02f2cfa8fe92
Status: Running (Booted)
Title:
Description:

Hypervisor Details

Hypervisor: KVM
Architecture: x86_64
Emulator: /usr/bin/qemu-system-x86_64
Chipset: i440FX
Firmware: BIOS

Via the hardware details menu we can tweak the settings of our VM. More RAM, better CPU and inserting discs, all via this handy interface.

Machine, select Local Install Media and click Forward. In the next screen navigate to the location of the downloaded ISO image by clicking Browse, and then Browse Local. Once you've selected the volume, set the operating system name to Generic Linux 2016 and click Forward. The RAM and CPU will be set to 2GB and two cores – this is based on our choice of "Generic Linux 2016" in the previous screen. This configuration is more than enough for Linux Mint 17. Click Forward and in the next screen click Finish to set the configuration and start the machine from the Live CD.

Now we follow a typical Linux Mint install, but there is a possibility that you might hit a snag. During our install process the live CD refused to start the X server. This required us to click the blue information icon in the viewer window, navigate to Video/QXL and change the video option to VGA. Apply and then power cycle the virtual machine. Everything went smoothly after this and we were even able to install software and browse the web via *Firefox*.

Running QEMU from the terminal

So far we've created hard drive images and installed distros using the *Virtual Machine Manager*. We can do the same thing using the terminal and *Crunchbang++* (**https://crunchbangplusplus.org**), a modern version of *Crunchbang Linux*.

First, we need to create a virtual hard drive. In this case it'll be a 10GB .img image called **cbpp.img** and it'll be stored in the current directory:

```
qemu-img create -f qcow2 cbpp.img 10G
```

We have a blank image ready for our install, so now we need to configure our machine. This is a long command, so we'll go through each option.

The first command invokes *QEMU* with 64-bit support, then we use the `-cdrom` option to tell *QEMU* where to find our *Crunchbang++* ISO. Using `-hda` we tell *QEMU* that we wish to use the **cbpp.img** virtual drive, which is in the current directory. We then set the RAM to 2GB using `-m 2048`, before setting up networking, enabling the virtual machine to access the internet.

```
qemu-system-x86_64 -cdrom /path/to/iso -hda ./cbpp.
img -m 2048 -netdev user,id=mynet0,hostfwd=t
cp::8080-:80 -device e1000,netdev=mynet0
```

Pressing Enter will run this command and start the boot. For a first boot we recommend running a live session to test that everything works. After that you can reboot the virtual machine and install to the virtual hard disk.

Linux today: The Undiscovered Country

Today, we're truly spoilt for choice. New Linux distros are coming out at an accelerated pace, including specialist distros for media production, networking and scientific projects. Older hardware is supported by lightweight distros such as MXLinux, which prolongs the life of older equipment. Power users aren't forgotten: we have bespoke gaming distros that use *Wine*, *Proton* or *DOSBox* to run Windows games with the latest RTX and RX GPUs. By looking to the Linux distros of the past we can see how our much-loved distros have grown. From the humble text-based installer of Debian to the slick guided installation process of modern distros, our Linux experience is very different now – but we can still see the rock-solid foundation under the hood. LXF

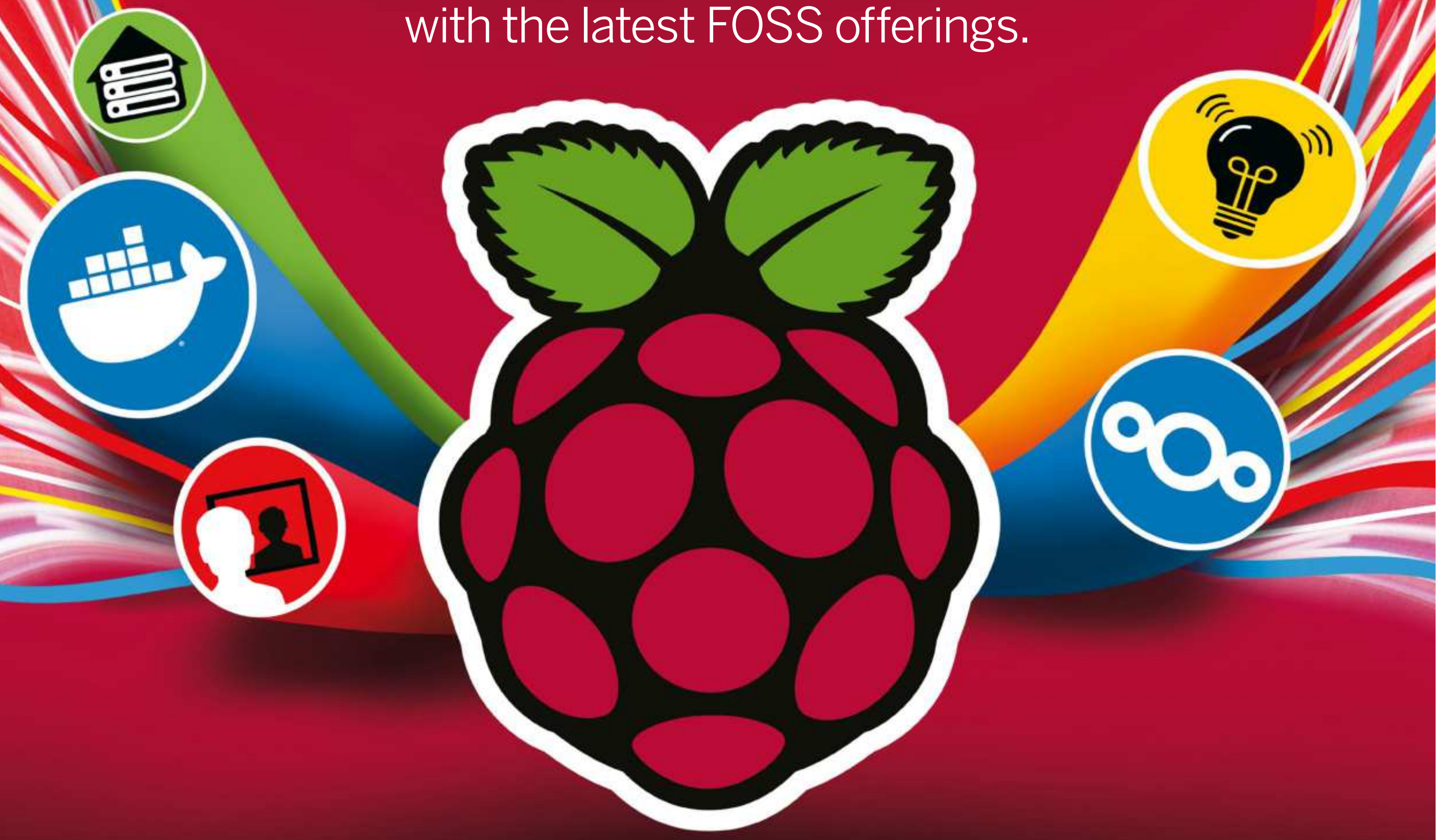
QUICK TIP

By default, Virtual Machine Manager will store all of the hard disk images in /var/lib/libvirt/images, but we can also specify our own destination when creating an image. Using the terminal it'll default to the current directory from which command is invoked.

Linux Mint 17 is a great balance of form and function. It looks great, and it came with everything we needed to enjoy our computer.

BUILD A SMART HOME OFFICE

Wiseguy **Jonni Bidwell** has some tips on making our homes and home offices smarter, with the latest FOSS offerings.



Creative ingenuity and the ability to cram a network stack into the smallest of things have gifted us, over the past few years, some pretty funky home innovations. Legacy (classical?) home dwellers may see little merit in upgrading to a smart home, but we have some uniquely Linux ideas that may persuade them.

Having a smart home needn't mean having Alexa (the voice of Amazon's Echo) eavesdropping on your every word. Nor does it mean the nameless entity within a Google Assistant reporting back to the mothership whenever you leave the bathroom light on. The Mycroft home assistant can do all the good bits of these voice devices, but without the more chilling data-collection aspects. Best of all, you can run Mycroft on a Raspberry Pi.

But there's more to smart homes than shouting at small robots. Once the colder months settle in, good citizens will be collectively donning long johns and firing up their boilers, heatpumps and sacrificial pyres. Some may find themselves having to shoulder the expense of a new boiler, others will want to see if their current one can be hacked and tweaked to make for a more economical winter. Through the magic of OpenHAB, we'll show you how.

For many of us, the COVID-19 pandemic has seen our homes shift and transform into unwitting makeshift extensions of the workplace, and depending on your line of work, that may be the case for the foreseeable future or maybe even permanent fixtures from now on. So we've got a couple of projects for smartening your home office setup, too. Recuse yourself from *Zoom* meetings and banish Google's office suite with the open source powerhouses *Jitsi Meet* and *Collabora Online*. And since Docker is all the rage these days, we will look at how we can harness its power better using the Portainer management engine. This will make it easy for you to add any service you can imagine to your self-hosted smart home office, so let's get started!

A smart home office?

Take a look at some great tools that can make a smart home smart, and a home office, er, official.

To the cynic, all this new smartification of everyday items seems to involve putting networked computers in places where they're not needed, and needing so many companion apps on one's smartphone to keep tabs on all these smart things. In so doing, one opens all these devices (and possibly your phone too) up to attack.

Some manufacturers are better than others at keeping their IoT things secured, and many attacks could be prevented by users changing default credentials. Since 2016, Raspberry Pi devices no longer ship with the SSH service enabled because it's too easy to spot them when they're exposed to the internet.

But there's a lot of smart home projects that are fun, safe and will bring some degree of joy to your home. Lots of these are variations on standard Pi maker projects, we really liked, for example Elio Struyf's offering documented over at **www.eliostruyf.com/diy-building-busy-light-show-microsoft-teams-presence**. This project utilises an LED array as a status indicator for outside your home office.

Watch this!

The pandemic has gotten lots of us quite used to video conferencing, and with that comes the terrible fear of children (or grown up children) interrupting your work chatter. Perhaps a brightly coloured warning light, that in this case hooks in to your Microsoft Teams account (it could be easily adapted to other services) to see if your status is "busy".

Home security systems are all the rage too. Again, the Raspberry Pi is excellent at this, you could adapt our Motion tutorial from **LXF266** to set this up, rather than going with an off-the-shelf product. Besides security concerns, such products could at any point stop being supported by the manufacturer, which makes them useless. This happened when Google bought Nest in 2014. Then in 2019 they announced they were shutting down the Works With Nest program, effectively bricking all the legacy Nest hardware that wasn't produced by Google.

In theory this could happen with open source projects too, but if it does, then at least the code is all there for someone else to take the reins.

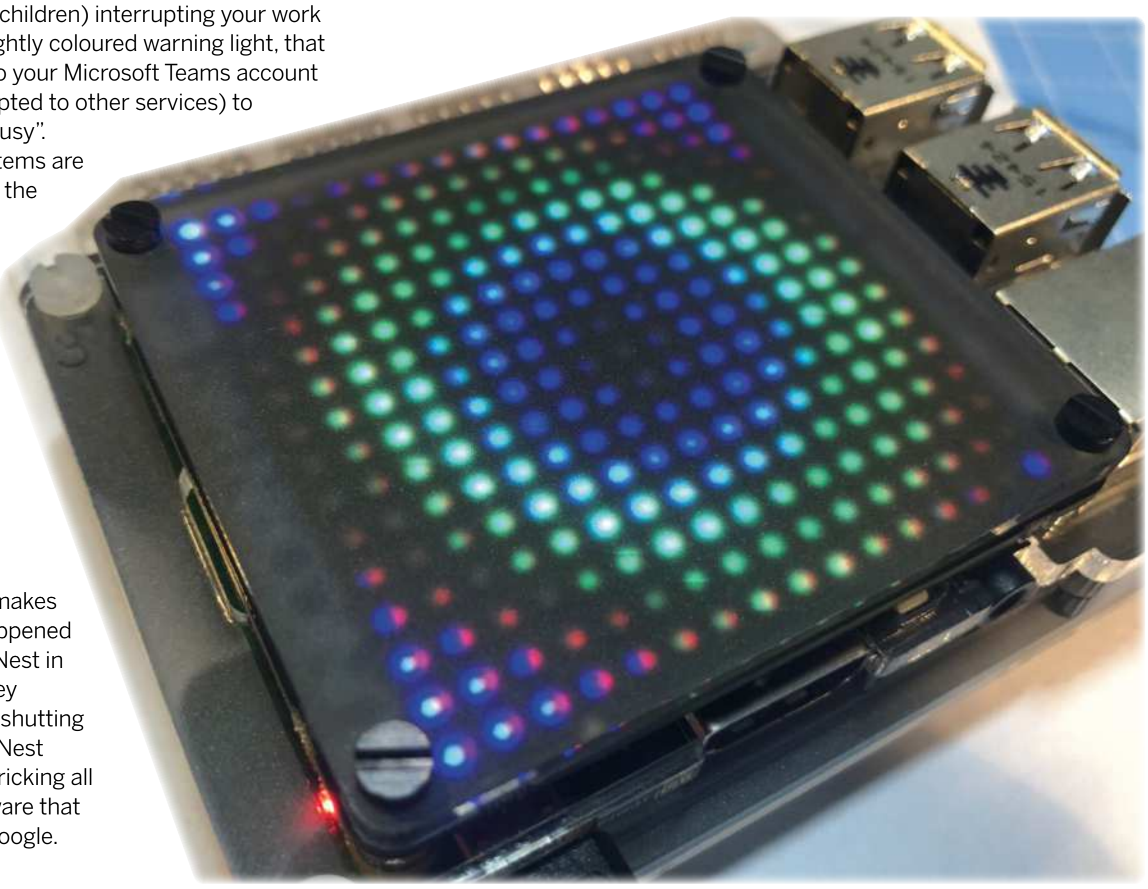
Hot smart stuff

Utility companies are keen for everyone to have smart meters now, and these go in tandem with smart boilers, smart thermostats, smart fridges. In theory, these can save you money, and it's certainly satisfying being able to pre-warm your house as you travel home.

We'll show you how you can connect all of these together through openHAB, the open source Home Automation Bus. Additionally, no smart home would be complete without a Voice Assistant, so we'll revisit our old friend Mycroft over the page.

We're also embracing Docker for this feature so we'll look at using that to smarten up your home office set up, using the Nextcloud cloud productivity suite. Then we've got some tips on containing this container overload with Portainer, which is a management engine used to constrain your container overload!

We'll also show you how to set up your very own video-conferencing service with *Jitsi Meet*. It's going to be an action-packed few pages, so brew yourself a nice cup of tea and get settled!


The Unicorn HD, with its 256 RGB LEDs and Python interface, could make an excellent warning light.

»

OpenHAB and Mycroft

Use the power of Ubuntu's openHAB appliance and the Mycroft Voice Assistant to voice-enable your home.

The open Home Automation Bus (openHAB) can connect all the diverse IoT appliances that may or may not be beginning to proliferate around your home. Typically, these devices all have their own, often proprietary applications, but often under the bonnet they use open-source protocols – and when they don't, people have figured them out enough to plug them into openHAB. We've covered installing it (and indeed Mycroft over the page) in the pages of **Linux Format** before, but this time we're going to do it a little differently because Canonical has released the openHAB Ubuntu Appliance for the Raspberry Pi. This means we can write an SD card image, boot it and have a fully functional web-based home automation server.

In order to use these Ubuntu Appliances (at the time of writing, there were four others besides openHAB and more are on the way), you'll need to sign up for Ubuntu SSO (Single Sign On) at **https://login.ubuntu.com**. It's free and you can use it all over the Ubuntu web ecosystem. Once you've done this, you'll want to generate an SSH keypair and upload the public key so that you can use your SSH-reinforced SSO to login to your appliance. You may already have generated an SSH key, in which case you probably don't want to overwrite it. Check by running

```
$ ls ~/.ssh/id_rsa.pub
```

If no such file exists, generate it with

```
$ ssh-keygen -t rsa
```

and press Enter to accept the default location. Once the key is generated, run `cat ~/.ssh/id_rsa.pub` and copy the contents to the clipboard. Log into Ubuntu SSO and go to the SSH keys section. Paste the contents into the Public SSH Key box and hit the import button. You can now SSH into Ubuntu services and appliances (from this machine only) using this key. But we still need to log our appliance into Ubuntu's cloud to fetch it, so follow the steps below and do that.

Once setup is complete, take a note of your Pi's IP address and connect to it from a browser by entering, for example, **http://192.168.0.2:8080**. The openHAB interface is a little confusing at first, especially given that the first step is 'choose an interface'. You can always return to this index page to explore the others. Choose PaperUI and peruse what Bindings (in the Add-ons section) are available, and if any of them match your hardware.

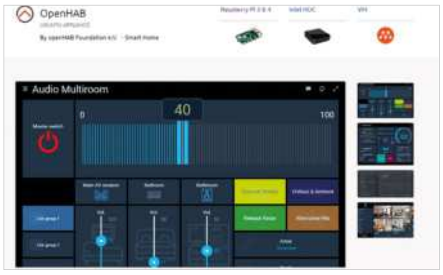
From here, your best friend is the openHAB documentation at **www.openhab.org/docs**. You can customise your appliance however you like. It's running Ubuntu Core so you can add Snaps from the repos. Instead of following our later section on Nextcloud, you can get a working install by SSHing into the appliance (the username is the first part of your Ubuntu SSO email) and running:

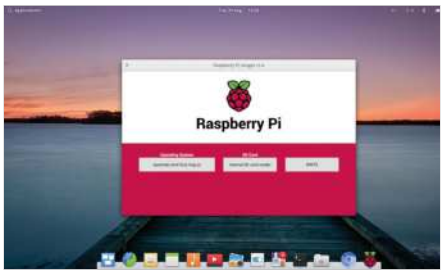
```
$ snap install nextcloud
```

Visit the Snapcraft store (**https://snapcraft.io**) and let us know how you augmented your appliance. Oh, and do check out the others; currently *Plex*, Nextcloud, *Mosquitto* and *AdGuard* have been applianced, and more will surely follow.

You might already know that Sherlock Holmes had an older brother called Mycroft – and that's where the

INSTALL OPENHAB





1 Download

Visit **http://bit.ly/LXF268openhab** and download the image. It's about 500MB, but compressed. You'll need to decompress it with `xz -d openhab-core18-pi.img.xz` if you're going to write it out manually with *dd* in the next step, but the official Raspberry Pi Imager program can do this for you.

2 Write the image to SD card

At least a 4GB card will be required, and possibly more if you go all out with extensions. On a Debian-based distro (including Ubuntu) you can download the Raspberry Pi Imager from **https://downloads.raspberrypi.org/imager** and install it with `dpkg -i` or graphically with *Gdebi*. Use this to write out the image file.

3 Boot your Pi

Connect a monitor, keyboard and network cable (optional) to your Pi, insert the freshly minted card, and fire it up. You will see some configuration choices and eventually will be asked for your Ubuntu cloud email address (see above) so that the appliance can be authenticated against your SSH keys.

Mycroft Home Assistant gets its name. The Mark I device was a friendly, ET-looking voice assistant that was powered by a Raspberry Pi 3. We were big fans, but sadly that device is no longer available, and its successor has not yet been released. But that's fine, because Mycroft's brains (or perhaps we should say OS?) are open source, and you can run them right now on a Raspberry Pi 3 or 4.

Buster rhyme

It's based on the official Raspberry Pi Linux Buster image, and you can download it ay **https://github.com/MycroftAI/enclosure-picroft**. Use the Raspberry Pi Imager again (or use something else, if you like) to write the image to an SD card. This time you'll need at least an 8GB card. When it's done, put the SD card into the Pi and fire it up. As with the openHAB appliance, you'll need a display and keyboard connected to the Pi to run the initial setup, but afterwards these will not be needed unless something goes wrong. On first boot, Picroft will resize its partition to use all available space on the SD card. When it finishes booting a second time, it will ask if you would like some help setting up your system, which you may as well accept.

You'll need a microphone and speakers, and there's a list of supported hardware over at the Mycroft wiki (**http://bit.ly/LXF268picroft**). Some of this will work out of the box, but some devices will need manual intervention. We used a Seeed Respeaker 4-microphone array HAT, which was easy to set up. Other devices will follow a similar pattern. Just run

```
$ git clone https://github.com/respeaker/seeed-voicecard.git
$ cd seeed-voicecard
$ sudo ./install.sh
```

If your network works (try rebooting if it doesn't!) you could also SSH into your Pi to do this (the default username is 'mycroft' with password 'pi', which you should change soon).

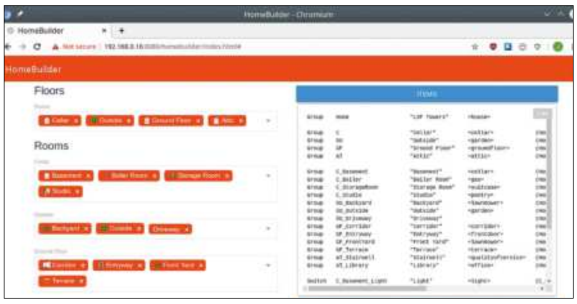
Then, if you're using the 3.5mm jack for audio, you may need to run `sudo raspi-config`, select '7. Advanced Options', 'A4 Audio', and finally '1. Force 3.5mm jack'. Now reboot to enact the changes. If you run into difficulties with the Respeaker (or other Seeed models), the driver documentation is at **https://github.com/respeaker/seeed-voicecard**.

Reboot and you can pair your device to the Mycroft Home service. You need to set up an account here to use the software, but hey, at least it's not Facebook. Only your queries are collected, and these are used to better train speech recognition. Just follow the prompts to set this up at **https://home.mycroft.ai** from another device; if your audio is working, you should hear some actual verbiage at this point. If not, you can rerun the setup wizard at any time with `mycroft-setup-wizard`.

Once everything's working, the Mycroft CLI client will start, and you can talk and type at it to your heart's content. Shout the magic wakeword ,"Hey Mycroft," to prick up it's digital ears, then try, "Tell me a joke", "What is my IP address?" or "How much wood would a woodchuck chuck if a woodchuck could chuck wood?" He likes it when you say "Thank you", too. Responses are again both verbal and on screen/terminal. You can quit the CLI at any point with `:quit` and then you can run anything you'd run on a normal Raspberry Pi Linux install. If you want to get back into the CLI, run

```
$ mycroft-cli-client
```

There's a custom audio setup script called **audio_setup.sh** and don't forget to check the excellent documentation at **https://mycroft-ai.gitbook.io/docs**. Mycroft can be installed on a regular Linux PC either as a Snap Package, through Docker, or straight from their GitHub. The wake word engine, known as Precise, uses TensorFlow for cutting edge accuracy, which requires the host to have AVX extensions; most machines from the last decade will.



The HomeBuilder UI allows you to build your own virtual LXF Towers, complete with mysterious library.

An openHAB skill is available for Mycroft, so you can have your whole house respond to your vocal utterances and edicts.

» POWER AND PRIVACY

There's still a couple of first generation Raspberry Pis keeping things ticking over here at **Linux Format**. These older models are great – such as our Pi 2 running *Volumio* – but lots of projects (such as Mycroft) will require more CPU power, so for a lot of projects you'll see at least a Pi 3 is recommended. If you have a Pi 4, you can even run a *Jitsi* instance that can handle a few callers with no problems. The trouble is, these later models require much more power, so it's important to use a power supply (ideally the official model) that can provide a steady 5.1V supply at up to 2.5A. Cheap USB power supplies can cause all kinds of hard-to-diagnose problems, so if something's behaving oddly, the power is a good thing to check.

Also we're doing a lot of setting up of web services here, which has the potential to invite the whole world into your machine. So beware of leaving things running with default credentials, and do look at firewalling things that only need to run on the local network. You may want to run things in a virtual machine, or on a 'disposable' virtual private server to get a feel for them first.

It's your responsibility to maintain these services and keep things up to date, too. The journey is far from over once things are up and running. Many companies offer, for example, hosted Nextcloud storage, and this may be a better option for small businesses that don't want the burden of administering a server.

»

Jitsi Meet

Set up, secure and self-host your way to video conferencing nirvana, and you'll never again have to use Zoom.

J*itsi Meet* is a completely free, open-source video conferencing program that you can connect to via a browser or smartphone app. You can use it right now by visiting **https://meet.jit.si** and choosing a name for your virtual meeting (or indeed accepting one of the amusing ones the website randomly generates). There are no downloads or configuration required by clients, they just require a link to be clicked which you can share by email from within the meeting room. Or attendees can dial in and connect with a PIN. Simple.

Obviously, Jitsi's free service has some limitations, but these will be gone when we self-host our own *Jitsi* installation. You don't need any particularly special hardware to do this, and thanks to the excellent Debian and Ubuntu packaging, you don't really need to do much in the way of configuration at all. You will need a domain name, though; a free one from a dynamic DNS service such as DuckDNS or Dyndns will work.

As you can imagine (and we'll look at this in a moment if you can't), under the bonnet *Jitsi* is a rather complicated affair. Not only does it have to wrangle the vagaries of streaming through NAT gateways (such as home routers), it also needs to keep all this secure with SSL, so if you cheat and use a self-signed certificate, your users will see scary warnings.

If you're happy for you and your users to see these warnings (or brave enough to add an exception to your browser), and your users are all local, you can get away with using a bogus domain name and fudging users' **/etc/hosts** files to resolve this to your machine. Or even just use IP addresses directly. But it's probably easier in that case to have an in-person conversation rather than involving the machines... but where is the fun in that? If you have a domain, the install process can automatically generate Let's Encrypt certificates, and then no one needs to see any security warnings.

A *Jitsi* install is, in fact, a collection of several components, all working in harmony to bring bright smiley faces closer in a world that grows ever more distant. Firstly, it relies on a web server (and some pretty wild JavaScript) to provide the web frontend. You can use Nginx or Apache here. Most of the software is written in Java, so keen-eyed dependency spotters will note the OpenJDK runtime making an appearance.

The magic of connecting more than two participants is handled by Jitsi VideoBridge (JVB). This uses the XMPP (like Jabber) and WebRTC (like Google Meet) protocols to swiftly route video to call participants. Deciding which participant is the speaker, and hence who users see 'focused' on their screens, is handled by Jicofo (Jitsi Conference Focus). Note that this is as close to video processing as *Jitsi* gets – all the encoding and decoding is handled on the user side. At the heart of it all, the Prosody XMPP server keeps all these components and users connected. You'll find more information about these and everything else *Jitsi* in the Handbook at **https://jitsi.github.io/handbook**.

Jitsi? Probably too much coffee

In order for *Jitsi* to work for users stuck behind NAT, firewalls or other obstacles in the stack, XMPP messages and web traffic must be able to flow freely to your server. So you'll need to open UDP port 10000 and TCP ports 80 and 443. TCP port 4443 can be used for fallback communications (if UDP connections are blocked for a user). If you're doing this behind a home router, you'll need to forward these ports.

In an ideal world, one-to-one calls should bypass the JVB, and connect directly to one another (so-called p2p mode), but this may be tricky to negotiate – as anyone who ever tried to share files over *MSN Messenger* in the early aughts will tell you. Prosody can work with a TURN (or STUN) server to facilitate this, but we won't cover this setup here.

We will, however, get started with a simple setup after one final, but pertinent, warning: the default *Jitsi* configuration allows anyone to start or join a meeting on your server. If you leave your server running in this way for any length of time, sooner or later people or bots may discover your machine and potentially cause trouble. The handbook covers how to set up authentication, which you should do. You should also check the philosophy behind this openness at **https://jitsi.org/blog/security**. You'll be fine using this guide to

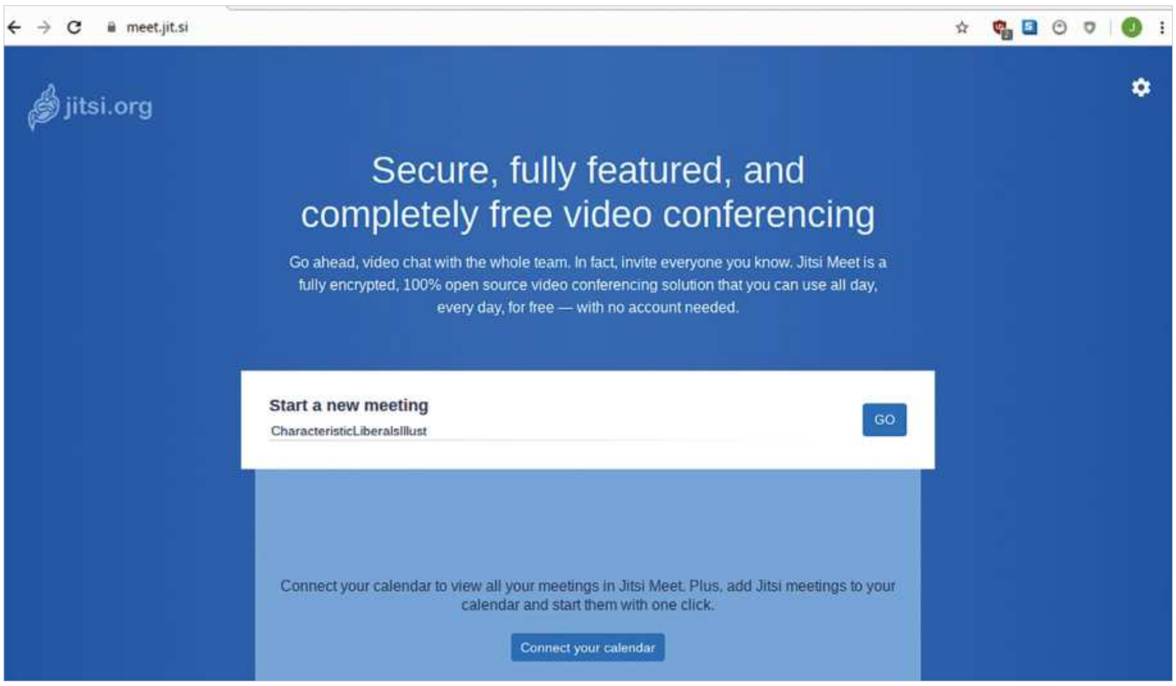
» WHAT IF IT DOESN'T WORK?

If you can join a room from multiple locations but don't see any other participants and are stuck with a message telling you you're the only one here, port forwarding is almost certainly the culprit. In some configurations, the extra port 4443 will be necessary. Your next stop should be the log files. JVB and Jicofo's are located in **/var/log/jitsi** and Prosody's is in **/var/log/prosody**. It's handy to delete these after a configuration change so that you don't get confused by previous logs. Alternatively, use `tail` to view the last few entries. After any configuration change, you'll want to ensure that you restart all the services, which is a matter of:

```
$ sudo systemctl restart jitsi-videobridge2 jicofo prosody
```

You might want to add `nginx` to that list, particularly if you've changed anything that will affect the frontend, such as the authentication we added earlier. And if it all goes wrong, you can purge all *Jitsi* data and configuration with:

```
$ sudo apt purge jigasi jitsi-meet jitsi-meet-web-config jitsi-
meet-prosody jitsi-meet-turnserver jitsi-meet-web jicofo jitsi-
videobridge2
```



Secure, fully featured, and completely free video conferencing

Go ahead, video chat with the whole team. In fact, invite everyone you know. Jitsi Meet is a fully encrypted, 100% open source video conferencing solution that you can use all day, every day, for free — with no account needed.

Start a new meeting

CharacteristicLiberalsIllust

GO

Connect your calendar to view all your meetings in Jitsi Meet. Plus, add Jitsi meetings to your calendar and start them with one click.

Connect your calendar

Jitsi is free, easy to use, and its random name generation algorithm has been a constant source of amusement throughout otherwise unamusing times.

get things up and running, but just make sure you stop the services when they're no longer needed.

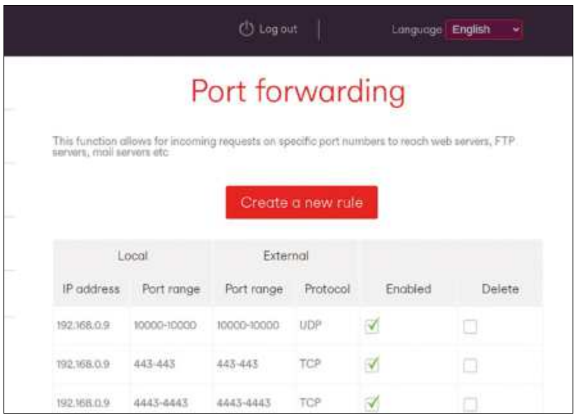
And now we can get on with installation. We're more or less following the Self-Hosting Guide in the Handbook, so do check there if you run into difficulty. First add the *Jitsi* repo with:

```
$ https://download.jitsi.org/jitsi-key.gpg.key | sudo sh -c 'gpg --dearmor > /usr/share/keyrings/jitsi-keyring.gpg'
$ echo 'deb [signed-by=/usr/share/keyrings/jitsi-keyring.gpg] https://download.jitsi.org stable/' | sudo tee /etc/apt/sources.list.d/jitsi-stable.list
$ sudo apt update
```

Then let the fireworks begin with:

```
$ sudo apt install jitsi-meet
```

During package configuration, you'll be asked for your Pi's hostname, and whether you want to generate a new, self-signed certificate or use an existing one. Choose the first option if you don't already have a valid one for this domain. We'll replace it with a Let's Encrypt certificate later. At this point, you should be able to log in to your server using your web browser. Just point it to **https://meet.yourdomain.com** and you'll see the scary certificate warning we warned you about earlier. You can



Port forwarding

This function allows for incoming requests on specific port numbers to reach web servers, FTP servers, mail servers etc

Local		External			
IP address	Port range	Port range	Protocol	Enabled	Delete
192.168.0.9	10000-10000	10000-10000	UDP	✓	☐
192.168.0.9	443-443	443-443	TCP	✓	☐
192.168.0.9	4443-4443	4443-4443	TCP	✓	☐

If you don't get your ports right, Jitsi will not play nice.

skip this for now (just this once) and you will be greeted by the *Jitsi* welcome screen, from where you can start a new meeting. Your web browser will ask for permission to use the camera and microphone, and you should see a message saying that you are the rightful moderator of this room.

Being alone in a room is no fun, so on another device (or get a friend to do this, if you have such a thing) visit the meeting by appending a slash followed by the room name to the server's URL.

With that working, we can now address our server's slightly too Open for Business attitude and add some authentication. Edit the prosody configuration at **/etc/prosody/conf.d/meet.yourdomain.com.cfg.lua** (replacing the domain name as appropriate). In the section beginning `VirtualHost "meet.yourdomain.com"` change the authentication line, currently set to anonymous, so that it reads:

```
    authentication = "internal_hashed"
```

Then add (and adjust the domain in) the following at the end of the file to set up a guest domain:

```
VirtualHost "guest.meet.yourdomain.com"
    authentication = "anonymous"
    c2s_require_encryption = false
```

Don't worry, the extra subdomains configured here are only used internally so they don't need a bona fide DNS record. Now we'll go from Lua to JavaScript and add the guest domain to the frontend. Edit **/etc/jitsi/meet/meet.yourdomain.com-config.js** and add the following lines to the hosts section:

```
    anonymousdomain: 'guest.meet.yourdomain.com',
```

Finally we should make focus requests only available to registered users, which is done by editing **/etc/jitsi/jicofo/sip-communicator.properties** and adding:

```
org.jitsi.jicofo.auth.URL=XMPP:meet.yourdomain.com
```

Now restart the services and add a user with the `prosodyctl register` command. You can now generate a Let's Encrypt certificate with:

```
$ sudo /usr/share/jitsi-meet/scripts/install-letsencrypt-cert.sh
```

Nextcloud

We've been relying on Nextcloud to keep our files in sync for ages, and we think you should too.

Regular Linux Format readers will be familiar with our ranting and raving about how great Nextcloud is. And it keeps getting better – the latest edition of the self-hosted storage and sharing platform, Nextcloud 19, sees it transformed into a complete Collaboration Hub.

If you already have a LAMP (Linux, Apache, MySQL, PHP) or equivalent stack set up on a home server or VPS, you can easily add a Nextcloud instance to it, either by extracting an archive to your web server's root directory, or by copying the Web Installer PHP file there and running it directly. Setting up a LAMP stack (or indeed a LEMP stack with Nginx and MariaDB) is easy enough (although it's a bit of a faff having to create databases by hand) in Ubuntu or Debian – we've covered it before and there are countless guides a mere DuckDuckGo search away. This time around, we're going to install Nextcloud differently, via Docker. In theory, this will mean our instance will be portable, so you could go from running on your VPS to running on a home server, or vice versa, with relative ease. Data and configuration are stored in volumes outside the Docker container (which you should consider a fairly ephemeral, fungible thing), so migrating the service just involves copying those volumes and connecting them to a new instance of the container.

CONTAINERS ARE GREAT

"Multiple instantiations of the same container aren't really a space issue, since the layers of the union filesystem are shared between them."

First on the agenda is to install the Docker engine which requires adding their GPG key to the *apt* keyring:

```
$ curl -fsSL https://download.docker.com/linux/
ubuntu/gpg | sudo apt-key add -
$ sudo apt-key fingerprint 0EBFCD88
```

You should check the fingerprint of that key matches the one beginning 9DC8 and given in full in the documentation at **https://docs.docker.com/engine/install/ubuntu** (where you can also copy and paste these lengthy commands). Next add the Docker repos (change `focal` to `bionic` if you're using 18.04), update caches and install Docker with:

```
$ sudo add-apt-repository "deb [arch=amd64] https://
download.docker.com/linux/ubuntu focal stable"
$ sudo apt update
$ sudo apt install docker-ce docker-ce-cli containerd.io
```

Check that it's working by running the Hello World image, then fire up the Nextcloud image:

```
$ sudo docker run hello-world
$ sudo docker run -d -p 8080:80 nextcloud
```

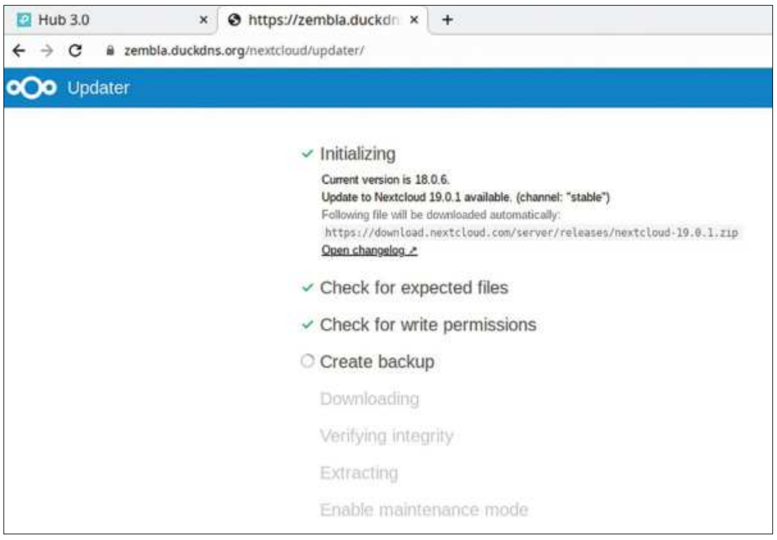
The first switch tells Docker to run the container detached (in the background), and the second tells it to expose the container's port 80 (where the webserver is running) to port 8080 on our server. So now – once a few hundred megs are downloaded and the container springs in to life – if you browse your way to **http://yourserver: 8080** (with your address or hostname) you should find the Nextcloud welcome screen. You'll see a warning about the container using SQLite; this lightweight database will be fine for a few users, but more popular setups will need a proper MySQL or MariaDB instance. Go back to the terminal, find your container's name (from the last column of the first command) and stop it with:

```
$ sudo docker ps -a
$ sudo docker stop yourcontainername
```

At this point, we could connect Nextcloud to a MariaDB container and have both store their data in named volumes, which will provide portability. The resulting volumes will be available in the **/var/lib/docker/volumes/** directory. We can also add boring names to our containers.

```
$ sudo docker run -d -p 8080:80 –name nextcloud -v
nextcloud:/var/www/html nextcloud
$ sudo docker run -d -v db:/var/lib/mysql –name mydb
-e MYSQL_ROOT_PASSWORD=secret1 mariadb
```

Getting these two containers to talk to each other is tricky, so we'll refer you to the Docker Compose section in the documentation here, but we'll return to this idea soon. Containers still continue to exist after they have been stopped. Multiple instantiations of the same one



✓ Initializing

Current version is 18.0.6.
Update to Nextcloud 19.0.1 available. (channel: "stable")
Following file will be downloaded automatically:
`https://download.nextcloud.com/server/releases/nextcloud-19.0.1.zip`
Open changelog ↗

✓ Check for expected files

✓ Check for write permissions

○ Create backup

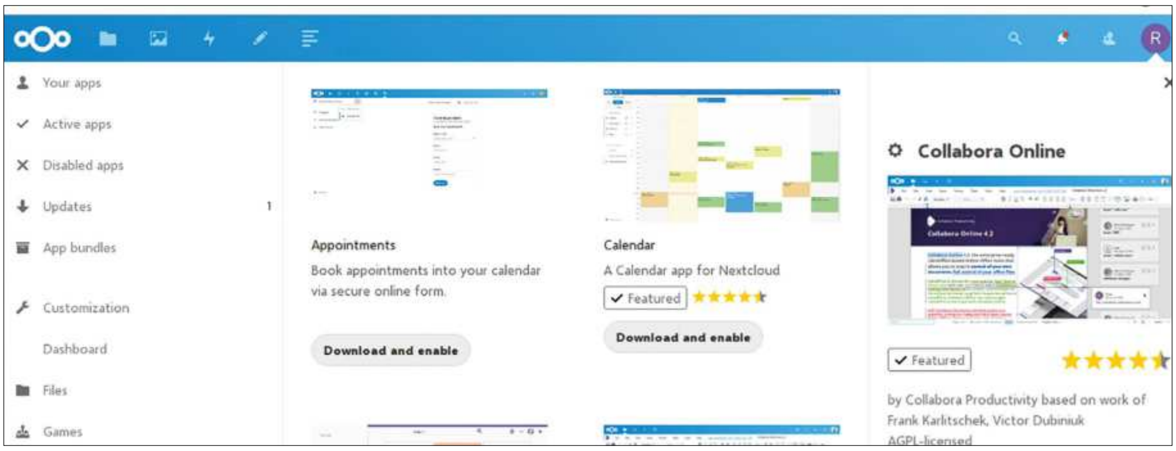
Downloading

Verifying integrity

Extracting

Enable maintenance mode

If you're already running an older edition of Nextcloud you can upgrade from the comfort of the web interface.



Jazz up your Nextcloud install with a dizzying array of add-ons. We're big fans of the News app, which is making RSS great again.

aren't really a space issue, since the layers of the union filesystem are shared between them. But it's nice to tidy up after yourself, so check your containers occasionally with `docker ps` and clear unwanted ones, such as our database above, with:

```
$ sudo docker rm mydb
```

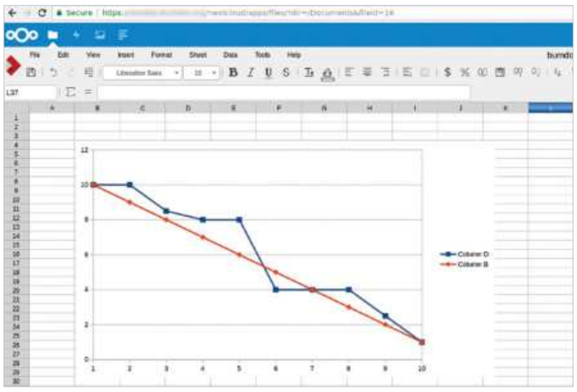
If you want to get rid of everything (danger, beware), you can do so with:

```
$ sudo docker rm $(sudo docker ps -aq)
```

You could at this point go back to the web installer and enter some administrative credentials, doggedly proceeding with the SQLite database, and reusing the first volume. But we're going to borrow from the community and use the example Docker Compose setup, skimming over a few details which you should check in the documentation. Running Docker this way uses four containers, one of which is a reverse proxy running Nginx, which can provide SSL encryption (which is essential if you make this installation available to the internet at large). Also existing here is a Let's Encrypt host to auto-update certificates. So if you've got a domain name setup, the whole thing will be seamless.

Grab the Compose recipe and environment from the Tutorials download, or fetch it from the nextcloud/docker repo on GitHub. Edit the **docker-compose** file, setting in particular the MySQL password, the virtual host (your domain name) and Let's Encrypt variables. Then follow the official documentation to install Docker Compose at **https://docs.docker.com/compose/install**, as this process may change. If something doesn't work, you can peer into the logs of each container, for example:

```
$ sudo docker logs compose_proxy_1
```



Spreadsheets and charts within web applications and without Google. We're pretty sure Collabora is onto a good thing here.

One way or another, you'll find your way back to the Nextcloud setup screen, and then all further configuration will be free of terminal commands. You just need to choose an admin password, use `db` for the database name (it uses Docker's internal networking so doesn't need a domain name) and wait a few moments for everything to be initialised. Your first step should be to create a new user, which you can do from the Users section in the drop-down menu in the top-right of the display. It's better to keep the admin account for, well, admin. While you're here, you can use your useful administrative privileges to see how apps are added to Nextcloud. Select Apps from the user menu in the top right, and you'll find a categorised list of them.

It's good practice to keep your containers updated with the command:

```
$ docker-compose pull
```

from the directory with the **docker-compose.yml** file.

COLLABORA ONLINE

At the time of writing Collabora has released the latest version of its enterprise edition of *LibreOffice*: *Collabora Office 6.4* (see **www.collaboraoffice.com**). Alongside that offering is another enterprise release, *Collabora Online*, which provides an online version of the office suite that can be plugged into a file sync and share host via the Web Application Open Platform Interface (WOPI).

A free version of *Collabora Online* – the Development Edition, aimed at home users – is available, and it can be connected to your Nextcloud instance to provide functionality not entirely dissimilar to Google Docs.

The easiest way to set it up is to use the Docker image and follow the instructions at **https://nextcloud.com/collaboraonline**. The CODE docker image listens on **localhost:9980**, so for this to be accessible to the outside world the instructions use an Apache reverse proxy. It's recommended to have this proxy run on a different domain (the same machine is fine) than your Nextcloud host, which is a little complicated because for this to work smoothly you need to have valid SSL certificates for both.

It's possible to use the same domain (at a cost to security), or self-signed certificates, but you may have to be creative in your approach here. Once you've done the hard work, install the *Collabora Online* app using your Nextcloud admin account, and then configure it with the domain where the proxy is listening.

We haven't heard the term 'groupware' since Novell Netware, but we're glad Collabora is bringing back the term with such style.

Portainer & Docker

Contain your sprawling smart home office with Portainer, then ruminate on a fully featured email server with Mailcow.

When you're running a home server or VPS, Docker is pretty great because you can add services without messing (too much) with the underlying system configuration. As we saw with Nextcloud though, even with just four Docker containers things were starting to become unwieldy.

EMAIL MADE EASY

"It's often been said (including by us) that setting up an email server properly is hard. This is true, but we can set up a rough and ready container."

Portainer is a Docker management system that aims to restore order to your containers.

It comes in two parts, server and agent, which both ship as Docker containers. We'll only need the server image for this initial foray. The agent is used to marshall Docker Swarms, which are another way of connecting containers together. We'll start by creating a volume for data, and then firing up the image.

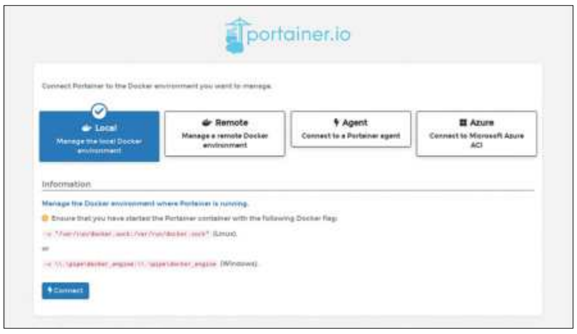
```
$ docker volume create portainer_data
$ docker run -d -p 9000:9000 --name=portainer
--restart=always -v /var/run/docker.sock:/var/run/
docker.sock -v portainer_data:/data portainer/
portainer
```

Note that images can be added according to their listing on Docker Hub, so you can find out more about the Portainer image at **https://hub.docker.com/r/portainer/portainer**. You should find Portainer's admin panel now waiting for you on port 9000, so point your browser there, choose a password, select 'Manage the local Docker environment' and enjoy the friendly interface. If you installed Nextcloud through Docker earlier, you should see all those containers listed in the Local Endpoint on the Home screen. If you click Containers on the menu on the left, you'll see these (and our latest Portainer container) in more detail. The quick actions column allows you to see each container's logs, info, stats and also log in to each one directly from the browser. This is a great way to do some troubleshooting, but recall that containers in general are quite minimal, so you won't find any fancy utilities.

Adding a new container is easy. For example, if you want to add Collabora's CODE and connect it with a Nextcloud install, hit the Add Container button and enter a meaningful name. In the Image field, enter `collabora/code`, and in the 'Manual network port publishing' section enter `127.0.0.1:9980` in the host field and `9980` in the container field. In the Advanced container settings below, go to the Env section and set the following environment variables:

```
domain: share\\yourdomain\\.com
username: admin
password: yourpassword
```

Finally in the Restart policy column select Always, so the container restarts on configuration change. Hit 'Deploy' and the container will spring into life. For guidance connecting it to your Nextcloud instance,


portainer.io
Local
Manage the local Docker environment
Remote
Manage a remote Docker environment
Agent
Connect to a Portainer agent
Azure
Connect to Microsoft Azure ACI
Connect

Connect Portainer to the Docker environment you want to manage.

Information

Manage the Docker environment where Portainer is running.

Ensure that you have started the Portainer container with the following Docker flag:

`-v "/var/run/docker.sock:/var/run/docker.sock"` (Linux).

or

`-v \\.\pipe\docker_engine:\\.\pipe\docker_engine` (Windows).

Besides running locally, Portainer can be used to manage remote systems running its agent.

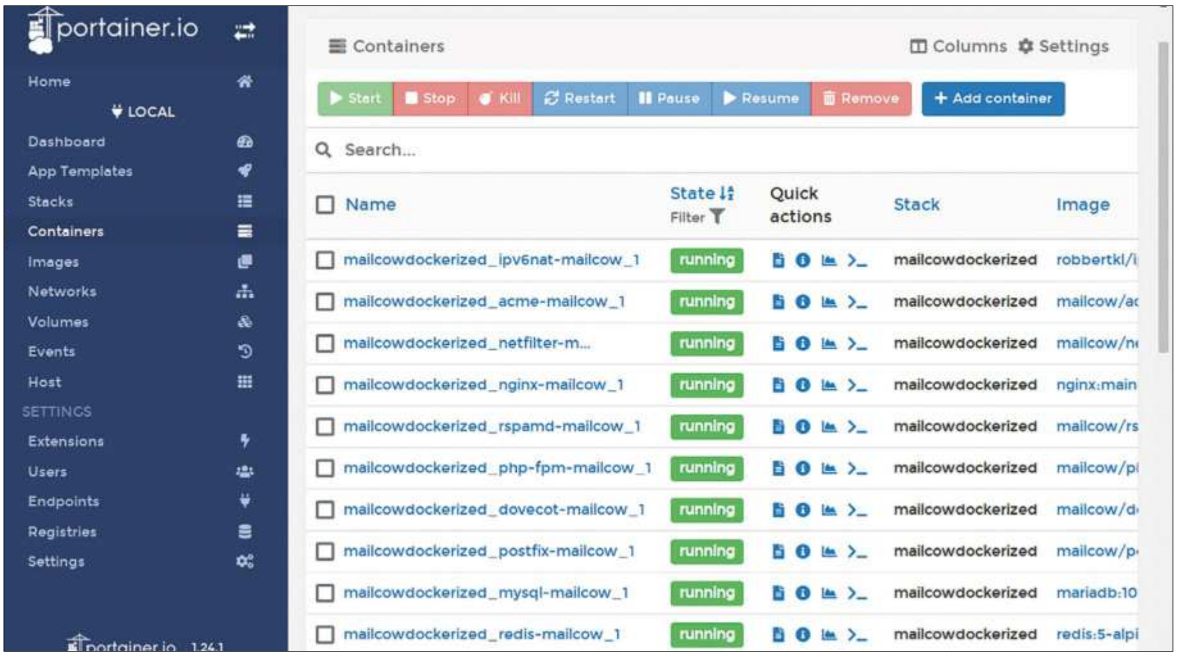
» JITSI Pi IN THE SKY/CLOUD

If you're feeling adventurous, it is possible to install *Jitsi Meet* on a Raspberry Pi 4 with 4GB (the 8GB version will perform better though). This isn't officially supported and setup is a little complicated. When we were writing this feature, our Pi 4s were all locked up in head office until such time as it was deemed Covid-safe.

We were feeling extravagant so we took advantage of Mythic Beasts' commitment-free, per-second billing and rented ourselves a remote Pi 4. Mythic Beasts' offerings use PXE booting and mount the root filesystem via NFS from their own storage arrays. This means you're not at the mercy of unreliable SD cards, and also allows you to rent up to 250GB of storage. We were more frugal, and went with the minimum 10GB; this costs just over £10 if it's provisioned for a whole month. With monthly billing it's £7.50.

If you're setting up *Jitsi* on your own Raspberry Pi locally, you might have to do a little detective work (or plug in a monitor) to find your Pi's IP address on your LAN. And if you use Mythic Beasts' offering, be aware that its Pi servers are connected by IPv6 only. A gateway is available so that IPv4-only users (that is, a large chunk of the UK's population) can still connect to it, but you'll need a real DNS record (DuckDNS doesn't count here) to use it for anything other than as a standard web host. We were a bit embarrassed that it took us a while to figure this one out.

If you're keen you can find a recipe to get everything working on the *Jitsi* GitHub at **http://bit.ly/LXF268jitsi**.



Name	State Filter	Quick actions	Stack	Image
mailcowdockerized_ipv6nat-mailcow_1	running		mailcowdockerized	robbertkl/i
mailcowdockerized_acme-mailcow_1	running		mailcowdockerized	mailcow/ac
mailcowdockerized_netfilter-m...	running		mailcowdockerized	mailcow/ne
mailcowdockerized_nginx-mailcow_1	running		mailcowdockerized	nginx:main
mailcowdockerized_rspamd-mailcow_1	running		mailcowdockerized	mailcow/rs
mailcowdockerized_php-fpm-mailcow_1	running		mailcowdockerized	mailcow/pl
mailcowdockerized_dovecot-mailcow_1	running		mailcowdockerized	mailcow/d
mailcowdockerized_postfix-mailcow_1	running		mailcowdockerized	mailcow/p
mailcowdockerized_mysql-mailcow_1	running		mailcowdockerized	mariadb:10
mailcowdockerized_redis-mailcow_1	running		mailcowdockerized	redis:5-alpi

Mailcow: dockerized involves about 20 containers, but you can disable a few of them.

check the documentation at **https://nextcloud.com/collaboraonline**. All going well, you should be able to specify it from Nextcloud, specifically from the Collabora Online section in Settings. You may need to tweak the reverse proxy container to direct traffic to the new container, if you used the Docker Compose example on the previous page.

In that case you'll find that example in the Stacks section with a warning about it being created outside of Portainer. In Portainer parlance, groups of containers are called Stacks, and it can only do limited things with externally created Stacks. However, it's easy to create a new Stack; just paste, upload or point Portainer to the URL of a **docker-compose.yml** file.

Nextcloud's own groupware, its Calendar, Talk and Mail applications, can be used for collaboration among users. The Mail app is just a client, so you set it up with a provider just as you would in a desktop email client. It's often been said (including by us at **Linux Format**) that setting up an email server is hard. This is true, but we can set up a rough-and-ready container running *Postfix*, *Dovecot* and *Roundcube* and have a working (but not necessarily well-configured) email set up in just a few keystrokes thanks to the Mailcow:Dockerized image.

Setting this up might offend the sensitivities of those adverse to running things as root, but the image uses extended attributes and requires it. So let's be a superuser and check our umask is set correctly (it should print 0022).

```
$ sudo -i
# umask
```

Now we'll clone the repository into **/opt**:

```
# cd /opt
# git clone https://github.com/mailcow/mailcow-dockerized
# cd mailcow-dockerized
# ./generate_config.sh
```

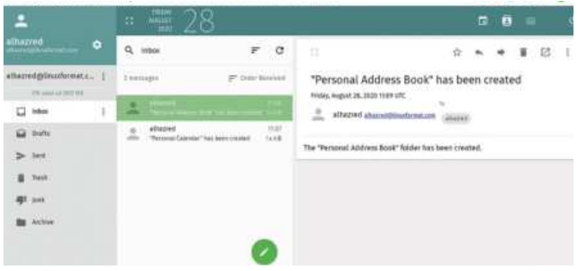
You'll be asked for your domain name; this is the host that it's running on, not the domain of any email addresses you want to set up. If your machine has limited memory the script will suggest you disable Solr (the heavy-duty Java search platform), which you should. The script generates **mailcow.conf** which you may wish to tweak. We'll just plough right ahead. First, in case they get in the way, stop Nextcloud and any other containers now. In Portainer do this by choosing Containers from the menu on the left, selecting all of them except Portainer (because that would be silly), and pushing the stop button. There's a Kill button next to it in case they don't shut down gracefully, and you can also remove them entirely if you don't like them.

Now return to the terminal (still using the root account) and let's fetch and unleash the Mailcow:

```
# docker-compose pull
# docker-compose up -d
```

At this point, you technically have a working mail server. You should now be able to log in from the web interface, using the credentials `admin:moohoo`. But there's still a lot of configuration to do. Not least of this involves changing that password, which can be done by clicking Edit next to the admin user to do this. If you want to receive mail you'll need to properly set up MX records for your domain(s) with your DNS provider, otherwise mail won't be routed correctly.

Once you've done that, you can configure them from Mailcow's Domains section, then you can configure the actual addresses in the Mailboxes section. In a world of endless spam, it's important to do your bit by enabling DKIM (Domain Keys Identified Mail), which signs messages originating from your server. DKIM-enabled servers will verify this key when they receive mail from your domain, preventing malfeasance. LXF



"Personal Address Book" has been created

The "Personal Address Book" folder has been created.

Once you've set up your mailboxes, you and your users can log into the SOGo webmail client by adding /SOGo to Mailcow's domain name.

ULTIMATE CODING PROJECTS »

PYTHON

Coding a Space Invaders clone

Calvin Robinson will always extend the hand of friendship – unless menacing aliens are descending from the skies, in which case… blast 'em!

OUR EXPERT

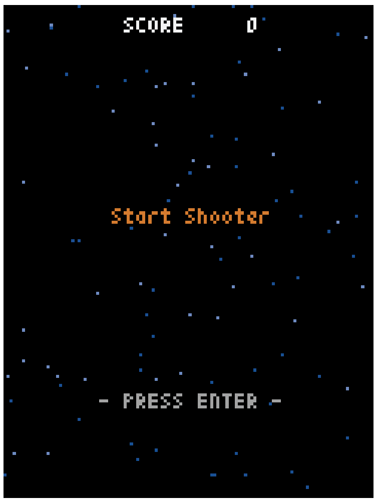
Calvin Robinson is a former assistant principal and computer science teacher and has a degree in Computer Games Design and Programming.

Space Invaders is probably the most famous space shooter of all time, released way back in 1978 for arcades. Designed by Tomohiro Nishikado for Taito in Japan, *Space Invaders* was an iconic fixed shooter and paved the way for shooting games as a viable genre. The idea being that the player controlled a space ship at the bottom of the screen and it was their job to move left and right to avoid enemies, while shooting ordnance up the screen to destroy them. Nice and simple fun, so let's have a go at creating our own take on this retro classic.

We'll be using Python to code our game, since it's one of the most versatile, accessible high-level programming languages on the market. Python is completely free and should be installed on most distros by default. However, there are multiple versions of Python, so let's make sure we're running the correct version by launching a terminal and typing `sudo apt-get install python3` if you're on a Debian based distro. For other distributions, check your software download tool.

Once installed, we can either use a text editor or the built-in Python IDLE to code. If using Python IDLE, remember to press File>New File to start an editable script document because by default Python IDLE opens up a shell window, which won't save any changes. If you'd prefer to use your favourite text editor, just be sure to save the file with the .py extension, and it can be run from terminal with the command `python3 filename.py`.

We often use *PyGame* in these tutorials because it offers a toolset that enables us to begin coding our game pretty much immediately, without messing around creating basic fundamental shapes or physics. This time, we're using a retro game engine called *pyxel*. *pyxel* is an advanced toolset, complete with image banks, tilesets, and editors for images and sound. It's fantastic for creating retro games. We're not going too deep this time around, but it's a module worth exploring further. To install *pyxel*, open a terminal and run `pip3 install pyxel`.



An intro screen, ready for the game to begin.

Let's begin by importing the random module built into Python, along with our newly installed *pyxel* toolset, and then declare and initialise all of our global variables:

```
from random import random
import pyxel

SCENE_TITLE = 0
SCENE_PLAY = 1
SCENE_GAMEOVER = 2
STAR_COUNT = 100
STAR_COLOR_HIGH = 12
STAR_COLOR_LOW = 5
PLAYER_WIDTH = 8
PLAYER_HEIGHT = 8
PLAYER_SPEED = 2
BULLET_WIDTH = 2
BULLET_HEIGHT = 8
BULLET_COLOR = 11
BULLET_SPEED = 4
ENEMY_WIDTH = 8
ENEMY_HEIGHT = 8
ENEMY_SPEED = 1.5
BLAST_START_RADIUS = 1
BLAST_END_RADIUS = 8
BLAST_COLOR_IN = 7
BLAST_COLOR_OUT = 10
enemy_list = []
bullet_list = []
blast_list = []
```

We've set up some placeholders and basic settings, most of which are self-explanatory. We have set values for our scene, points, the player, ammo and our enemies.

Sticking with best practice, we'll be using Object Oriented Programming throughout this tutorial. Not only will we be using classes from *pyxel*, but we'll be creating our own functions so that we don't need to repeat code. OOP is considered the most efficient way of programming video games.

Let's get our code sorted for drawing, updating and erasing lists, since everything in the game will be run through arrays we declared and initialised earlier:

```
def update_list(list):
  for elem in list:
    elem.update()

def draw_list(list):
  for elem in list:
    elem.draw()

def cleanup_list(list):
  i = 0
  while i < len(list):
    elem = list[i]
    if not elem.alive:
      list.pop(i)
    else:
      i += 1
```

Now we can begin setting up the game environment. Let's add a new class for the background:

```
class Background:
  def __init__(self):
    self.star_list = []
    for i in range(STAR_COUNT):
      self.star_list.append(
        (random() * pyxel.width, random() * pyxel.
height, random() * 1.5 + 1)
      )

  def update(self):
    for i, (x, y, speed) in enumerate(self.star_list):
      y += speed
      if y >= pyxel.height:
        y -= pyxel.height
      self.star_list[i] = (x, y, speed)

  def draw(self):
    for (x, y, speed) in self.star_list:
      pyxel.pset(x, y, STAR_COLOR_HIGH if speed > 1.8
else STAR_COLOR_LOW)
```

Upon initialisation, we're randomising the appearance of the stars with a for loop and the random module, appending them into our star list. We've also added functions and drawing and updating the stars in our background.

Player one has entered the game

Next, we need to introduce our player character:

```
class Player:
  def __init__(self, x, y):
    self.x = x
    self.y = y
    self.w = PLAYER_WIDTH
    self.h = PLAYER_HEIGHT
    self.alive = True

  def update(self):
    if pyxel.btn(pyxel.KEY_LEFT):
      self.x -= PLAYER_SPEED

    if pyxel.btn(pyxel.KEY_RIGHT):
      self.x += PLAYER_SPEED

    if pyxel.btn(pyxel.KEY_UP):
      self.y -= PLAYER_SPEED

    if pyxel.btn(pyxel.KEY_DOWN):
      self.y += PLAYER_SPEED
    self.x = max(self.x, 0)
    self.x = min(self.x, pyxel.width - self.w)
    self.y = max(self.y, 0)
    self.y = min(self.y, pyxel.height - self.h)

    if pyxel.btnp(pyxel.KEY_SPACE):
      Bullet(
        self.x + (PLAYER_WIDTH - BULLET_WIDTH) /
2, self.y - BULLET_HEIGHT / 2
      )

      pyxel.play(0, 0)

  def draw(self):
    pyxel.blt(self.x, self.y, 0, 0, 0, self.w, self.h, 0)
```

Upon initialisation we're setting the x and y-coordinates. We're pulling the width and height in from the global variables we set earlier, and most importantly, we're declaring our player character to be alive. Like in the other class, we're setting a function to draw and then update our player character. The update function will move our player character when the appropriate key is pressed, at a speed based on the speed variable we declared at the beginning.

That's our player sorted, let's get our bullet ammo set up, much in the same way as our player class:

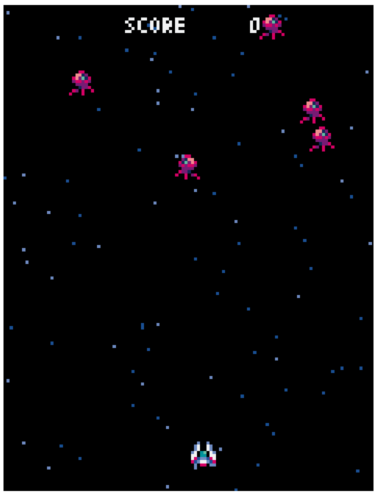
```
class Bullet:
  def __init__(self, x, y):
    self.x = x
    self.y = y
    self.w = BULLET_WIDTH
    self.h = BULLET_HEIGHT
    self.alive = True

    bullet_list.append(self)

  def update(self):
    self.y -= BULLET_SPEED

    if self.y + self.h - 1 < 0:
      self.alive = False

  def draw(self):
    pyxel.rect(self.x, self.y, self.w, self.h, BULLET_
COLOR)
```

We'll need a class for our enemies, too. The beauty of using classes, in an Object Oriented Programming methodology is that we can spawn multiple instances (objects) of the same thing, without duplicating any code.

```
class Enemy:
  def __init__(self, x, y):
    self.x = x
    self.y = y
    self.w = ENEMY_WIDTH
    self.h = ENEMY_HEIGHT
    self.dir = 1
    self.alive = True
    self.offset = int(random() * 60)

    enemy_list.append(self)

  def update(self):
```

QUICK TIP

You can create a standalone executable version of your space shooter. Run pip3 install pyinstaller to get the dependencies sorted, then pyxelpackager shooter.py to turn shooter.py into an executable.



Our player controls the rocket ship at the bottom of the screen, shooting the aliens.

»

QUICK TIP

Create your own image sprites in the pyxel engine editor by running pyxeleditor. It's an intuitive pixel image editor, useful not only for this project, but any retro creation you might be working on.

```
        if (pyxel.frame_count + self.offset) % 60 < 30:
            self.x += ENEMY_SPEED
            self.dir = 1
        else:
            self.x -= ENEMY_SPEED
            self.dir = -1

        self.y += ENEMY_SPEED

        if self.y > pyxel.height - 1:
            self.alive = False

    def draw(self):
        pyxel.blt(self.x, self.y, 0, 8, 0, self.w * self.dir, self.h, 0)
```

The last of these series of classes is the destructive blast itself:

```
class Blast:
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.radius = BLAST_START_RADIUS
        self.alive = True

        blast_list.append(self)

    def update(self):
        self.radius += 1

        if self.radius > BLAST_END_RADIUS:
            self.alive = False

    def draw(self):
        pyxel.circ(self.x, self.y, self.radius, BLAST_COLOR_
IN)
        pyxel.circb(self.x, self.y, self.radius, BLAST_COLOR_
OUT)
```

That's all of our game elements set up, we can now begin coding the application window itself, by tapping into the *pyxel* module. The rest of the code features in this tutorial belongs in the App class, so let's create it:

```
class App:
    def __init__(self):
        pyxel.init(120, 160, caption="Retro Shooter Game")
```

Of course, we could change the caption to anything we like, and that would appear in the titlebar of our game window.

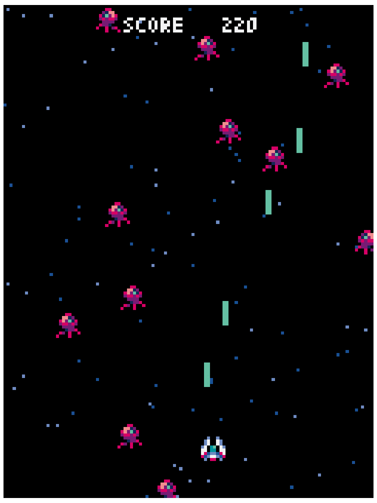
We're now going to use the *pyxel* image sets to create our sprites:

```
        pyxel.image(0).set(
            0,
            0,
            [
                “00c00c00”,
                “0c7007c0”,
                “0c7007c0”,
                “c703b07c”,
                “77033077”,
                “785cc587”,
                “85c77c58”,
                “0c0880c0”,
            ],
        )

        pyxel.image(0).set(
            8,
            0,
            [
                “00088000”,
                “00ee1200”,
                “08e2b180”,
                “02882820”,
                “00222200”,
                “00012280”,
                “08208008”,
                “80008000”,
            ],
        )
```

Likewise for the game sounds:

```
        pyxel.sound(0).set("a3a2c1a1”, “p”, “7”, “s”, 5)
        pyxel.sound(1).set("a3a2c2c2”, “n”, “7742”, “s”, 10)
```

The image sets and sound effects are completely customisable, but it does take further experimentation with the *pyxel* game engine.



Killing unfriendly aliens results in explosions.

Background information

Set the scene and draw a background and our player:

```
        self.scene = SCENE_TITLE
        self.score = 0
        self.background = Background()
        self.player = Player(pyxel.width / 2, pyxel.height -
20)
        pyxel.run(self.update, self.draw)
```

This will need updating – as with everything we draw onto the screen – in order to ensure the player always sees the latest version when something changes:

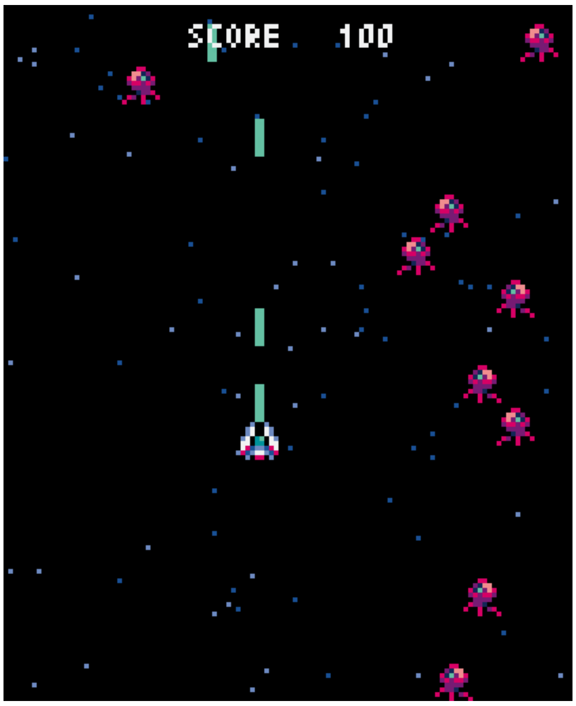
```
    def update(self):
        if pyxel.btnp(pyxel.KEY_Q):
            pyxel.quit()
        self.background.update()
        if self.scene == SCENE_TITLE:
            self.update_title_scene()
        elif self.scene == SCENE_PLAY:
            self.update_play_scene()
        elif self.scene == SCENE_GAMEOVER:
            self.update_gameover_scene()
```

Updating our title scene is simple:

```
    def update_title_scene(self):
        if pyxel.btnp(pyxel.KEY_ENTER):
            self.scene = SCENE_PLAY
```

Updating our main game content, on the other hand, is a little more complicated:

```
    def update_play_scene(self):
        if pyxel.frame_count % 6 == 0:
            Enemy(random() * (pyxel.width - PLAYER_
WIDTH), 0)
        for a in enemy_list:
            for b in bullet_list:
                if (a.x + a.w > b.x
                    and b.x + b.w > a.x
                    and a.y + a.h > b.y
                    and b.y + b.h > a.y):
                    a.alive = False
                    b.alive = False
                    blast_list.append(Blast(a.x + ENEMY_WIDTH
/ 2, a.y + ENEMY_HEIGHT / 2))
                    pyxel.play(1, 1)
                    self.score += 10
        for enemy in enemy_list:
            if (self.player.x + self.player.w > enemy.x
                and enemy.x + enemy.w > self.player.x
```



Roam the screen with the arrow keys to shoot those pesky aliens.

```
            and self.player.y + self.player.h > enemy.y
            and enemy.y + enemy.h > self.player.y):
            enemy.alive = False
            blast_list.append(
              Blast(
                self.player.x + PLAYER_WIDTH / 2,
                self.player.y + PLAYER_HEIGHT / 2,))
            pyxel.play(1, 1)
            self.scene = SCENE_GAMEOVER
        self.player.update()
        update_list(bullet_list)
        update_list(enemy_list)
        update_list(blast_list)
        cleanup_list(enemy_list)
        cleanup_list(bullet_list)
        cleanup_list(blast_list)
```

A lot is going on here. We're spawning enemies at random locations, creating blasts when they're destroyed and cleaning up the sprites (remove the resources) when they're no longer visible.

We'll need to update our game over scene, too:

```
    def update_gameover_scene(self):
        update_list(bullet_list)
        update_list(enemy_list)
        update_list(blast_list)
        cleanup_list(enemy_list)
        cleanup_list(bullet_list)
        cleanup_list(blast_list)
        if pyxel.btnp(pyxel.KEY_ENTER):
          self.scene = SCENE_PLAY
          self.player.x = pyxel.width / 2
          self.player.y = pyxel.height - 20
          self.score = 0
          enemy_list.clear()
          bullet_list.clear()
          blast_list.clear()
```

Here we're just doing some housekeeping, restoring the lists, deleting unused sprites, and waiting for the enter key to be pressed so we can restart the game.

Finally, we need to code the draw functions. It's usually easier to draw something than to keep it updated, so these are the program's shortest functions:

```
    def draw(self):
        pyxel.cls(0)

        self.background.draw()

        if self.scene == SCENE_TITLE:
          self.draw_title_scene()
        elif self.scene == SCENE_PLAY:
          self.draw_play_scene()
        elif self.scene == SCENE_GAMEOVER:
          self.draw_gameover_scene()

        pyxel.text(39, 4, "SCORE {:5}".format(self.score), 7)
```

We'll also need to draw the title scene, play scene and 'game over' scene.

```
    def draw_title_scene(self):
        pyxel.text(35, 66, "Start Shooter", pyxel.frame_
count % 16)
        pyxel.text(31, 126, "- PRESS ENTER -", 13)
```

That's the text to display on the start screen, but we can change the text to something more suitable: "Shooter Game – Press Enter to begin", for example. The final two functions will prepare our bullets, enemies and blast effects for the game and game over scenes:

```
    def draw_play_scene(self):
        self.player.draw()
        draw_list(bullet_list)
        draw_list(enemy_list)
        draw_list(blast_list)
    def draw_gameover_scene(self):
        draw_list(bullet_list)
        draw_list(enemy_list)
        draw_list(blast_list)
```

And finally, the text for our game over screen, again, entirely customisable:

```
        pyxel.text(43, 66, "GAME OVER", 8)
        pyxel.text(31, 126, "- PRESS ENTER -", 13)
```

That's it. One more line left, make sure this one isn't indented. `App()` will call the App class we've just finished, and kick everything off. Now press F5 to save and run our program, and if all went to plan we should have a functioning graphical retro shooter game. Good luck, have lots of fun! LXF

They think it's all over... it is now!

QUICK TIP

Pyxel github page features a comprehensive wiki guide on how to use the game engine. Try playing around with different image sets and sound effects to alter the game experience: https://github.com/kitao/pyxel

DAST

Build a dynamic app security pipeline

Dynamic Analysis Security Testing takes centre stage in this coding tutorial with **Tim Armstrong**.

OUR EXPERT

Tim Armstrong is a former Lead Engineer turned Developer Advocate specialising in networking, software development, and security. You can find him on Twitter as **@omatachyru** or via his website at **www.plaintextnerds.com**.

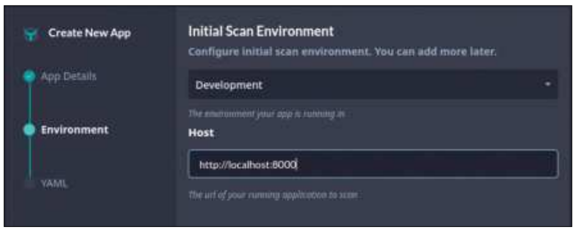
The battle between developers and malicious hackers is one that developers have been losing. A lot of the time, it comes down to mentality and company priorities. Hackers, like burglars, only need to find a single open window or unlocked door to get in. You wouldn't check that you've locked your door only once every few months, yet this is the exact approach many companies take to security.

Dynamic Analysis Security Testing (DAST) is perhaps the most overlooked stage of any security pipeline, frequently relegated to a check-up every six months by an outside consultancy that does an automated scan with *Burp Suite* or *Zed Attack Proxy* (*ZAP*) and provides you with a (hopefully short) report and an invoice in the range of £3,000-30,000, mostly depending on the scope. In most cases, the consultants don't go further than the automated scan because at that point they already have enough to write a multi-page report.

But here's the thing: when malicious actors (aka hackers) attack your web app, site or API, they aren't checking if your code is neatly formatted, they're essentially doing dynamic analysis. They're looking for a place where you've not validated the input, an endpoint that you've forgotten to protect, cookie slack, a vulnerable login system, leaked credentials and hundreds of other things that are very difficult to detect statically. If you're relying on a spot test every six months then odds are you've got security holes that you're not aware of.

Building DAST into your CI/CD only takes a few minutes and gives you effectively that same information that you'd get from a pen-test where all they did was run an automated scanner. The main difference is that instead of it only occurring every six months, the scan happens every time someone merges a PR to the main branch – meaning you find out about the vulnerability when it gets merged. Ultimately this means that when you do bring in the external consultants for the six-month check-up, you actually get your money's worth!

In this tutorial, you'll be adding DAST to a GitLab CI/CD pipeline. We have covered this before in *Linux Format* magazine (**LXF279 & 280**, if you have them to hand), and it's a good idea to check those out first, but if you just want to dive in at this point, then you can pick up a copy of the progress so far at **https://gitlab.com/plaintextnerds/web-app-security-tutorial2-lxf280**.



Initial Scan Environment

Configure initial scan environment. You can add more later.

Development

The environment your app is running in

Host

http://localhost:8000

The url of your running application to scan

Adding an app in StackHawk is pretty straightforward, with a clean workflow and UI.

Too many acronyms

When it comes to DAST there are a growing number of solutions in the market. Most of them use hosted scanners that run on a periodic schedule, so either you need to expose a test environment to the internet and have the app scan that, or you need to set it up to scan production – which is a bit late (and could lead to instability of the production environment).

The goal here is, as it was when you added static analysis and software composition analysis to the pipeline, that you know about any vulnerabilities before the code goes live so that you can fix them before it gets deployed.

What you need, then, is something that can run inside the pipeline to test the service on commit. So what are your options? Well, you could build your own solution around an open-source tool like *ZAP*, or you could pick up an off-the-shelf solution. Unfortunately, there aren't many DAST solutions that you can build directly into your CI pipeline, with the leaders in this space being StackHawk and GitLab. Both based their scanners on *ZAP*, meaning that they can run in a Docker container in your pipeline (or even locally).

StackHawk's scanner, *HawkScan*, is a little more advanced than GitLab's version and has support for multiple authentication methods and makes it easy to customise the scanner. While StackHawk holds the lead in scanning capabilities, GitLab is ahead in pricing (for proprietary/closed-source software) as it's included in GitLab Ultimate, and obviously, it's also ahead in its integration with the rest of the GitLab platform.

What really separates these two solutions from others in this space, however, is their dedication to open-source, as both companies have decided to make their solutions free to open-source projects. Which, considering that they are both built on *ZAP* (which itself is an open-source solution), should perhaps not be a surprise. Unfortunately, however, it is not that common, as even though a number of their competitors are also likely based on *ZAP*, the vast majority of them do not extend the same offer.

Street – er, StackHawk

Kicking things off with StackHawk, the first thing you'll want to do is register an account over at **http://stackhawk.com** so that you can set up your app and get an API key-pair for your *HawkScan* instance with which to push the scan results.

When you open a Developer account, you get to use it for free for one app so you can follow along even if you're not working on an open-source project. If you are working on an open-source project, make sure you contact the StackHawk team to unlock that free upgrade for your own team (they also help out start-ups with special deals).

With an account set up, make sure you're on the Applications dashboard and hit the 'Add an App' button. This will open up a modal dialogue box where you can give it a name, configure the environment type, set the hostname, and generate the Application ID and **stackhawk.yml** config file.

Next, you need to generate your API key. Do this by clicking your profile picture at the bottom-left and then going to Settings. From here go to API Keys and create a new key and copy it into your clipboard. Then, head over to the Settings page for the CI/CD in your GitLab project and add it as a new variable with the key **HAWK_API_SECRET** and with both the 'Protect variable' and 'Mask variable' boxes ticked. While you're on this page you'll also want to add the **app_id** as **HAWK_APP_ID**; however, you don't need to tick the boxes for this one.

Next, you want to edit the **.gitlab-ci.yml** to add a new stage and of course, the new job. To do this add the line

```
- dynamic-analysis
```

to the `stages` section, then add the job as follows:

```
hawkscan:
  stage: dynamic-analysis
  image: docker:20
  services:
    - docker:20-dind
  before_script:
    - docker pull stackhawk/hawkscan
  script:
    - |
      docker run -v $(pwd):/hawk:rw -t \
        -e API_KEY="hawk.${HAWK_API_ID}.${HAWK_
API_SECRET}" \
        -e NO_COLOR=true \
        stackhawk/hawkscan
```

Something you might notice is that this job is quite different from those defined in previous instalments of this series; this is because *HawkScan* needs to run in a DinD (Docker in Docker) environment. But how does it know how to run your app so that it can test it? The answer to that question of course is that it doesn't, so that's what you'll need to define next.

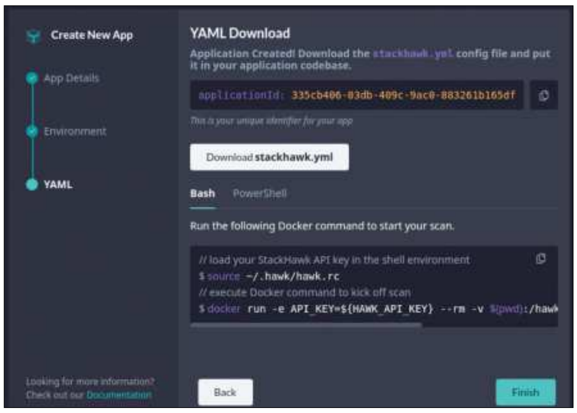
To run the app in the CI/CD pipeline, you'll need it running in a Docker container. This means you need to define a build stage that makes a Docker image from the source code of the merge request. To do this, you'll need a Dockerfile to build and a stage in the pipeline that will build it.

When it comes to building a Dockerfile for a Django project you only really need it to be a handful of lines long, like this:

```
FROM python:3.9
WORKDIR /usr/src/app
COPY requirements.txt ./
RUN pip install -r requirements.txt
COPY ./i_am_vulnerable/. .

EXPOSE 8000
CMD ["python", "manage.py", "runserver",
"0.0.0.0:8000"]
```

Place this **Dockerfile** in the project's **src** directory next to the **requirements.txt**.

To build this Docker image in the CI/CD pipeline you'll need to add a new stage called **build** to the list of stages, placing it directly before the **dynamic-analysis** stage. Then you'll need to add a build job, that will look something like this (code on the next page):



Application Created! Download the stackhawk.yml config file and put it in your application codebase.

```
applicationId: 335cb406-03db-409c-9ac0-883261b165df
```

This is your unique identifier for your app

Run the following Docker command to start your scan.

```
// load your StackHawk API key in the shell environment
$ source ~/.hawk/hawk.rc
// execute Docker command to kick off scan
$ docker run -e API_KEY=${HAWK_API_KEY} --rm -v $(pwd):/hawk
```

Looking for more information? Check out our Documentation

Make sure you download the stackhawk.yml file and make a note of the App ID as you'll need it later.

QUICK TIP

CI/CD pipelines can be optimised by adding rules that dictate when a stage or job is to be run, which can reduce costs and improve the developer experience.

» DEFENCE IN DEPTH

Modern security practice involves accepting that, at some point, you will be breached. That doesn't mean that you give up; it means that you plan for the eventuality. Once you have been breached you need to detect and prevent lateral movement through the system so that you know what has been leaked.

If you've been hit by ransomware, you'd better hope that you have off-site backups. If your administrator credentials for your hosting provider get stolen, you'd probably wish you had your infrastructure defined as code so that it can be rebuilt quickly. If your database gets leaked, then you'd want to be certain that you've encrypted sensitive data so that your customers don't get exposed.

This kind of planning for disaster is essential in today's world, and it's not going to change any time soon. Laws won't stop criminals and to put salt in the wound, at the moment governments around the world seem to be single-minded in making it easier for them (with proposals like backdoored encryption, and online 'real identity' verification being promoted by politicians).

»

```
build-docker:
 stage: build
 image: docker:20
 services:
  - docker:20-dind
 script:
  - cd src
  - docker login --username $CI_REGISTRY_USER
--password $CI_REGISTRY_PASSWORD $CI_
REGISTRY
  - >
   docker build
   --tag $CI_REGISTRY_IMAGE:$CI_COMMIT_
SHORT_SHA
   .
  - docker push $CI_REGISTRY_IMAGE:$CI_
COMMIT_SHORT_SHA
```

So what does this do? Looking through the script section of this job, you can see that it will login to the GitLab Docker repository for the project, then build the **Dockerfile** found in the root of the project directory tagging it with the current commit tag, after which it pushes the container up to the GitLab Docker repository for the project.

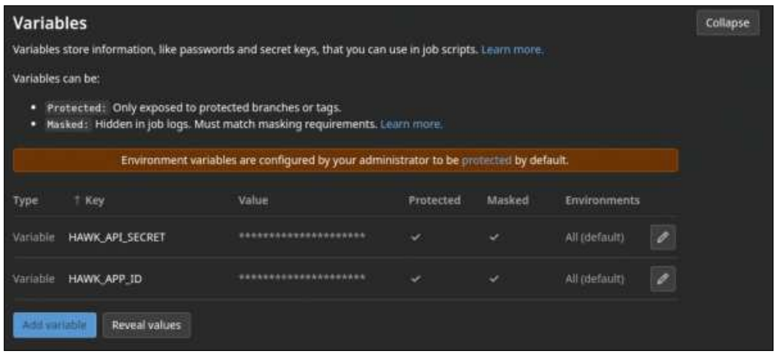
This is good to go, so you'll need to make it available to HawkScan. To do so, you'll want to replace the `script` section of the HawkScan job with the following code:

```
- docker login --username $CI_REGISTRY_USER
--password $CI_REGISTRY_PASSWORD $CI_
REGISTRY
- docker run --name djangoapp -d $CI_REGISTRY_
IMAGE:$CI_COMMIT_SHORT_SHA
- |
 docker run --link djangoapp -v $(pwd):/hawk:rw -t \
  -e API_KEY="hawk.${HAWK_API_ID}.${HAWK_API_
SECRET}" \
  -e NO_COLOR=true \
  stackhawk/hawkscan
```

When you've finished, the first few lines of your **.gitlab-ci.yml** should look something like this:

```
stages:
- static-analysis
- composition-analysis
- build
- dynamic-analysis
hawkscan:
 stage: dynamic-analysis
 image: docker:20
 services:
  - docker:20-dind
 before_script:
```

Once you've generated your StackHawk API secret and APP ID you need to put them where HawkScan can find them.


Variables

Collapse

Variables store information, like passwords and secret keys, that you can use in job scripts. Learn more.

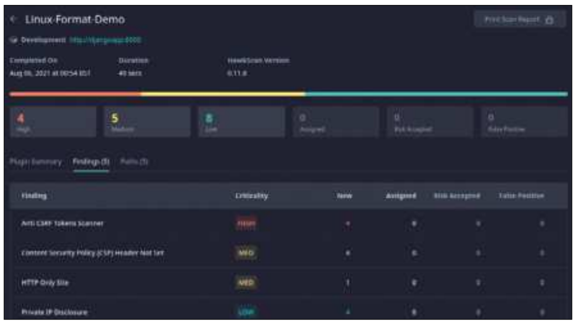
Variables can be:

- `Protected:` Only exposed to protected branches or tags.
- `Masked:` Hidden in job logs. Must match masking requirements. Learn more.

Environment variables are configured by your administrator to be protected by default.

Type	↑ Key	Value	Protected	Masked	Environments
Variable	HAWK_API_SECRET	********************	✓	✓	All (default)
Variable	HAWK_APP_ID	********************	✓	✓	All (default)

Add variable · Reveal values


Linux-Format-Demo

Print Scan Report

Development [illegible]

Completed On	Duration	HawkScan Version
Aug 06, 2021 at 00:54 BST	40 secs	0.11.8

4 High · 5 Medium · 8 Low · 0 Assigned · 0 Risk Accepted · 0 False Positive

Plugin Summary · Findings (5) · Paths (5)

Finding	Criticality	New	Assigned	Risk Accepted	False Positive
Anti CSRF Tokens Scanner	HIGH	4	0	0	0
Content Security Policy (CSP) Header Not Set	MED	4	0	0	0
HTTP Only Site	MED	1	0	0	0
Private IP Disclosure	LOW	4	0	0	0

From the scan results view, you can see how many of the issues are new and how many still remain since you last triaged the results.

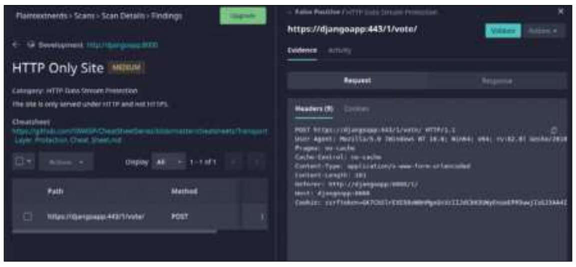
```
  - docker pull stackhawk/hawkscan
 variables:
  APP_HOSTNAME: $CI_REGISTRY_IMAGE
 script:
  - docker login --username $CI_REGISTRY_USER
--password $CI_REGISTRY_PASSWORD $CI_
REGISTRY
  - docker run --name djangoapp -d $CI_REGISTRY_
IMAGE:$CI_COMMIT_SHORT_SHA
  - |
   docker run --link djangoapp -v $(pwd):/hawk:rw -t \
    -e API_KEY="hawk.${HAWK_API_ID}.${HAWK_
API_SECRET}" \
    -e NO_COLOR=true \
    stackhawk/hawkscan
build-docker:
 stage: build
 image: docker:20
 services:
  - docker:20-dind
 script:
  - cd src
  - docker login --username $CI_REGISTRY_USER
--password $CI_REGISTRY_PASSWORD $CI_
REGISTRY
  - >
   docker build
   --tag $CI_REGISTRY_IMAGE:$CI_COMMIT_
SHORT_SHA
   .
  - docker push $CI_REGISTRY_IMAGE:$CI_
COMMIT_SHORT_SHA
```

Then you'll need to add the **stackhawk.yml** file that was generated at the start to the root of the project. Finally, you'll need to edit a couple of lines in that file, so change the host field of `http://djangoapp:8000` to `http://djangoapp:8000/bad_sql` and uncomment the `antiCsrfParam` field and set it to be `csrfmiddlewaretoken`. Then `git add`, `git commit` and `git push` those changes up to the GitLab project.

This will trigger the pipeline which, in addition to running the SAST and SCA stages defined in previous instalments, will now build a Docker image, push that to the GitLab project's Docker Registry and then run *HawkScan* against the image, posting the result up to your StackHawk account when it's complete.

Scan results

Results are broken down into three main categories. High category should be fixed immediately as they pose



The finding details view provides a very useful overview of the issue, the evidence, and the paths where it was detected so that you can tackle it efficiently.

an immediate danger to business continuity; Medium are generally issues with known exploit paths, but might not be a direct risk to business. Low category tend to be informational leaks (such as server versions) that could make an attacker's job easier.

Clicking one of the findings brings up a summary of what the specific vulnerability is, including some notes on how a malicious hacker might abuse it.

Next to the findings is a complete list of the paths scanned, so you know if *HawkScan* was able to find and scan a particular path in your application. This can be really helpful for ensuring that you have full coverage of the application. Web crawlers are rarely perfect, and the one utilised by *HawkScan* is no different; however, if the application you're scanning has a GraphQL or OpenAPI/Swagger schema (or even a SOAP descriptor) then *HawkScan* won't need to use a crawler and should hit 100 per cent of the paths every time.

It should be noted that just because a vulnerability isn't detected doesn't mean that it isn't there. At the time of writing, the SQL Injection vulnerability in the *bad_sql_practices* Django app that is used as the base example for this example was not detected by *HawkScan* (or any of its competitors).

Triage and false positives

Once we know of a vulnerability we need to triage it. The first step to a good triage process is to discard known false positives. When testing developer environments, like the one configured in this tutorial, it's common to exclude things like TLS/SSL certificates. It's no surprise then that *HawkScan* detected it as an 'HTTP Only site'.

Because of this, the StackHawk UI wishes to inform you of the dangers of HTTP Only websites. However, as this is a development-grade deployment this is not actually a concern. To mark this finding as a false positive, open up the finding, and at the top-right of the page you'll see a button labelled Validate and a drop-down called Actions. From the drop-down you can select 'False Positive' and provide a description as to why it should be ignored.

Once you've filtered out known false positives, the next step of triage is to ensure that you have tickets in your project management for all of the remaining risks. Any high-risk vulnerabilities should be expected to break sprint and receive immediate attention, as failure to do so would mean knowingly leaving the door wide open to attackers.

Medium-risk vulnerabilities are commonly scheduled into the next sprint. This isn't ideal as, while they aren't normally as big of a problem as the high-risk ones, when coupled with other vulnerabilities they can be just as bad as a high-risk vulnerability. However, if you attempt to treat everything as a sprint-breaking priority then management might start to ignore you as if you were crying wolf – ultimately leading to the opposite of the desired outcome.

Making a plan

If you've followed this series thus far, you now have a pipeline containing SAST, SCA and DAST. Once all three stages do not find any vulnerabilities, you should be in good shape moving forward.

However, it's important to remember that this just means there are no vulnerabilities found – not that they aren't there.

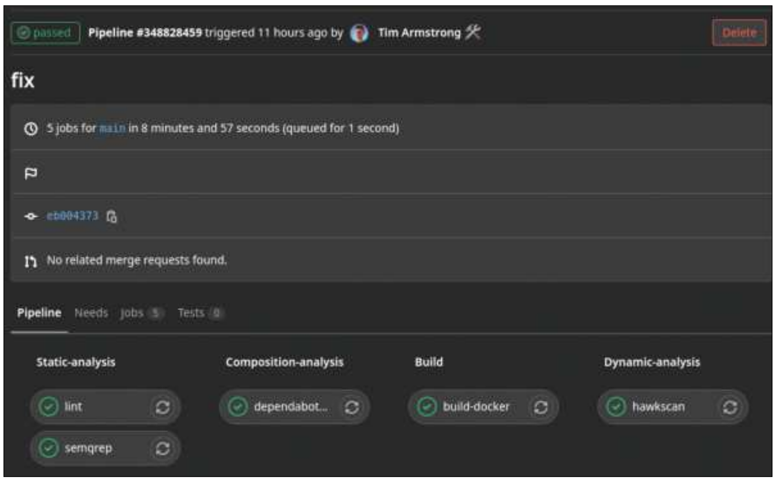
This is why it's important to take time to make a security breach response plan and to build your defence in depth. When developing a complex stage to a CI/CD pipeline, it can be a good idea to comment out the existing stages – otherwise, you can find yourself spending a lot of time waiting. LXF

» MAKING A POINT EFFECTIVELY

If you're struggling to advocate for fixing a particular vulnerability, one of the best tools is learning how to demonstrate the risk. If you can show the vulnerability in action and the damage that can be done, you will find the conversation shifts very quickly to how quickly you can fix it. If it doesn't, you might need to speak to someone higher up in the organisation. If even that doesn't help, then the organisation as a whole has a serious culture problem and will likely have a big breach in the future.

Demonstrating vulnerabilities is a far more effective way to get time to fix vulnerabilities than just discussing them because it makes things far more tangible for someone who isn't as capable or knowledgeable as you are.

This is one of the main reasons that hacker conferences like Black Hat and DEF CON are important, as they disseminate both defence and attack strategies, giving a more rounded exposure to the field. After all, without knowing how to exploit something, understanding the potential risk and work out how to protect it is very difficult, and expressing the risk to others effectively is even harder.


passed Pipeline #348828459 triggered 11 hours ago by Tim Armstrong

fix

5 jobs for main in 8 minutes and 57 seconds (queued for 1 second)

eb004373

No related merge requests found.

A completed pipeline can take up to 15 minutes to run (or longer in more advanced projects), so combining stages can sometimes be necessary.

TURING MACHINES

Programming a Turing Machine

It was the computer that started it all, albeit in theory. **Mike Bedford** shows you how to program a Turing Machine and put it through its paces.

OUR EXPERT

Mike Bedford is fascinated by how simple computer architectures – such as a Turning Machine – can do anything the most powerful supercomputers can do.

QUICK TIP

Alan Turning might not have built a physical realisation of his theoretical computer, but Mike Davey has. See www.aturingmachine.com. It works on a real tape, erasing old symbols and writing new ones as it goes. Behind the scenes it's controlled by a Parallax Propeller microcontroller.

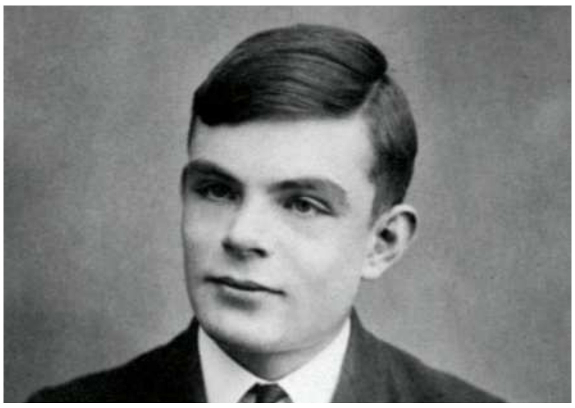
Alan Turing is probably best known for his pioneering work on code breaking at the Government Code and Cypher School at Bletchley Park. In playing a key role in developing the electro-mechanical Bombe device that was used to crack the Enigma cipher, Turing had a major impact on shortening World War II by an estimated two years and saving as many as 14 million lives.

Despite having been dubbed the "Father of Modern Computing", however, his contributions to general-purpose computing are less well-appreciated. And here it's interesting to note that his design for the ACE computer, a cut-down version of which was eventually built by the National Physics Laboratory in 1950, predated the Manchester Baby, the world's first stored program computer, by three years. Arguably, though, his biggest contribution to computing was his vision for a machine that was never actually built, and would have been totally impractical had it ever become a physical reality. This was the so-called Turning Machine and here we look at this model of computing and see how to program it using a couple of simulators.

Turning Machines

Alan Turing's concept of the computing architecture that now carries his name dates back to 1936, which was almost a decade before anyone figured out how to build a universal computing device. But since this was a so-called thought experiment, a theoretical concept that he used to develop his theory of computability, it didn't matter that it only ever existed on paper. What makes it so interesting to today's programmers, though, is how simple it is compared to modern day computers, despite being capable of computing any function that the PC on your desk could compute.

If you've ever delved into historic computers you'll no doubt have encountered some extremely minimalistic instruction sets. The Manchester Baby, for example, despite being a universal computing device, had just seven instructions, compared to many hundreds or thousands in today's processors. Turning Machines are even simpler, not in having even fewer instructions, but in using an architecture that doesn't really have



Alan Turing's famous Turing Machine was never intended as a physical computer, although today's simulations have brought it to life.

(Credit: www.turingarchive.org/viewer/?id=521&title=4)

instructions in the normal meaning of the word. So, possibly for the first time ever, you'd be writing programs that aren't sequences of instructions. Instead, a Turning Machine is a special type of Finite State Machine, as we're about to see.

A Turning Machine (TM) operates on an infinite tape, divided into squares that we can think of as memory locations, each of which can contain a symbol from the machine's alphabet of symbols. We can think of this as the equivalent of the memory in a PC, because the TM can read and write symbols on the tape, albeit only at the position occupied by the TM's read/write head.

At any time, the TM is in one of a finite number of states. These can be as numbered, although it makes more sense to give them names. Although TM's don't execute instructions, they do operate sequentially. Each operation involves reading the symbol at the current position of the read/write head on the tape and then, depending on the combination of that symbol and the TM's current state, it writes a specified symbol to the tape (which can be the same as the symbol it had read), it adopts a new state (which can be the same as the current state), and it then moves its read/write head one position left or right along the tape.

So, a particular TM is defined by its alphabet of symbols, its number of states, and a transition table that defines the action in each step depending on the tape symbol and the state. With this information defining the TM, it can then operate on an initial sequence of symbols on the tape –which is its input data, and the start position on the tape. So, following that brief introduction, we can try out a TM in our first simulator.

The Tursi simulator

To start, download the Java code for first TM simulator, which goes by the name of *Tursi*, at **https://github.com/schaetzc/tursi**. Although a TM can be designed to perform any function that can be expressed logically, even the simplest examples can take a huge amount of time to execute, with the machine appearing to move almost endlessly back and forth along the tape. So to limit the number of steps required to execute our first example, we'll carry out an arithmetic addition but, to keep things as simple as possible, we're going to use unary instead of binary arithmetic.

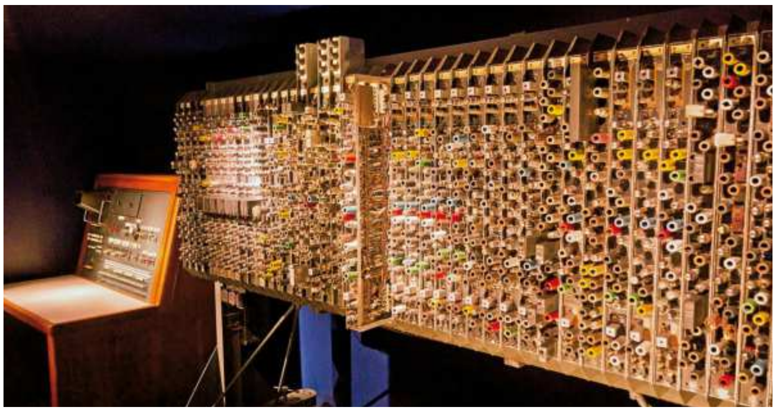
You might not have come across the unary numbering system before so here's a quick introduction. Otherwise known as base 1, it follows the same rules we've learned for interpreting decimal (base 10), binary (base 2) or hexadecimal (base 16) numbers. So, starting from the right-most digit and moving left we have positions for one to powers of zero, one, two, three, etc. Uniquely among number systems, these all equate to the same number, specifically one, which means that the equivalent in unary of a decimal number is that number of ones. So, for example, decimal 3 equates to unary 111, and decimal 6 equates to 111111.

Although *Tursi's* interface allows the operation of a TM to be controlled, including writing the initial pattern to the tape and specifying the initial position, the remainder of the definition is done externally. So, here's the definition for unary addition, which you should enter using a text editor:

```
#! start 0
#! end 4
#! fill *
0 0 0 R 1
1 0 1 R 2
1 1 1 R 1
2 0 * L 3
2 1 1 R 2
3 0 0 L 3
3 1 0 L 4
```

The first two lines define the start state and any states that cause the TM to halt, and the third line specifies a symbol to be written to all positions on the tape as an initialisation. The remaining lines, of which there should be one for each combination of the state and read tape symbol (but only those that are expected to be encountered), dictate the symbol that should be written to the tape, the direction to move along the tape (L or R), and the new state. So, for example, the line `2 0 * L 3` means that whenever the TM is in state 2 and reads a 0 from the tape, it should write an asterisk to the tape, move one space to the left, and enter state 3.

Now start up *Turi* and load your unary addition program, and you'll then see the transition table shown as the various rules at the left of the screen. At the top you'll see that every position on the tape contains an asterisk,and a couple of lines below, the Current State is shown as 0, both of which we defined. However, we


The TM predated Turing's ACE computer – initially built at the National Physics Laboratory in 1950 as the Pilot ACE – by 14 years. Photo by Antoine Taveneaux.

can't do any arithmetic on an infinite string of asterisks so ensure that the Head Position is 0, type `01110111110` in the Tape box, and press the Enter key.

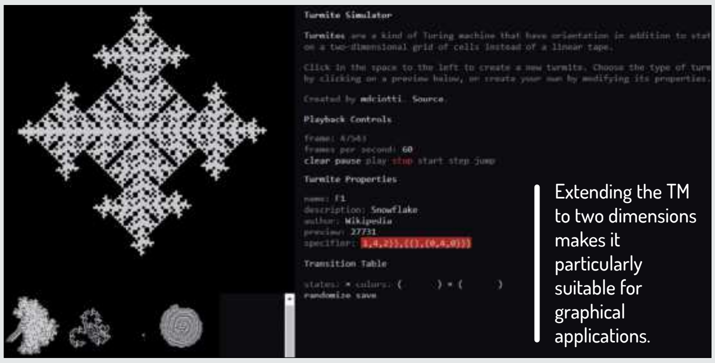
You'll notice that your data – which represents the unary numbers for 3 and 5, each delimited by zeros – appears on the tape, starting at position 0, with the head on the left-hand zero. Now try executing the TM one step at a time using the "find and execute next rule" button which is the third button above Current State, and is the one that looks like a footstep with a forward

» 2D TURING MACHINES

Although 2D Turing Machines are no more powerful than the original ones, they're an interesting diversion. That's because the grid of cells enable graphical patterns to be calculated and displayed, especially if colours are written to those cells instead of letters or numbers.

Most of what we've learned about ordinary TMs applies to 2D TMs, the only difference being that, instead of moving only left or right, the read/write head is able to move left, right, up or down. In some implementations the movement is defined absolutely, so we can think of it as N, S, E and W, while in others it's relative to the current direction of the head so the angle of a turn is defined as 0, 90, 180 or 270 degrees, after which the head moves forward. If you're looking for simulations of 2D TMs they're often called Turmites and, while there are plenty online, there don't seem to be too many native Linux applications so that could be an opportunity for you.

To get a feel for Turmites, go to **https://mdciotti.github.io/turmites** and try out the Turmite defined by `{{{1,8,1},{1,2,0}},{{1,4,1},{1,4,2}},{{},{0,4,0}}}`. This is intended as a curiosity as opposed to for serious coding, and it seems to be used mostly to try out random definitions to appreciate the range of behaviours that can result from just repeating a simple set of rules.


```
Turmite Simulator

Turmites are a kind of Turing machine that have orientation in addition to stat
on a two-dimensional grid of cells instead of a linear tape.

Click in the space to the left to create a new turmite. Choose the type of turm
by clicking on a preview below, or create your own by modifying its properties.

Created by mdciotti. Source.

Playback Controls

frame: 4/543
frames per second: 60
clear pause play stop start step jump

Turmite Properties

name: F1
description: Snowflake
author: Wikipedia
preview: 27731
specifier: 1,4,2}},{{},{0,4,0}}}

Transition Table

states: * colors: (       ) * (       )
randomize save
```

Extending the TM to two dimensions makes it particularly suitable for graphical applications.

QUICK TIP

"How fast is a Turing Machine?" is surely a daft question since it's a theoretical computer. For a mechanical implementation, though, Mike Davey (see page 92) came up with some estimates about his creation – from 3.1 operations per second to 6.2 seconds per operation.

pointing arrow. As you execute steps, or enable automatic execution – that's the green triangle button – you'll see the steps listed as they're executed. For each step, you're shown the initial state, the symbol read from the tape, the symbol written to the tape, the direction moved and the new state. Hopefully you'll find that it halts on step 12, having changed the zero between the two numbers to a one, the final zero to an asterisk, and the final one to a zero. The result is, therefore, `0111111110`, which is the unary representation for eight, again delimited by zeros.

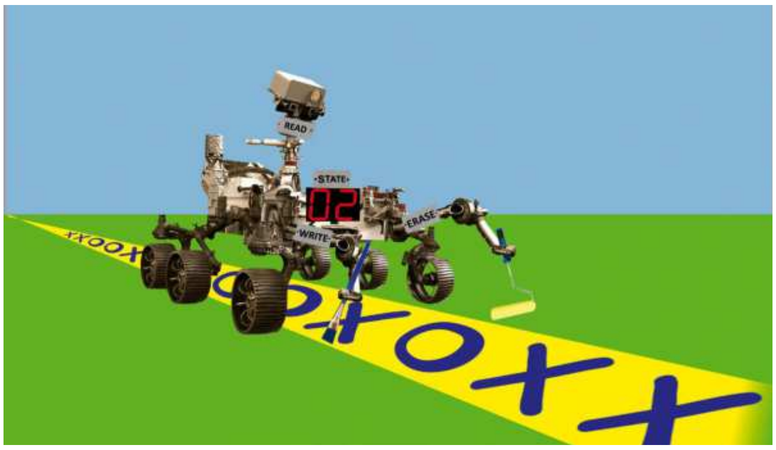
Vizualisation machine

Our second TM simulator is an online utility that you can find at **https://turingmachine.io** and which calls itself *Turing Machine Visualization*. This simulator has the benefit of a built-in editing facility so you don't have to use a separate text editor, it shows the TM as a state diagram, and it has lots of example TMs that you can try out. The state diagram is interesting, and represents a graphical alternate to the transition table as the definition of a TM, as well as enabling you to follow progress graphically. Arguably, though, Turin provides more information that could be useful in debugging, so there might be some benefit in using both. That would involve maintaining two versions of the source code, though, and keeping them in step with each other, because the syntax used to define their TMs is different.

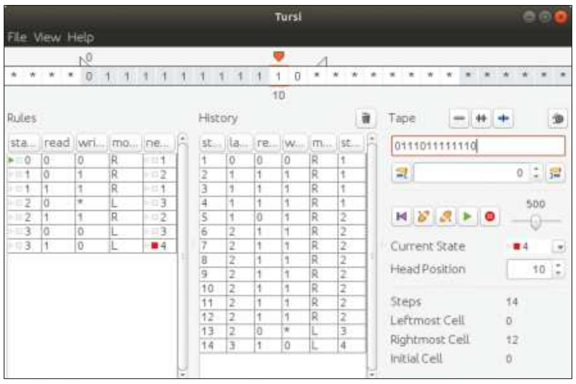
To get a feel for the operation of the *Turing Machine Vizualization*, we suggest you start with a simple example and, although it uses binary arithmetic instead of unary, it's still trivially simple. This is a TM to increment a binary number which is the second example from the menu in the title bar. Since we're now into the realm of binary numbers, and therefore we need both zeros and ones, the number to be incremented is delimited by spaces.

If you step through this example you'll notice that the current state is always highlighted on the state diagram and the arrows representing the transitions are highlighted as they occur. You can easily change the start pattern on the tape – that is, the binary number to be incremented – by editing the code in the panel at the right and clicking Load Machine. It should be fairly obvious how this TM works.

To understand how laborious it can be to carry out on a TM what would be trivially simple on any conventional processor, how about trying out another of



This might be a somewhat fanciful representation of a Turing Machine, but it illustrates the key principles.



```
*  *  *  *  0  1  1  1  1  1  1  1  1  1  1  0  *  *  *  *  *  *  *  *  *  *  *  *  *  *
                                             10
```

Rules

sta...	read	wri...	mo...	ne...
0	0	0	R	1
1	0	1	R	2
1	1	1	R	1
2	0	*	L	3
2	1	1	R	2
3	0	0	L	3
3	1	0	L	4

History

st...	la...	re...	w...	m...	st...
1	0	0	0	R	1
2	1	1	1	R	1
3	1	1	1	R	1
4	1	1	1	R	1
5	1	0	1	R	2
6	2	1	1	R	2
7	2	1	1	R	2
8	2	1	1	R	2
9	2	1	1	R	2
10	2	1	1	R	2
11	2	1	1	R	2
12	2	1	1	R	2
13	2	0	*	L	3
14	3	1	0	L	4

Tape

0111011111110

0

500

Current State	4
Head Position	10
Steps	14
Leftmost Cell	0
Rightmost Cell	12
Initial Cell	0

The Tursi simulator offers an excellent way of creating a Turing Machine and putting it through its paces.

the binary arithmetic examples, namely the one for addition? Even with the two four-bit binary numbers defined in that example, it takes no fewer than 92 steps to carry out the addition!

Before moving on, we'll suggest an example for you to try yourself from scratch with little in the way of guidance. The only bit of advice we'll provide is that, below the state diagram and the editable code panel, the syntax for defining the TM is described. The exercise we're suggesting is a traffic light controller. The one we implemented followed the sequence used in the UK (red>red+amber>green>amber>red ...) but you could implement the sequence used in other countries.

Although your Turing Machine might venture into a blank cell, depending on how you implement it, your machine should write to just four positions on the tape – the left-hand one which defines the phase of the lights, IE which combination of lights should be lit on the next change, and three others that can contain either a zero or a one to represent whether the red (top), amber (middle) and green (bottom) lights are illuminated. This TM shouldn't halt, but should execute forever, as do real traffic lights.

The traffic light problem makes a nice change from arithmetic operations, but it isn't a particularly difficult exercise. After all, despite rarely having delved into TM programming previously, we soon got it up and running. However, at the expense of being accused of coming up with something we didn't have the time or confidence to try out ourselves, we're suggesting a more taxing problem: adding together two decimal numbers. We'll give a bit of advice though. The normal solution involves decrementing one number until it reaches zero and, for each such decrement, increment the other number. There's an example for incrementing a binary number in the *Turing Machine Vizualization*, and that could provide a starting point for incrementing a decimal number and, although there's no binary decrement, they're not in short supply so you'd easily be able to find one.

Universal machines

So far we've seen several examples of TMs that were designed to do a particular job. But although you can define a TM to execute any logical or arithmetic function, this isn't the same as the general purpose computers we're all familiar with. However, Alan Turing did envisage exactly such a device, long before such a machine ever materialised. The machine that did that –

another hypothetical thought experiment – was the Universal Turing Machine that we'll refer to as a UTM.

A UTM is a TM, in just the same way as all the other TMs we've seen so far. Where it differs, though, is that the tape isn't initialised only with the data on which it would operate, but also with data that defines the function of a particular TM that will operate on that data. The parallel with today's stored program computers is evident. Just as a PC stores its data and a program in memory, a UTM holds both data and the definition of a TM (a close parallel to a program) on its tape. UTM simulators have been written but, given how many steps it took on a TM to add two four-bit binary numbers, you can probably get a feel for how laborious a UTM would be in its operation. For that reason, we're not going to recommend that you try one out yourself, but simulators do exist if you really want to delve more into the hypothetical machine that paved the way to pretty much all today's computers.

It's interesting to think about how a TM, including a UTM, could be improved to make it more efficient and various people have done exactly that. One example is a two-tape TM. We can imagine that this would be an effective approach for a UTM since the data could be held on one tape and the program on the other, thereby reducing the amount of stepping back and forth between the data and the program areas of a single tape UTM.

Going one stage beyond multiple tape TMs is the 2D TM which, instead of operating on a one-dimensional tape, operates on a two dimensional grid. This takes us a bit closer to random access memory, as opposed to the original TM's sequential memory, and again it would result in a significant reduction of the movement of the read/write head on the tape or grid. If you fancy trying out a 2D Turning Machine, be sure to take a look at the Try a 2D Turing Machine box (box on page 83), but there's one important thing we need to stress.

The fact is that a multiple tape TM or a 2D TM is no more powerful than the original single tape TM. All other things being equal, these derivations and others would be faster, perhaps considerably so, but they wouldn't be more powerful. And going even further, your PC or even the world's fastest supercomputer, the 514 PFLOP/s Fujitsu Fugaku with its 7,299,072 cores, is no more powerful than a standard TM. An apparently bizarre statement, you might think, but in the realm of the theory of computation, power refers to what a computer architecture is cable of doing. And the fact is that the humble Turing Machine can solve any problem that can be solved by the world's fastest supercomputers.

As we draw to a close, however, we should point out the corollary of our last statement, namely that if a TM can't solve a given problem then it couldn't be solved on a multi-million core beast of a machine. What's more, Alan Turing came up with a problem – the so-called Halting Problem – that can't be solved on a TM. Today, of course, most PC applications run forever until the user chooses to close it. If you're a programmer, however, you've no doubt written code that's supposed to solve a particular problem and then halt. If you get it wrong, however, it's all too common for that code to run forever, never halting, at least for some combination of input data. So it might be quite handy for debugging, therefore, if you had a utility that analyses a program and its data and indicates whether it will halt or run forever.

The Halting Problem is supposed to answer this question using a TM, which analyses the definition of another TM and its data. And it wasn't just that Turing couldn't figure out how to solve the problem. Using a rather clever logical argument, he proved that it's uncomputable. So, you couldn't solve the Halting Problem for a TM on your PC, and nor could you determine whether a program written in Python or Java will get stuck in an infinite loop. In at least one sense, therefore, we'd have to conclude that we've really not progressed since 1936. Or, more positively, we surely have to recognise how future-looking Alan Turing's concept was when he devised it all those years ago. LXF

» A TM-ISH PROGRAMMING LANGUAGE

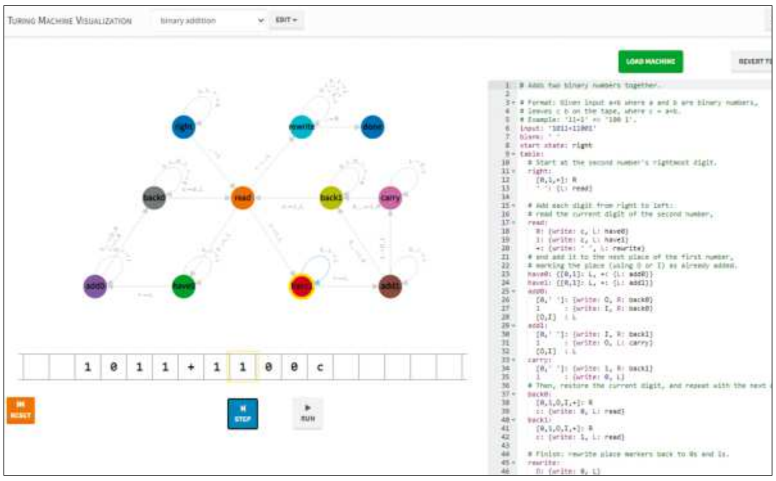
If you fancy trying out another minimalistic form of computation you could try Brainf**k (we've censored the official name). It differs from a TM in that it's a programming language so you don't have to deal with states, but in common with TMs, it operates on a one-dimensional list of memory locations that a pointer can move along, one location at a time. Like a TM, it has just eight instructions, which are represented by single ASCII characters. The instructions move the pointer left or right, increment or decrement the value at the pointer, output or input the value at the pointer, and implement a loop. Despite its simplicity, like the TM, this language is universal.

Brainf**k interpreters for Linux are common, as are compilers, both as source code and executables, which can be smaller than 200 bytes. However, one of the easiest solutions is to head over to **https://bit.ly/lxf271bf** and enter the program `,>,[<+>-]<------------------------------------------------.` which prompts for two single digit decimal numbers, adds them together and outputs the result, but only for single digit answers.

To help you understand it, the commas are inputs, `>` and `<` increment and decrement the pointer respectively, `+` and `–` increment and decrement the value at the pointer respectively, a `[ ]` paring executes the instructions between them until a zero result is encountered, and a full stop is the output instruction. Oh, and the 48 minus signs are needed because the values input and output are ASCII characters, in which code the figure zero is 48, one is 49, and so on.

QUICK TIP

Go to www.redfrontdoor.org/turing-mandelbrot. to see how the Mandelbot Set has been plotted on a TM. Amazing stuff!



```
# Adds two binary numbers together.

# Format: Given input a+b where a and b are binary numbers,
# leaves c b on the tape, where c = a+b.
# Example: '11+1' => '100 1'.
input: '1011+11001'
blank: ' '
start state: right
table:
  # Start at the second number's rightmost digit.
  right:
    [0,1,+]: R
    ' ': {L: read}

  # Add each digit from right to left:
  # read the current digit of the second number,
  read:
    0: {write: c, L: have0}
    1: {write: c, L: have1}
    +: {write: ' ', L: rewrite}
  # and add it to the next place of the first number,
  # marking the place (using O or I) as already added.
  have0: {[0,1]: L, +: {L: add0}}
  have1: {[0,1]: L, +: {L: add1}}
  add0:
    [0,' ']: {write: O, R: back0}
    1      : {write: I, R: back0}
    [O,I]  : L
  add1:
    [0,' ']: {write: I, R: back1}
    1      : {write: O, L: carry}
    [O,I]  : L
  carry:
    [0,' ']: {write: 1, R: back1}
    1      : {write: 0, L}
  # Then, restore the current digit, and repeat with the next
  back0:
    [0,1,O,I,+]: R
    c: {write: 0, L: read}
  back1:
    [0,1,O,I,+]: R
    c: {write: 1, L: read}

  # Finish: rewrite place markers back to 0s and 1s.
  rewrite:
    O: {write: 0, L}
```

The Turing Machine Visualization provides a different representation to Tursi by showing the TM's state diagram.

PYGAME

Classic pseudo-3D racing road effects

With a bag of silver coins in hand, **Andrew Smith** whisks us back to the classic arcade days to recreate pseudo-3D racing games.

OUR EXPERT

Andrew Smith is a software developer @ NHS Digital, has a Bachelors degree in Software Engineering and an MSc in Computer Networks.

Cue the budget wibbly-wobbly 'going back in time' special effects, as for this coding tutorial we're going to look at some the old-school techniques used in some of the classic racing video games such as *Road Rash* (1991), *OutRun* (1986) and *Pole Position* (1982). Designed to work smoothly on low-powered hardware, these are smart visual tricks that create a fake 3D effect.

What is known as pseudo-3D techniques were used to create a simulated 3D racing effect. The games would often be played by a single player or two players against computer opponents. The Pseudo 3D Road project (created by Ray Tomely) that we will be looking at, even though not a full video game, is a selection of examples of pseudo-3D techniques demonstrating ways to generate the 3D effect. You will see that once the project has been downloaded, these programming techniques are located in different folders that each demonstrate a pseudo-3D effect.

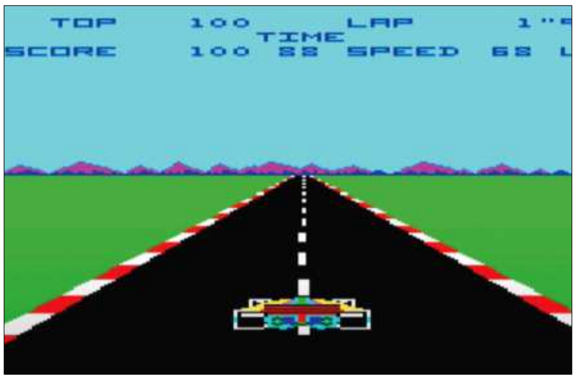
Getting started

To get started, we will need a few things: Python, PyGame and the Pseudo 3D road project. To install Python, open a terminal window (Ctrl+Alt+T) and type `sudo apt-get python3` followed by `sudo apt-get install pip3`. Then install the PyGame module by typing `pip3 install pygame`.

Finally, grab a copy of the Pseudo 3D road project from **https://raytomely.itch.io/p** by clicking the Download button towards the bottom of the page. Once everything has downloaded, extract the contents of the **pseudo_3d_road_collection.rar** file into an accessible location on your system.

As an example, the whole project has been put into a folder called **PythonProjects** which was created before downloading the project. If you are already an *LXF* fan, the source code and project can be retrieved from the **LXF281** DVD (find it here **bit.ly/lxf281**). This tutorial will focus on the source code located in the folder called **simple_road**. Type `cd simple_road` to get into it.

To edit and view the source code, you can either use a default text editor installed on your flavour of Linux (Ubuntu, for example) or you could use something more



The absolute classic racing game Pole Position by AtariSoft.

specific such as *Notepad++*, *PyCharm* or *VS Code*. The choice is entirely up to you. For this tutorial, we will be using *gedit* to view and edit the source files. When using this method to view/edit source files, it maybe helpful to open up two console windows where one will be used for editing/viewing source files and the other is a terminal window for executing the PyGame code.

Within the **simple_road** folder, the source code file that we want is **simple_road_curve_segment_demo.py**. Before we look at the source code, navigate into the **simple_road** folder and type the following to execute the script.

```
$ python3 ./simple_road_curve_segment_demo.py
```

On successful execution you should get the output as seen in the image on page 88. The script executed has only been implemented with a forward control using the Up cursor key, and you will need to close the program with a mouse to end the program. As you hold down the Up cursor key, you will see the generated road go from being straight to having a bend in it and then back to being straight again.

Getting curvy

To view/edit this source file type the following into another terminal window:

```
$ gedit ./simple_road_curve_segment_demo.py
```

You will see that it's commented throughout to indicate what the variables are used for. As with any other Python/PyGame script, at the top of the source code the libraries needed to run the script are declared. Also notice that continuing on from this, global variables for the script are declared to define the colours for the other parts of the road. See the following below:

```
import pygame,sys
from pygame.locals import *
BLACK=pygame.color.THECOLORS["black"]
WHITE=pygame.color.THECOLORS["white"]
RED=pygame.color.THECOLORS["red"]
GREEN=pygame.color.THECOLORS["green"]
BLUE=pygame.color.THECOLORS["blue"]
YELLOW=pygame.color.THECOLORS["yellow"]
SCREEN_WIDTH=640
SCREEN_HEIGHT=480
HALF_SCREEN_HEIGHT=int(SCREEN_HEIGHT/2)
```

The screen resolution settings `SCREEN_WIDTH` and `SCREEN_HEIGHT` are set to relatively low values in regards to today's device screen capabilities. Please feel free to change these for a higher resolution screen setting if you're not happy with them. In addition to changing this, you will need to add two lines of code to ensure that the images loaded will always meet the screen resolution specified.

Add the following lines just after the images **light_road.png** and **dark_road.png** have been loaded, as shown below.

```
light_road=pygame.image.load('light_road.png').
convert()
light_road = pygame.transform.scale(light_road,
(SCREEN_WIDTH, SCREEN_HEIGHT))
dark_road=pygame.image.load('dark_road.png').
convert()
dark_road = pygame.transform.scale(dark_road,
(SCREEN_WIDTH, SCREEN_HEIGHT))
```

You will also need to alter the following line of code in the script. Change

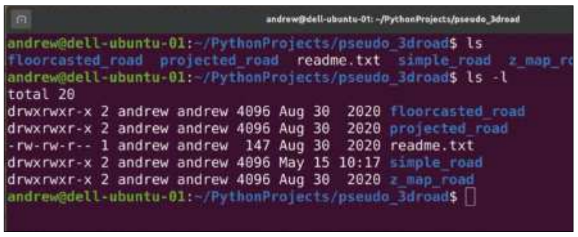
```
bottom_segment={'position':240,'dx':0}
```

to

```
bottom_segment={'position':SCREEN_
HEIGHT/2,'dx':0}
```

If the `SCREEN_WIDTH` and `SCREEN_HEIGHT` are changed and the above changes are not done, the display will look odd when running the program.

The main operation of the script is carried out in a defined `main()` function which is shown in part here:



```
andrew@dell-ubuntu-01:~/PythonProjects/pseudo_3droad$ ls
floorcasted_road  projected_road  readme.txt  simple_road  z_map_ro
andrew@dell-ubuntu-01:~/PythonProjects/pseudo_3droad$ ls -l
total 20
drwxrwxr-x 2 andrew andrew 4096 Aug 30  2020 floorcasted_road
drwxrwxr-x 2 andrew andrew 4096 Aug 30  2020 projected_road
-rw-rw-r-- 1 andrew andrew  147 Aug 30  2020 readme.txt
drwxrwxr-x 2 andrew andrew 4096 May 15 10:17 simple_road
drwxrwxr-x 2 andrew andrew 4096 Aug 30  2020 z_map_road
andrew@dell-ubuntu-01:~/PythonProjects/pseudo_3droad$ 
```

The final folder structure once the project has been downloaded.

```
def main():
  pygame.init()
  #Open Pygame window
  screen = pygame.display.set_mode((640, 480),) #add
RESIZABLE or FULLSCREEN
  #Title
  pygame.display.set_caption("simple road")
  #font
  font=pygame.font.SysFont('Arial',30)
  #images
  light_road=pygame.image.load('light_road.png').
convert()
  dark_road=pygame.image.load('dark_road.png').
convert()
  light_strip=pygame.Surface((SCREEN_WIDTH,1)).
convert()
  dark_strip=pygame.Surface((SCREEN_WIDTH,1)).
convert()
  light_strip.fill(light_road.get_at((0,0)))
  dark_strip.fill(dark_road.get_at((0,0)))
  #variables
  texture_position=0  #this is used to draw the road
```

As with all PyGame scripts, PyGame is initialised with a `pygame.init()` function call as can be seen from the above code. Also notice that when the screen display is set up, where there is the line:

```
screen = pygame.display.set_mode((640, 480),) #add
RESIZABLE or FULLSCREEN
```

it misses out using the variables `SCREEN_WIDTH` and `SCREEN_HEIGHT` declared above. Replace the value of 640 with `SCREEN_WIDTH` and the value of 480 with `SCREEN_HEIGHT` so that the screen resolution can be adjusted to suit your device.

Notice that there are just two images loaded to create the intended effect and the rest of the effect is created by use of internal colour schemes to create alternating strips.

» CREATING THE ROAD

The road is created with a combination of two images loaded near the beginning of the script, **light_road.png** and **dark_road.png**, and also two colour strips created with light and dark tones for the grass area outside of the road. The colour of the grass is picked from the light and dark images respectively for the grass colour. Let's look at the code that generates the graphics for the road.

```
  if texture_position<half_texture_
position_threshold:
          screen.blit(light_
strip,(0,i+HALF_SCREEN_HEIGHT))
          screen.blit(light_road,(curve_
value,i+HALF_SCREEN_
HEIGHT),(0,i,SCREEN_WIDTH,1))
        else:
          screen.blit(dark_
strip,(0,i+HALF_SCREEN_HEIGHT))
          screen.blit(dark_road,(curve_
value,i+HALF_SCREEN_
HEIGHT),(0,i,SCREEN_WIDTH,1))
```

As can be seen, when rendering the road, an `if/else` statement is used to determine whether to render a dark part of the road or a light part of the road. The grass outside of the road is rendered at the same time the road is. What is also worth pointing out is that when the strips are drawn, they stretch from one side of the window to the other; the road is overlaid on top of the strips to give the illusion the grass is at each side of the road. To help create the curve effect in the road (left-hand bend), only a part of the image for light and dark road are redrawn with an offset x-position by `curve_value`. Each image segment which is rendered on the screen is incremented slightly more than the last to give the road a left-bend.

»

In every PyGame program there needs to be a main loop that renders the graphics used in the program and that also controls input for the game via keyboard and/or mouse depending on the PyGame application being developed. Let us have a look at the main loop implemented in **simple_road_curve_segment_demo.py**.

```
while True:
    #loop speed limitation
    #30 frames per second is enough
    pygame.time.Clock().tick(30)
    for event in pygame.event.get():    #wait for events
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
    #Movement controls
    keys = pygame.key.get_pressed()
    if keys[K_UP]:
```

As can be seen from the code above, the main `while` loop ends on a Boolean condition to identify when the program has finished. In the case of the above, the loop ends when a False condition is raised. Continuing on from this, the frame rate is declared at 30 frames a second and further on from this the input/keyboard control is set up.

Finally, after doing all this, we come to the final part of the loop as shown below:

```
    pygame.display.flip()
```

From the above we update the contents of the entire display. The main focus of the render processing is done in the PyGame main `while` loop as shown below:

```
while True:
  # Setup / Control code
  #Movement controls
  keys = pygame.key.get_pressed()
  if keys[K_UP]:
    road_pos+=road_acceleration
    if road_pos>=texture_position_threshold:
      road_pos=0
    top_segment['position']+=curve_speed
    #if we reach the curve's end we invert it's
incrementation to exit it
    if top_segment['position']>=curve_map_lenght:
      top_segment['position']=0
      bottom_segment['dx']=top_segment['dx']
      top_segment['dx']-=0.01  #+0.01 to exit a left curve
and -0.01 to exit a right curve
      top_segment['dx']*=-1
```

As can be seen from the above code segment, when the Up cursor key is pressed, the road position is increased as per the `road_acceleration value` (currently set to 80) and when the value of `road_pos` exceeds the value of `texture_position_threshold` (currently set to 300), the value of `road_pos` is set back to 0 and the process is repeated. If you think of a conveyer belt or treadmill mechanism where the track used is on a continuous cycle or loop, this is a very similar idea here for the pseudo-3D effect. The road is continually re-generated after a certain point is reached.

To help control when the bend on the road appears, there is the following line of code after that which deals with the Up cursor key and after the initial road processing code:

```
curve_map_index+=curve_increment
```

Initially, the value of `curve_map_index` is set to -1, however the value is incremented by `curve_increment` (currently set to 2) on each cycle that the player has the Up cursor key pressed.

As the Up cursor key is pressed, you should start to see the road go into what looks like a left bend and then back to being a straight road again. Let's look at the code that controls this.

```
if curve_map_index>=curve_map_lenght:
   curve_map_index=curve_map_lenght
   curve_increment*=-1
#if we exit, we invert its incrimintation to enter again
 #we invert the curve's direction to change the way
  elif curve_map_index<-1:
    curve_increment*=-1
    curve_direction*=-1
```

The point at which the left bend appears and goes back to being a straight road again is controlled by an `if else` statement, or in Python `if elif`, as the Up cursor key is pressed. The first part of the `if` statement deals with the situation where if the curve has reached its maximum curvature it starts to make the road look straight again. The second part of the statement (`elif`) deals with the case of making the road bend to the left.

For those that are new to incrementing variable values in Python, it may be worth pointing out that `curve_increment *= -1` is equivalent of writing `curve_increment = curve_increment * -1`. This is again the same with the variable `curve_direction = curve_direction * -1`.

Graphics rendering

To help learn about how this program works further, you may find it useful to play around with the following variables:

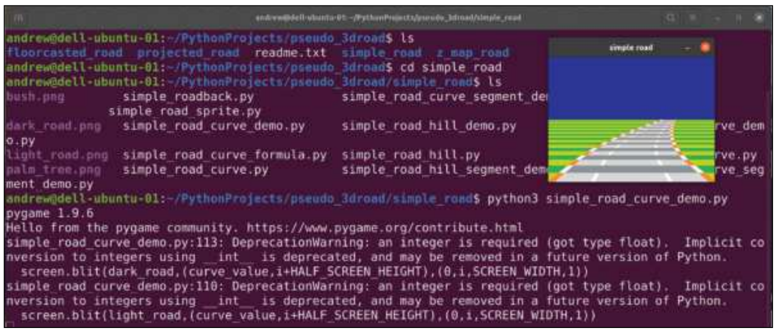
```
road_pos=0  #remembers our position on the road
road_acceleration=40  #the speed we traverse the road
texture_position_acceleration=8  #strip "stretch" value
texture_position_threshold=300  #strip division value
half_texture_position_threshold=int(texture_position_
threshold/2) #define drawing light or dark road
```

To save time scrolling through the code in getting to what you want, use the IDE search facility usually brought up by pressing Ctrl+F and then type the name of the variable or function you are looking for.

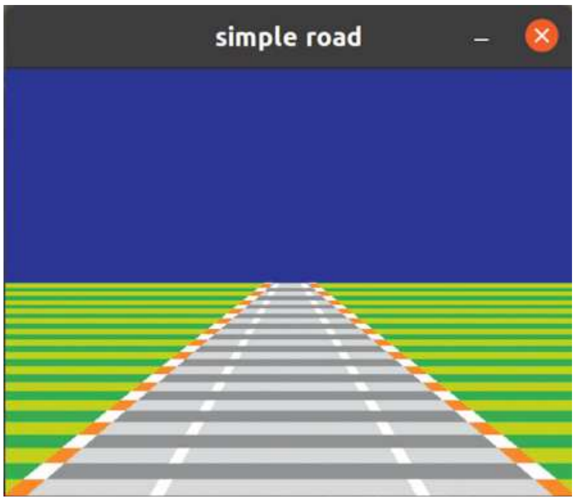
The two variables that might first be of interest to play around with values first could be `road_acceleration` and `texture_position_acceleration`.

Change the values of each variable and then run the program as shown before to see the effect this has on the running of the program. You may want to do this a

This is what a successful execution of program should look like.



```
andrew@dell-ubuntu-01:~/PythonProjects/pseudo_3droad$ ls
floorcasted_road  projected_road  readme.txt  simple_road  z_map_road
andrew@dell-ubuntu-01:~/PythonProjects/pseudo_3droad$ cd simple_road
andrew@dell-ubuntu-01:~/PythonProjects/pseudo_3droad/simple_road$ ls
bush.png        simple_roadback.py           simple_road_curve_segment_de
              simple_road_sprite.py
dark_road.png   simple_road_curve_demo.py    simple_road_hill_demo.py          rve_dem
o.py
light_road.png  simple_road_curve_formula.py  simple_road_hill.py              rve.py
palm_tree.png   simple_road_curve.py         simple_road_hill_segment_dem      rve_seg
ment_demo.py
andrew@dell-ubuntu-01:~/PythonProjects/pseudo_3droad/simple_road$ python3 simple_road_curve_demo.py
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
simple_road_curve_demo.py:113: DeprecationWarning: an integer is required (got type float).  Implicit co
nversion to integers using __int__ is deprecated, and may be removed in a future version of Python.
  screen.blit(dark_road,(curve_value,i+HALF_SCREEN_HEIGHT),(0,i,SCREEN_WIDTH,1))
simple_road_curve_demo.py:110: DeprecationWarning: an integer is required (got type float).  Implicit co
nversion to integers using __int__ is deprecated, and may be removed in a future version of Python.
  screen.blit(light_road,(curve_value,i+HALF_SCREEN_HEIGHT),(0,i,SCREEN_WIDTH,1))
```



Our retro road running the basic straight road effect.

number of times to get used to the effect of using different values.

In every PyGame program there has to be a part that is written to render all the graphics so after user input they can be updated. This is done with a `for` loop situated in the second half of the `while` loop as seen in part below:

```
for i in range(HALF_SCREEN_HEIGHT-1,-1,-1):
  if top_segment['position'] < i:
     dx = bottom_segment['dx']
  else:
      dx = top_segment['dx']
  ddx += dx
  current_x += ddx
  curve_map[i] = current_x
  curve_value = curve_map[i]
  if texture_position<half_texture_position_threshold:
      screen.blit(light_strip,(0,i+HALF_SCREEN_
HEIGHT))
      screen.blit(light_road,(curve_value,i+HALF_
SCREEN_HEIGHT),(0,i,SCREEN_WIDTH,1))
  else:
     screen.blit(dark_strip,(0,i+HALF_SCREEN_
HEIGHT))
     screen.blit(dark_road,(curve_value,i+HALF_
SCREEN_HEIGHT),(0,i,SCREEN_WIDTH,1))
```

Going further

The rendering of the graphics is by default set to take place in the bottom half of the screen and not the top half, hence the use of `HALF_SCREEN_HEIGHT`. The above code renders both the dark and light sections of the road onto the bottom half of the screen. The top half of the screen is kept clear for the blue sky effect.

After you have gained more confidence with the code shown, you may want to add further functionality to the existing code. As you're currently forced to close the program by closing the window with a mouse, it may be a good idea to end the program when the Escape key is pressed instead.

The colour of the sky is currently a plain blue. A possible addition here would be to create an image of a white cloud in an image editor such as *Gimp* and then create an animation of clouds moving across the top half of the screen.

» GENERATING HILLS

An example of how to create a hill effect is given in the Python script file **simple_hill_road.py**. From viewing the code, you will see that new variables have been added to create the hill effect. `hill_position`, `hill_velocity`, `hill_acceleration` and `hill_sharpness` are the main ones which have been added. The code that deals with the Up arrow key event is the same but the rendering process is slightly different even though similar in some ways to how the left-bend was created in **simple_road_curve_segment_demo.py**.

Again, you will see that a `for` loop is used for rendering the graphics to the bottom half of the screen. The hill effect for the road is created with the following code.

```
hill_position+=hill_velocity*hill_sharpness
```

To increase how steep the hill is, increase the value of `hill_sharpness` which is currently set to the value of 4. To make the road appear flat again, set the value to 0. The position of the hill is continually tracked as well to determine whether to regenerate a light part of the road or a dark part of the road.

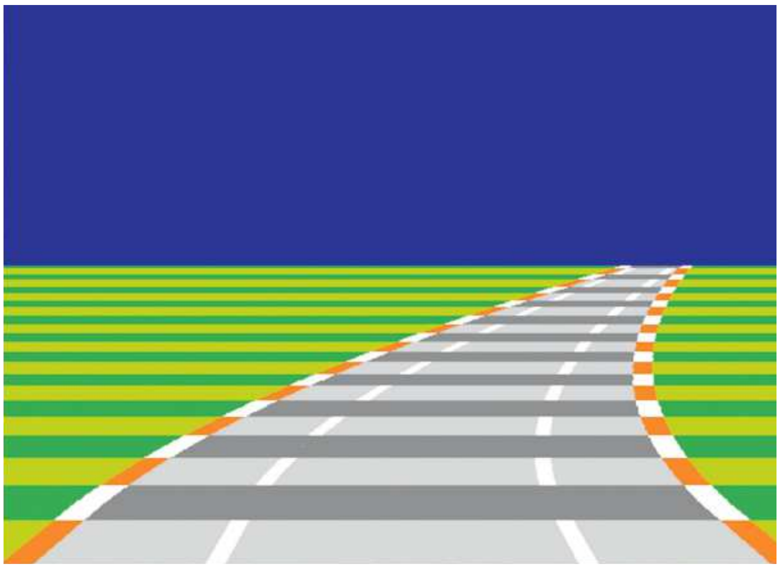
There is currently recorded a current position of the hill and old position which the difference between the current hill position and the old hill position is continually worked out.

As can be seen in the script, the acceleration value is set to a constant 80 per Up cursor key press. Those of you who from a physics background may want to create a more realistic acceleration algorithm where the rate of acceleration may gradually increase and decrease.

Even though it may involve some planning and some re-construction of the existing code, since components of a race track have been demonstrated – straight roads, bends and hills – it maybe a good exercise to build a small race track.

In other source code examples, images of vehicles have been added. It maybe a good idea to take the existing image(s) and modify them in an image editor to create an image of a vehicle turning left or turning right and implementing into the script, writing code that will handle the left and right cursor keys. Instead of scrolling through the code line by line looking for what you want, instead press Ctrl+F which will usually bring up a search facility and type in a variable or function name. LXF

With a bit of maths the basic road can be made to bend.



MERMAID

How to code diagrams, graphs and pie charts

Mihalis Tsoukalos explains how to use Mermaid to create beautiful technical graphs and diagrams you can reuse and update anywhere.

OUR EXPERT

Mihalis Tsoukalos is a systems engineer and a technical writer. He's also the author of Go Systems Programming and Mastering Go, with a third book in the works!

The subject of this tutorial is Mermaid, which is a Markdown dialect. Put simply, Mermaid code looks like Markdown code and enables you to create flows, diagrams, charts and more. It's based on JavaScript that renders the Markdown as code to generate the output, because Markdown doesn't offer support for drawing. Mermaid is supported by many editors including *Microsoft Visual Code, typora* and online editors which we'll cover here.

Mermaid code is delivered in blocks. The first line of a Mermaid block specifies the type of the plot. Valid values include graph, pie, sequence diagram and flowchart. Each plot type has its own syntax, and it's much easier to include Mermaid code in Markdown files that use the md file extension (**filename.md**), which is what we'll be using for the examples of this tutorial.

A Mermaid block in a Markdown file begins with `` ```mermaid `` and ends with `` ``` `` – this is the Markdown way of telling that the embedded code has a given property, which in this case is Mermaid. However, Mermaid code can also be saved on its own in files that use the mmd file extension (**filename.mmd**).

You can find the code used in this tutorial at **https://www.linuxformat.com/includes/download.php?PDF=LXF274.code.tgz**.

QUICK TIP

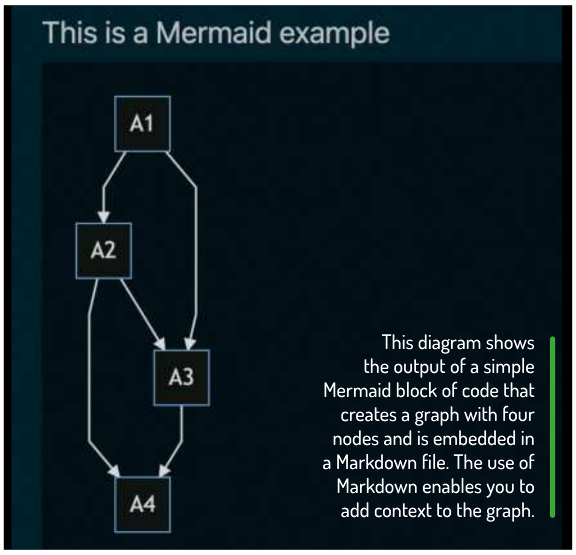
If Mermaid isn't to your liking, try Vega and Vega Lite, which enable you to create graphs and plots. Both grammars are based on D3.js and JavaScript. Learn more about Vega and Vega Lite at https://vega.github.io/ and at https://github.com/vega/vega.

Installing Mermaid... or not!

You don't need to install Mermaid. All you need is an editor that supports Mermaid to see the output from a Mermaid block. The following is a Markdown file, saved as **mermaid.md**, with a block in Mermaid format:

~~~
## This is a Mermaid example

```mermaid
graph TD;
  A1-->A2;
  A2-->A4;
  A1-->A3;
  A2-->A3;
  A3-->A4;
```
~~~

The first line is plain Markdown and generates a title in the output. Then, we declare that we're creating a graph that's going to be rendered from the top to the bottom (TD: Top-Down) of the page. Next, we add nodes and we connect them with other nodes. The `-->` notation shows that the node on the left is connected to the node on the right. The graph contains four nodes, named A1, S2, A3 and A4. If you want to include more nodes and connections between them, just declare what you want – Mermaid takes care of the output.



This diagram shows the output of a simple Mermaid block of code that creates a graph with four nodes and is embedded in a Markdown file. The use of Markdown enables you to add context to the graph.

This example illustrates one of the simplest uses of Mermaid; it can do many more tasks. However, it also illustrates the simplicity of the Mermaid dialect.

The output of the previous code can be seen in the diagram above with the help of *Microsoft Visual Code* – more on this next.

Microsoft Visual Code

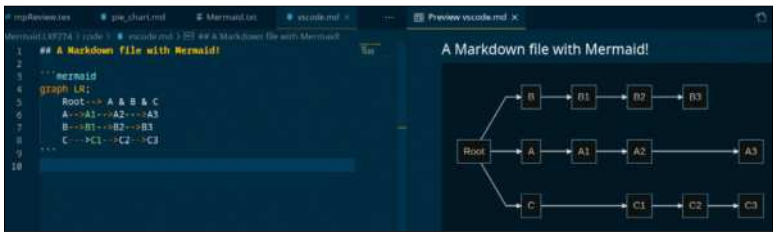
Microsoft Visual Code has a extension for rendering Mermaid files and previewing Mermaid output inside *Visual Code*. Go to the Extensions tab, search for the "Markdown Preview Mermaid Support" extension and install it. It would also be helpful to install the "Mermaid Markdown Syntax Highlighting" extension so that you can see your Mermaid code beautifully coloured.

Next, we render the Markdown file **vscode.md** inside *Microsoft Visual Code*:

~~~
## A Markdown file with Mermaid!

```mermaid
graph LR;
  Root--> A & B & C
  A-->A1-->A2--->A3
  B-->B1-->B2-->B3
  C--->C1-->C2-->C3
~~~
```

This code presents an alternative way of creating graphs. Instead of writing multiple `A--><node>` lines, you can simply write `A--><n1> & <n2> & <n3>` or `A--><n1>--><n2>--><n3>` in order to make multiple node connections. Using `--->` instead of just `-->` creates longer edges between nodes, as in `C--->C1`.

The screenshot opposite shows the preview of **vscode.md** using the Mermaid extension for *Visual Code*.

Creating a pie chart

Mermaid can easily plot pie charts and can even do the calculations for you so that the percentages of the pie chart are correct. To create a pie chart, you need to begin the Mermaid block with `pie`. The next line should begin with `title` followed by the title text. After that, you need to write the dataset that's used for generating the pie chart. Each line of the dataset is a different entry and contains a label, followed by a colon, which is followed by a positive numeric value. Note that the label text should be included in double quotes, which isn't the case with the pie chart title text.

The following block shows the definition of a pie chart in the Mermaid dialect:

```
## This is a Pie Chart in Mermaid

```mermaid
pie
  title Linux Distributions
  "Debian" : 42.96
  "Ubuntu" : 50.05
  "Arch" : 10.01
  "CentOS" :  5
```
```

The previous code is saved in the **pie_chart.md** file – render it on your own to see its output.

Creating a flowchart

Flowcharts in Mermaid start with the flowchart keyword, followed by their orientation (LR or TD). Then you add the nodes of the flowchart as you would with graphs. If you want to add text in an edge, you should use the `<n1> -- Text --> <n2>` notation. You can add text to any node by including it after the node name. If you include the text in square brackets, the node is drawn as a rectangular; use curly braces to draw the node as a diamond.

The following Mermaid code, which can be found as **flow_chart.md**, describes a flowchart:

```
## Flow chart

```mermaid
flowchart TD
  A[Choose OS] --> B{Do you want?};
  B -- Yes --> C[UNIX or Windows];
  C -- Windows --> D[Good Luck!];
```



~~~
## A Markdown file with Mermaid!

```mermaid
graph LR;
    Root--> A & B & C
    A-->A1-->A2--->A3
    B-->B1-->B2-->B3
    C--->C1-->C2-->C3
~~~
```


mpReview.tex
pie_chart.md
Mermaid.txt
vscode.md
Preview vscode.md
A Markdown file with Mermaid!
Root
A
A1
A2
A3
B
B1
B2
B3
C
C1
C2
C3


This shows Microsoft Visual Code rendering a Mermaid document. The code is previewed in a separate window from the Markdown file.

```
  C -- UNIX --> E{Linux or macOS?}
  E -- Linux --> G{Good Choice!}
  E -- macOS --> H{apple.com!}
  B -- No ---> F[End];
```
```

When rendered, the code generates a flow chart that can help you choose an Operating System. The diagram on the next page shows the flowchart generated by this code with the help of *Microsoft Visual Code*.

Sequence diagrams

A sequence diagram describes interactions and how things are carried out. They're helpful when you want to describe situations that depend on many factors.

A sequence diagram begins with the `sequenceDiagram` keyword followed by the list of participants, which are defined using the participant keyword (each participant is defined in a separate line). After that, you need to define the connections and interactions between participants, which in Mermaid terminology are called Messages.

This Mermaid code produces a sequence diagram – it's part of the **seq_diagram.md** Markdown file:

```
  sequenceDiagram
    participant User
    participant CTC
    participant CSV
    participant Auth as Authorization
       loop Every minute
         CTC-->CSV: Keep updating!
       end
    User-->>Auth: Send Authorization details
    Auth-->>User: Get back authorization token
```

» MERMAID VS GRAPHVIZ

Another useful tool that can help you create graphs and plots is *Graphviz* (**http://graphviz.org**). It's is a collection of tools for manipulating graph structures and generating graph layouts that supports both directed and undirected graphs. It offers both graphical and command-line tools. *Graphviz* uses its own language.

The good thing is that *Graphviz* is open source and its language is easy to learn and use. As with Markdown, you can write *Graphviz* code using a simple plain text editor such as *vi*. A wonderful side effect of it is that you can easily develop scripts that generate *Graphviz* code!

So, Mermaid and *Graphviz* have many things in common including using plain text files and producing output in various formats. However, with Mermaid, you don't need to install anything whereas for *Graphviz*, you need to install the *Graphviz* suite of tools and learn its language. Furthermore, Mermaid uses Markdown, which is more popular than the language of *Graphviz* and very handy if you're already using it for blogging or writing. As is usually the case with most tools, only after using both *Graphviz* and Mermaid for some time will you know which option is best for you.

»

QUICK TIP
The official web site for Mermaid is at https://mermaid-js.github.io/mermaid. The GitHub repository for the Microsoft Visual Studio Mermaid extension is at https://github.com/vstirbu/vscode-mermaid-preview. Learn more about the Markdown language at https://en.wikipedia.org/wiki/Markdown.

There are four participants: User, CTC, CSV and Authorization. The Authorization participant is accessed as Auth inside the Mermaid code but is displayed as Authorization for clarity – this is implemented with the `Auth as Authorization` statement. The loop block represents actions that happen continuously.

You should be able to see the output of **seq_diagram.md** either in *Microsoft Visual Code, typora* or any other online Mermaid editor such as **https://mermaid-js.github.io/mermaid-live-editor**.

Use Cases in Mermaid

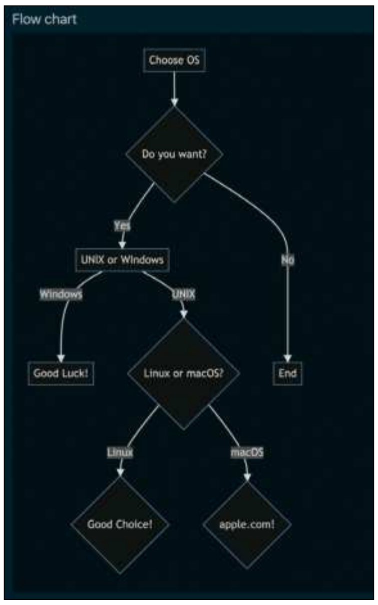
Let's see how Mermaid can describe the use cases of a project. Use cases are implemented using sequence diagrams, which means they're defined using the `sequenceDiagram` keyword. We want to use the Mermaid code to describe how a user interacts with a program and the components involved in the process. This is shown in the following Mermaid code:

```
sequenceDiagram
  participant U as User
  participant CTC
  participant CSV
  participant CTC_Auth as CTC/Auth
  autonumber
  rect rgb(0, 100, 0)
   par Get Data
    CSV->>CTC_Auth: A Database that keeps Usage
data
    Note right of CSV: REST API
    CTC_Auth-->>CSV: Return Usage data
   end
  end

  U->>CTC: Logs in
  CTC-->>U: Returns Security Token
  CTC->>CSV: Opens CSV Window
  U->>CSV: Asks for system visualisations
  CSV-->>CSV: Query Internal Database
  CSV-->>U: Displays Visualisations
```

The autonumber keyword that's shown here attaches a sequence number to each arrow in the sequence diagram. The `rect rgb(0, 100, 0)` command puts colour in the output using RGB values and colours a rectangle. Inside that rectangle, you can write the Mermaid commands you want. The par block shows actions that happen in parallel. Both `rect` and `par` blocks end with the `end` keyword.

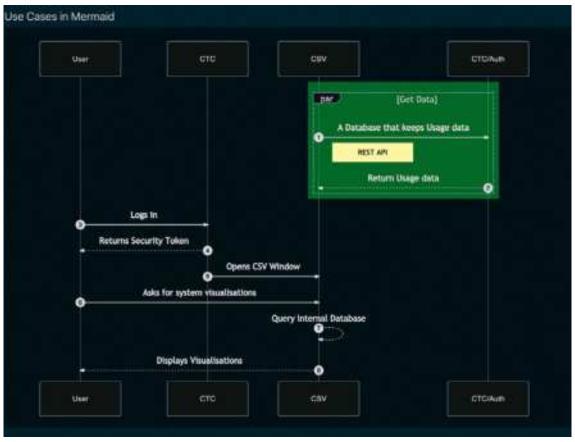
The diagram above displays the output of the previous Mermaid code – the Markdown source file with the Mermaid code is called **use_cases.md**. The use of `autonumber` enables you to specify the part of the sequence diagram you want by using a number.

Here's a flowchart generated in Mermaid, which can be drawn from left to right or top to bottom.



Mermaid and Hugo

The Hugo (**https://gohugo.io**) framework offers support for Mermaid code. This section presents such as example where we embed Mermaid code into a Hugo blog post that we publish. In order to support Mermaid rendering in Hugo, you need to add a small shortcode inside the **layouts/shortcodes** directory – Hugo uses shortcodes to add extra functionality that can't be implemented in Markdown. The name of the shortcode file should be **mermaid.html** and must have the following content:



This shows how Mermaid can describe use cases and therefore offers a handy way of generating documentation for your software projects.

```
$ cat layouts/shortcodes/mermaid.html
<script async src="https://unpkg.com/mermaid@8.2.3/
dist/mermaid.min.js"></script>
<div class="mermaid">
{{ .Inner }}
</div>
```

The previous file defines a shortcode named `mermaid` – in order to use that shortcode you should embed Mermaid commands in {{<mermaid>}} and {{</mermaid>}} blocks. A part of the Markdown code for the Hugo blog post is as follows:

```
Using [Mermaid](https://mermaid-js.github.io/
mermaid/) in Hugo.

{{<mermaid>}}
graph TD;
  A-->B;
  A-->C;
  B-->D;
  C-->D;
{{</mermaid>}}
```

The first line is Markdown code. After that you need to put all Mermaid-related code in a block that begins with {{<mermaid>}} and ends with {{</mermaid>}}.

The diagram on the next page shows the code inside the Hugo blog post – the entire Markdown file for that blog post is called **useMermaid.md**. You can see the blog post live at **www.mtsoukalos.eu/2021/01/using-mermaid**. Bear in mind that the Mermaid code is rendered with the help of the JavaScript library used in the shortcode file (**https://unpkg.com/mermaid@8.2.3/dist/mermaid.min.js**) and not by Hugo itself.

Go scripts

This section presents a Go utility that generates Mermaid code, but you can use any programming language you want to produce Mermaid code. The utility reads a CSV file with two columns and creates a Markdown file with a title and a pie chart in Mermaid format. The first column in the CSV file is the label and the second column is the value, which should be an integer. The filename of the input file becomes the title of the Markdown document as well as the title of the pie chart and the contents of the file generate the elements of the pie chart.

The most interesting code of **pieChart.go** is

```
for _, line := range lines {
  _, err := strconv.Atoi(line[1])
  if err != nil {
    continue
  }
  fmt.Printf("\t\"%s\" : %s\n", line[0], line[1])
}
```

The input file is read all at once and kept in the lines slice. The previous code processes the elements of the lines slice one by one to obtain the data and print it on the screen. The presented code ignores all rows that don't contain a valid integer with the help of the `strconv.Atoi()` Go function.

The input file that's processed by **pieChart.go** has the following format:

```
Linux,75
Windows,10
...
```

Running **pieChart.go** with the aforementioned input file produces the following output:

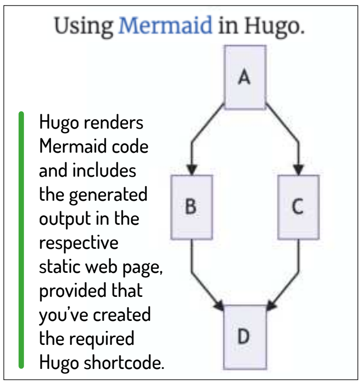
~~~
## data.csv
```mermaid
pie
    title data.csv
    "Linux" : 75
    "Windows" : 10
    ...
```
~~~

The easiest way to collect the output is to redirect it into a file: run **pieChart.go data.csv** and then **filename.md**. After saving the output you can process it in *Microsoft Visual Code* or *typora* and obtain the type of output you're looking for.

The key takeaway here is that once you understand the format of the Mermaid file you want to create, you can generate Mermaid files dynamically using input from multiple data sources, including plain text files, database servers and web services.

Python 3 scripting

Finally let's look at how Python 3 utility that generates Mermaid output on the fly using an external service. For the purposes of this tutorial, the Mermaid code is hardcoded and kept in a Python variable. If you have experience in the Python programming language, you can add some versatility to the presented utility and write code that reads the Mermaid code from a plain text file or a database server instead. Note that you might need to install some Python 3 modules for the script to work. The presented Python 3 script saves the output into a file that's named **nodes.png** – the name of the output file is hardcoded.



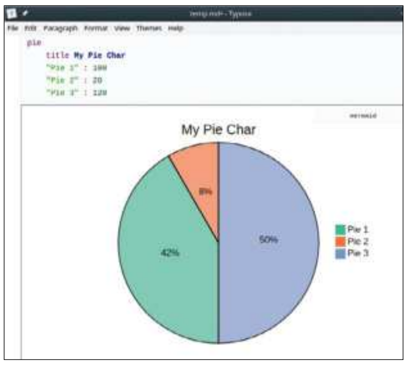
Hugo renders Mermaid code and includes the generated output in the respective static web page, provided that you've created the required Hugo shortcode.

The core functionality of the **script.py** Python 3 file can be found in this code:

```
img = Image.open(io.BytesIO(requests.get('https://
mermaid.ink/img/' + base64_string).content))
plt.imshow(img)
plt.savefig('nodes.png', dpi = 300)
```

The first line uses an external web site (**https://mermaid.ink/img**) to render the Mermaid input. The second statement displays the image output – if you're running the Python 3 script from the command line, then this statement might not generate any output on screen – and the final statement saves the generated image into a PNG file. The biggest difference between the Go script and the Python 3 one is that the Python 3 script generates the output image by itself whereas the Go script generates Mermaid code that needs to be rendered first.

Don't get fooled by the use of Markdown and plain text files and think that Mermaid is limited in function. It's a professional tool that can help you create beautiful technical diagrams and plots, and include them in your blog posts, documents or reports. Experimenting with Mermaid is going to make you more productive – play around with it a little before using it in real projects. LXF



```
pie
    title My Pie Char
    "Pie 1" : 100
    "Pie 2" : 20
    "Pie 3" : 120
```

Here's typora in action, which can render Mermaid code as well as any other Markdown code and works on most popular operating systems including Linux. If you're not a fan of Microsoft Visual Code, give typora a try.

» USING TYPORA WITH MERMAID

Typora is a stable and easy-to-use Markdown editor and reader that also supports Mermaid and works on many operating systems including Linux. On an Arch Linux machine, you can install *typora* by running `sudo pacman -S typora`. Adjust the command to match your Linux variant to install *typora* on your own Linux machine.

The biggest advantage of *typora* is that it's an editor. You can see the real output as you write it without needing to take any extra steps unlike *Microsoft Visual Code*. Additionally, you get to see syntactic errors in your Mermaid code as soon as possible and you can export your files in various formats including PDF, LaTeX and HTML. However, you can't run *typora* from the command line because it's a graphical application, which means that *typora* can't be used for automation tasks.

The screenhow below shows *typora* in action when rendering Mermaid code – the UI of typora is simple and without distractions which makes it a pleasure to use. Therefore, if you want something lighter and faster than *Microsoft Visual Code*, *typora* looks like a sensible option. Additionally, *typora* is much more elegant than *Visual Code*. However, if you're already using *Microsoft Visual Code* on a daily basis then you might want to stick with it – the choice is yours! Find more about *typora* at **https://typora.io**.

MICRO:BIT

Build radio walkie-talkies with Python

Using a pair of BBC micro:bit devices, **Calvin Robinson** creates walkie talkies in Python.

OUR EXPERT

Calvin Robinson is a former assistant principal and Computer Science teacher with and a degree in Computer Games Design and Programming BSc (Hons).

The BBC micro:bit is an open-source micro computer. Launched in 2015/16, the BBC gave UK secondary schools the opportunity to apply for one free unit for each Year 7 pupil. The idea was that every child could keep their micro:bit for free and develop a taste for coding.

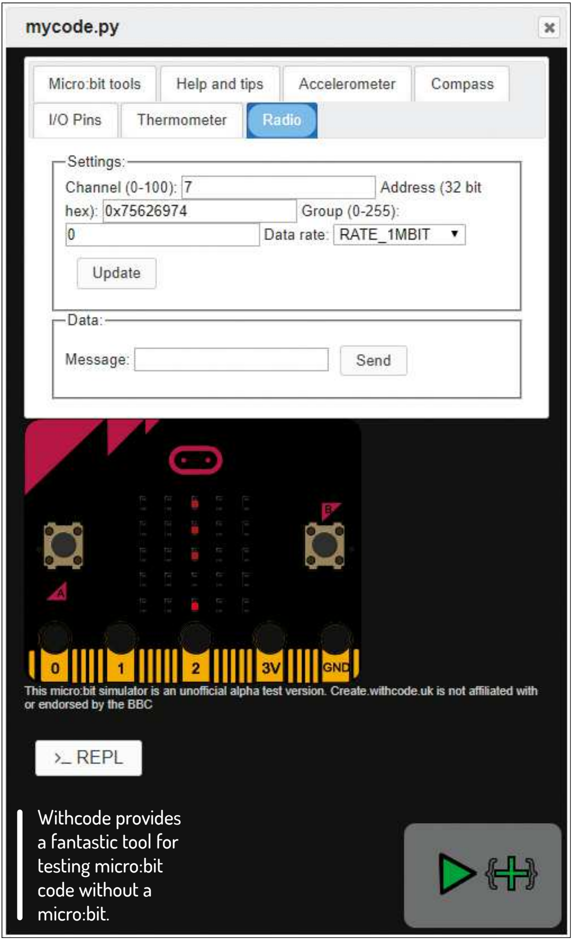
Certain UK readers might remember the original BBC Micro from our school days (1981-1994), an Acorn computer used for programming in BASIC. The micro:bit is its spiritual successor, designed to be cheap, adaptable, and a fraction of the size, ideally to be used alongside a PC or Raspberry Pi.

The micro:bit features 25 LEDs, two programmable buttons, as well as a non-programmable button, an accelerometer, magnetometer, Bluetooth Smart Technology and five Input/Output rings that are ideal for attaching things like motors and buzzers with crocodile clips. The unit is specifically designed for Blocks (a JavaScript editor) and Python, although it is also possible to program it with Free Pascal, Simulink, C__, Forth, Lisp, Rust, Ada and Swift. You can even run Zephyr OS on it.

For this tutorial, we're going to be coding in Python. There are a number of editors available for the micro:bit, including some simulators that will emulate the micro:bit hardware, so owning a unit isn't a barrier to entry. Visit **https://microbit.org/code/** and click 'Let's Code' at the top of the page for access to the Python Editor. The editor Mu is a great native GUI interface (**https://codewith.mu/**), whereas uFlash and microrepl are command line options. There's also a mobile app for iOS/Android from the official micro:bit website. For this tutorial we'll be using the platform at **https://create.withcode.uk/** as it provides a fancy simulated micro:bit tool to check if our code is functioning before exporting it onto the physical device.

We're going to program some basic walkie-talkie devices that send messages from one to the other. We can send pre-coded images and lines of scrolling text between micro:bits at the press of a button.

The micro:bit supports Bluetooth (BLE) and radio transmission, but BLE isn't supported by MicroPython (the version of Python running on the micro:bit), so for that reason we'll be using radio implementation to make our micro:bits communicate with each other. There's no limit to the number of micro:bits that can send/receive over radio transmission, other than the usual 2.4GHz signal traffic congestion. We would therefore recommend experimenting with no more than a handful of micro:bits to begin with (two is fine), and perhaps keep them in the same room until the code is fully functioning, so you can rule out the possibility of walls or objects interfering with the signal.



Settings:
Channel (0-100): 7
Address (32 bit hex): 0x75626974
Group (0-255): 0
Data rate: RATE_1MBIT

Data:
Message:

This micro:bit simulator is an unofficial alpha test version. Create.withcode.uk is not affiliated with or endorsed by the BBC

Withcode provides a fantastic tool for testing micro:bit code without a micro:bit.

QUICK TIP

If you make a mistake in your code, keep an eye on the LEDs. Micro:bits have their own debugger built in, so the error will scroll across the LEDs, starting with the line number.

Coding for the micro:bit

To get started on withcode, we'll need to import the micro:bit module and the specific functions for displaying images and using the two buttons, labelled 'a' and 'b':

```
from microbit import display, Image, button_a,
button_b
```

Now, to display text, we can either 'scroll' or 'show' a string of text. The display function contains two options, scroll and show, which will display the text accordingly – either from left to right, or one letter at a time:

```
display.scroll("Hello")
display.show("world")
```

Displaying images is a similar process. There is a whole host of built-in images that the micro:bit will recognise, including basic emojis:

```
display.show(Image.HAPPY)
sleep(1000)
display.show(Image.SAD)
sleep(1000)
```

But to draw your own images we'll need to highlight precisely which LEDs we'd like lit up, and how bright we'd like them on a scale of 0-9 – 0 being off and 9 being full brightness:

```
i = Image("00000:"
      "22222:"
      "44444:"
      "66666:"
      "88888")
display.show(i)
sleep(1000)
```

For an easy life, or to save space, we can narrow this down to a single line. The colon remains in place between lines of lights, but we narrow the code down to one pair of speech marks:

```
LEDS = Image("00000:11111:22222:33333:44444")
display.show(LEDS)
sleep(1000)
```

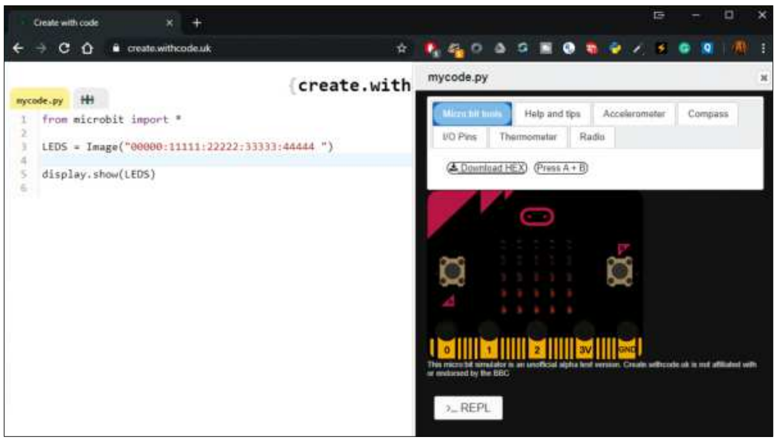
Here we've used a slightly brighter LED on each line to highlight the difference in hue.

Since this article was originally written in December in a festive mood, why not design a Christmas tree for our first image to send to another micro:bit later:

```
xmastree = Image("00500:00500:05550:55555:00500:")
display.show(xmastree)
```

It's all well and good being able to display our images on the LEDs, but before we're able to send them to another device we're going to need to work out the buttons. Here's a basic counter that can be used as an example of how the buttons function. We've used an infinite loop, with a small pause (sleep) so as not to cause a heavy load on the processor.

```
from microbit import *

counter = 0
new_value = 0

while True:

  # decrease by 1 if button A was pressed
  if button_a.was_pressed():
    new_value = counter - 1

  # increase by 1 if button B  was pressed
  if button_b.was_pressed():
    new_value = counter + 1

  # reset to 0 if you touch pin 0
  if pin0.is_touched():
    new_value = 0

  # stop counter from going less than 0
  if new_value < 0:
    new_value = 0

  # scroll new value if it's changed
  if new_value != counter:
    counter = new_value
    display.scroll(str(counter))

  sleep(50)
```



```
from microbit import *

LEDS = Image("00000:11111:22222:33333:44444 ")

display.show(LEDS)
```

Lighting up your life and LEDs, row by row.

QUICK TIP

The micro:bit uses Micro Python. More info and tutorials are available at https://microbit-micropython.readthedocs.io – including help with text, images, buttons, I/O, music, gestures, directions, and also storage.

» MORSE CODE

Micro:bits are perfect for Morse code communication. Instead of using images or strings to light to LEDs, we can use the *flash* command to light up the entire LED grid for a set amount of time. The following code will display a full flash for 10ms:

```
display.show(flash, delay=10, wait=False)
```

We could then take this and introduce a selection to make the micro:bit flash for different lengths, depending on what letter we want to represent. We can represent dots with short flashes and dashes with slightly longer flashes. For the purposes of this example, a dot is 5ms and a dash is 10ms:

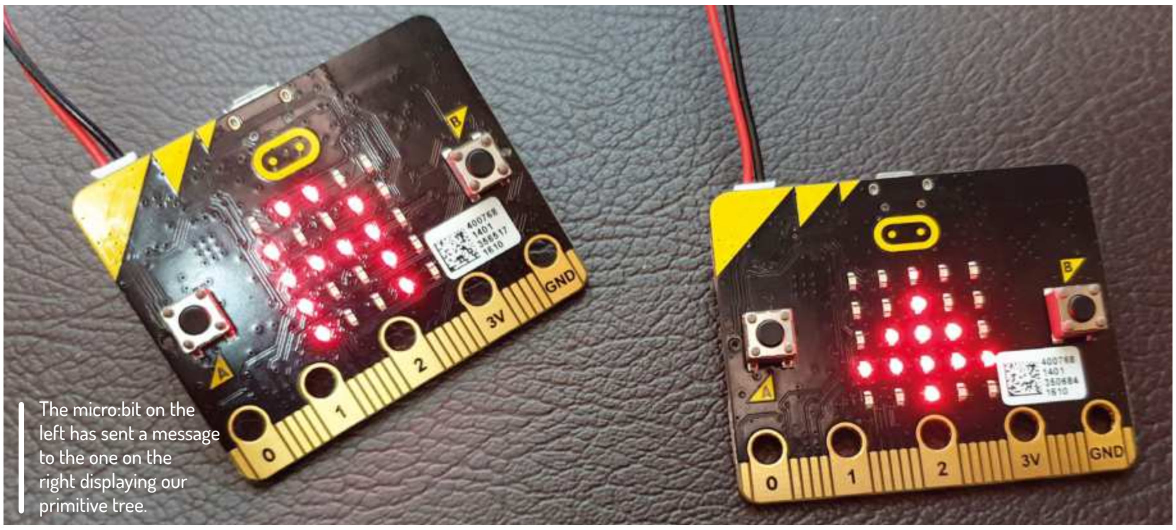
```
if incoming == 'A'
        display.show(flash, delay=5, wait=False)
        display.show(flash, delay=10, wait=False)
elif incoming == 'B'
        display.show(flash, delay=10, wait=False)
        display.show(flash, delay=5, wait=False)
        display.show(flash, delay=5, wait=False)
        display.show(flash, delay=5, wait=False)
```

After developing Morse code statements for each letter of the alphabet, we can then set up a loop to recognise each incoming letter, or only send single letters at a time. As we only have two buttons on the micro:bit, perhaps use one button to rotate through the 26 letters of the alphabet, showing them on the LEDs, and the other button to send the letter. This way we'd be able to send messages on the fly, without having to re-program the micro:bit every time we want to say something new.

»

QUICK TIP

Using crocodile clips on 0 and GND, you can attach a pair of headphones or speakers and introduce sound to your micro:bit. Import the music module and play music.


The micro:bit on the left has sent a message to the one on the right displaying our primitive tree.

Setting up the radio

In order to send these images or strings with our buttons, we'll need to connect the radio. Alter the top of your code to reflect the following:

```
from microbit import *
import radio
radio.on()
```

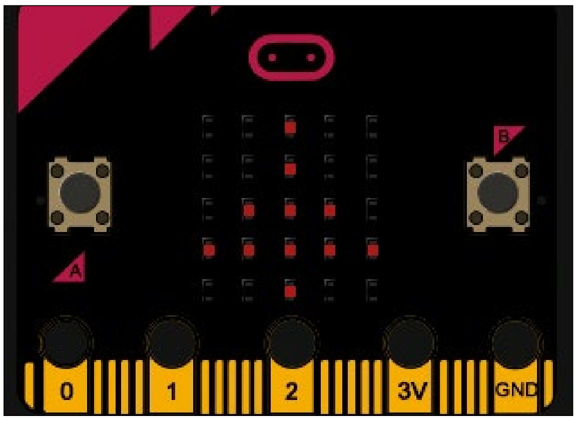
Now that the radio is turned on, we can send or receive messages over it. However, we might also want to be more specific and set a channel before turning the radio on:

```
radio.config(channel=10)
```

Sending messages

Let's start off by sending and receiving some basic messages. We'll set our A button to ask for a handle, and the B button can reply with it – a throwback to CB radios, if you will.

```
import radio
from microbit import *
radio.config(channel=0)
radio.on()

while True:
  if button_a.was_pressed():
    display.show('A')
    radio.send('What is your handle?')
  if button_b.was_pressed():
    display.show('B')
    radio.send('My handle is Calvin')
  incoming = radio.receive()
  if incoming:
    display.scroll(incoming)
  sleep(100)
```

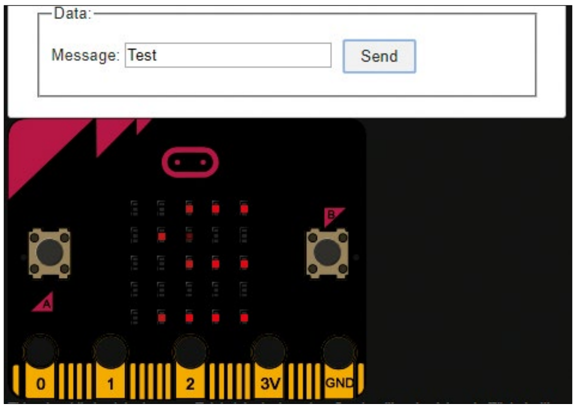

As basic as pixel art goes – a 2-bit pine tree.

If you're using two micro:bits for this tutorial, don't forget to set a different handle on each device for a real-world example of two-way communication. If you're using more than two micro:bits, however, you may need a way to change channels on the fly, while the device is running. While we don't have any more programming buttons, we do have the accelerometer functionality. Using the following code, we can shake the micro:bit to randomly change channel between 1 and 10, in iterations of 1:

```
while True:
  if accelerometer.was_gesture('shake'):
    newChan = random.randrange(1,10,1)
    display.scroll(str(newChan))
    radio.config(channel=newChan)
  if button_a.was_pressed():
    display.show('A')
    radio.send('What's your handle?')
  if button_b.was_pressed():
    display.show('B')
    radio.send('My name is Calvin)
  incoming = radio.receive()
  if incoming:
    display.scroll(incoming)
  sleep(100)
```

Of course, we could change these strings to say anything, or swap them out for variables. If we were going to replace them with our Christmas tree, it'd look something like this:

```
import radio
from microbit import *
radio.config(channel=0)
radio.on()

xmastree = Image("00500:00500:05550:55555:00500:")

while True:
  if button_a.was_pressed():
    display.show('A')
    radio.send(xmastree)
  if button_b.was_pressed():
    display.show('B')
    radio.send(xmastree)
```



Simulation of two-way communication over the micro:bit radio.

```
incoming = radio.receive()
if incoming:
  display.scroll(incoming)
sleep(100)
```

Our 'while' look is constantly looking out for an A or B button press but also for any incoming messages. We've set these messages to automatically scroll across our LEDs. It might be beneficial to clear the screen first with display.clear(). You may also decide to display.show() the message instead of scrolling it, depending on whether you plan to show images or text. We've found that text scrolls nicely but images are usually better to just pop up straight away with a display.show(). Again, add a sleep timer to the end of our loop to save processing power.

To take things to the next level, we could introduce a transition – a message to let the user know an incoming message has been received, before showing it. That way, the user has less chance of missing the message, which could easily happen if the text starts scrolling over the LEDs without prior warning.

```
if incoming:
  display.show(alertImage)
  sleep(10)
  display.scroll(alertText)
  sleep(10)
```

This would require a little setting up, before the 'while' loop:

```
alertImage = Image("00500:00500:00500:000000:009
00:")
alertText = "Incoming message:"
```

If you haven't got two micro:bit devices, you can test all of your code on the **https://create.withcode.uk** website, simulating communication both ways. Provided your code has imported the micro:bit module, turned on the radio and has a loop waiting to receive and display the incoming message, you can send yourself a message with the radio test tool. Make sure you're using the same channel number (anything between 0 and 100), and send some test data (preferably a simple string). It should scroll across your simulator micro:bit LEDs. You can also press the fake buttons on the micro:bit image and they should function as intended.

When you're happy with your code, download the HEX file. Once that's done, plug your micro:bit device into your computer using a Micro USB cable; it should automatically mount. Next, simply drag the HEX file onto your micro:bit's drive space. It should override any previously installed HEX file, since the micro:bit has extremely limited storage capacity and only room for one single HEX file.

As soon as the copying process is complete, the micro:bit will be running your Python code. Either keep the micro:bit plugged in via USB or provide power with a portable battery pack. Then press either the A or B button to begin. LXF

» SIGNAL STRENGTH

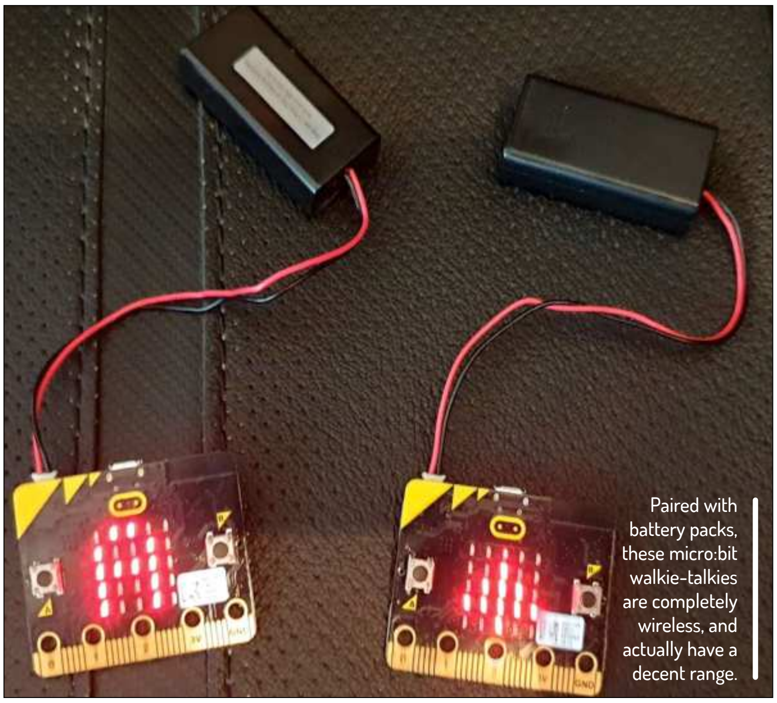
Micro:bits transmit some specific information every time we send or receive data over the radio. If we use radio.receive_full() instead of radio.receive(), we'll get the message as a tuple, including the message content, signal strength and a time-stamp. The signal strength (RSSI) will be displayed as a value between 0dBm and -255dBm, with 0 being the strongest and -255 the weakest signal.

Now we can create all kinds of treasure-hunt games by triangulating the radio signal. By setting one or multiple micro:bits up so that they constantly send a radio message on low power, we can create a receiver that can parse the incoming message and then display the signal strength:

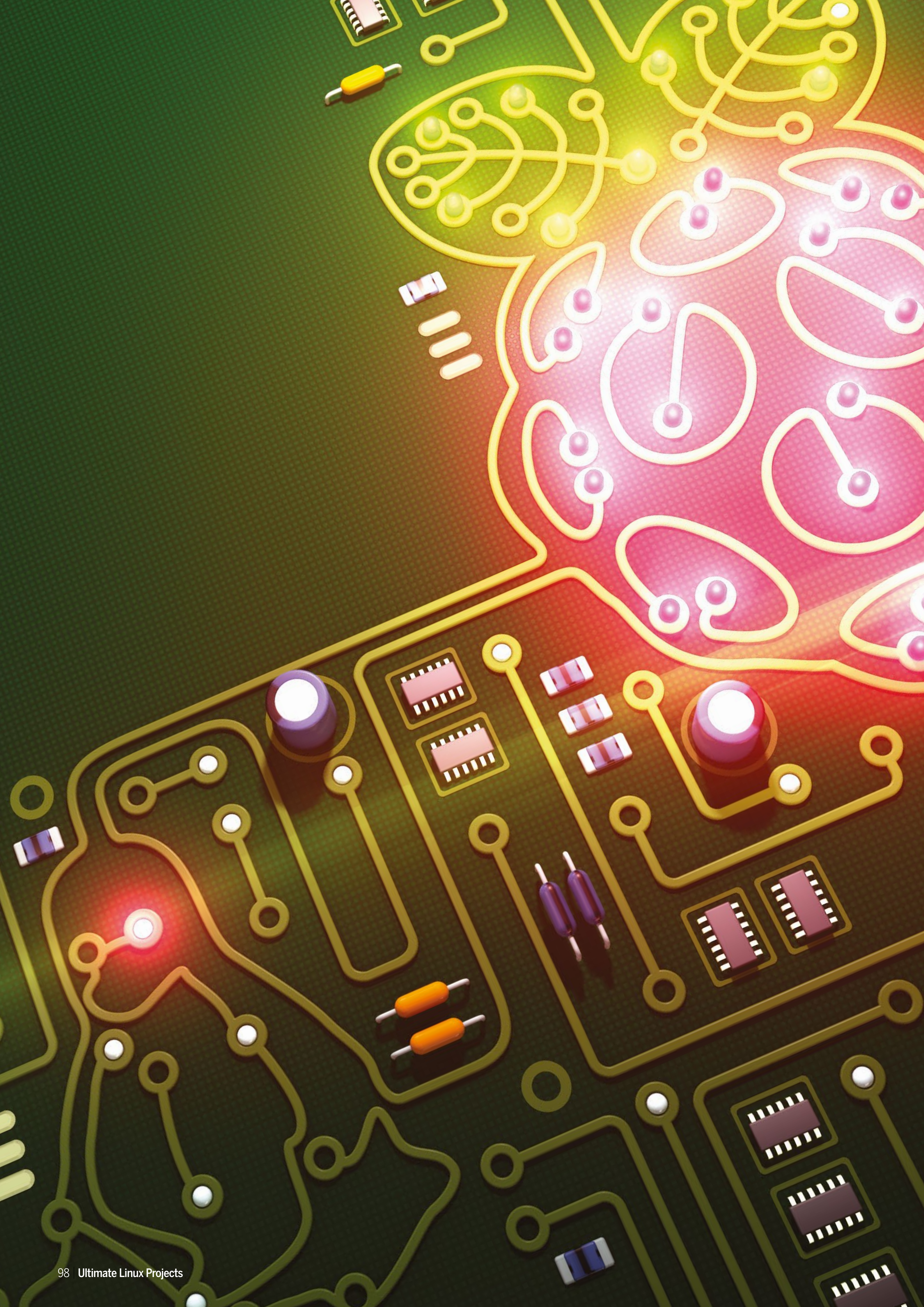
```
while True:
  message=radio.receive_full()
  if message:
    strength = message[1]+100
    displaystrength = (int((strength/10)+1))
    display.show(str(displaystrength))
    sleep(200)
  else:
    display.show(Image.NO)
```

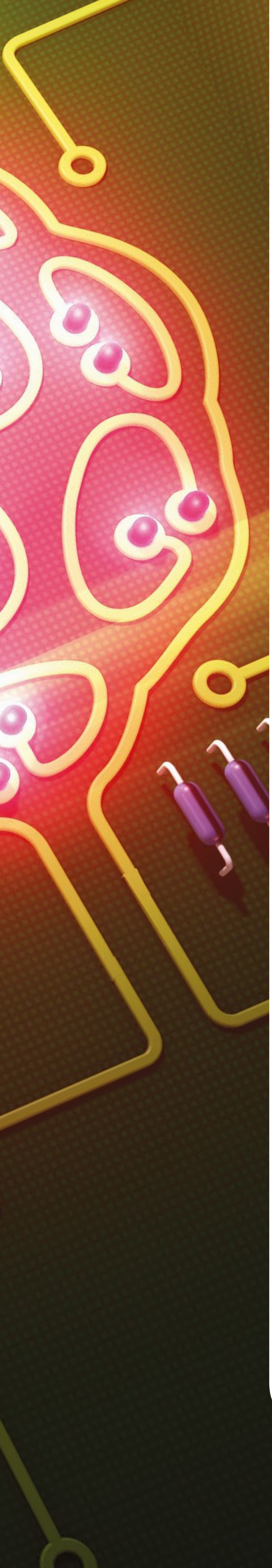
Now, the closer we get to the transceiver the higher our signal strength will be. We could make this into a game by designing an image for hot and another one for cold, then displaying these images as we move around. Every time the number rises we could display the hot image, signalling that we're getting closer, then displaying the cold image when we're moving away. The build-in commands Image.YES and Image.NO would display a tick and a cross, to help make things even easier.

```
Display.show(Image.NO)
```



Paired with battery packs, these micro:bit walkie-talkies are completely wireless, and actually have a decent range.





ULTIMATE RASBERRY PI PROJECTS »

PLUG & PLAY Pi PROJECTS

Jonni Bidwell with the help of Tom's Hardware bring you the best in Raspberry Pi-based fun.

We do love the Raspberry Pi at *Linux Format*, and so do our digital colleagues at tomshardware.com. So with their help we've selected the finest Pi projects for you to enjoy. We've got something to inspire everyone, from retro gaming to home security, or even artificially intelligent object classification.

The Raspberry Pi is the perfect device for learning and getting familiar with Linux. Its official operating system is based on Debian Linux and there's even a build of Ubuntu for it. The Pi 4 is powerful enough that it can happily replace your desktop, and small enough that you can hide it behind the back of your monitor. If that's not small enough, there's always the Pi Zero, which is perfect for budding 'Internet of Things' enthusiasts. And now there's the Pi 400, where the keyboard is the computer.

Of course, there's no need to stop at these projects, or any others you might find online. We want you to boldly go exploring new physical computing frontiers, and we hope that this offering inspires just that. With a bit of practice and experience, you'll be soldering up your own hardware, controlling robots and be limited only by your own imagination.

In association with tom'sHARDWARE

Get your kit ready

Prepare your USB cables, SD cards and a cup of tea. It's Raspberry Pi project time...

With more than 34 million units sold, the Raspberry Pi is not only one of the world's most popular computers – it's also one of the most important. Originally designed to help kids learn about technology, this inexpensive, single-board system is the leading choice for makers, developers and hobbyists who want to do everything from building industrial robots to setting up retro arcade machines. Whether you're aged eight or 80, if you love technology then the Raspberry Pi is made for you – and there are models ranging from £5 to £65 to suit any budget.

Whatever you do with your Raspberry Pi, you're going to need an SD card with an operating system on it. The official Raspberry Pi OS (formerly Raspbian) is a popular choice and in many cases is a great start for many projects. It's available in Desktop and Lite flavours at **www.raspberrypi.org/software/operating-systems**. There's an official *Raspberry Pi Imager* utility that you can use to write this (or a selection of other OSes) to an SD card, or you can use the *NOOBS* tool. We favour *Balena Etcher*, which has the advantage of being able to handle compressed images.

Troubleshooting

Throughout this feature we'll use a few different OS images, so we've made a handy guide to writing them below. If you run into difficulties, it's always worth remembering that a faulty SD card might be to blame, so it's worth trying a different one if things go wrong.

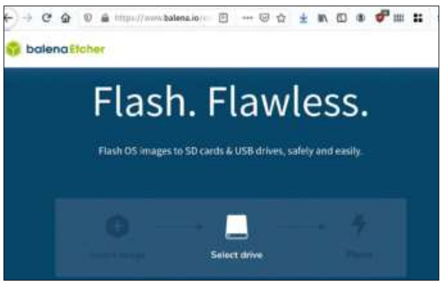
The latest incarnation of the Pi, the 400 model, embodies a Pi 4, also pictured, within a keyboard.

Cheap power supplies are another cause of errors. For older model Pis these are less of a problem, but the Pi 3 and above really do need a good 5.2V or they'll be subject to CPU throttling and instability. The Pi 4 has switched its power input from micro USB to USB-C, and unfortunately this new port isn't USB-C compliant. So if you use a fancy smart charger, it probably won't power the device, and might possibly damage it.

You won't run into these issues if you use the official charger, so do that. If you're opposed to that, we're told that cheaper USB-C cables (ones that lack the smart charge chip) work fine. Cheaper chargers should work too, but they may not be so reliable, so caveat emptor.

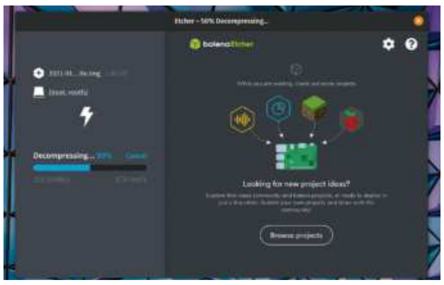
We've handpicked some fun, useful and interesting projects to showcase the tremendous scope of things that can be achieved with the humble Pi. From retro gaming to machine learning and Bitcoin, the possibilities are truly endless...

WRITING SD CARDS WITH ETCHER


balenaEtcher
Flash. Flawless.
Flash OS images to SD cards & USB drives, safely and easily.
Select drive

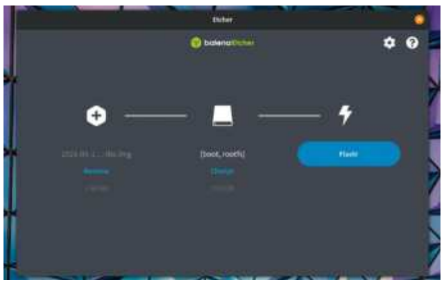
1 Download Etcher

You'll find Linux, macOS and Windows builds at **https://etcher.io**. On Linux it ships as a zipped AppImage, so once you've unzipped it, run `chmod 755` on the file (or select Properties> Permissions and check the box to allow execution). You can now double-click the AppImage to run it (or do so from the CL).



2 Prepare the target

Insert your SD card, select the Flash from file option and navigate to your downloaded image. Click Select target and a menu will show available devices. Hard drives are hidden to prevent you from accidentally overwriting them. Select your SD card reader, and check to make sure there's nothing important on the card.



3 Write the image

Hit Flash, enter your password and wait patiently for the progress bar. You'll be offered the chance to write the same image again, which you might want to do if you're preparing a fleet of Pi clones. But otherwise your SD card is ready for action. Move it from your PC to your Pi, power on and let the adventures begin!

»

Image recognition

Get into machine learning and train your Pi to recognise and classify other Pis, without having to write a single line of code.

YOU NEED

- **Raspberry Pi 3, 4 or 400**
- **8GB (or larger) microSD card**
- **Raspberry Pi Camera or a USB webcam**
- **Power supply for your Raspberry Pi**
- **Smartphone for taking photos**
- **Selection of Pis or other objects for classification**

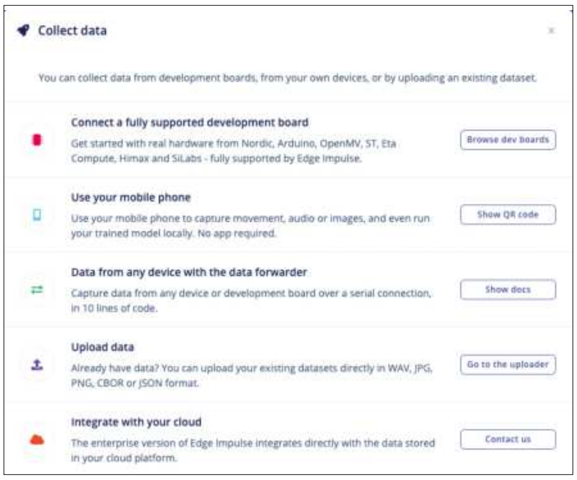
We're going to train our Raspberry Pi to identify other Raspberry Pis (or other objects) with machine learning (ML). Why is this important? An example of an industrial application for this type of ML is identifying defects in circuit boards. As circuit boards exit the assembly line, a machine can be trained to identify a defective circuit board for troubleshooting by a human.

Tom's Hardware has some neat machine learning and artificial intelligence tutorials, including facial recognition and face mask identification. In those projects, all of the training images were stored locally on the Raspberry Pi and the model training took a long time because it was also performed on the Pi. In this tutorial, we'll use a web platform called Edge Impulse to create and train our model to alleviate a few processing cycles from our Pi. Another advantage of Edge Impulse is the ease of uploading training images, which can be done from a smartphone (and without having to involve an app).

We'll use BalenaCloudOS instead of the standard Raspberry Pi OS since the folks at Balena have prebuilt an API call to Edge Impulse. This project eliminates all terminal commands and instead utilises an intuitive GUI.

Training on the Edge

Head over to **https://edgeimpulse.com** and create a free account (or login if you already have one), from a browser window on your desktop or laptop. Select Data Acquisition from the menu bar on the left. You can either choose to upload photos from your desktop or scan a QR code with your smartphone and take photos. In this tutorial, we'll opt for taking photos with our smartphone. Select Show QR code and a QR code should pop up on your device's screen. Scan it and select Open in browser and you'll be taken to a data collection site. You won't need to download an app to collect images.



You can collect data from development boards, from your own devices, or by uploading an existing dataset.

Connect a fully supported development board
Get started with real hardware from Nordic, Arduino, OpenMV, ST, Eta Compute, Himax and SiLabs - fully supported by Edge Impulse.

Use your mobile phone
Use your mobile phone to capture movement, audio or images, and even run your trained model locally. No app required.

Data from any device with the data forwarder
Capture data from any device or development board over a serial connection, in 10 lines of code.

Upload data
Already have data? You can upload your existing datasets directly in WAV, JPG, PNG, CBOR or JSON format.

Integrate with your cloud
The enterprise version of Edge Impulse integrates directly with the data stored in your cloud platform.

Besides collecting images from your mobile device, Edge Impulse can connect to all kinds of data sources.

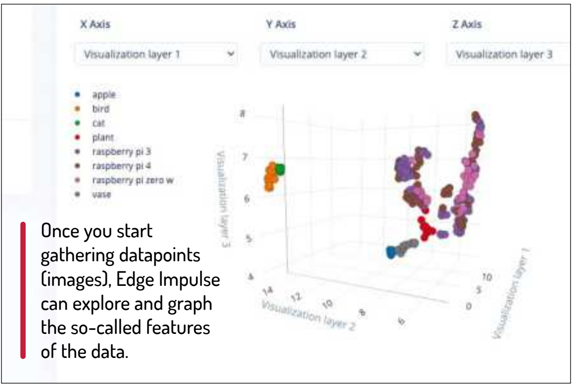
Accept permissions on your smartphone and tap Collecting images? in your phone's browser screen. Tap Label and enter a tag for the object that you'll take photos of. Take 30-50 photos at various angles. Some photos will be used for training and other photos will be used for testing the model. Edge Impulse automatically splits photos between training and testing. Repeat the process of Entering a label for the next object and taking 30-50 photos per object until you have at least three objects. We recommend three to five identified objects for your initial model. You'll have a chance to re-train the model with more photos and/or objects later on.

From the Data Acquisition tab in the Edge Impulse browser window, you should now see the total number of photos taken (or uploaded) and the number of labels (type of objects) you've classified. (You may need to refresh the tab to see the update.) You can click any of the collected data samples to view the uploaded photo.

Impulse design

Next, click Create impulse from Impulse design in the left column menu. Click Add a processing block and select Image to add an image to the second column from the left. Click Add a learning block and select Transfer Learning. Click the Save Impulse button on the far right. Click Image under Impulse design in the left menu column. Select Generate features to the right of Parameters near the top of the page. Click the Generate features button in the lower part of the Training set box. This could take around five to ten minutes (or maybe even longer) depending on how many images you have uploaded.

Select Transfer learning within Impulse design, set your Training settings (keep the defaults, check Data augmentation box), and click Start training. This step will also take five minutes or more depending on your amount of data. After running the training algorithm, you will be able to view the predicted accuracy of the model. For example, in this model the algorithm can only identify a Raspberry Pi 3 correctly 64.3 per cent of the time and will misidentify a Pi 3 as a Pi Zero 28.6 per cent of the time.



Once you start gathering datapoints (images), Edge Impulse can explore and graph the so-called features of the data.

Select Model testing in the left column menu. Click the top check box to select all and press Classify selected to test your data. The output of this action will be a percentage accuracy of your model. If the level of accuracy is low then we suggest going back to the Data Acquisition step and adding more images or removing a set of images. Select Deployment in the left menu column and select WebAssembly for your library. Scroll down (the Quantized option should be selected by default) and click the Build button. This step may also take three minutes or more, depending on your amount of data.

Setting up BalenaCloud

Instead of the standard Raspberry Pi OS, we'll flash BalenaCloudOS to our microSD card. The BalenaCloudOS is prebuilt with an API interface to Edge Impulse and eliminates the need for attaching a monitor, mouse and keyboard to our Raspberry Pi.

Create a free BalenaCloud account at **https://dashboard.balena-cloud.com/signup** and once you're in, go to **https://dashboard.balena-cloud.com/deploy** to open the Create and Deploy page and create our *balena-cam-tinyml* application. Click Deploy to Application. After creating your application, you'll land on the Devices page. Don't create a device yet!

In Balena Cloud, select Service Variables and add two variables. First, add to the service `edgeimpulse-inference` a variable named `EI_API_KEY` and in the Value field paste the API key from the Keys section of the Edge Impulse Dashboard. Add a second variable to the service named `EI_PROJECT_ID` and paste the Project ID value from the Dashboard.

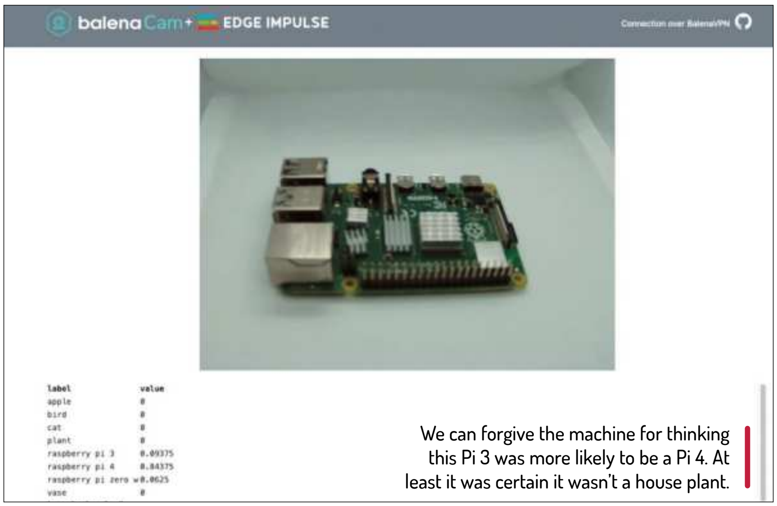
Select Devices from the left column menu in your BalenaCloud, and click Add device. Select your Device type (Pi 4, Pi 400 or Pi 3).

Select the radio button for Development. If using Wi-Fi, select the radio button for Wi-Fi+Ethernet and enter your Wi-Fi credentials. Download your customised balenaOS image and write it to an SD card (you can do this using our guide to *Balena Etcher* on the first page)

Connect the hardware

Remove the microSD card from your computer and insert it into your Raspberry Pi. Attach your webcam or Pi Camera and then power up your Pi. Allow 15 to 30 minutes for your Pi to boot up and BalenaOS to update. Only the initial boot requires the long update. You can check the status of your Pi Balena Cloud OS in the BalenaCloud dashboard.

Identify your internal IP address from your BalenaCloud dashboard device. Enter this IP address in a new browser Tab or Window. Place an object in front of the camera. You should start seeing a probability rating for your object in your browser window (with your internal IP address). Try various objects that you entered into the model and perhaps even objects you didn't use to train the model.



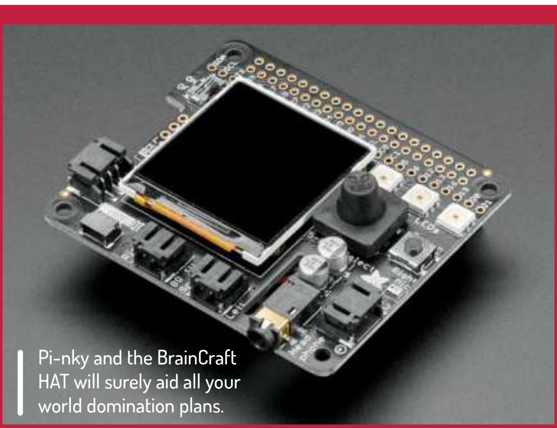
```
label               value
apple               0
bird                0
cat                 0
plant               0
raspberry pi 3      0.09375
raspberry pi 4      0.84375
raspberry pi zero w 0.0625
vase                0
```

We can forgive the machine for thinking this Pi 3 was more likely to be a Pi 4. At least it was certain it wasn't a house plant.

» REFINING THE MODEL

If you find that the identification isn't very accurate, first check your model's accuracy for that item in the Edge Impulse Model Testing tab. You can add more photos by following the Data Acquisition steps and then selecting Retrain model in Edge Impulse. You can also add more items by labelling and uploading in Data Acquisition and retraining the model. After each retraining of the model, check for accuracy and then redeploy by running WebAssembly within Deployment.

If you want to go further with machine learning, you might want to check out the Pimoroni's new BrainCraft HAT for the Pi 4. It features a 240×240 TFT IPS display for inference output, slots for camera connector cable for imaging projects, a five-way joystick, two microphones, audio outputs and much more. Most importantly, it has a controllable fan to keep the thing cool, what with all the never-ending TensorFlow computations.



Pi-nky and the BrainCraft HAT will surely aid all your world domination plans.

CREDIT: Pimoroni

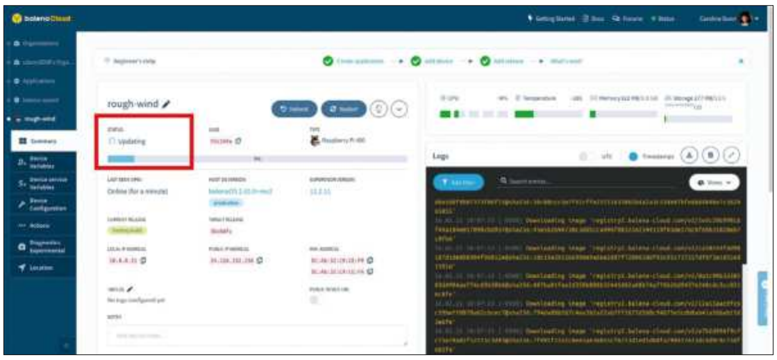
»

Multi-room Pi audio

Get your Raspberry Pis in sync, audibly throughout your home.

Multi-room audio systems can cost a pretty penny, but who needs to buy an expensive set of wireless speakers when you can use some Raspberry Pis and any 3.5mm wired speakers to achieve the same effect? We'll use our Pi 4s and speakers/receivers (with 3.5mm audio inputs) to play music from a streaming service perfectly in sync via Bluetooth from our phone, tablet or computer. We'll show you how to do this with a single speaker and Raspberry Pi and then replicate it in other rooms. This is an excellent project for repurposing old receivers. If you have a non-Bluetooth receiver with a 3.5mm jack input, you can connect your speakers to your receiver and your receiver to your Raspberry Pi.

Instead of Raspberry Pi OS, we'll use BalenaSoundOS to make the Bluetooth connection visible (to our other devices) and eliminate the need for a monitor, keyboard and mouse for our Raspberry Pi. Create a free BalenaCloud account at **https://dashboard.balena-cloud.com/signup** and log in. Open the Deploy a Balena-sound application page at **https://sound.balenalabs.io/docs/getting-started**. You must already be logged into your Balena account for that link to work. Click Deploy to Application and select Devices from the left column menu in your Balena-Sound Dashboard. Click Add device and choose Raspberry Pi 4, Raspberry Pi 400 or Raspberry Pi 3. Select the radio button for Development. If using Wi-Fi, select the radio button for Wi-Fi+Ethernet and enter your Wi-Fi credentials. Click Download balenaOS and a zip file will start downloading. Write this to an SD card, using *Balena Etcher* or otherwise.



Your musical Pi can be monitored at all times through the Balena Cloud dashboard.

YOU NEED

- **Raspberry Pi 4 or Pi 400 (one unit per room)**
- **8GB (or larger) microSD card, U3 cards recommended. One per room**
- **3.5 mm audio cables**
- **Amplified speakers or a receiver with 3.5mm/phono audio input**
- **Power supplies for your Raspberry Pis**
- **Optional: Smartphone or tablet**

Connect and update

Remove the microSD card from your computer and insert into your Raspberry Pi. Connect the speaker to your Pi via the 3.5mm audio cable. Power up your Pi. Allow 15 minutes for your Pi to boot up and BalenaOS to update. Only the initial boot requires the long update. You can check the status of your Pi Balena Sound OS in your BalenaCloud dashboard. If you're using a Raspberry Pi 400, connect a USB speaker instead of the 3.5mm audio jack. Wait for your Raspberry Pi BalenaOS update and confirm all systems are running.

From your smartphone, tablet or computer, navigate to your Bluetooth settings and look for BalenaOS XXXX and pair it to your device. Connect the device as the sound output for your smartphone, tablet or computer. Go to your streaming service and play a song. The music should stream via Bluetooth to your Raspberry Pi and play from the attached speaker.

» MULTI-DEVICE/MULTI-ROOM SYNC

For multi-device sync, repeat the setup stages we described above for each extra device. You can flash the same image from earlier if you're using the same model of Raspberry Pi. With the same Bluetooth device connected as mentioned earlier, music can now stream to all Raspberry Pis and their connected speakers. You will only need to connect one Pi via Bluetooth to your phone, tablet, or computer to stream to all Raspberry Pi Balena Sound speakers.

That's it! You can now create multi-room/multi-device sound experiences in your home with your Raspberry Pi devices while DJing from your smartphone, tablet, or computer like a pro.

If you're an audiophile you probably won't care much for Bluetooth audio, or indeed the headphone output on the Raspberry Pi. Fortunately there are a variety of DAC (digital analogue converter) HAT boards that enable faithful audio reproduction for not much money. IQaudio has been part of the Pi community since 2015 and we've long been a fan of its range of DAC and amplifier boards. Now IQaudio is part of Raspberry Pi and those boards are being added to the range of Raspberry Pi products.

Breathe life into unused bluetooth speakers and old amplifiers with Balena Sound.

Pi-based KVM over IP

Use your Pi as a remote control for your PC, even if it won't boot to an OS.

Those of you who have needed to access a PC remotely will probably have tried VPN or other applications such as *TeamViewer*. However, this kind of software only works within the remote computer's OS, which means that it can't access the BIOS, reboot, install an operating system or power on the computer. There are several solutions that make it possible for you to remote control a PC independently of its operating system, but using a KVM over IP is one of the most convenient and affordable approaches.

While a store-bought KVM over IP device can cost hundreds of pounds, it's easy to use a Raspberry Pi to create your own. A developer named Maxim Devaev designed his own system, Pi-KVM (**https://pikvm.org/**), which he's planning to sell as a $130 kit. However, if you have the right parts, you can use the software he's developed and your Pi, to put it together for far less.

Here, we'll show you how to build your own Raspberry Pi-powered KVM over IP that can output full HD video, control GPIO ports and USB relays, configure server power using ATX functions and more. You'll be able to control the whole setup via a web browser from another device on your local network.

YOU NEED

- **Raspberry Pi 4 or Zero**
- **16GB or larger microSD card**
- **HDMI-to-CSI bridge or HDMI-to-USB capture dongle**
- **USB female-to-dual male Type-A splitter**
- **USB C to Type-A cable**
- **5V, 3 Amp power supply with USB Type-A output. You'll be plugging a type-A cable into it so the official Raspberry Pi power supply won't do**

Download the image

The first thing we will need is to download the SD card image from **pikvm.org**. Note that there are different versions, depending on which Raspberry Pi you use and whether you use the HDMI-to-CSI bridge or an HDMI-to-USB capture dongle. The file is bz2-compressed, but *Etcher* can handle this automatically, so follow the guide from earlier and write it to an SD card.

Now we can move on to installing the HDMI-to-CSI bridge or USB-to-HDMI dongle and prepping the OTG USB-C cable. Connect the CSI ribbon cable from the HDMI-to-CSI bridge to the Raspberry Pi's CSI camera port. Make sure that the blue marking faces the black clamp. If you're using an HDMI-to-USB dongle instead, connect it to a USB port on your Pi. If you're using a Pi Zero then you'll need a microUSB-to-USB Type-A hub.

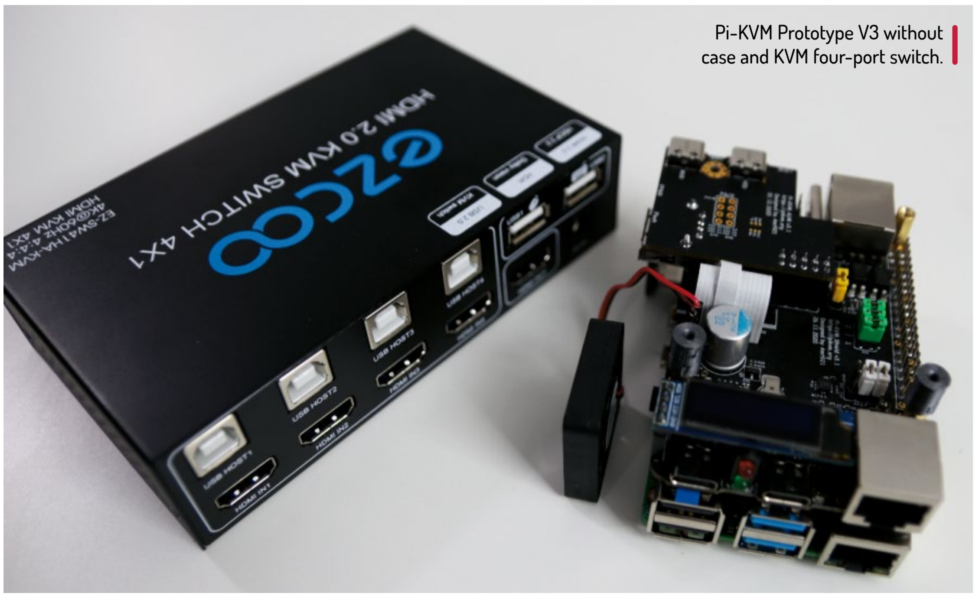
Disable the 5V pin on one of the USB Type-A male connectors from your splitter. The easiest way to do this is to place a small piece of Kapton tape over the right-most pin on the connector. You could also try cutting the lead to that pin, but that's more complicated. This will be the connector that attaches to a USB port on the PC you wish to control. If you don't disable that 5V pin then it'll back feed the power from your wall power to the PC, possibly causing damage to its USB port in the process.

Connect the USB C-to-A cable to the Type-A female connector on the splitter. This will provide power to the Pi. Connect the USB-C cable to the Raspberry Pi 4's USB-C port and connect the unmodified Type-A male to your power supply. Attach the USB Type-A connector and HDMI to the PC that you wish to remote control.

The Pi-KVM software

At this point we're ready to start using the Pi-KVM. On first boot it'll take longer then expected due to the initial process for enlarging the microSD card. Just be patient – it'll boot eventually. Navigate to the Pi's IP address (you can look it up in your router's control panel or use *Nmap*) in a browser on your client computer (the one you're using to control the other PC). You'll be redirected you to the login page, which you should log into using `admin` for both username and password.

Click the KVM icon. You should now be presented with a screen providing you with access to the remote PC and a number of other menus. Further options and instructions are available from the Pi-KVM GitHub.


Pi-KVM Prototype V3 without case and KVM four-port switch.

Gaming on RetroPie

Turn your Raspberry Pi 4 (or 3 or Zero) into a retro gaming rig with RetroPie and dubious internet downloads.

Back in the 1980s and 1990s, the arcades were the place to be. The latest video games from Namco, Sega, Konami and more ate our ten-pence coins as we pursued that high score. Those days are now sadly behind us, but retro gaming has seen massive growth in the past decade.

Using a Raspberry Pi, you can run a variety of emulators that enable you to play not only old-time arcade games, but also your favourite titles from many old consoles, including the Atari 2600, NES, Nintendo 64, Sega Mega Drive and Game Boy. There are a number of emulation platforms available, but RetroPie is by far the most popular and arguably the best.

TUCK INTO A SLICE OF RETROPIE

“Using a Raspberry Pi, you can run a variety of emulators that enable you to play arcade and console games.”

Up until recently, installing RetroPie on a Raspberry Pi 4 was a pain. Even though Raspberry Pi 4 came out in June 2019, RetroPie didn't officially support it for nearly a year and there were some manual steps you needed to take to make it work. Now, not only can you install RetroPie on a Raspberry Pi 4, but you can do it using the Raspberry Pi imager. It couldn't be easier. You can also grab the latest image from **https://retropie.org.uk/download** and use *Etcher*. Either way, there are separate editions for first-gen devices (the original Raspberry Pi and Pi Zero), multicore devices (the Pis 2 and 3) and the latest Pi 4 or 400 models. Eject the microSD card and insert it into your Raspberry Pi 4. Next, plug in a controller. Xbox One and PlayStation 4 pads are compatible with RetroPie via Bluetooth, but require a USB connection until Bluetooth is configured.

Power on the Raspberry Pi and wait for your controller to be detected. Configure your controller. Note that you probably will have fewer buttons on your controller than *RetroPie* asks you to configure. Just hold down a button that you've already configured to skip past any options you don't need. Press the A button on your controller to close the configurations menu and open the main menu.

Adding games To RetroPie

All the games exist as files called ROMs, which are dumps of real boards or cartridges. The easiest method to add games is to use a USB stick to transfer the files from your computer to your Raspberry Pi. It's important to note that games from retro consoles are under copyright no matter their age. If you don't personally own the cartridge/board and rip a copy of your own, downloading and distributing these ROMs may violate laws in your country, so proceed at your own risk.

On your computer create a folder called **retropie** on a FAT32/exFAT-formatted USB flash drive. Plug the USB flash drive into the Raspberry Pi 4 and RetroPie will create a folder structure inside the **retropie/** folder on the stick. Remove the flash drive and insert it into your computer. Copy your games to the correct system or console folder in **retropie/roms/**. For example, put Atari 2600 games in the **/atari2600** folder. Eject the flash drive and place it back into a USB port of the Raspberry Pi 4. How long this will take depends on how many games are transferred over: a matter of minutes for a few games; hours if transferring a large amount.

When complete, remove the USB flash drive and press START. From the menu select Restart Emulationstation/Restart System. The games list will now update and show the available computers/consoles for which there are now games. Bear in mind that, when it comes to playing retro arcade games from actual arcade machines, not all ROMs are compatible with all emulators. RetroPie can run with ten different versions of *MAME* (Multiple Arcade Machine Emulator), the most popular arcade emulator, and some ROMs will run on some versions of *MAME* and others will work on others. RetroPie maintains a handy chart over at **https://retropie.org.uk/docs/MAME**, but if your arcade games don't load in one *MAME*, try another.

» WELCOME TO THE PICADE

Pimoroni's Picade arcade cabinets have a long history of quality. Coming as a laser-cut, self-assembly kit in eight- or ten-inch screen sizes, Picade is easy to build and can be used with all models of Raspberry Pi, but for best results you'll want a Raspberry Pi 4.

Because Picade uses common arcade components, it can be customised to meet your playing style and aesthetics. If you like microswitched buttons and joystick then you can easily swap to get the edge in your favourite game. The Picade X HAT attaches to the GPIO of your Raspberry Pi and provides connections for the controls, USB-C power management and an audio DAC connected to a 5W speaker for authentic arcade audio. Everything you need to build your own arcade cabinet – just add a Raspberry Pi! A DIY kit with step-by-step instructions to build, wire and configure a custom cabinet offering a slice of retro gaming heaven.

The Picade is the perfect desktop or bookshelf distraction.

Environment sensors

With sensors and data and neat tidy cables, that's how our garden grows.

You can use your Pi to display your PC's CPU temperature and graph it on an LCD display. However, for this project we'll take that idea a step further, and measure a range of environmental variables. Thermistors and most other components that measure such factors, are analogue in nature. So they're not immediately suitable for the Pi, which only has digital (GPIO) inputs. However, thanks to HATs like the GrovePi, it's easy to get your Pi talking to analogue sensors. We used a high accuracy temperature and humidity sensor, a soil moisture meter, a light sensor, but much more are available and getting them working with GrovePi is super easy.

We won't cover setting up the Pi (any model will do), but you'll probably want to get it set up for SSH and Wi-Fi access, especially if you're planning on installing this in your greenhouse. One thing we will note is that you might have to coerce your Pi into using Python 3 by default (if it doesn't already). This is a matter of

```
$ sudo update-alternatives --install /usr/bin/python
python /usr/bin/python2.7 1
$ sudo update-alternatives --install /usr/bin/python
python /usr/bin/python3.7 2
```

Now we can set up the GrovePi board, and Dexter Industries provide a script that will do this in one line:

```
$ curl -kL dexterindustries.com/update_grovepi | bash
```

Vision on

You'll need to reboot once everything's done, at which point we can start setting up our display. There are a huge range of TFT, OLED or LCD screens available for the Pi so this depends on your hardware. We used Adafruit's 1.3-inch Pi Bonnet display and to set that up we had to install *Pip* and then install its Blinka library, which enables interfacing with their CircuitPython APIs:

```
$ sudo pip3 install --upgrade adafruit-python-shellwget
$ https://raw.githubusercontent.com/adafruit/
Raspberry-Pi-Installer-Scripts/master/raspi-blinka.
pysudo python3 raspi-blinka.py
```

Then reboot and you should be ready to go.

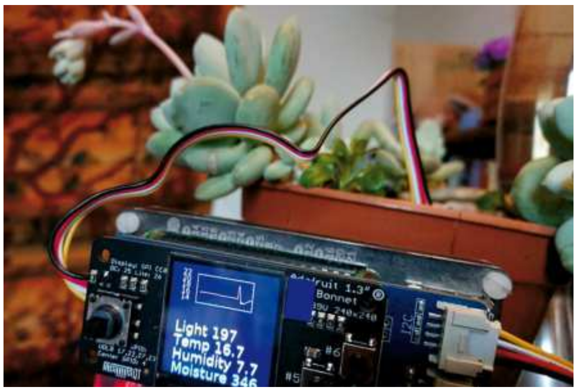
Reading from our combined temperature/humidity sensor required a bit of copying and pasting from **https://github.com/ControlEverythingCommunity/TH02**, since the bit-banging is a little complex. But our moisture and light sensors can be read with one liners:

```
import grovepi
moisture = grovepi.analogRead(1)
light = grovepi.analogRead(2)
```

Having got the data from the sensors, we need to get it on the display. The Adafruit Python libraries enable images to be sent directly to the display, so we can use the Pillow libraries to help us. Then there's some boilerplate to initialise the display (which we'll omit here, but you'll find it at **https://learn.adafruit.com/adafruit-1-3-color-tft-bonnet-for-raspberry-pi/python-stats-example**).

YOU NEED

- **Raspberry Pi (any model)**
- **GrovePi or GrovePi Zero**
- **Adafruit 1.3-inch Bonnet (or other display)**
- **GrovePi Sensors (temperature, moisture, light)**



Stack the display atop the GrovePi daughterboard, plug in some sensors and you'll be monitoring the environment in no time.

```
from digitalio import DigitalInOut, Direction
from PIL import Image, ImageDraw, ImageFont
import adafruit_rgb_display.st7789 as st7789
```

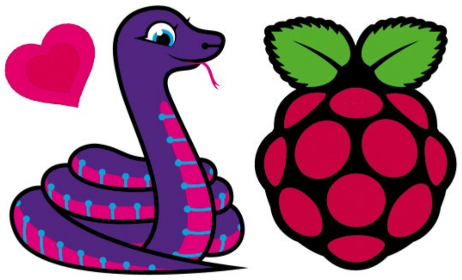
Then you can use regular PIL image and text commands to draw the info on screen

```
height = disp.height
width = disp.width
image = Image.new("RGB", (width, height))
rotation = 90
image.draw = ImageDraw.Draw(image)
```

For this project to be useful, we need the display to continuously update. So we'll make a loop, which clears the screen, updates readings, pauses and repeats:

```
fnt = ImageFont.truetype("/usr/share/fonts/truetype/
dejavu/DejaVuSans.ttf", 30)
while True:
  moisture = grovepi.analogRead(1)
  light = grovepi.analogRead(2)
  draw.rectangle((0, 0, width, height), outline=0, fill=0)
  draw.text((20,120), "Light {}".format(l), font=fnt,
fill="#FFFF00")
  draw.text((20,210), "Moisture {}".format(m), font=fnt,
fill="#00FFFF")
  disp.image(image)
  time.sleep(0.5)
```

You can go further with this project, such as logging and graphing temperatures with *Gnuplot*. Or you could connect a watering system via a relay and never have to worry about your peonies drying out again!



Adafruit's Blinka library enables Python to talk CircuitPython, a variety of MicroPython used in tiny gadgets and robots.

Bitcoin Node with RaspiBlitz

Make the Bitcoin network a bit bigger and better with a Pi and a large storage device.

You're not going to get rich mining Bitcoin on a Raspberry Pi. Even with dedicated ASIC mining hardware it's hard to compete with industrial operations in countries with subsidised electricity supplies. The story is the same for other cryptocurrencies too – see www.tomshardware.com/uk/how-to/mine-cryptocurrency-raspberry-p.

Yet you can still use *RaspiBlitz* to turn your Pi into a full Bitcoin node. This helps the Bitcoin network be more decentralised, and by extension more secure. If you run your own node, and store your bitcoins locally, then you're essentially self-sovereign – you haven't entrusted your keys to others and you aren't relying on other nodes to supply correct information. To quote the *RaspiBlitz* manual: "Not your node, not your rules".

Put some storage in place

Running a full node involves regularly syncing the Bitcoin blockchain, which at the present time is 350GB and growing, so you'll need some large external storage before you begin. You can get away with a mechanical hard drive, but life will be much easier if you can get hold of a 1TB or larger external SSD. Ideally, you want your node to be online all the time, but if you have a fast internet connection then it's easy enough to catch up after some downtime. Bitcoin nodes verify new transactions before sending them to miners, and record new blocks as they're mined, so they can't do anything useful until they're synced.

It's recommended to use a 3.5-inch LCD touch display to run *RaspiBlitz*, to display status information, but you can also run it headless. It's also advised to use a heatsink to avoid throttling or damage from overheating. Fetch the SD card image from **https://github.com/rootzoll/raspiblitz**, write it to an SD card (minimum 16GB), and use it to boot your Pi. Don't bother connecting a monitor (it won't work), but do connect an Ethernet cable. If you have the recommended XPT2046 display then this will be used automatically and will display the device's IP address. If not you can use a tool like *Nmap* to get this information and then connect from another device with, say

```
$ ssh admin@192.168.178.47
```

and using the default `raspiblitz` password.

The setup wizard will walk you through setting up passwords and storage, and ultimately syncing the blockchain, which will take hours. Once you log in again you'll be presented with the options screen where you can tweak settings and, optionally, set up as a node on the Lightning Network (see box, below). You can also see the status screen that would be displayed if you had a display attached. Note that for others to connect to your node, you'll need to open port 8333 on your router (and use a dynamic DNS service if you don't have a static IP at home). For Lightning you'll also need ports 9735, 10009 and 8080 to be open. You can also open a command prompt from the main menu, but don't be tempted to run `apt update` or such. There are instructions for updating *RaspiBlitz* on the project page and if you don't follow them there will be tears.

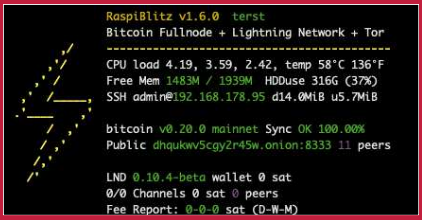
You'd be hard pushed to find a large enough SD card to store the blockchain, so get yourself an external SSD.

» THE LIGHTNING SEEDS

Besides running a Bitcoin node *RaspiBlitz* enables you to run a node on the "layer 2" Lightning network. The Lightning network permits Bitcoin transactions to take place off the main blockchain so they can be confirmed instantly rather than subject to the median six-minute confirmation time that is part of the Bitcoin fabric.

It's possible to generate some income by running a Lightning node, but not necessarily something you should jump into unawares. Since the network lacks the democratic (but lengthy) consensus mechanism of the main network, you could operate a rogue node and attempt to double-spend coins or otherwise cheat the system. As such there are ominous-sounding Watchtower services that detect improperly operating Lightning nodes and penalise them.

Lightning nodes are connected to blockchain wallets, and these must contain sufficient funds to cover any off-chain business. If malfeasance is detected, a so-called Justice Transaction is initiated and funds are docked. If your node accidentally goes offline while a channel is open, then you may initially lose funds, but these can be reclaimed (subject to both party's approval) once your node is back online.



```
RaspiBlitz v1.6.0  terst
Bitcoin Fullnode + Lightning Network + Tor
------------------------------------------
CPU load 4.19, 3.59, 2.42, temp 58°C 136°F
Free Mem 1483M / 1939M  HDDuse 316G (37%)
SSH admin@192.168.178.95 d14.0MiB u5.7MiB

bitcoin v0.20.0 mainnet Sync OK 100.00%
Public dhqukwv5cgy2r45w.onion:8333 11 peers

LND 0.10.4-beta wallet 0 sat
0/0 Channels 0 sat 0 peers
Fee Report: 0-0-0 sat (D-W-M)
```

You can use Onion (Tor) routing to keep the IP address of your Lightning Node a secret.

MotionEyeOS

Learn how to build your own motion-triggered home security camera.

YOU NEED

- **Raspberry Pi 3B+, 4 or Pi Zero W**
- **8GB (or larger) microSD card**
- **Raspberry Pi Cam, HQ Camera, Infrared Camera or webcam**
- **Monitor/power supply/HDMI cable**

Home security camera systems have exploded in popularity while decreasing in price over the past few years. Yet there are some drawbacks: first, vendors often charge a monthly fee to store your data; and second, you might not want video and photos from inside your home being shared with a third party.

MotionEyeOS is a distro that enables you to turn a Raspberry Pi with a camera into a home video-monitoring system, where the photos and videos can either stay on your device (and home network) or, if you choose, be uploaded automatically to a cloud-storage service such as Google Drive or Dropbox.

Here, we'll show you how to set up a Raspberry Pi security camera with MotionEyeOS. This software works with almost any Raspberry Pi (connected to the internet) and almost any webcam or Pi camera. There's no fancy coding to be done in this project; it just works.

Download the latest version of MotionEyeOS corresponding to the model of Pi you're using from **https://github.com/ccrisan/motioneyeos/releases** and write it to the SD card. When the process completes, physically remove and then reinsert your microSD card. We do this because the software automatically ejects the microSD card when the process completes, but we need to add one file before the next step.

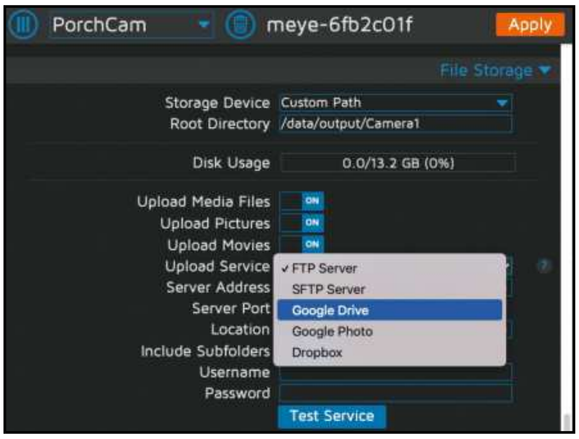
Create a new file named **wpa_supplicant.conf** with the this text, replacing "YOUR_NETWORK_NAME" and "YOUR_NETWORK_PASSWORD" with your details:

```
country=gb
update_config=1
ctrl_interface=/var/run/wpa_supplicant
network={
 scan_ssid=1
 ssid="YOUR_NETWORK_NAME"
 psk="YOUR_NETWORK_PASSWORD"
}
```

Save the file, eject the SD card and insert it into your Pi. Connect your camera, monitor and power supply to your Raspberry Pi and power up. If you have a monitor connected, your Pi's internal IP address will be displayed on the Pi screen. Alternatively, you can find this out from *Nmap* or your router's configuration page.

Enter your internal IP address into a browser window and *MotionEye* should start streaming.

This Raspberry Pi security camera can be used to record porch pirates, monitor children or pets, or to watch out for burglars.



Storage Device: Custom Path
Root Directory: /data/output/Camera1
Disk Usage: 0.0/13.2 GB (0%)
Upload Media Files: ON
Upload Pictures: ON
Upload Movies: ON
Upload Service
Server Address
Server Port
Location
Include Subfolders
Username
Password

Don't rely too much on SD cards for storage. Cloud saves will make files easier to access too.

Configuring MotionEye

Click the Profile icon within your browser menu to pull up the Login screen. Log in using the default credentials. The username is admin, and the password field should be blank. Select your Time Zone from the drop-down menu in Time Zone. Click Apply. *MotionEye* will reboot which will take a few minutes. This step is important as each photo and video is timestamped.

The Frame Change Threshold setting determines the proportion of pixels that change before recording starts. The intent is to set your percentage low enough to pick up the movement you're tracking, but high enough to avoid recording a passing cloud. In most cases, this is achieved through trial and error. Start with your default 4% Frame Change Threshold and then move up until you reach your optimal setting. Click the down arrow to the right of Still Images to reveal extra settings. Do the same for Movies. Set Capture Mode and Recording Mode to Motion Triggered and choose an appropriate period to preserve both. A week is reasonable if you're working with an 8GB card. Click Apply to save changes.

Set your Camera Name, Video Resolution and other options in the Video Device section. Click Apply to save your changes. On the live feed view you'll see new buttons for viewing saved images and video.

In the current setup this media won't leave your local network. But if you're okay having your media in the cloud then it's easy to set MotionEyeOS to upload to Google Drive or DropBox. Alternatively, you may want to enable remote access to your Pi security cam. You'll need to configure port forwarding on your router to do this. If you do this there are official iOS and Android apps so you can check your home when you're out. LXF

HQ CAMERA

Take better photos with your Raspberry Pi

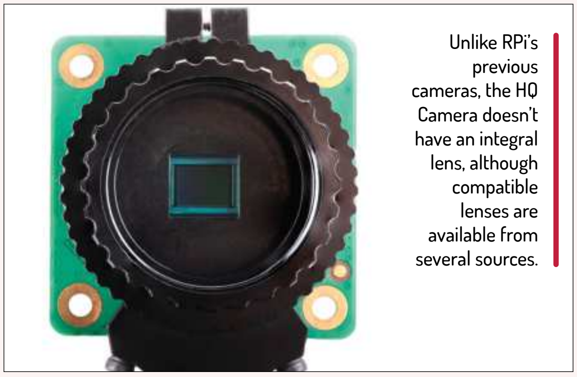
Mike Bedford explains how to get the most from the Raspberry Pi HQ Camera and learn the basic principles of photography at the same time.

OUR EXPERT

Mike Bedford has embraced all that digital photo techniques bring, but does like to play with manual controls, so the HQ Camera is surely a winner.

When Raspberry Pi launched its HQ Camera in 2020 (which we scored a commendable 8/10 in **LXF264**), it introduced something very different from their previous two cameras. While the other two camera modules are made up of a sensor and its interfacing electronics together with a lens, the HQ Camera doesn't have an integral lens. This means that you need to buy one or more lenses separately, and Raspberry Pi offers a couple of alternatives that we look at here. In that way, it's similar to digital SLR (DSLR) cameras, but different from the cameras in smartphones or compact standalone cameras.

Our aim here is to introduce the HQ Camera for those who haven't yet taken the plunge, and to follow this up by investigating how to use some of its less-familiar features, and at how to choose lenses as alternatives to those provided by Raspberry Pi. Armed with this information, we trust that many of you will decide to try your hand with the HQ Camera, and that it will form the basis of some fascinating new projects that you can formulate yourself.



Unlike RPi's previous cameras, the HQ Camera doesn't have an integral lens, although compatible lenses are available from several sources.

CREDIT: Raspberry Pi Foundation

The new sensor has a higher resolution – specifically 12MP, which compares to 5MP for the original camera module and 8MP for the second iteration. But although that megapixel figure is the one that most users concentrate on, the size of the sensor is also significant because the larger the sensor, the more light it gathers, and this also improves the image quality. So we should point out that while the original and v2 camera modules both had 1/4-inch sensors, the HQ Camera has a 1/2.3-inch sensor. These figures don't relate directly to the sensors' width, height or diagonal, but the area of the sensor in the HQ Camera is about three times greater than that of the earlier RPi cameras. For reference, consumer DSLRs have sensor that has a 12-times larger area again, and that increases by a factor of another 2.3 in professional full-frame DSLRs.

The HQ Camera supports lenses with C or CS mounts, which are standards commonly used in CCTV cameras. While these are widely available – although not always affordable – Raspberry Pi has picked a couple of third-party lenses that are the "official" offerings. One has a focal length of 6mm and the other 16mm – see the boxout (on the left) for more details.

Manual controls

All cameras except the most basic point-and-shoot models offer a degree of manual control over focusing, shutter speed and aperture. The HQ Camera's lenses, on the other hand, don't support automatic focusing or

» FOCAL LENGTH EXPLAINED

The focal length is proportional to the magnification provided by the lens and is inversely proportional to its field of view. However, direct comparisons can't be made between different lenses if they're used in conjunction with differently sized sensors. For that reason, a 35mm equivalent focal length is commonly quoted because it's familiar to photographers and provides an easy way of making comparisons.

The 35mm equivalent focal length is calculated by multiplying the focal length by something called the crop factor, which depends on the sensor size. For the HQ Camera this about 5.5. This means that the 35mm equivalent focal lengths of the 6mm and 16mm lenses are 36mm and 88mm, respectively. A lens with a 35mm equivalent focal length of 50mm is considered to be a "standard lens", which means that it has roughly the same field of view as the human visual system, so the 6mm lens is a slightly wide angle lens, and the 16mm lens is slightly telephoto.

Beyond these two official lenses, some Raspberry Pi dealers offer other lenses – for example the £36 8-50mm f/1.4 zoom lens from the Pi Hut or the £15 2.8-12mm f/1.4 zoom lens from Pimoroni – but knowing that the crop factor of the HQ Camera is 5.5, and bearing in mind that there are lots of C or CS mount lenses available, you should be able to choose other lenses that will meet your needs.

automatic aperture control. This might sound like a drawback, but being forced to use manual controls means the user learns about some of the basics of photography and gains the artistic benefits that aren't available by using a fully automatic alternative.

Next up, we need to consider exposure and this is controlled by three factors: shutter speed, aperture and ISO rating. Shutter speed is exactly what it sounds like, how long the shutter remains open. Typically this can vary from as little as a thousandth of a second to several seconds, and the amount of light admitted into the camera is simply proportional to the shutter speed.

Next is the aperture, which can be thought of as the effective diameter of the lens, and which is controlled by an iris shutter. This is specified as a so-called f-number and might vary from f/2.0 through f/2.8, f/4, f/5.6, f/8, f/11 and f/16, to f/22. To cut a long story short, the larger the number the smaller the aperture, and each f-number in the above sequence represents a halving of the amount of light admitted. (See below for more)

Finally there's the ISO rating. This is a measure of how much the camera's electronic circuitry amplifies the signal. It might range from 100 to several thousand, but to 800 for the HQ Camera.

Balancing act

Since there are three factors that affect the exposure, there will be several combinations that will provide a correct exposure. So, for example, if a correct exposure is achieved with 1/60 seconds, f/8 and ISO 100, it will also be correctly exposed with 1/120 seconds, f/5.6 and ISO 100, or with 1/240 seconds, f/5.6 and ISO 200. However, the results of these various combinations will not all be the same because these factors each impact the photo in various ways.

Shutter speed is probably the most obvious. The slower the shutter speed, the more likely it is that there will be some blur due to motion, either because you moved the camera while the shutter was open, or because something in the scene moved. The likelihood of blurring due to camera shake is almost eliminated if you use a tripod, although as a rule of thumb if you're shooting handheld you need a shutter speed faster than 1/60 second with a standard lens, and faster with a telephoto. Blurring due to something moving in the scene is quite different, and while this is usually a bad thing, there are exceptions. For example, if you're shooting a waterfall, you might choose to use a shutter speed of 1/30 or slower so the blurring of the water imparts a feeling of motion.

Aperture effects something called depth of field, with the depth of field increasing as the aperture decreases – in other words, as the f-stop figure increases. A small depth of field means that only objects close to the distance you're focused on will be in focus, but a large depth of field means that objects over a greater range of distances will be sharp. A large depth of field would often be suitable for landscape photography, so that everything from the foreground to the far distance is in focus. Conversely, a small depth of field might be preferred for portraits, to ensure that the subject is in focus while a distracting background is blurred.

Finally, we come to the ISO rating, and here the rule is that the greater this figure, the more noise – which manifests itself as a graininess to the image – is imparted to the picture. While there are pros and cons to slow/fast shutter speed, and large/small aperture, it's hard to think of a reason that you wouldn't choose the smallest ISO rating required to enable your ideal shutter speed/aperture combination.

While the aperture is controlled manually using a ring on the lens, the shutter speed and ISO rating are controlled by the *raspistill* software, albeit as specified

CREDIT: Raspberry Pi Foundation

Raspberry Pi's HQ Camera product offering comprises the camera module, a 6mm lens and a 16mm lens, which can be bought independently.

QUICK TIP

The HQ Camera has a blanking cap that you have to remove to attach a lens, but when there's no lens attached, replace the cap to prevent dust contaminating the sensor. The lenses are supplied with front and rear lens caps which have to be removed when the lens in use and these should also be replaced whenever possible to guard against damage or fingerprints on the glass.

» F-STOPS EXPLAINED

It's easier to remember something if you understand the principles, rather than just learning by rote. Because f-stops, which are the measure of aperture, aren't immediately understandable, here's what you need to know.

The way f-stops are written might seem strange. However, it's really very simple. An aperture of f/4 means that the diameter of the lens – as reduced by the iris shutter – is the focal length divided by 4. So, in the case of the HQ Camera's 16mm lens, at f/4 the iris shutter will have a diameter of 4mm. This, of course, explains the apparently contradictory fact that a large f-stop number implies a small aperture and vice versa. Because the amount of light a lens can admit depends on its area, and the area is proportional to the square of its diameter, we can now see why at f/8 a lens admits a quarter of much light as f/4, and why the sequence of f-stops relating to a halving of the amount of light is the odd-looking sequence f/1.4, f/2, f/2.8, f/4, f/5.6, f/8, f/11, f/16, f/22.

The fact that the f-stop is a ratio of the diameter to the focal length means that, to have the same aperture, a telephoto lens must have a greater diameter than a wide-angle lens. You can see this by comparing the two official HQ Camera lenses. Despite the 6mm lens having a slightly larger maximum f-stop of f/1.2 than the 16mm lens's f/1.4, the 6mm lens is smaller in diameter.

QUICK TIP

Many C and CS lenses have a megapixel rating and it might sound odd that the 6mm official lens has a 3MP rating whereas the camera is 12MP. However, although optimal performance is achieved if the lens has a rating at least equal to that of the sensor, it would be wrong to believe that 12MP sensor will only provide 3MP performance when used with a 3MP lens.

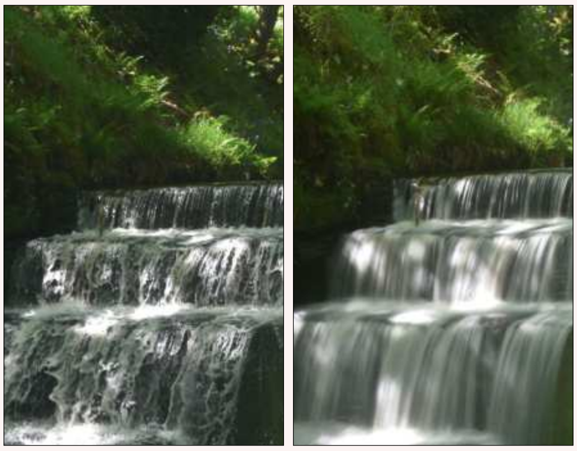
by the user in the case of ISO. In the default automatic mode, therefore, the HQ Camera will operate in what's generally referred to as aperture priority automatic mode. This means that you select the aperture to achieve a desired depth of field, and the camera software will adjust the shutter speed to ensure the correct exposure.

Many cameras also offer a shutter speed priority automatic mode, but this isn't possible with the HQ Camera. That is because, although you can specify the shutter speed in *raspistill*, the software isn't able to adjust the aperture accordingly because it can only be controlled manually on the lens. However, although you can't choose a specific shutter speed and expect the aperture to change to compensate, knowing that reducing the aperture will decrease the shutter speed – assuming the default auto exposure hasn't been overridden in *raspistill* – you can change the shutter speed by altering the aperture. This might involve an element of trial and error, but you'll soon learn how to achieve the effect you want, and don't forget that the shutter speed selected by *raspistill's* automatic exposure will be recorded in the EXIF data in the image file.

Getting started

Setting up your HQ Camera to work with a Raspberry Pi board involves plugging its ribbon cable into the camera socket of the devoce and enabling it using the *raspi-config* utility. This is all well described in the official user documentation so we'll limit our advice to what might not be as well documented. If you're going start out using the HQ Camera on your desk, perhaps as a learning exercise or to get to grips with the camera before building it into a casing as part of a project, it makes sense to think about mechanical stability.

If you're experimenting with the HQ Camera, it makes sense to attach it securely to your Raspberry Pi to prevent strain on the ribbon cable, and to mount it on a tripod.


Motion blur is controlled by the shutter speed. Left: sharp image/fast shutter speed, Right: motion blur/slow shutter speed.

The HQ Camera has a tripod socket, and it would be good to mount it on a tripod. A small cheap tripod might be perfectly adequate for use on your desk while testing, although if you want to use it in the great outdoors, as discussed later, you'll need a taller one. If the HQ Camera is mounted on a tripod, the associated RPi will end up dangling from the ribbon cable.

Since that cable is nowhere as robust as a USB or HDMI lead, and nor are the associated on-board sockets, this is best avoided by bolting the camera module and the Raspberry Pi board together. Hardware is available to achieve that and it doesn't cost much – see, for example, mounting plate from The PiHut that costs £3 – but if you order the camera from a supplier that doesn't have the mounting hardware it might not be difficult to make one yourself, depending on what you have available. You'll need a small sheet of plastic, onto which you can mount the HQ Camera on one side and the Raspberry Pi on the other. You'll also need eight short bolts, eight nuts, and spacers of some sort – you can see the one we made in the photo (on the left).

This will be enough to get the camera up and running on your desk, where you'll have access to a mains power

» CLOSE-UP PHOTOGRAPHY

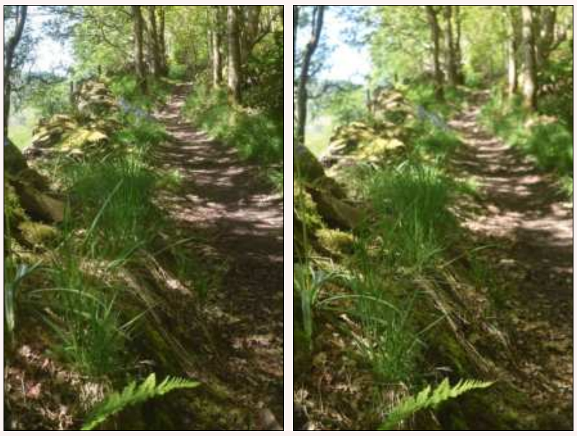
Commonly, lenses will focus to infinity, but they won't enable you to focus on close-up objects, which makes photographing small things difficult. We tested out the two official lenses. The 6mm was a surprise since it'll focus down to just a few millimetres from the front of the lens. That's not very useful, though, because you can't illuminate anything so close to the lens, although focused to a practical minimum distance of 10mm, the field of view is 15mm wide. The 16mm lens is more typical, having a minimum focus of about 55mm, and at that distance the horizontal field of view is 22mm.

Like most C/CS lenses, the official lenses have filter threads at the front which means that you can screw on a close-up filter, which is a misnomer because it's a simple lens that enables you to focus on closer objects. They come in various strengths, measured in diopters. 1, 2, 4 and sometimes 10 diopter filters are common and can be bought in sets of the three or four strengths, which can used singly or in combination. The 16mm lens needs filters with a 37mm thread. We tested out the 16mm lens with a combination of a 2 and 4 diopter close-up filter – because we had them at hand – and were able to focus down to 11mm, at which point the horizontal field of view was about 5mm.

Yet another option is to use C/CS extension tubes, which separate the lens further from the sensor, thereby decreasing the minimum focussing distance.

A set of close-up filters makes it possible to photograph small objects. This set has too large a thread for the official 16mm lens, although if you already have larger filters for use with a DSLR, you could use a converter.

CREDIT: Joe Ravi.


Depth of field is controlled by the aperture. Left: large depth of field/small aperture, Right: small depth of field/large aperture.

supply and a monitor, and room for a keyboard and mouse, but if you want to start photographing the world at large, you'll need to devise a more portable solution. For a start, you'll need a small LCD panel that will act as the viewfinder, and you'll need some sort of battery power supply. All of that will need building into an ergonomic case, of course, and you'll probably want to connect a push-button to act as the shutter release. Many such projects have been published based on the earlier camera modules, and you could adapt one of these, the main change being the mechanical arrangement required to house the larger HQ Camera.

The other thing you need to know is how to use *raspistill*, the command line utility that enables you to control any of the official Raspberry Pi cameras. There's no shortage of documentation, but to start we suggest using `raspistill -t 0` which won't actually take a photo but which will display a preview so you can try out the manual controls on the lenses. You can terminate the preview using Ctrl-C.

The official lenses

Of the two official lenses, the 6mm one has a CS mount, while the 16mm one has a C mount. Confusingly, they both have the same thread, so it's quite possible to attach either directly to the HQ Camera. However, because C and CS lenses focus to a sensor at a different distance behind the lens, if you're using a C mount lens you must use the adapter that's supplied with the camera. Failure to use the adapter with the 16mm lens, or using it with the 6mm lens will mean that you can't focus properly.

Before attaching either of the lenses to the HQ Camera, you'll probably find it useful to get some practice using the lenses' manual controls. We'll start with the 16mm lens which is the most intuitive. You'll notice there are two rings, each of which has a thumbscrew. Unlike an ordinary camera such as DSLRs, with which you'll regularly be changing the aperture and focus, these lenses are intended for CCTV use, so they enable the rings to be locked to prevent accidental adjustments. Unless you're using it for some unattended application, though, it makes sense to use the 16mm lens with both the thumbscrews unscrewed, but not so far that they'll fall out.

The focusing ring is the one with the FAR and NEAR labels, and as you rotate it, the lens will become longer or shorter as it focuses on near and far objects, respectively. The other ring controls the aperture and has f-stops from f/1.4 to f/16 marked on it. To see how it works, remove the front and rear lens caps, screw on the C/CS adapter, hold the lens by that adapter and then try rotating the aperture ring while looking through the lens from the rear. You'll be able to see the iris shutter open and close.

You might expect the 6mm lens to work in the same way but it doesn't. Specifically, the focus ring isn't a ring in the sense that it can be rotated independently. Instead, it's immovable from the thread that screws into the camera. Trying to rotate it will actually tighten or unscrew the lens from the camera. Instead, to adjust the focus, tighten the thumbscrew on the aperture ring – the one labelled with OPEN and CLOSE – and unscrew the thumbscrew on the focus ring.

Now, while holding the thread or the aperture ring, rotate the aperture ring. As with the 6mm lens, this focusing action will cause the lens to become longer or shorter. To adjust the aperture, tighten the thumbscrew on the focusing ring and unscrew the thumbscrew on the aperture ring. Now rotate the aperture ring but this time, it'll adjust the aperture and again you can observe that by looking through the lens. Unlike the 16mm lens, the f-stops aren't labelled, and the minimum aperture is with the iris shutter fully closed.

Once you know how to operate the lenses, you can attach them to the HQ Camera so you can see how adjusting the focus alters the distance at which objects will be sharp, and how the aperture adjusts the depth of field. Both these effects can easily be seen if you use the preview facility in *raspistill*. However, if you have both lenses, we suggest you start with the 16mm to better see depth of field in action, because telephoto lenses have a smaller depth of field than wide-angle lenses. You won't be able to preview blurring due to the consequential change in shutter speed this way though, because the preview window shows a moving image. This is quite different from the still images that *raspistill* can capture. To see motion blur in action, therefore, you'll have to capture some still images. LXF

QUICK TIP

We explained how moving from an aperture of f/8 to f/5.6 doubles the amount of light entering the camera, and moving from a shutter speed of 1/60 to 1/120 halves it. Such changes are often described in terms of EVs (exposure values) or, colloquially, stops. A change of 1 EV involves a doubling or a halving of the amount of light.

If you look through a lens while you adjust the aperture control, you'll clearly see the iris shutter open and close to increase or decrease the amount of light admitted to the camera.

BACKUPS

Back up and clone your Raspberry Pi

From disk cloning to backing up specific directories and partitions, keeping your Pi backed up saves time and heartache, says **Christian Cawley**.

OUR EXPERT

Christian Cawley is known as The Gadget Monkey because he eats a banana for breakfast, microwave chips for his lunch, and a Raspberry Pi for dinner.

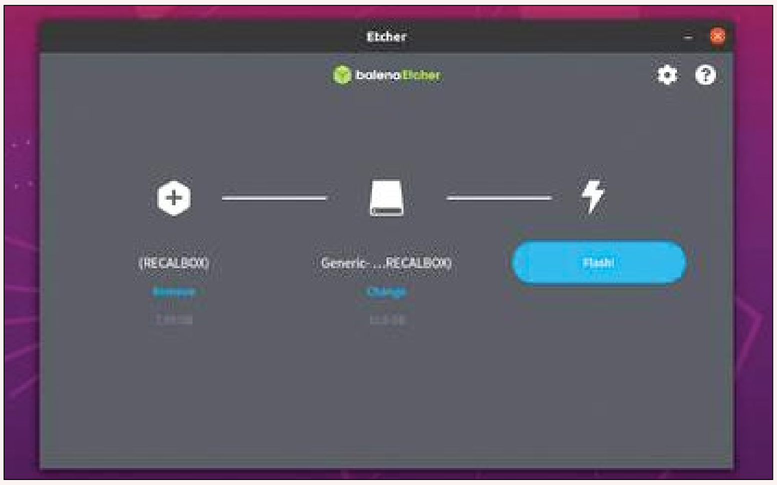
While a flexible, affordable and much-loved piece of computing hardware, the Raspberry Pi isn't without its problems. One particular bugbear is the device's reliance on flash storage – potentially unreliable SD and Micro SD cards that are prone to data corruption following an unscheduled shutdown or restart.

One way to avoid this is to ensure that the Pi is shut down carefully. But it also pays to have a backup of your data, just in case the worst happens. Raspberry Pi has several options for backup, from cloning the Micro SD card to remotely backing up over SSH.

Clone Wars

The ability to switch Raspberry Pi projects simply by swapping Micro SD cards means that if a card is found to be corrupt, you can flash an image of its contents to a replacement. Cloning the SD card is a smart option that enables you to keep a library of ready-to-use Pi images on your PC's hard drive, ready for deployment.

Just think of the time you can save. All your favourite software installed just once; Wi-Fi networks already configured; SSH enabled, ready to use. Where once you would reinstall and reinstall, changing numerous settings to get things just right, cloning the SD card means that you can customise Raspberry Pi OS, get things working how you like them, and then make a copy to use over and over again.



Rather than fiddle around with the odd command in the terminal, you can clone Raspberry Pi SD cards with Etcher.

This approach isn't without its shortcomings, however. For a start, there's the volume of storage space required for multiple images. You should also consider the amount of time required for the initial image creation. Simply having a clone of a fresh Raspberry Pi OS install is pointless, because that's essentially what you can download directly from **raspberrypi.org**. Instead, the image should be the underlying operating system together with the key software that you want to use, along with the data you have saved to the device.

Some challenges might be faced with cloning your Raspberry Pi. For example, a 16GB disk image can't be cloned and then written to an 8GB SD card. However, you may also run into problems cloning a 16GB image to another 16GB SD card, because of differences in quality-control standards.

There may also be some difficulty booting a cloned disk image on a different system. For example, a backup that's been created on a Raspberry Pi 4 will almost certainly not work on the Raspberry Pi Zero. However, if you keep in mind basic issues of compatibility and file size, then cloning your Raspberry Pi for back-up purposes should be plain sailing. Cloning the SD card can be done on a desktop PC or on a Raspberry Pi.

The clone heads

To create a clone of your Pi's SD card on a desktop computer, all you need is the cross-platform flashing tool, *balenaEtcher* (**http://bit.ly/LXF287-etcher**). Download and install this on your desktop system and then insert the Pi's Micro SD card into the PC's reader.

You should also have a second Micro SD card connected ready to write to. Failing this, a USB flash stick. Writing to a system drive isn't possible without first creating a dedicated partition.

Click Clone drive and select a Source. Click Select, then Select target to choose the target drive. This should be the second Micro SD card. Once again, click Select to confirm, then Flash to commence. Wait while the software writes and verifies the data. You'll notice that it's possible to flash the data to multiple devices if you need to. You can also clone the Pi's SD card directly to your HDD. This is useful if you only have one SD card but plan to run multiple Raspberry Pi projects.

With the SD card inserted in your desktop computer, open the terminal, and enter the following:

```
sudo fdisk –l
```

Identify the SD card and make a note of its name. This can be tricky, but it should be obvious from the size of the SD card. It will typically be sda or sdb followed by a number.

Next, use the `dd` command to create a clone. You can name the target file however you like, but it should have the .img extension:

```
sudo dd if=/dev/sdb of=~/raspberry_pi_clone.img
```

Wait as the process completes. When you're ready to write the disk image to a new Micro SD card, insert a fresh card and enter the following:

```
sudo dd if=~/raspberry_pi_clone.img of=/dev/sdb
```

You might also use the *Raspberry Pi Imager* tool, the mouse-driven Raspberry Pi disk imaging software available from **raspberrypi.org** (you can find it here: **http://bit.ly/LXF287-piimager**) for Ubuntu, macOS, and Windows. After installing the program, click Choose OS, then scroll to Use Custom and browse for the cloned . img file before writing it to a fresh SD card.

Clone your Home

For a simpler approach that's less time-intensive, archive the Pi's Home folder. This method enables you to browse and then restore directories and files to a reimaged Raspberry Pi as and when required. Open a terminal and enter:

```
cd /home/
sudo tar czf pi_home.tar.gz pi
```

With the archive created, copy **pi_home.tar.gz** to another device for safekeeping. The next time you need to reimage a Micro SD card, once complete simply unzip the **/home/** directory and copy the files you need to the Raspberry Pi.

Network backup

If you prefer using the Raspberry Pi remotely and have SSH enabled, then it's possible to back up the device across your network.

This isn't ideal for backing up the entire SD card, however. Some files that are in use while the Raspberry Pi is running won't be correctly cloned, leaving you with a disk image that may not boot correctly. The best option with this network approach is to back up only data that you deem is vital, such as the home directory.

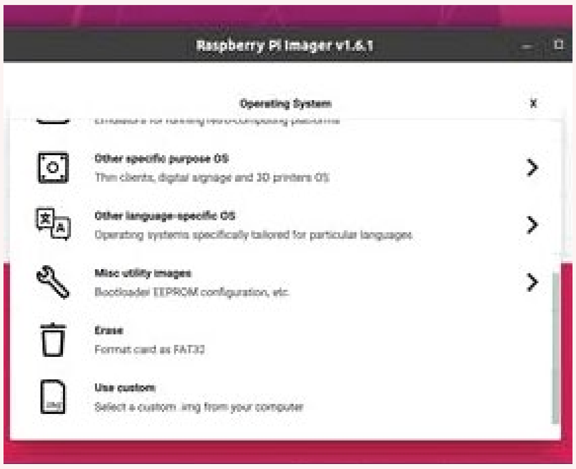
Begin by starting an SSH session and connecting to the Raspberry Pi. Then enter the command:

```
ssh pi@[IP_ADDRESS] "sudo dd if=/dev/mmcblk0
bs=4M | gzip –" | dd of=~/Desktop/[BACKUP_NAME].gz
```

Be sure to replace `[IP_ADDRESS]` and `[BACKUP_NAME]` as appropriate.

Wait while the backup completes. Of course, the backup cannot be restored across the network. The Pi would lose connectivity part-way through. To restore the disk image, you would need to insert the blank SD card into your computer's card reader and then use the `dd of` command:

```
gzip -dc ~/Desktop/[BACKUP_ NAME].gz | sudo dd of=/
dev/rdisk1 bs=4M conv=noerror,sync
```


Raspberry Pi Imager v1.6.1
Operating System
Other specific purpose OS
Thin clients, digital signage and 3D printers OS
Other language-specific OS
Operating systems specifically tailored for particular languages
Misc utility images
Bootloader EEPROM configuration, etc
Erase
Format card as FAT32
Use custom
Select a custom .img from your computer

The Raspberry Pi Imager tool has various hidden features. Among them is the ability to write any IMG file to a microSD card.

Remote backups using SSH will save time and wear and tear on the Pi's SD card. It's also a useful option if you run multiple Raspberry Pis.

When in clone...

Backing up your Raspberry Pi files through cloning can take up quite a bit of disk space, especially if you're making regular disk images. By default, the disk images are the exact same size as the disks, rather than the data used. So, an 8GB cloned SD card would take up 8GB of hard disk space. Fortunately, the files can be compressed. All you need to do is pipe the output from *dd* to *gzip* to create a compressed .gzip file.

```
sudo dd bs=4M if=/dev/sdb | gzip > raspberry_pi_clone.
img.gz
```

To restore:

```
gunzip --stdout raspberry_pi_clone.img.gz | sudo dd
bs=4M of=/dev/sdb
```

The cloned file is smaller using this technique and will take up less space on your HDD. This, of course, means you can make even more Pi backups... LXF

QUICK TIP

Use the fastest rated microSD cards for your Pi, with A1 performance. This will ensure fast, reliable system cloning, which will help when it comes to writing a fresh disk image. Also, avoid unknown brands and stick to well-known names: SanDisk, Kingston and Samsung, say.

» JUST BACK UP THE CONFIG FILE

If your Raspberry Pi setup is unusual, perhaps due to the hardware you've connected to the device, but you don't feel the need to regularly back up, there is an alternative. Instead of backing up the full SD card, or even archiving the **/home** directory, why not simply make a copy of **config.txt**? Existing in place of a system BIOS, **config.txt** stores various system parameters required for the correct running of the Raspberry Pi. Every Raspberry Pi operating system features a **config.txt** file, which is read by the system's GPU as the device boots.

While the Raspberry Pi is running, **config.txt** can only be edited as root. You'll find it in the **/boot** directory as **/boot/config.txt**. With the microSD card inserted in other systems, **config.txt** can be edited using a standard text editor. A whole bunch of options can be stipulated in **config.txt**. It enables the use of options for every possibility, from RAM use to camera settings, display options, overclocking, GPIO, and much more. For example, if you used a Raspberry Pi Zero with a small display, you would need to make some changes to the default options in **config.txt**. By backing up the file after the changes are saved, you can save time after a reinstallation by simply replacing the default **config.txt**.

SENSEHAT

Turn a Pi HAT into a Magic 8-ball project

Les Pounder blasts off with another tutorial showing how a versatile HAT board can answer all your questions, and even be run in space!

OUR EXPERT

Les Pounder is associate editor at Tom's Hardware and a freelance creative technologist.

The Raspberry Pi Foundation released its Sense HAT add-on back in 2015. Yet this board still packs a full scientific platform and an 8x8 RGB LED matrix for a little fun. In this tutorial we'll introduce the board, show text on the LED matrix and learn how to read accelerometer data for a classic game of chance.

Installing Sense HAT is straightforward. With the Raspberry Pi powered off, connect the Sense HAT to all of the GPIO pins, ensuring that the Sense HAT perfectly overlaps the Raspberry Pi. Use the included stand-offs to securely mount the SenseHAT. Now attach your keyboard, mouse, HDMI, micro SD and finally power to boot the Raspberry Pi to the desktop. Because we're using the latest version of Raspberry Pi OS, there's no need to install any software or Python packages.

Project 1: Hello World

The first project with any new piece of tech or software is "Hello World". It proves that our kit is working, and that everything is ready for us to move further. The first project is a simple scrolling message that will prove our software and hardware works. In your preferred Python editor, *Thonny, IDLE, Mu* or a text editor, create a new file and call it **text_scroll.py**. Remember to save often!

Our first two lines of Python are an import that imports the `sense_hat` class from the **SenseHat** module. We then create an object that enables us to easily use the class.

```
from sense_hat import SenseHat
sense = SenseHat()
```

The next two lines are tuples – data storage structures which can be created and destroyed but never updated. These tuples store the RGB colour values for a particular colour, in this case red and white.

```
red = (255, 0, 0)
white = (255, 255, 255)
```

The message to scroll across the screen is stored as a string inside a variable called `message`.

```
message = "Hello World"
```

We now enter a loop, and to start the process we start a `try` and `except` handler. Any code inside the try section of the loop is what our project will attempt to run. In this case it'll launch an infinite loop.

```
try:
  while True:
```

Designed for all 40-pin models of Raspberry Pi, Sense HAT is an old board that's still relevant in the classroom and space station!

Inside the loop we have a single line of code that will scroll our message, with red text and a white background. Each step in the scroll lasts 0.1 seconds.

```
    sense.show_message(message, text_colour=red,
back_colour=white, scroll_speed=0.1)
```

We now reach the `except` part of the code. This is an exception handler that is activated if we press Ctrl+C to end the code. When that happens we instruct the Sense HAT board to clear:

```
except KeyboardInterrupt:
  sense.clear()
```

Save the code and run via your Python editor. *Thonny* and *Mu* have a run/play button. *IDLE* uses F5 or via the Run menu. 'Hello world' should now scroll across the LED matrix. When finished, press Ctrl+C – this clears the matrix.

Project 2: Magic 8-Ball

The Magic 8-Ball is a classic child's toy. Ask a question out loud ("Will my online delivery be left in the hedge again?", "Am I the only one who can empty the dishwasher in this family?" or "Is there anything worth watching on TV tonight?"), then shake the 8-ball. In a few seconds a message floats to a viewing portal, ready to read. With the Sense HAT we can make a modern-day version, which uses raw data from the accelerometer to determine that the 8-ball has been 'shaken'.

Create a new Python project in your favourite editor, call the project **8ball.py** and remember to save often.

YOU NEED

- **Any 40-pin Pi model**
- **SenseHAT**
- **Raspberry Pi OS**
- **Code: https://github.com/lesp/LXF273-SenseHAT/archive/main.zip**

We start by importing two modules. The first is our Sense HAT module, and the second is called **random** and from it we import a function that will be used to pseudo randomly choose a response:

```
from sense_hat import SenseHat
from random import choice
```

Next we create the sense object to enable the ease of handling with the Sense HAT module:

```
sense = SenseHat()
```

Again we create two tuples to store the RGB colour data for the text and background:

```
red = (255,0,0)
white = (255,255,255)
```

The next line of code is a list, sometimes known as an array in other programming languages. A list stores data using an index that starts at zero. The first item in a list is at position zero, and subsequent items have their own numerical position. Lists can be created, destroyed and updated. The list we are using stores five text strings that are the answers to our eager player's questions.

```
answers = ["Not likely","Chances are
slim","Maybe","Quite possibly","Certainly"]
```

Another try and infinite loop process handles running the code:

```
try:
  while True:
```

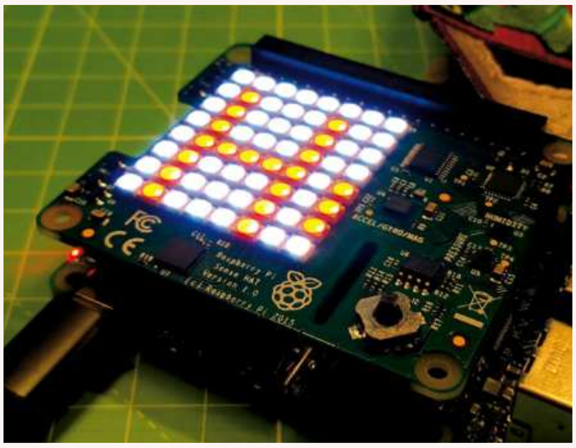
We obtain the raw accelerometer values from the Sense HAT and store them in an object called "acceleration". From that object we create three variables that store the data for each axis:

```
    acceleration = sense.get_accelerometer_raw()
    x = acceleration['x']
    y = acceleration['y']
    z = acceleration['z']
```

The values stored inside the variables x,y,z can be positive or negative values. It all depends on the orientation of the Sense HAT. Our code won't need to understand negative numbers, just that the numbers can go over a threshold that will trigger the answers to appear.

Using `abs()` we can update the variable to store an absolute value that ignores the value if it's either positive or negative:

```
    x = abs(x)
    y = abs(y)
    z = abs(z)
```


The 8x8 LED matrix is bright and easy to control. Shown here is a message across the matrix.

Sense HAT is packed with sensors and is a capable platform for scientific experiments and it can be used with many different programming languages.

A simple if conditional test checks the value stored inside the three variables x,y,z and if any of the values are greater than 1 then the indented line of code is run:

```
    if x > 1 or y > 1 or z > 1:
```

The indented line of code is a scrolling message, the answer to our question. But this time, instead of using a static message in a variable we set the message to be a random choice from the answers list. Then we set the text colour to red, background to white and speed up the scrolling text speed a little.

```
        sense.show_message(choice(answers), text_
colour=red, back_colour=white, scroll_speed=0.05)
```

If the Sense HAT hasn't been shaken, or the player has put it down on a table, then the else condition is activated and the LED matrix is cleared.

```
    else:
        sense.clear()
```

The final two lines of code handle the user pressing Ctrl+C to end the game. The LED matrix is cleared ready for use again.

```
except KeyboardInterrupt:
  sense.clear()
```

Save aLnd run the code. Pick up the Sense HAT and give it a little shake. The answer to your questions is but a shake away. LXF

QUICK TIP

If you intend to use the official Raspberry Pi camera with Sense HAT, you'll need to thread the flat flex cable through the slot next to the joystick before attaching it to the Raspberry Pi.

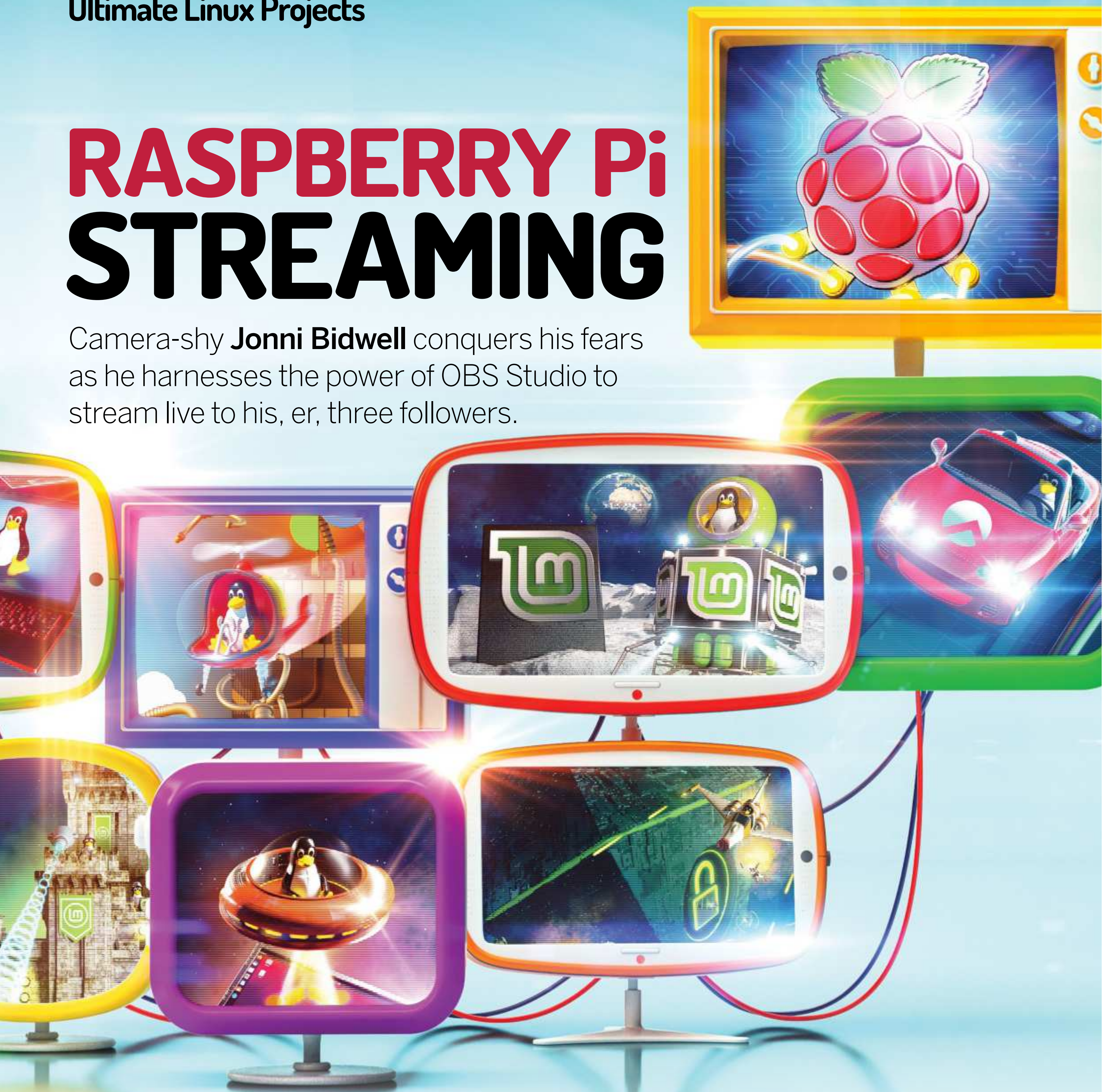
» PiS IN SPACE!

The Sense HAT is packed with sensors for temperature, humidity, acceleration, orientation and air pressure. And we have a great RGB LED matrix and a joystick that can be used in projects. To control the Sense HAT we used Python, but we can also use Scratch and Node-RED. Sense HAT was developed alongside a project called AstroPi which saw two Raspberry Pi B+ boards being sent to the International Space Station. These two Pis had their own Sense HAT boards, official Pi cameras and a custom aluminium case designed to protect and cool the Pis in space. Projects written by children across the world were run on those two Raspberry Pis and that project still exists today and you can take part via **https://astro-pi.org**.

The power of Sense HAT lies in its simplicity and flexibility. In our time working with the Raspberry Pi Foundation for its Picademy training courses, we used the Sense HAT to control robots via the joystick and accelerometer, and one bright spark created a *Minecraft* world that would change as the temperature changed. A cold room produced snow across the realm. Sense HAT retails for around £30 and while this is quite expensive for a HAT, you will get plenty of use out of this great board.

RASPBERRY Pi STREAMING

Camera-shy **Jonni Bidwell** conquers his fears as he harnesses the power of OBS Studio to stream live to his, er, three followers.



Being an online broadcaster has been a legitimate profession for some years. But the pandemic fuelled a surge in aspiring home streamers. And despite the ongoing relaxation of measures, the home streaming wave shows no sign of breaking. Whether it's repairing washing machines on YouTube, playing *Doom* on Twitch or just ranting on Facebook Live, there's abundant diversity in the things people are broadcasting, and tuning into. Setting up a professional recording studio at home is an expensive undertaking, but you really don't need any fancy hardware at all to get started. Many a YouTuber started out using just their laptops.

Software-wise, the open source *OBS Studio* is by far the most popular choice. It provides an intuitive interface that makes it easy to manipulate scenes and add effects during recording. And it can stream to all the most popular platforms (as well as a whole bunch we'd never heard of, and some we could have done without hearing about). We'll show you how to get the most out of that and start broadcasting to the world. You can even run it on a Raspberry Pi, and it works great in tandem with the Pi Foundation's High Quality Camera.

But why stop there? Thanks to *Owncast*, it's remarkably easy to self-host your streaming. All you need is enough bandwidth to support your audience. With even a modestly fast upload speed you'll be able to host a few dozen viewers. And if you don't, then take *Owncast* off-world and run it on a virtual private server. As long as your stream can get from *OBS* to *Owncast* smoothly, then the mass transport all takes place in the cloud.

Streaming basics

The hardware and infrastructure you'll need to get your stream live.

Adam Curtis's seminal 2002 documentary *The Century of the Self* highlighted how rampant consumerism in the west, driven by careful marketing, has engendered a society focused on the individual. Curtis's production was prescient, given that the decade that followed saw the rise of social media, selfies and targeted advertising. And in the past decade we've seen this appeal to the ego used to direct not just purchases, but politics too. We offer no salve for this societal narcissism, and as the old saying goes: if you can't beat em, write a feature all about live video streaming.

Well, maybe our list of old sayings suffered a misprint. And just because we'll show you how to put a camera and a microphone in front of yourself and broadcast signals to the internet at large, it doesn't mean we want you to become a product-placing, falsehood-selling influencer type. There are plenty already, and we would rather you didn't contribute to the further erosion of society. Indeed, you could use this power to foster community values instead, sharing arts and crafts and stories of the old country.

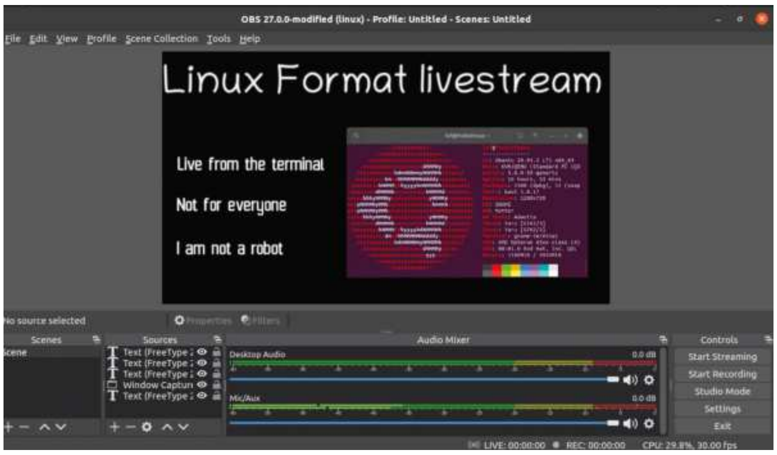
Hardware considerations

Whatever you're planning to broadcast, you'll probably need a camera and a microphone. Your laptop's ensemble will be just fine to get started. If the hardware works elsewhere in Linux then it will work on *OBS* (*Open Broadcasting Software*) *Studio*. This gem of open source software is hugely popular, both on Linux and proprietary OSes.

As you get more into this you may want to upgrade your hardware. Microphone pops are annoying on the ears, and poor lighting makes your skin look bad. For high-quality game streaming you might want a video capture card, if you're streaming live audio you might want to invest in a mixing desk or better soundcard. If you want to appear super-imposed on a funky backdrop you'll need a green screen. Basically, there's no shortage of directions to invest in this field, but let's not get ahead of ourselves. We still haven't covered how this streaming thing works.

OBS sends video streams (whether they're from your desktop, a camera, or the internet) to a streaming platform. This platform (whether it's YouTube, Twitch, Facebook, or a server in your basement) sends the livestream to viewers, often recompressing it on the fly to accommodate different viewers' bandwidth situations. So *OBS* only needs to have sufficient upload bandwith to get the data to the streaming platform, which ought to have more.

Streaming platforms often incorporate some kind of interactivity, such as chat windows, so your viewers can express their approval or otherwise through colourful language and hundreds of emoji. We'll see later how to set up our own self-hosted streaming server too, sticking it to the tech giants. But for now let's turn the page and get started with *OBS*.



Even if you don't have a camera, you can livestream your terminal tutorials using just a microphone or a text-to-speech engine.

STREAMING VS RECORDING

We're going to concentrate on 'live' livestreaming in this feature, but you might prefer to record your show, do some edits and retakes, then upload it. *OBS Studio* is just at home recording as it is streaming – you just need to make sure you have enough disk space. At the default bitrate (2,500kbits/sec) an hour of video will occupy just over a gigabyte (plus more for audio). If you're recording to another machine (say, over your LAN via *Samba*) then make sure there's enough bandwidth for this, especially if you're streaming and recording simultaneously. This is mostly a concern for patchy Wi-Fi connections.

The *OBS* setup wizard can optimise for recording from the get-go, which means it'll drop frames from the stream before it does so for the recording. Once you've got your video recorded, the non-linear video editor *KDEnlive* is great for editing it. You'll need plenty more space for this though as working with large videos tends to produce lots of temp files, as well as the finished product. You might also want to treat audio separately, using *Audacity*, for example.

Never let an opaque data collection policy get in the way of a quality audio editor. Audacity is widely available.

Introducing OBS Studio

The gold standard for home streaming software is open source, powerful and really easy to learn. It's a triple whammy threat!

Lets start by trying desktop OBS before moving on to running it a Raspberry Pi. Largely as *OBS* is in the Ubuntu repositories, and probably in those of whatever distro you're using too. But it's always worth getting the latest version (27.0.1 at the time of writing). *OBS* maintains its own Ubuntu repository (the wiki covers this, as well as distro-specific instructions, over at https://obsproject.com/wiki/install-instructions#linux), but since there's a Snap package you'd probably do well to use that. The Snap package includes support for hardware-assisted video encoding, virtual cameras and all sorts of other treats you probably wouldn't get if you compiled it yourself – something Pi users will have to do as we'll see over the page... You don't have to be on Ubuntu to use Snaps (but if you are, you'll find the *OBS* Snap in the *Software Centre*), so once you've got the Snap daemon installed it's just a matter of:

```
$ sudo snap install obs-studio
```

Well, that's not strictly true. In order to talk to the appropriate interfaces and hardware, the Snap needs to be connected, like so (this happens automatically if you install from Ubuntu's *Software Centre*):

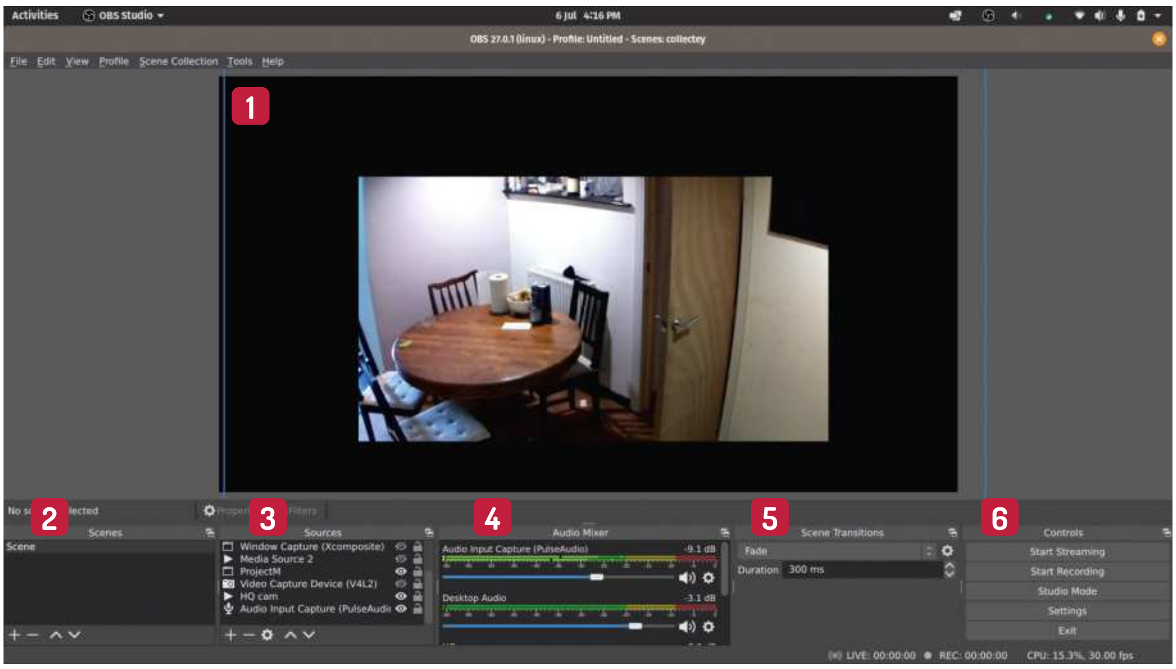
```
$ sudo snap connect obs-studio:{alsa,audio-record,avahi-control,camera,jack1,kernel-module-observe}
```

If you don't like Snaps there's a Flatpak available too. That's what we used, with great results, on our Pop!_OS laptop. Pop supports installation from Flathub straight from its own software centre (*Pop! Shop*), and so might your distro. But if it doesn't, then install the Flatpak daemon and then install and run *OBS* with:

```
$ flatpak install flathub com.obsproject.Studio
$ flatpak run com.obsproject.Studio
```

When *OBS* starts you'll be greeted by a setup wizard (and if you use the Snap, some weird questions about resetting the UI and setting a password for the Websocket server, which you shouldn't worry about). Choose "Optimize for streaming", and go with the default Base resolution of 1280x720, unless you have

GET TO KNOW THE THE OBS INTERFACE


Activities
OBS Studio
6 Jul 4:16 PM
OBS 27.0.1 (linux) - Profile: Untitled - Scenes: collectey
File Edit View Profile Scene Collection Tools Help
1
2
3
4
5
6
Scenes
Scene
Sources
Window Capture (Xcomposite)
Media Source 2
ProjectM
Video Capture Device (V4L2)
HQ cam
Audio Input Capture (PulseAudio)
Audio Mixer
Audio Input Capture (PulseAudio) -9.1 dB
Desktop Audio -3.1 dB
Scene Transitions
Fade
Duration 300 ms
Controls
Start Streaming
Start Recording
Studio Mode
Settings
Exit
LIVE: 00:00:00 REC: 00:00:00 CPU: 15.3%, 30.00 fps

1 Preview window
This displays the current Scene. Set the scene by dragging and resizing sources with their frames.

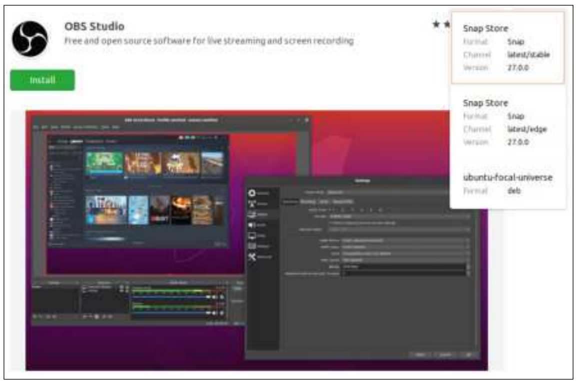
2 Scenes
A Scene is a template for a section of your broadcast. Different Scenes might use different sources, layouts or captions.

3 Sources
These might be cameras, microphones, desktop windows or something else. Each scene has its own sources.

4 Audio controls
The VU meters will react to audio input. They shouldn't go into the red. If they do then use the sliders to bring levels down.

5 Scene transitions
Here you can figure a stylish effect, for example a fade, to make switching between scenes look pleasing to the eye.

6 Stream controls
Start or stop your stream (or recording) with these buttons, or jump into Studio Mode if you feel like some live VFX.



The Snap version of Open Broadcasting Software Studio comes with all kinds of additional bells and whistles.

strong feelings otherwise. Unless you're confident your machine can handle it (and your viewers will notice the difference) dial down the FPS to 30, at least for now.

Next you'll need to choose a streaming platform. This will require you to have an account, and bind you to the terms and conditions of that platform. A key is used to make sure it's you and not a stranger connecting their camera to your streaming platform. Click the Get Stream Key button, log in to your platform, copy the private stream key from the web page and paste it into *OBS*. Leave the Estimate bitrate checked. *OBS* will see what it thinks of your bandwidth situation and CPU power and set the encoder settings accordingly. Your platform will then take this stream and serve it, as well as offering lower bitrates, to the masses. Go with the settings *OBS* suggests (you can always change them later) and you'll find yourself in *OBS's* main interface. If you have a Linux-friendly camera and microphone attached, then these will be automatically set and you will see what the camera sees in the preview window. See the next page for a guide to what's going on here.

Getting started with your hardware

It's good to keep things simple to start with, at least until you're sure everything's working. So start with just a camera and microphone. These will be listed as Video Capture Device (V4L2) and Audio Input Capture (Pulseaudio) in the Sources section. We set the base canvas resolution to 720p earlier, which might be different to your camera's native resolution. You can change this by right-clicking the V4L2 source and choosing Properties. You might not want to fill the canvas with camera input, in which case choose a lower resolution here. You can put some decorations in the blank bits of canvas later.

The currently selected source appears surrounded by a red frame (unselected ones have a blue frame) and it's possible to drag and resize this (even during a live broadcast) using the handles. Bear in mind that rescaling video is an expensive operation, and if you're planning on using the same source at different sizes (to go fullscreen with a game, say) it's more efficient to make use of Scenes. That way you can set the source up at different resolutions for different Scenes.

If you're only interested in one video source, you can right-click its preview frame and choose Resize Output (source size). This will set the base and output resolutions to that of the camera, so that it appears full screen. Video is sent to the streaming platform at the output resolution. If your computer handled it during set up it'll be the same as the base resolution (the size of the canvas), otherwise it will be less.

You might be planning on doing a purely audio stream (for example, to *Mixcloud*) in which case there's one less class of media to worry about. Either way, make sure that your recording levels are good and be sure to familiarise yourself with the ways of muting the microphone. Given the pandemic and its plethora of virtual meetings, you might already know this particular drill.

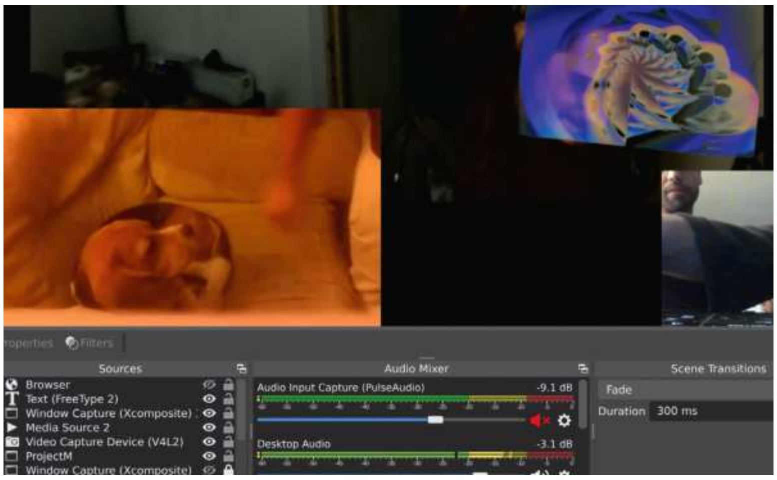
OBS saves your sources and scenes automatically, so it's easy to jump back in after a streaming break. Still, it's worth coming up with a better name than "Scene" for your first scene, which you can do by right-clicking and selecting Rename. You might want to rename some of your sources, too.

In addition to getting hold of the stream key, your platform will offer options for your broadcast from its dashboard. Twitch, for example, enables you to mark your content as mature, set latency settings or store broadcasts. YouTube's Studio section makes it possible for you to edit your video online, or add a soundtrack.

AUTOMATIC CONFIGURATION

"OBS will see what it thinks of your bandwidth situation and CPU power, and set the encoder settings accordingly."

We won't tell you how to make a good broadcast. To be honest, we don't know, but if it's anything like making a good feature, then plan what you're going to cover and probably don't wing it all at the last minute. Clicking in the Scene Collection box is easy during live broadcasts, so if your Scenes are all set up nicely before you start you won't have to fumble around switching between sources and clumsily repositioning them on the fly. Once you've got the hang of Studio View then it's easy to create smooth and professional-looking inter-scene transitions too.



In no time you'll be adding beagles and interdimensional portals to your livestream.

Pi-assisted streaming

Use one of your many spare Raspberry Pis as a remote camera or go crazy and run everything from a Raspberry Pi 4.

While we were planning this feature, a little voice at the back of our head muttered, "Can we get a Raspberry Pi angle in?". We like to keep that little chap happy so we've come up with a couple of ways to quieten his screams for Raspberry Pi streams.

We're always looking for new Pi camera projects, so first we reached for our trusty Pi Zero and attached a camera – you can read more on the HQ camera back on page 110. Remember that if you're setting up a headless Pi from scratch, the SSH service is disabled by default. If you're running wirelessly, you'll need to tell it about that too. This you can do by putting an empty file named **ssh** and another file full of wireless network info (**wpa_supplicant.conf**) on the boot partition of your Pi's SD Card. Then you'll need to run `raspi-config` on it to enable the camera in the Interfaces section. Again, a quick search engine query will help you if you get stuck.

The HQ Camera has a standard tripod mount. Handy because clumsy cabling will easily topple whatever else you balance it with.

The *raspistill* and *raspivid* programs which ship with the Pi are commonly used to test that camera-related things are working. But it turns out you can use *raspivid* to make your Pi into a rough 'n' ready network source in *OBS*. Just SSH into it and run:

```
$ raspivid -t 0 -l -n -w 640 -h 360 -o tcp://<ip-
address>:3333
```

replacing `<ip-address>` with that of your Pi. There are a thousand and one options for the program, and the ones we use tell it (in order) to keep streaming forever, listen on a TCP socket, not display a preview (although that would be tricky without a monitor) and serve a 640x360 stream on port 3333 the interface specified.

You can specify all interfaces with **tcp://0.0.0.0:3333**, and it's also possible to stream over UDP, too. In order to use this in *OBS*, add a Media Source, uncheck the local file box and use the same **tcp://<ip-address>:3333** URL that you used in the command above in the input box. Once you've added the source it'll initially be blank for a few seconds, and then you should see the video stream. If you're like us and using a Pi Zero (or even an OG Pi from nine years ago), this delay will be significant because the Pi's SoC has to do some work encoding the stream to h.264. This effect is compounded over wireless, so if you were hoping to perform some kind of multi-camera choreographed line-dancing routine, this probably isn't the best solution for you. However, that said, it can make for some pretty amusing and

» TAKE THE DIY Pi WEBCAM APPROACH

Obviously, neither Pi camera is a microphone too, so on its own it's not much use if you want audio in your streams. Dell recently moved the webcam on its popular XPS13 laptops to the top centre. But if you're like us, and using an older model with its unique up-the-left-nostril focal plane, then you might be tempted to use the Pi Camera as a surrogate, and use your laptop's microphone.

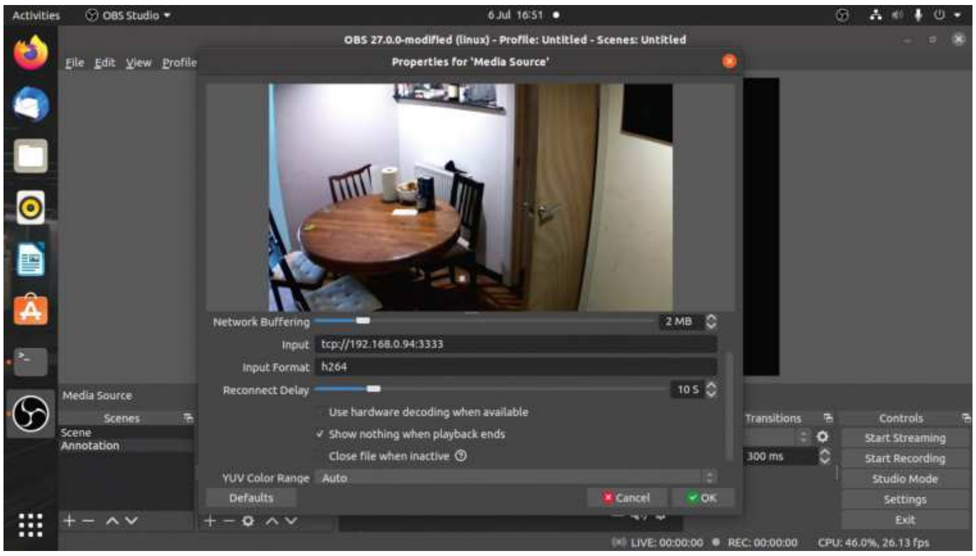
If you use our slightly Heath Robinson esque `raspivid` method on the Pi 4, then latency is certainly reduced, but not to zero. You can compensate for this in *OBS* by clicking the cog next to the Audio Input Capture meters, going to Advanced Audio Properties and adding a delay using the Sync Offset box. This might be tricky to get right, since the actual latency might vary quite a bit depending on the compressibility of the current scene, as well as network conditions.

But all is not lost. Thanks to the Pi Zero and Pi 4's OTG ports, you can set those models up as USB webcams. David Hunt's blog post at **www.davidhunt.ie/raspberry-pi-zero-with-pi-camera-as-usb-webcam** has all the details. Or, if you want a more automated approach see Jeff Geerling's GitHub for an Ansible playbook at **https://github.com/geerlingguy/pi-webcam**.

Another approach is to use Ubuntu Core to stream using MJPEG, and Canonical has an excellent blog post at **https://ubuntu.com/blog/how-to-stream-video-with-raspberry-pi-hq-camera-on-ubuntu-core**.

occasionally paradoxical multi-room antics.

Next, we approached the Raspberry Pi Foundation which kindly sent us an 8GB Raspberry Pi 4, a High Quality Pi Camera and a 6mm lens. There's another official lens (and a whole bunch of unofficial ones): the telephoto model, which is slightly pricier, but also less suited for streaming unless you are shooting distant objects. It's easy enough to set all this up (again, check out page 110) but if you're using the 6mm lens then remember to remove the CS-mount adapter that comes attached to the HQ camera module, otherwise you'll never be able to get it in focus.



OBS 27.0.0-modified (linux) - Profile: Untitled - Scenes: Untitled

Properties for 'Media Source'

Network Buffering: 2 MB
Input: tcp://192.168.0.94:3333
Input Format: h264
Reconnect Delay: 10 S
Use hardware decoding when available
✓ Show nothing when playback ends
Close file when inactive
YUV Color Range: Auto

Media Sources has all kinds of options. We couldn't get the hardware acceleration box to work, though.

Stay focused

This lens has two manual adjustment rings – focus and aperture – which lock closed with the helpfully provided screwdriver. If you're not used to manual lenses, it can be tricky to get the hang of, but you can set up *raspivid* exactly as we did earlier and see if you're going in the right direction. Our advice is to start by pointing the camera in the direction of where you want to film, then adjust the aperture ring until it looks like enough light is getting in (and do take care to ensure that there's enough light, too; it's not a night vision camera). Then screw the aperture ring to fix it in place and adjust the focus until it's crystal clear.

Speaking of focus, remember that it's possible to adjust the focus on the standard Pi camera, too. On older models you need to be a bit brave scraping off the glue around the lens, but then you'll find it rotates quite freely. Then you can get it focused right up close for a macrovision stream of your bird box. Pimoroni sells a handy lens adjustment tool for this, as well as all kinds of lenses for both camera models.

To finish, we thought we'd do something a little adventurous. After all, we had one of the most powerful Raspberry Pis in the known universe in our possession. There's no official build of *OBS* for the Pi, but it's open source so we figured let's just have a go at compiling it ourselves. You might be able to do this on a Pi 3 as well, but you'll need a swap file because 1GB of RAM isn't going to cut it. And it'll take a very long time. There's a thread all about this on the *OBS* forums at **https://obsproject.com/forum/threads/obs-raspberry-pi-build-instructions.115739**, so check there if you run into any difficulties. Things tend to break often, because updates in both Raspberry Pi OS and the *OBS* source code itself stop the two from playing nicely together.

An awful lot of packages need to be installed before we can begin, and it wouldn't be terribly useful (for our paper readers who can't copy and paste) to list them here. Instead we'll use Xavier Alonso's rather excellent script which you can find at **https://github.com/xbelanch/OBS4Pi**. You'll need to upgrade to the just-released Bullseye edition of the Pi OS to start. To do this edit **/etc/sources.list** and change all references of `buster` to `bullseye`. Then do the ol' update, upgrade *apt*-dance and machine reboot. Welcome to the future. Now download the **build.sh** file from the GitHub, make it executable, and run it with

```
$ wget https://github.com/xbelanch/OBS4Pi/raw/main/build.sh
$ chmod 755 build.sh
$ ./build.sh
```

Your Raspberry Pi will take an awfully long time to do this, and you might find that it gets fairly hot. Besides *OBS*, the script compiles custom versions of *FFMPEG* (SRT transport support isn't included in the regular version) and all the associated codecs. If it completes without error, you should be able to start *OBS* from the command line. We didn't download the launcher from Xavier's GitHub, so the instructions to use it won't work. Try running the `obs` command directly, though – you'll see an error about invalid OpenGL contexts, because *OBS* isn't familiar with the Pi's video capabilities. To get around this, force the OpenGL version to something respectable with

```
$ MESA_GL_VERSION_OVERRIDE=3.3 obs
```

or indeed download Xavier's launcher.

THINKING BIG WHILE GOING SMALL

"There's no official build of OBS Studio for the Raspberry Pi, but it's open source so we figured let's just have a go at compiling it ourselves"

And there you have it – streaming from a credit-card-sized mini-computer. Why not see if you can get it working with a microphone HAT. Or a thermal camera. Oh, and big thanks to Alex Blake and The Raspberry Pi Foundation for sending us all that useful equipment. If we become a famous influencer streamer-type we'll be sure to give you some exposure and a shout-out in any upcoming awards ceremony.

»

Further, faster, free-er streaming

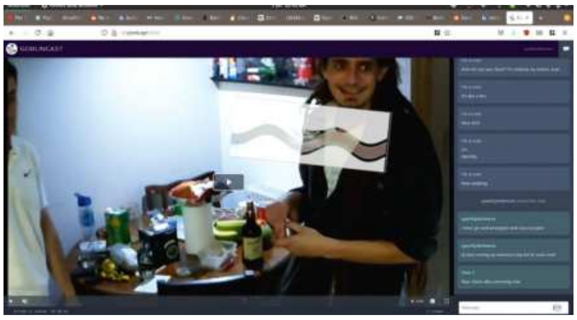
Discover what else OBS can do, then cut the corporate cord and host your own stream with Owncast.

Now that we've got basic streaming covered it's time to look further at what *OBS* can do. We won't tell you how to make a good broadcast. Like we mentioned before, we're far from experts, so we'd encourage you to watch some popular streamers yourself and take some notes on what they do well that you enjoy.

We're big fans of DistroTube on Twitch, so if you're planning a Linux-themed channel you could do a lot worse than following his example. Clicking in the Scene Collection box is easy during live broadcasts, so if your Scenes are all set up nicely before you start you won't have to fumble around switching between sources and clumsily repositioning them on the fly. Once you've got the hang of Studio View then it's easy to do smooth inter-scene transitions on the fly, too.

To see what that's about, hit the Studio View button and you'll see two panes. Preview, on the left, shows the Scene that you're planning, and Program on the right shows what you're streaming (if indeed you're doing that). Some basic transitions (Cut, Fade and Fade to Black) are available by clicking the cog icon next to the Transition button. Add some transitions, then rearrange some sources in the preview window. Use the slider to overlay the new scene manually, or hit the button to do it smoothly. Simple. We haven't covered what kind of other media sources you might be fading into or out of, so now would be a good time to look at that.

You can add sources by clicking the + in the Sources panel. The simplest source is Text (Freetype2) which enables you to add captions to your videos. Choose that option, select Create New, give the source a sensible


It's so easy to get started with Owncast – even your garden variety kitchen goblins can do it.

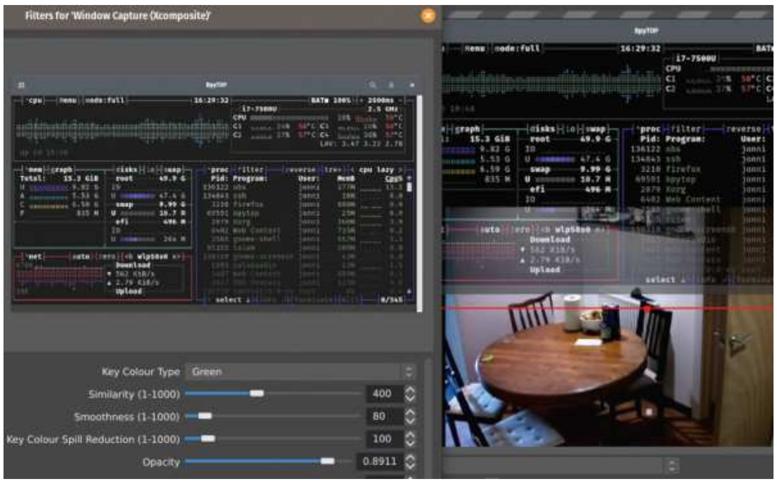
name (the text of the caption itself is reasonable), and then type in some text. And format it to your heart's content. If you use Comic Sans then someone somewhere will judge you for it, but please, stick to your guns. The chat log mode enables recent IRC (or other) logs to be displayed, but if your streaming platform has its own chat platform, this can be added through View>Docks>Custom Browser Docks. Alternatively, if you set up *Owncast* (which we'll get to in a moment) then that has its own chat interface.

It's also possible to capture your desktop, or a particular window and use that as a source in *OBS*. To capture the whole screen, use the Screen Capture (XSHM) source. You can capture individual screens or (in the advanced settings) a separate X server entirely. To capture a window, use Window Capture (Xcomposite). You can crop captures arbitrarily, which is useful for showing the currently playing Scene. Remove any sources you don't need with the minus button, and use the Scene Collection to collate sources into chapters.

Thanks to *OBS's* plugin architecture, it's easy to add custom features too. If you wanted to jazz up your game stream have your favourite search engine look up "OBS overlays", which will provide you with all kinds of jazzy borders for your sources. You might be concerned that all these sources sound very X-centric and might not work on Wayland. In the latest release you don't need to worry about this. *OBS* runs using XWayland, but behind the scenes Pipewire (and the slightly mythical sounding Desktop Portals) is used to make the magic happen.

Host your own stream

Proprietary streaming platforms (and their copious bandwidth) are all well and good, but we're fans of self-hosting here. So we went hunting for something


Adding a transparent BPYTop window to your stream is perhaps not the best monitoring solution, but it looks cool.

Filters for 'Window Capture (Xcomposite)'

Key Colour Type: Green
Similarity (1-1000): 400
Smoothness (1-1000): 80
Key Colour Spill Reduction (1-1000): 100
Opacity: 0.8911

that would enable us to take our broadcasts in-house, and it didn't take us long to find *Owncast*, which is fantastically simple to use. The demo on its website (**https://owncast.online**) shows how you can be up and running in less than a minute. We're more than happy to show you something similar here.

Owncast comes with its own online installer, which you can invoke with

```
$ curl -s https://owncast.online/install.sh | bash
```

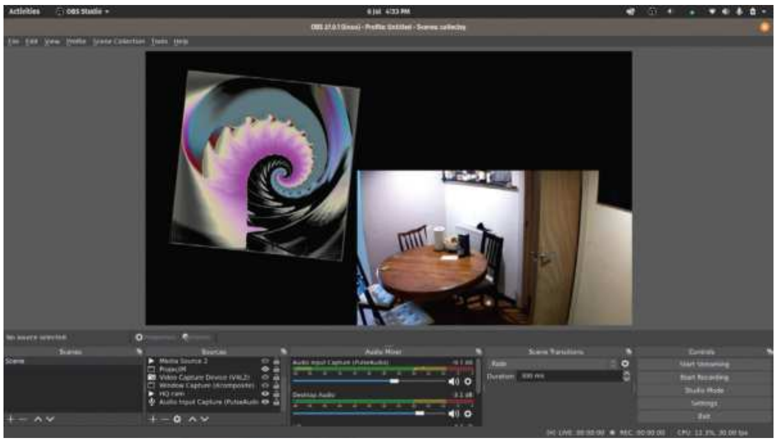
It's good practice to check scripts before running them, but since this one installs locally and doesn't use the root account it's less of a concern. The script will also install *FFMPEG* if it's missing, because that's what's going to be doing all the transcoding. *Owncast* will now be installed in your home directory and you can start it as follows:

```
$ cd owncast
$ ./owncast
```

Now you can point your browser to **http://localhost:8080** (or the hostname or IP of the remote machine you installed it on). You should see the *Owncast* interface, complete with chat. The first thing you should do is change the default streaming key (it's `abc123`), otherwise anyone on your network could hijack your channel before it's even begun. Or anyone on the internet if your machine is accessible from there. So go to the admin page at **http://localhost:8080/admin** and change the stream key (which is also the password) in the Server Settings section. You can also change the site's name, logo and description to suite your personality or purposes.

In order to get *OBS* talking to it, go to *OBS's* Settings, and in the Stream section set the service to Custom. For the server use **rtmp://localhost/live** (replacing localhost with wherever *Owncast* is running) and use the stream key you set up earlier. Now if you start streaming in *OBS*, and head over the *Owncast* page, you should see yourself. Plus you'll get some idea of the latency you're dealing with.

Speaking of which, you can configure latency in *Owncast's* settings panel – it's tempting to turn everything down to microseconds, but unless you have a very low-latency connection to your viewers this will result in buffering. For most purposes it really doesn't matter that things happen a few seconds before the internet sees them. You'll get used to people in the chat taking a while to respond to your superbly delivered jokes and witticisms (we're still waiting).



Use ProjectM to produce glorious visualisations (à la Winamp) for your audio.

In order for viewers to connect to your stream, port 8080 of the machine running *Owncast* will need to be open. This means that you'll need to brave your home routers settings here. Port 1935 needs to be open for *OBS* to connect to it over RTMP. You can change these in the settings, but remember that we're running *Owncast* as a user here, so you're not allowed to claim port 80.

If you want HTTPS – and you probably do for anything other than testing purposes – then it's a little complex. The recommended way is to use a reverse proxy. You might already have one set up, but if not you'll find instructions for doing so on Apache and Nginx at **https://owncast.online/docs/sslproxies**. It's also possibly to use the Site.js server administration tool to install *Owncast* as a service with a simple flourish at the command line.

Exactly how many users you can stream to depends on your bandwidth and chosen resolution. We found that a 1,500kbps (kilobits, not kilobytes) stream was sufficient for our purposes, and since we allegedly have a 20Mbs upload speed (fibre is good, traffic-shaping policies might not be) that should accommodate a dozen or so users. That turned out fine for our party, but if you have more than 12 friends (lucky you!), or would like to stream more bits to them and faster, then you can install *Owncast* on a VPS and have *OBS* connect remotely to it. *Owncast* also supports object storage providers, so (for a small investment) you can outsource the heavy lifting to the likes of Amazon S3, Backblaze or Digital Ocean Spaces. LXF

COPYRIGHT CONCERNS

Budding disc jockeys, mix masters, selektas and soundpeople may be tempted by the self-hosting option because it obviates the problem of automated DMCA takedowns that plague other platforms. Many a Twitch streamer has found their streams muted or stopped altogether because they had something that sounded like copyrighted music playing in the background. There are fair use exceptions to these things and copyright strikes can be contested, but the hope of being vindicated in a couple of days is little recompense for your broadcast being shut down.

We're not lawyers, but if you host a stream and only tell people you know about it (and certainly if it's password protected) then that seems pretty much like an extension of your front room and you'd expect to be able to play what you like there. But setting up a massive cluster of *Owncast* instances, publicly advertising it on Facebook and then broadcasting Major Lazer on repeat would very much be public performance. And if you look on Major Lazer (or anyone's) CDs it says "not licenced for public performance". In the UK you'd need to apply to PPL PRS for an appropriate license. In the US ASCAP, BMI and SESAC hold performance rights to most pre-recorded music. Oh and there's different licences for performance and synchronisation. Did we mention that we're not lawyers?

GAMING

Connect, configure and use gamepads

Planning some Raspberry Pi gaming? **Christian Cawley** shows you how to set up a game controller either natively or with an emulator.

OUR EXPERT

Christian Cawley
A freelance writer and editor specialising in consumer electronics, who is still finding pine needles in his Raspberry Pi.

YOU NEED

- > **Raspberry Pi 3 or later**
- > **Popular game controller**
- > **Raspberry Pi OS**
- > **USB cable**

Planning to play some games on your Raspberry Pi? *Minecraft: Pi* is far easier to play with a game controller, but you might have your eye on *Doom* or other shareware classics. Alternatively, you could be hooking up a controller to your Raspberry Pi-based retro gaming system.

A good selection of game controllers can be connected to your Raspberry Pi using USB. Furthermore, some well-known controllers can also be linked up using Bluetooth.

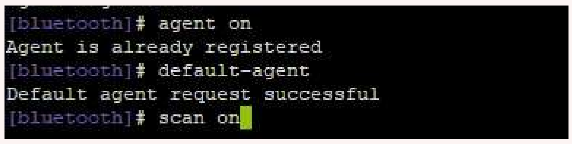
In theory, all controllers should work with a Raspberry Pi. This covers everything from generic USB joysticks and joypads to the latest Bluetooth devices. So, you can expect to be able to connect an Xbox One controller and a PS4 controller to your Raspberry Pi. Controllers designed for the PlayStation 3 and Xbox 360 will also work, as will Nintendo gamepads.

Own a PlayStation 5? The new Sony console features a major revision of the much-loved game controller. But despite being fresh out of the box in 2020, the PS5 controller will easily connect to a Raspberry Pi over Bluetooth, just like its predecessor.

Meanwhile, Xbox Series S and X controllers are backward compatible, and can be used on an Xbox One console. The new controller design should also work with the Raspberry Pi.

In this tutorial, we'll look at what you need to do to connect the most widely used game controllers to a Raspberry Pi: those intended for the Xbox One, PS4, Xbox 360 and PS3 consoles. We'll begin with the Xbox One, which boasts one of the most popular game controllers available. Also compatible with PC games,

Most game controllers will connect to the Raspberry Pi using Bluetooth or USB. For wireless Xbox One controllers, however, you'll need a USB receiver dongle.



```
[bluetooth]# agent on
Agent is already registered
[bluetooth]# default-agent
Default agent request successful
[bluetooth]# scan on
```

Configure Bluetooth to accept connections from Bluetooth game controllers, then trust the MAC address to enable easy reconnection.

this is a well-designed, multi-purpose controller that can be easily connected to a Raspberry Pi, either using USB or Bluetooth.

To connect the Xbox One controller to your Pi using USB, start by updating and upgrading Raspberry Pi OS:

```
sudo apt update
sudo apt upgrade
```

Next, connect the controller. Try it out with a game, perhaps *Minecraft: Pi*. The driver for Xbox One controllers should be built in – if not, use

```
sudo apt install xboxdrv
```

Follow the instructions to install the driver, reboot the Raspberry Pi, and try again.

Using a wireless Xbox One controller with the Raspberry Pi is a little more complicated. Two types of wireless Xbox One controller have been released. One uses wireless, while the second requires Bluetooth. How can you tell which is which?

If your Xbox One controller has Wi-Fi, the plastic around the Xbox button is the same colour as the top of the controller where the triggers and shoulder/bumper buttons are. Bluetooth Xbox One controllers, meanwhile, have a single-coloured face. So, the plastic around the Xbox button is the same colour as the rest of the controller face.

To go in-depth you can open the battery compartment. Here, the model number is listed: 1697 means it is a wireless controller, whereas 1708 indicates a Bluetooth model.

If you have the 1697 wireless model, you'll need to connect the official Microsoft Xbox Wireless Adapter to your Raspberry Pi. This is a standard USB dongle that should work out of the box. Simply hold the pairing buttons on the adapter and the Xbox One controller to sync, then start playing.

For the 1708 Bluetooth controller, you'll need the **xboxdrv** driver, as described earlier. You'll also need to disable ERTM (Enhanced Re-Transmission Mode). While enabled, this Bluetooth feature blocks syncing between the Xbox One controller and your Raspberry Pi.

```
echo 'options bluetooth disable_ertm=Y' | sudo tee -a /
etc/modprobe.d/bluetooth.conf
```

Follow this by rebooting the Pi

```
sudo reboot
```

Start Bluetooth tools

```
sudo bluetoothctl
```

At the [Bluetooth]# prompt, enable the agent and set it as default:

```
agent on
default-agent
```

Next, power up the Xbox One controller and hold the sync button. Quickly enter:

```
scan on
```

to search for the controller. The MAC address should appear, comprising six pairs of letters and numbers followed by "Xbox Wireless Controller." Use the MAC address to connect the Xbox controller:

```
connect [YOUR MAC ADDRESS]
```

When you see "Connection successful" you're ready to play. To save time connecting the controller manually in future, use the trust command:

```
trust [YOUR MAC ADDRESS]
```

But what if you prefer to use a PS4 controller with your Raspberry Pi? The steps described earlier for the Xbox One controller also apply, and you can connect with USB or Bluetooth.

Using older or different controllers

If you don't have more recent controllers (or the budget to buy them because they can be quite pricey these days), it might be easier for you to grab a controller from an older generation of consoles, such as the Xbox 360, or PlayStation 3.

Connecting an Xbox 360 controller again requires the **xboxdrv** driver. While the USB model should work out of the box, the wireless model will require a dedicated wireless receiver (the type that is developed for PC use).

The Xbox 360 controller is simple to sync with a Raspberry Pi, but connecting a PS3 controller is more involved. Although the USB option is straightforward, Bluetooth access requires some compiling.

After updating and upgrading the Raspberry Pi, install the *libusb-dev* software. This ensures the PS3 can communicate with the Raspberry Pi over Bluetooth:

```
sudo apt install libusb-dev
```

Next, create a folder for the *sixpair* software, switch to that folder, and download it:

```
mkdir ~/sixpair
cd ~/sixpair
wget http://www.pabr.org/sixlinux/sixpair.c
```

Compile the code with *gcc*:

```
gcc -o sixpair sixpair.c -lusb
```

Now you need to connect the PS3 controller to the Pi using its USB cable and run *sixpair* to configure the Bluetooth connection:

```
sudo ~/sixpair/sixpair
```

USB game controllers work out of the box, but if you're not using RetroPie, Recalbox, or similar, you'll need to manually install the xboxdrv.

Make a note of the MAC code, then disconnect the PS3 controller. You're now ready to start a Bluetooth connection as explained earlier in the instructions for the Xbox One.

Note that you aren't limited to console controllers – almost any controller you can think of with a USB connector can be hooked up to a Raspberry Pi. From a cheap £5 controller to the top-end USB pads for PC, if there's a suitable USB driver available then the controller will work. The advantage of this is that if you're building a retro gaming system, pretty much any controller you connect over USB will work.

For other Bluetooth controllers, meanwhile, generic connections should work. This means that anything – smartphone game controllers, for example – can conceivably be connected using `bluetoothctl`, but some calibration may be required.

Whatever device you're using, you may need to test it. To do this, simply use the testing tool in the Linux joystick utility:

```
sudo apt install joystick
```

Then run *jstest*

```
sudo jstest /dev/input/js0
```

This will help you to spot unresponsive buttons and thumb sticks – useful for troubleshooting. LXF

QUICK TIP

Don't lumber an old Raspberry Pi Model B/B+ board with a Bluetooth dongle – it will slow things down. Instead, connect Bluetooth controllers to a Raspberry Pi 3 or later. For the very best results, use a Raspberry Pi 4.

» CONTROLLERS AND RETRO GAMING

While you'll need to manually install drivers on the Raspberry Pi for native gaming with game controllers, using controllers with RetroPie, Recalbox and so on, is easier. Whether you're using a full disk image or have manually installed the retro gaming suite on your operating system, the drivers are included.

However, this doesn't mean you can simply plug and play. Rather, you'll need to take the time to detect and calibrate the controller. To illustrate, let's look at configuring a game controller in Recalbox.

Start by connecting the USB game controller. If you're using Bluetooth, put the device into pairing mode, then with a USB controller, open Start>Controller Settings and choose Pair a Bluetooth Controller. Select the controller from the list, then confirm.

Once detected, Recalbox will prompt the button mapping process. This involves pressing buttons for each displayed option. For instance, consider the ABXY buttons, triggers, D-pad and so on. It's worth noting that if you make a mistake, you'll need to restart the process for the controller in question.

Usually this doesn't take too long – once the buttons and direction controls are mapped you'll be ready to start playing retro games.

Paint & Draw Watercolours

Simple Steps Crochet

Practical Painter

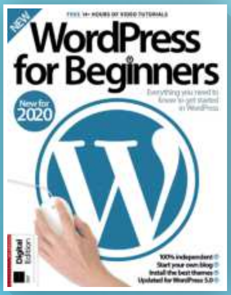
Concept Artist

Ultimate Wildlife Experiences

Paint Like the Masters

Ultimate Road Trips

Paint & Draw Acrylics


WordPress for Beginners

Coding Made Simple

101 dream Travel Locations

Create Crochet Amigurumi

Paint & Draw Anatomy

Paint & Draw Pastels

Animation Artist

100 Best Typefaces Ever

FROM THE MAKERS OF THE NUMBER ONE LINUX MAGAZINE

ULTIMATE LINUX PROJECTS

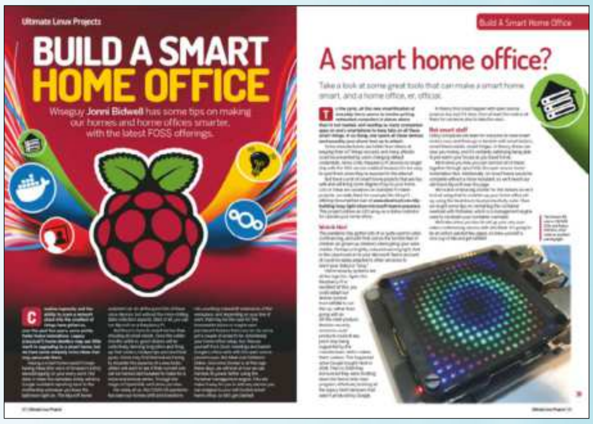
Expert guides for open-source enthusiasts

INSIDE 132 PAGES OF EXPERT ADVICE FROM LINUX FORMAT

Ultimate Linux Projects
CUSTOMISE MINT 20!
Linux Mint is fantastic but it's also flexible, malleable and tweakable. Jonni Bidwell shows you how to truly make it your own.
Customise Mint 20
The joy of tinkering

» GET THE MOST FROM YOUR LINUX DISTRO

» FREEWARE ALTERNATIVES TO POPULAR SERVICES



Ultimate Linux Projects
BUILD A SMART HOME OFFICE
Wiseguy Jonni Bidwell has some tips on making our homes and home offices smarter, with the latest FOSS offerings.
Build A Smart Home Office
A smart home office?

» SUPERCHARGE YOUR WORKING ENVIRONMENT

» HARNESS THE POWER OF RASPBERRY PI

BOOKAZINE

9021

9000