PROFESSIONAL CONTROL SYSTEM DEVELOPMENT TOOL

Home projects made easy.

CDP Studio, a great software development tool for your home projects. Build systems for Raspberry Pi, use C++ or NoCode programming, open source libraries, out of the box support for GPIO, I2C, MQTT, OPC UA and more. Create beautiful user interfaces. Built for industrial control system development, **FREE for home projects.**
love programming. I tapped out my first piece of code on a ZX-81 computer in junior school and I’ve been smitten ever since. Code controls what happens on the screen in front of you and increasingly in the world around you. Programming is an engaging process that requires thought and rewards patience. Code is good for the mind.

Here at The MagPi, we see evidence of Raspberry Pi computers making the world a better place every day. Raspberry Pi brings together code with projects: weather stations, video games, robots, websites, and server systems. The possibilities with a smattering of code are mind-blowing. Which, perhaps, is why it’s often so hard to get started. But it shouldn’t be!

This month, we’ve put together a feature on Practical Programming – your starter guide to picking a language, installing the tools you need, and finding the best courses to learn the basics. Then, we’ll guide you towards interesting starter projects.

I believe everybody should learn to code. It teaches you a new way of thinking. And the best way to learn is with a practical project. I hope you enjoy this issue!

Lucy Hattersley Editor

PS: It’s #MonthOfMaking next issue. Get your thinking caps on!
LCD & OLED

Various Device & System Support:
Raspberry Pi / Pico / Jetson nano / Arduino / PC / STM32, etc.

Multi-Size
from 0.96" to 15.6"

Touch Type
Resistive & Capacitive

Display Panel
IPS / TFT / QLED / AMOLED / PMOLED

Display Port
HDMI / USB-C / VGA / DSI / DPI / RGB / LVDS

More details, please check Waveshare website: https://waveshare.com/lcd
Contents

Cover Feature
34 Practical Programming

Regulars
92 Your Letters
97 Next Month
98 The Final Word

Project Showcases
08 DeMoor Orrery
12 Teasmade 2.0
14 RMS meteor tracker
20 Pneumonia Detection
22 GBA Remote Play
24 Sunrise Lamp
28 Technical Function
30 Rotating television

DISCLAIMER: Some of the tools and techniques shown in The MagPi magazine are dangerous unless used with skill, experience, and appropriate personal protection equipment. While we attempt to guide the reader, ultimately you are responsible for your own safety and understanding the limits of yourself and your equipment. Children should be supervised. Raspberry Pi (Trading) Ltd does not accept responsibility for any injuries, damage to equipment, or costs incurred from projects, tutorials or suggestions in The MagPi magazine. Laws and regulations covering many of the topics in The MagPi magazine are different between countries, and are always subject to change. You are responsible for understanding the requirements in your jurisdiction and ensuring that you comply with them. Some manufacturers place limits on the use of their hardware which some projects or suggestions in The MagPi magazine may go beyond. It is your responsibility to understand the manufacturer’s limits.
Tutorials
46  Create your own Teletext
50  Sensory World - part 4
54  Build a MIDI synth
58  LEGO® RC car
62  Make a retro platformer

The Big Feature
68  Maker tools for all ages

Reviews
78  Bangle.js 2
80  Maker HAT Base
82  10 amazing small projects
84  Resources: AI

Community
86  Alex Glow interview
88  This Month in Raspberry Pi

WIN A TONYPI
FREE SHIPPING ON ORDERS OVER £33 OR $50 USD*

Design starts here

More products
More suppliers
More NPI

0800 587 0991
DIGIKEY.CO.UK

Start with Digi-Key

*A shipping charge of £12.00 will be billed on all orders of less than £33.00. A shipping charge of $18.00 USD will be billed on all orders of less than $50.00 USD. All orders are shipped via UPS, Federal Express, or DHL, for delivery within 1-3 days (dependent on final destination). No handling fees. All prices are in British pound sterling or United States dollar. Digi-Key is a franchised distributor for all supplier partners. New products added daily. Digi-Key and Digi-Key Electronics are registered trademarks of Digi-Key Electronics in the U.S. and other countries. © 2022 Digi-Key Electronics, 701 Brooks Ave, South, Thief River Falls, MN 56701, USA.
Many people have a keen interest in astronomy, but before Chris de Moor created a project that’s truly out of this world, he didn’t count himself as being among them. “I didn’t know anything about astronomy and that sort of stuff,” he freely admits. Yet after visiting the Eise Eisinga Planetarium in Franeker in the Netherlands, he was inspired.

“Eise Eisinga completed an orrery in his house in 1781,” Chris says, of the oldest working orrery in the world, created by the Frisian amateur astronomer. “I saw it on the ceiling and thought I wanted one just like that, thinking it would fit perfectly in my living room. At first I believed it would be a weekend project – a simple art piece – until I decided it should work as well. And then the rest came.” Indeed, what followed would take him a year!

Chris set about building a replica, one that is similarly coloured and with planets fitted to copper tracks. But while Eisinga’s orrery was driven by a pendulum clock driving a host of mechanics in the space above the ceiling, Chris went one better: he used six Raspberry Pi Zero computers – one for each of the six planets he decided to concentrate on.

Planetary planning
“My first question was about scale and how big my orrery should be,” Chris recalls. Ultimately, he worked out that he only had room for Mercury, Venus, Earth (plus the Moon), Mars, Jupiter, and Saturn. The next step was then figuring how spaced apart they should be, the size of each representative planet and how they would move into position in real-time.

“I’d heard about Raspberry Pi computers so I decided to experiment with them,” Chris says. “I also know how to program and, while Python is not my favourite language, it was easy to learn. What I didn’t know was how stepper motors worked. They were new to me. But once I bought one Raspberry Pi Zero and a stepper motor and played around, I realised what was possible.”

Even so, there were lots of technical challenges along the way. “I got some plywood and started sawing, but a lot of things went wrong,” he laughs. “Trying to saw perfect circles by hand is impossible, so I had to go to a store that had a CNC cutting machine, which meant I needed a drawing and DXF vector file. I ended up in areas where I didn’t know anything.”
Raspberry Pi in the sky

Even so, he persevered and learned, ending up with two ways of moving the planets, each of which is connected to a Raspberry Pi Zero computer. Mars, Jupiter, and Saturn are attached to front-wheel drive, 3D-printed cars which run on tracks on the non-visible side of the project. The inner planets, Mercury, Venus, and Earth, are mounted to dishes. “The inner system is so small, you can’t have tracks there, so I mounted those planets on dishes and even connected Mercury directly to the axle of the stepper motor.”
The orrery is 30 centimetres high and suspended from the ceiling. Chris has attached it to a winch which allows it to be lowered for easy maintenance. A winding mechanism from a vacuum cleaner is used to ensure the power cord doesn’t get in the way.

This car drives Jupiter around its circular track (in reality, orbits are not perfect circles). It gets electricity via two sliding contacts mounted on the back.

Here’s a close-up view of the front of the Mars car with a Raspberry Pi Zero mounted on top.
The biggest headache was working out the planetary positions. “I looked into the mathematics of NASA, and the exact positioning of Mercury is a mathematical equation, with more than 40 pages of data. It’s very, very complex,” Chris says. “I had to make it more simple, and I found a beautiful JavaScript library called JSOrrery which is installed on a server. You give it a date and it plots the planets. Raspberry Pi computers are then connected by wireless LAN and they read their position before moving their connected planets to it.”

It’s not 100 percent accurate. “If you want to travel to Mars and you let yourself be guided by my orrery, then you’re definitely going to miss it,” Chris laughs. But as a working showpiece, it’s stunning and, what’s more, it has been made open-source so that anyone can try to make their own. “Maybe it will lead to something – perhaps people will contact me and it’ll involve some travelling and meeting new people,” he says. “If not, I’ve got a beautiful ceiling and that was the whole point.”

---

**Warning! Electrical Safety**

Please be careful when working with electrical projects around the home. Especially if they involve mains electricity.

[www.magpi.cc/electricalsafety](http://www.magpi.cc/electricalsafety)

---

**Recreating Eise Eisinga’s historic orrery**

Eise Eisinga spent seven years creating his orrery in the 18th century. Bought by King William I of the Netherlands in 1818, it was very detailed and also kept track of the phases of the moon. Credit: Erik Zachte, Wikipedia

---

**01**

Chris de Moor opted for a simpler system because of the smaller space he had. The outer planets are connected to 3D-printed cars running on tracks, controlled by a Raspberry Pi Zero.

**02**

This is a view of the inner planets attached to their propeller dish. The Sun is drawn on the ceiling, otherwise it’d be 2.85 metres in diameter!

**03**

Inner planets Earth and Venus, meanwhile, are connected to a viewing dish that’s mounted to a propeller dish. Mercury is mounted directly on to the stepper motor. It allows for tighter circling.
Teasmade 2.0

Here’s a fresh hack on an absolute classic. Hot beverage anyone? Nicola King admits to being old enough to remember the original Teasmades are machines for automatically making tea. They were hugely popular in the 1960s and 1970s, and often placed in bedrooms (for the convenience of having a piping hot cup as soon as you were awoken by the machine’s alarm). The Goblin Teasmade, in particular, is something of an icon in its field. This author fondly remembers her late grandmother’s Teasmade that sat proudly in a guest bedroom, waiting for the odd bed-and-breakfast guest that she occasionally took into her home. Although, we did always wonder the point of it when you still had to get up and go to the kitchen to get some milk from the fridge. But, despite our puzzlement, the Teasmade has basked in a form of legendary glory ever since.

The coffee stimulus

Hardly surprising then that, decades later, some makers are keen to revive that concept of an automatic hot drink maker, and what better base to use for it than a retro Goblin Teasmade? Martin Spendiff and Vanessa Bradley are two such makers, and this fun update uses a Teasmade along with Raspberry Pi Zero WH to produce their hot drink of choice… coffee.

Their motivation for the project was twofold, as Martin explains: “The Teasmade (inherited) is pretty impressive – it has no microprocessors and therefore has a workflow built with just a few switches and logic. I’ve long admired the design, so bringing the innards up to date by adding a little Linux machine to the mix was appealing. I don’t like to say cyberpunk, but cyberpunk.”

He also wanted to incorporate Google Calendar into the venture. “It’s nice to have a serious idea dressed up in a novelty application,” he tells us. “Getting Google Calendar entries to trigger events has crossed my mind a few times. The idea of automating a cup of coffee was enough motivation to get it over the finish line.”
Right royal cuppa
So, how exactly does the machine produce a regular caffeine hit? On the side of the Teasmade is a Raspberry Pi Zero with a Grove ReSpeaker HAT. “There is also a connection to a little cheap speaker and a relay switch, replacing the alarm switch on the inside,” says Martin. “The script is automatically started using systemd when it turns on. It starts a process that monitors a Google Calendar. Every minute, it runs a query on the calendar, looking for the trigger phrase. If it sees it, it starts the boil cycle.”

The most difficult challenge for the pair was working out how to get the tool that pulls alerts from a Google Calendar (gcalcli) to act as a trigger in the main code. In the end, they used the subprocess module in Python, and now the Teasmade 2.0 cleverly starts brewing their coffee ten minutes before the time it’s required.

What’s more, the speaker that Martin and Vanessa inserted belts out the God Save the Queen while refreshments are being prepared. “There is a recurring appointment in our Google Calendar for an 11am coffee. The sound of the tinny national anthem wafting from the kitchen is our signal that it is time to down tools for a break.”

![Raspberry Pi Zero is connected to a relay switch inside the Teasmade](image1)

Hidden in the Teasmade's light, the speaker plays the UK national anthem

Quick FACTS
- The project took Martin and Vanessa just a couple of days to complete
- The LEDs flash a patriotic red, white, and blue while the drink is prepared
- Find the code and instructions on GitHub: magpi.cc/teasmadegit
- Check out their other great makes here: magpi.cc/veebyoutube
- Martin recommends YouTuber James Hoffmann’s musings on bedside coffee makers: magpi.cc/barisieur

The sound of the tinny national anthem wafting from the kitchen is our signal that it is time to down tools for a break.
Mary and Mark McIntyre are so dedicated to astronomy that they had a mini-honeymoon at AstroFarm in France, and timed it so the moonlight wouldn’t affect whether they could spot stars. They even moved halfway across the UK in order to set up home close to an important stargazing spot, at which point they decided to replace their existing meteor tracking setup. They have now built an RMS (Raspberry Pi Meteor Software)-based meteor tracker, one of dozens of astronomy trackers that helped pinpoint the whereabouts of a very rare carbonaceous chondrite meteorite which landed in Somerset in February 2020.

Mary and Mark’s previous Raspberry Pi projects included a weather station and an all-sky camera which they used with an analogue meteor camera, and for which they adapted code to display findings on their website (see github.com/markmac99). Expense and maintenance issues with this setup meant they were only too happy to get involved with RMS Raspberry Pi meteor tracking instead. “The RMS project was conceived, and software is written collaboratively by astronomy academics who wanted a low-cost DIY system using off-the-shelf parts, to generate science-grade data about meteors as they burn up in our atmosphere,” says Mary.

This chimed with the interests of the McIntyres, who now run five meteor cameras and help coordinate the UKMON (UK Meteor Network). Its team of more than 100 UK citizen scientists has adapted RMS with a UK-specific toolset and data archive, making it far simpler for stargazers to identify events and objects they spot. “If more than one station captures the same event, then you can calculate velocity, mass, a more accurate orbit trajectory, and whether or not anything survived and fell as a meteorite,” Mary explains. This is what happened with last year’s Winchcombe fireball meteorite.
The camera is connected via Ethernet to a Raspberry Pi 3 with 128GB SD card to capture footage of passing meteors.

The meteor tracker once detected 20,000 objects in a single night (a spider had crawled over the lens).

Security cameras were chosen to track the night skies due to their reliable performance in low light.

RMS now has hundreds of members globally.

NASA uses RMS data to model impact hazards.

In November, a new meteor shower was identified.

The crucial data came from Mary and Mark's data.

It was the third meteor shower discovered by RMS.

Raspberry Pi stack running the four meteor tracking cameras.

Quick FACTS
A sun dog or parhelion, an atmospheric optical effect caused by ice crystals in high-level cirrus cloud, captured from the McIntyres’ home observatory.

Scanning the skies
The McIntyres’ meteor tracker uses a relatively inexpensive CCTV camera – chosen because it can detect objects in low light levels – which is pointing up at the sky and is clad in weatherproof housing. Handily, the camera can be positioned several metres from the couple’s home, while its Raspberry Pi is tucked away indoors. Mary’s version uses a Raspberry Pi 3, but she recommends the additional power of Raspberry Pi 4 if you’re keen to build your own. “We sometimes detect several hundred meteors per night on every camera, which can lead to heavy load on Raspberry Pi.”

Connectivity is in the form of PoE (Power over Ethernet), but there are other options: “The biggest advantage of Raspberry Pi is that the system can be installed virtually anywhere. Ideally, it should have an internet connection, but it can be run off-grid using batteries and solar power if necessary, provided the operator can...”

As well as meteorites, the RMS cameras can detect sprites – the extremely fast and faint upper atmosphere electric discharges that occur very high above thunderstorms.
WIZnet Ethernet HAT Contest 2022

for Raspberry Pi Pico and RP2040 projects

What can you achieve if RP2040 had access to Ethernet?

Hardware Giveaway!
Submit your idea and get one for FREE

Contest begins
December 15, 2021

Idea submissions & applications for Hardware close
January 31, 2022

Project submissions close
Mar 31, 2022

Winners announced
April 25, 2022

Sum of prize for H/W & S/W winners will be $30,000

$5,000 1st PRIZE

$3,000 2nd PRIZE

$1,000 3rd PRIZE

$500 Honorable Mention

More details at maker.wiznet.io
ACHTUNG!!!
The MagPi gibt es auch auf deutsch!

Tolles Angebot
10% Discount
und eine Überraschungsbox

Barcode scannen oder auf www.magpi.de/magpi-deutsch gehen um mehr zu erfahren.
Round up the components

For the camera, a Raspberry Pi 3 or 4, or an IP camera or Raspberry Pi HQ Camera, large-capacity microSD card, and an Ethernet connection. You’ll need a Raspberry Pi’s Ethernet port and a PoE injector, and run the script on Raspberry Pi to configure the camera for night vision.

01 Connect the camera via Raspberry Pi’s Ethernet port and a PoE injector, and run the script on Raspberry Pi to configure the camera for night vision.

Wonders of the universe

The free open-source Raspberry Pi Meteor Software (RMS) records the sky continuously from dusk until dawn, looking for anything that is moving. RMS recognises and discounts aircraft and satellite trails, fireworks, and most moths, bats, and owls. The camera is calibrated in advance with a ‘platepar’ file that identifies the star field in its field of view. Based on this information, it is able to work out the orbit of the material that caused a meteor event.

As well as meteorites, the RMS cameras can detect sprites – the extremely fast and faint upper atmosphere electric discharges that occur very high above thunderstorms. This is helping researchers who study upper atmospheric phenomena.

The open-source software is primarily written in Python 3 (and supports Python 2) with some performance aspects coded in C/C++. Downloadable from magpi.cc/rmsgit, its video capture and stability have recently been improved, while machine learning and detection accuracy are ongoing.

Mary has been impressed at how well Raspberry Pi has worked for their meteor detection setup. It cost around a third as much as a previous meteor system based on Windows. “Raspberry Pi has made it much more affordable, much easier to set up, and much simpler to maintain.”

Enable night mode

The RMS software can be downloaded from magpi.cc/rmswiki, while detailed build instructions are at magpi.cc/rmscamera. You’ll need a Raspberry Pi 3 or 4, IP camera or Raspberry Pi HQ Camera, large-capacity microSD card, and an Ethernet connection.

02 Install the camera in a weatherproof housing on the outside of a building with a good view of the night sky.

03 After the first night’s run, ‘calibrate’ the camera’s field of view against the stars – see the setup guide. After this, it should be set to run automatically every night.
Pneumonia is a respiratory condition that results from any number of viral or bacterial infections in people. It can also be fatal, which makes it very dangerous. Diagnosis is usually done via symptoms, and one of the more common ways of confirming it is via an X-ray. Young maker Arijit has been creating a device that scans these X-rays so it can more accurately detect it.

“My project Pneumonia Detection is a complete open-source system that uses a Raspberry Pi accompanied with a Raspberry Pi Camera to run a state-of-the-art embedded machine learning model,” Arijit explains. “[This] allows the device to scan chest X-rays and let the user know the amount of viral or bacterial pneumonia present in the chest of the patient. The entire system kit would cost you less than $100, including all of the hardware and the software needed, and it can practically run anywhere.”

Over the last two years, there have been more cases of pneumonia, and a great need for a low-cost system that could quickly and accurately detect it.

“There was also a massive number of people above the age of 60 who were affected by pneumonia [which] even caused death due to delay in detection,” Arijit mentions.

**Machine learning and X-rays**

For this project, Arijit decided Raspberry Pi was the way forward due to it being easily powered by a mobile battery, as well as the software support.

“The system consists of a Raspberry Pi 4, along with a Raspberry Pi Camera,” he says. “From the software side of things, we have two Docker containers running the balenaCAM and the Edge Impulse machine learning model on top of balenaOS running on Raspberry Pi. The user just has to place a chest X-ray or an image of it in front of the camera and open up [http://localhost:4912](http://localhost:4912) in a browser. After two-three seconds, the live camera feed will appear in the dashboard and show what type of pneumonia has the person [contracted] with percentages!”

The model was trained with over 3700 images of bacterial and viral pneumonia, which leads to an 80% accuracy in detections. According to a pulmonologist Arijit spoke to, human doctors only have about a 30% accuracy rate when looking at X-rays themselves.

**Positive science**

“My Twitter and LinkedIn feed was blasted out on the day I released this project publicly,” Arijit tells us. “People have shown very much interest in this.
Signs of pneumonia can be detected in X-rays. X-rays are used to try and see pneumonia in lungs, as it is faster than other methods. A simple USB webcam is used to look at chest X-rays from patients. Using machine learning, the image is analysed for any traces of bacterial or viral pneumonia. Pneumonia symptoms include coughing, breathing difficulties, and a fever. Other ways to confirm infections are blood and sputum tests. You can get vaccinated for pneumonia. Pneumonia has been known about since ancient Greece. X-rays look for fluid around the lungs.

Quick FACTS

- Pneumonia symptoms include coughing, breathing difficulties, and a fever
- Other ways to confirm infections are blood and sputum tests
- You can get vaccinated for pneumonia
- Pneumonia has been known about since ancient Greece
- X-rays look for fluid around the lungs

The system consists of a Raspberry Pi 4, along with a Raspberry Pi Camera. And to my surprise not a single result given by the model was wrong. So, it turns out the ML model has been trained well!”

With more testing and learning, the accuracy of the system will only increase, hopefully resulting in cheaper and more accessible pneumonia detectors.

“The project is completely open-sourced,” Arijit adds. “There are a lot of places wherein you can change and add a new feature to the project. It’s now up to the community to contribute and upgrade its features!”

Signs of pneumonia can be detected in X-rays.
Modding retro consoles is nothing new, but it can leave the original machine feeling a little bruised and battered. With Rodrigo Alfonso’s project, however, no classic console has to suffer harm. Instead, it’s possible to take a handheld Game Boy Advance (GBA), run PlayStation games on it, and not so much as take a screwdriver to the device itself.

Rodrigo has achieved this by creating a custom cartridge with a Raspberry Pi 3 computer tucked neatly inside. By installing and running the RetroPie emulator on the cart and engaging in a bit of jiggery pokery with the GBA’s Link Port, Rodrigo has been able to stream PSOne games on the Nintendo handheld.

“What inspired me the most was the fact that the whole project seemed impossible to achieve,” he says. “But I talked with a friend called Lucas Fryzek who knows a lot about embedded systems and he helped me figure out how to start.”

At that point, Rodrigo says he didn’t even know what a Serial Peripheral Interface (SPI) was, but he soon learned and realised its potential. The idea was to stream video from the cart at 240×160 resolution and send button-press inputs to it through the GBA’s Link Port. Nintendo included this interface so that two handheld consoles could be connected together for multiplayer gaming, but it has definitely come in handy here!

“My first step was to solder Raspberry Pi’s SPI ports to a GBA Link Cable, then test a multiboot tool to send a ROM [a file containing the game] to the Game Boy Advance,” Rodrigo explains. “I then ran a quick benchmark to see how packets could be transferred per second, though after optimising the code, the final transfer rate ended up being considerably higher.”

Rodrigo’s motivation was to create an impressive “superpowered” cartridge – “something you could take to a party and blow your friends’ minds with,” he says. He’d played lots of GBA games as a child and he still loves to check out new non-commercial releases for it. “But PlayStation was my first home console and I didn’t have too many games for it at the time.”

The project involved lots of trial and error. “I started to learn about bitmap modes on the GBA and frame buffers on the Raspberry Pi side and I tried to send a small image,” Rodrigo says. “After some trial and error time, the result was the GBA displaying one frame of Raspberry Pi OS’s desktop. Once I had that working, the remaining tasks were...
It’s something you could take to a party and blow your friends’ minds with.

Creating a virtual gamepad mapped to the GBA keys and improving the transfer rate.”

Rodrigo says there are limits. It’s not possible to transfer more than 1.6Mbps bi-directionally, for example, otherwise Raspberry Pi starts to receive “garbage” from the GBA. There’s also a one-way limit of 2.6Mbps, or 4.8Mbps if overclocked. Yet a better frame can be achieved if you forgo image quality by lowering the GBA’s native resolution from 240×160 to 120×80. Most games run smoothly.

Gamers are not limited to PlayStation games either. You could try SNES and Mega Drive, for instance, but Rodrigo says the whole point was to play 3D games. Ideally, he would have run Nintendo 64 games. “That was the original idea, but it requires more processing power and fine-tuning to get it working well.” He’s achieved his main aim, though. “There are several mods out there based on Raspberry Pi [boards] inside Game Boy Advance shells, but this is proper expandability,” he says.
The gloomier end of the year can be improved with pretty lights and sparkles, at least so goes the thinking behind purveyors of SAD (seasonal affective disorder) lamps, while the festive season also partly owes its timing to attempts to break up the monotony of winter. Teacher Russell Eveleigh’s impressive Sunrise lamp could easily do a turn as a mood-brightening SAD lamp, but was envisaged as an alternative to a Gro clock.

“Gro clocks serve to persuade children to stay in bed until a reasonable hour; in our house, the clocks we had been passed on were simply ignored,” Russell relates. Nonetheless, it was while on paternity leave that Russell began his first forays into Raspberry Pi project creation, using the Raspberry Pi Camera for a time-lapse of his young son sleeping in his Moses basket. “I’m lucky his first word wasn’t LED!” he jokes. More recently, Russell had been looking for an excuse to learn more about addressable LEDs and how they may be controlled via the GPIO pins on Raspberry Pi.

Snap to it
The Sunrise Lamp resulted from another of Russell’s nascent interests: Snapology, a technique in which origami shapes are created from rigid triangles. Having seen Ed Chew’s award-winning Tetra Lamp made from recycled cartons (magpi.cc/tetrapaklamp), Russell became convinced that a programmable light source in the centre of a similar [origami lampshade] “could create some interesting patterns and lead to lie-ins beyond 5 am!” Creating his version became a good way to
Raspberry Pi Zero W controls the colour-changing NeoPixel LEDs without the need for an external power pack.

At a preprogrammed time, the Sunrise Lamp slowly begins to brighten to indicate that it’s morning.

Quick FACTS

- Snapology guru Dave Honda’s designs inspired Russell’s project (snaporigami.weebly.com)
- It is best not to underestimate how long the origami aspect takes!
- Russell is going to trial the Sunrise Lamp as a doorbell indicator
- He’s already made a Moon Phase Lamp based on similar principles
- He hopes to track the ISS as it passes overhead
At night, the Sunrise Lamp functions as a reassuring night light, only reaching full brightness when it’s time to wake up.

A web interface allows the choice of LEDs and the illuminated effect to be changed instantly.

At night, the Sunrise Lamp functions as a reassuring night light, only reaching full brightness when it’s time to wake up.

There may be some kind of evolutionary link to our ancestors sleeping around camp-fires.

unwind of an evening, with mindfulness due to the repetition involved in folding 320 triangles and more than 600 connecting pieces for the lamp’s origami shade.

During the night, the light dims and the lamp functions as a night light, then at a predetermined time, the Sunrise Lamp slowly begins to brighten to indicate that it’s morning and time to get up, he explains.

Armed with Raspberry Pi Zero W, a plastic case, and £7 worth of NeoPixels from The Pi Hut, Russell followed the reseller’s detailed instructions for...
Embrace the light
To make your own Sunrise Lamp, you’ll need a shape such as an icosahedron that allows light through, Raspberry Pi Zero, microSD card, power supply, jumper leads, and a NeoPixel ring.

Use jumpers to attach the NeoPixel ring to the GPIO header on your Raspberry Pi Zero W, and then to the removable base section of the origami shade. Full instructions and code are at magpi.cc/sunriselamp.

It’s simplest to control the Sunrise Lamp from a web dashboard and to log in via SSH.

You’ll need around 50 sheets of stiff paper or card from which to cut and fold strips to assemble into a discernible shape. Video instructions for creating origami icosahedra can be found on YouTube: magpi.cc/icosahedron.

Embracing the light

Changing things up
Russell was pleased with his Sunrise Lamp and impressed by the intriguing patterns it creates on his ceiling, depending on whether one or more NeoPixels is active. He later revised the setup (seen in his follow-up Moon Phase Lamp) using Apache and PHP directly on the Raspberry Pi Zero. This meant the web page menu could instantly change the light sequence. Raspberry Pi Pico, or a different microcontroller, could also be used to replicate this project, but in this instance Russell was keen to learn about using Raspberry Pi with a web server. He was also advised to use red, not blue, lights so as not to interfere with anyone’s sleep patterns: magpi.cc/bluelight. “Apparently there may be some kind of evolutionary link to our ancestors sleeping around camp-fires,” he muses.

installing LED libraries (magpi.cc/usingneopixels) and NeoPixel maker Adafruit’s guide to installing CircuitPython on Raspberry Pi.

Having established that a modest number of LEDs could be controlled from Raspberry Pi with no need for an additional power source – its 5 V pin would suffice – Russell soldered the ring to the GPIO header on his Raspberry Pi and used SSH to log in and control the lamp.

Russell created a simpler version of the project using the same ideas for a colour-changing Christmas tree topper.

To make your own Sunrise Lamp, you’ll need a shape such as an icosahedron that allows light through, Raspberry Pi Zero, microSD card, power supply, jumper leads, and a NeoPixel ring.

Use jumpers to attach the NeoPixel ring to the GPIO header on your Raspberry Pi Zero W, and then to the removable base section of the origami shade. Full instructions and code are at magpi.cc/sunriselamp.

It’s simplest to control the Sunrise Lamp from a web dashboard and to log in via SSH.

Changing things up
Russell was pleased with his Sunrise Lamp and impressed by the intriguing patterns it creates on his ceiling, depending on whether one or more NeoPixels is active. He later revised the setup (seen in his follow-up Moon Phase Lamp) using Apache and PHP directly on the Raspberry Pi Zero. This meant the web page menu could instantly change the light sequence. Raspberry Pi Pico, or a different microcontroller, could also be used to replicate this project, but in this instance Russell was keen to learn about using Raspberry Pi with a web server. He was also advised to use red, not blue, lights so as not to interfere with anyone’s sleep patterns: magpi.cc/bluelight. “Apparently there may be some kind of evolutionary link to our ancestors sleeping around camp-fires,” he muses.

installing LED libraries (magpi.cc/usingneopixels) and NeoPixel maker Adafruit’s guide to installing CircuitPython on Raspberry Pi.

Having established that a modest number of LEDs could be controlled from Raspberry Pi with no need for an additional power source – its 5 V pin would suffice – Russell soldered the ring to the GPIO header on his Raspberry Pi and used SSH to log in and control the lamp.

Russell created a simpler version of the project using the same ideas for a colour-changing Christmas tree topper.

To make your own Sunrise Lamp, you’ll need a shape such as an icosahedron that allows light through, Raspberry Pi Zero, microSD card, power supply, jumper leads, and a NeoPixel ring.

Use jumpers to attach the NeoPixel ring to the GPIO header on your Raspberry Pi Zero W, and then to the removable base section of the origami shade. Full instructions and code are at magpi.cc/sunriselamp.

It’s simplest to control the Sunrise Lamp from a web dashboard and to log in via SSH.

Changing things up
Russell was pleased with his Sunrise Lamp and impressed by the intriguing patterns it creates on his ceiling, depending on whether one or more NeoPixels is active. He later revised the setup (seen in his follow-up Moon Phase Lamp) using Apache and PHP directly on the Raspberry Pi Zero. This meant the web page menu could instantly change the light sequence. Raspberry Pi Pico, or a different microcontroller, could also be used to replicate this project, but in this instance Russell was keen to learn about using Raspberry Pi with a web server. He was also advised to use red, not blue, lights so as not to interfere with anyone’s sleep patterns: magpi.cc/bluelight. “Apparently there may be some kind of evolutionary link to our ancestors sleeping around camp-fires,” he muses.

installing LED libraries (magpi.cc/usingneopixels) and NeoPixel maker Adafruit’s guide to installing CircuitPython on Raspberry Pi.

Having established that a modest number of LEDs could be controlled from Raspberry Pi with no need for an additional power source – its 5 V pin would suffice – Russell soldered the ring to the GPIO header on his Raspberry Pi and used SSH to log in and control the lamp.

Russell created a simpler version of the project using the same ideas for a colour-changing Christmas tree topper.

To make your own Sunrise Lamp, you’ll need a shape such as an icosahedron that allows light through, Raspberry Pi Zero, microSD card, power supply, jumper leads, and a NeoPixel ring.

Use jumpers to attach the NeoPixel ring to the GPIO header on your Raspberry Pi Zero W, and then to the removable base section of the origami shade. Full instructions and code are at magpi.cc/sunriselamp.

It’s simplest to control the Sunrise Lamp from a web dashboard and to log in via SSH.
I’s weird how eyes can be a bit creepy – whether they’re animated incorrectly or just sitting on their own. Anyway, here’s a very big, single eye that will keep track of you.

“This robot, named ‘Technical Function’, is used to test out offline computer vision, this time with the much more powerful Raspberry Pi 4,” Sean Glendinning, its creator, explains to us. “It also features a USB microphone array attached to its head, two speakers attached to the sides, and its most striking feature: an animatronic eye where a Raspberry Pi Camera is mounted. I originally wanted to test out offline speech recognition as well, but I haven’t had any good results. It also includes a real time clock, an accelerometer to sense orientation, and a power circuit that starts up and shuts down the robot on a button press.”

What’s the best way to test offline computer vision? An animatronic eye, of course. Rob Zwetsloot takes a look

**Electric eye**
The design is based around a two-eye concept by Will Cogley ([magpi.cc/cogleye](magpi.cc/cogleye)) which Sean cut down to be a single eye, and have space for the camera module as well.

“I then modelled (using Autodesk Inventor) an enclosure for the eye, Raspberry Pi, and the other electronics: including two speakers, amplifier, real time clock, servo controller, custom power circuit, OLED display, microphone, button, and two 5V power connectors (one micro USB and the other a DC power jack).” Sean tells us. “My first design was entirely 3D-printed, with a white shell covering up most of the exposed bolt heads. It was strongly based on the design for my microphone robot I built previously, with space made inside to mount the eye mechanism, replacing the large speaker with two smaller ones.”

It’s a very compact design, with a lot of components squeezed in.

“There were a number of drawbacks with this design,” Sean says. “The biggest one being that the eye was mounted too far forward. If the robot was dropped on its face, the eye would be damaged, and there was no way to move the eye back as the case dimensions were too small. I wanted to try building with sheet metal as a challenge. I designed a case with 3D-printed braces to hold together four pieces of 2mm aluminium sheet metal, cut with the help of my Dad. The braces hold the sides together with an assortment of nuts and bolts. Since the case was slightly bigger than the previous one, the eye mechanism could be mounted further back.”

Sean also replaced the power circuit, added an accelerometer, and changed a couple of other small bits to get the eye to where it is now.

**Technical Function**

What’s the best way to test offline computer vision?
An animatronic eye, of course. Rob Zwetsloot takes a look

**Sean Glendinning**
A student from Aberdeenshire studying robotics at Heriot-Watt University. He likes anything to do with robotics, including electronics and programming.

The components are well-packed into the case
The second, larger case is made from aluminium and is more sturdy.

I was able to make the eye track people in real time, and say their name if it saw their face.

Follow along

As well as servos that control the direction of the eye (pan and tilt), there’s also two that control the upper and lower eyelid. These are controlled via Raspberry Pi, which tracks faces using OpenCV.

“Also connected is an audio amplifier that powers the two side-mounted speakers, with the audio signal coming from the 3.5 mm audio jack on Raspberry Pi.” Sean continues. “The microphone array is connected through USB, while two USB ports and the Ethernet port are left exposed for programming. The power circuit connects to the 5 V input, the on/off button, and the micro USB input. All that’s left is the real time clock module and the accelerometer, which are both connected through I2C, as is the servo driver... using all of this, I was able to make the eye track people in real time, and say their name if it saw their face (it would have to have their face recorded first). It also blinks at random intervals.”

With all that, it works well, although apparently a little too well for some, as Sean explains peoples’ reactions: “Very positive! However, I have had people comment on the ‘uncanny valley’ of the single mechanical eye following them around, and blinking at them.”
When Samsung introduced the Sero, it literally turned television on its head. Rather than force viewers to watch in a widescreen format, this 43-inch 4K OLED TV is able to rotate on its stand, giving viewers the choice of horizontal or vertical video – perfect if you want to switch from a TV series such as The Tick to enjoying the latest shenanigans on TikTok.

Anton Suntinger was certainly impressed after catching sight of the television in a commercial. Rather than rush out and buy one, though, he looked to create his own version from scratch. Not only would his television show whatever was being played on his Android phone, it would rotate depending on how he was holding his device.

“It started as a kind of feasibility study rather than something practical,” Anton tells us. “But when I was proceeding with the project, I discovered it could be used for a more immersive media experience, reading e-books and enjoying a new perspective for social media. I also wanted to buy a Chromecast dongle anyway and I decided it would be the right time to get started with programming.”

Steering the project

For this project, Anton used a Raspberry Pi 4. “I needed both potent computational and graphical power to drive the screen mirroring,” he explains. “In addition, the GPIO pins and just one other driver board was all that I needed computationally.”

Of course, he also required a way of turning a screen – a 32-inch Sony television in his case. As luck would have it, he owned a Thrustmaster T150 steering wheel used for controlling racing video games. “I love to salvage used tech and give it a new function and, because this wheel was lying around, I didn’t have to design a gearing and mount from the ground up,” Anton says.

His first step was to decide which parts would be useful. “I kept the DC motor and unscrewed the components,” he continues. “The only challenge then was to somehow attach my heavy TV on to it. My eyes wandered around my room and stopped at my old monitor’s VESA mount. I gave it a try and, interestingly enough, it fit perfectly when secured with bidirectional bolts.”

Round it goes

An L298N motor controller sits between Raspberry Pi and the USB steering wheel. When Raspberry Pi is booted, it runs a script that establishes Bluetooth and a wireless ADB connection with an Android device. At this point, a fork of the open-
I love to salvage used tech and give it a new function.

source screen-mirroring app Scrcpy (which is available on GitHub) becomes active, and another script discovers the current state of the phone’s rotation via an ADB shell.

“As soon as the value equals the vertical value, a Python script is triggered which controls the motor via the GPIO outputs,” Anton says. “At the same time, the output of the range is adjusted to either the horizontal mode or the portrait mode using the command-line tool Xrandr.

“When the change in the phone’s rotation is registered via ADB, the same Python script is executed with inverted values so the motor’s polarity is flipped and triggers the rotation in the other direction. At the same time, it changes the video output of Raspberry Pi. And there’s your rotating TV.”

It’s certainly impressive and Anton has learned so much from the process. “The biggest challenge – other than reducing the delay which appears when using the ADB wirelessly – was to find a responsive way to get the phone’s rotation,” he says. “This project also introduced me to the Android developer options.”

Four jumper cables are used to connect Raspberry Pi to the L298N motor driver, which is powered by an external 20 V power brick and controls the DC motor.

Anton believes the steering wheel can cope with up to four newton-metres of torque.

All the components cost about $250.

The project took three weeks to create.

It repurposes a gaming steering wheel.

It’s operated using an Android phone.
SUBSCRIBE TODAY
FROM ONLY £5
SAVE UP TO 35%

Subscriber Benefits

▶ FREE Delivery
Get it fast and for FREE
▶ Exclusive Offers
Great gifts, offers, and discounts
▶ Great Savings
Save up to 35% compared to stores

Rolling Monthly Subscription

▶ Low monthly cost (from £5)
▶ Cancel at any time
▶ Free delivery to your door
▶ Available worldwide

Subscribe for 12 Months

£55 (£5) £90 (£USA)
£80 (£E) £90 (£Rest of World)

Free Raspberry Pi Zero 2 W with 12 Month upfront subscription only (no Raspberry Pi Zero 2 W with Rolling Monthly Subscription)

📞 Subscribe by phone: 01293 312193
💌 Subscribe online: magpi.cc/subscribe
✉️ Email: magpi@subscriptionhelpline.co.uk
JOIN FOR 12 MONTHS AND GET A

FREE Raspberry Pi Zero 2 W

WITH YOUR FIRST 12-MONTH SUBSCRIPTION

Subscribe in print today and get a FREE computer!

WORTH $15

- A full Raspberry Pi desktop computer
- Learn to code and build your own projects
- Make your own retro games console, media player, magic mirror, and much, much more

Buy now: magpi.cc/subscribe

This is a limited offer. Not included with renewals. Offer subject to change or withdrawal at any time.

SUBSCRIBE on app stores

From £2.29
Get to grips with coding and start making useful apps, games, and devices. By Lucy Hattersley
We’ve had a huge amount of fun learning to code with Raspberry Pi, and for 2022 we’re going to make it our mission to make things with code.

If you got a Raspberry Pi recently, then we’re pretty sure learning to code is high up on your list. Or, maybe you already have a smattering of code and are looking for inspiration. Either way, this feature is here to help.

In this article, we’re going to look at all the resources we found helpful when learning to code over the years, and explain how you can get coding quickly on Raspberry Pi. We’ll cover the key coding concepts and tools, and the resources we found most useful. Finally, we’ll brainstorm some great ideas for apps and games. And if you want to do things seriously, we’ll show you where to go to get certificates and prove you are a coder.

We’re convinced that 2022 is going to be a standout year! So, let’s take our code and put it to good use. Let’s make practical tools with our programming skills.
Congratulations! Raspberry Pi OS is the best environment to learn coding you can find. So, you’re off to a good start. Raspberry Pi OS is a custom build based on Debian Linux and it is packed with programming environments, useful tools, links to tutorials and projects, and online help.

In this feature, we’re going to use the stock Raspberry Pi OS (installed via Imager, magpi.cc/imager). If you want help setting up Raspberry Pi and Raspberry Pi OS, take a look at our Get Started with Raspberry Pi feature in The MagPi 113 (magpi.cc/113).

Click on the Raspberry Pi applications menu and choose Programming to reveal a bunch of pre-installed coding software. Most people program using an IDE (integrated development environment) and Thonny Python IDE is your place to start. In Thonny, you will be coding in Python, which is the best language for beginners; other programming languages are available (more on those in a bit).

The advantage of using an IDE over a text editor lies in its built-in features that help you type, debug, and run code. Your text syntax is colour-coded, making it easier to read and spot errors.

If you want to move beyond Python to other programming languages, or are ready for a more detailed experience, then the IDE of choice is Microsoft’s Visual Studio Code (or VS Code, magpi.cc/vscode). Visual Studio Code is packed with advanced features, extensive language support, and extensions are used to provide additional functionality.

Since VS Code arrived on Raspberry Pi OS, we’ve started using it as our primary coding environment. Microsoft’s Jim Bennett has a blog on getting Visual Studio Code up and running on Raspberry Pi (magpi.cc/vscodeblog).
PICK A CODING LANGUAGE
Speak the lingo by learning the right language

There are many different programming languages you can choose from, and during your coding journey, you’ll typically learn a few different languages. The good news is that most modern programming languages share similar concepts: variables, if-while loops, function definitions, objects, and so on. So, once you’ve learnt the first language, the second is much easier.

As well as core languages, there are tools of the trade that you will need: SQL databasing, networking, GUI implementation, and so on.

Which languages you should learn is an ever-shifting conversation, with new languages coming along and evolving every year. Having said that, there are a few solid choices that you should keep on your radar. If you’re an absolute beginner, then avoid the chatter and go straight to Python.

**C & C++**
The venerable grandparent of coding languages. C is a superb language and we think everybody should learn C at some point. Most of Linux is built in C. Beyond C lies C++, an extension that packs in modern features such as object-oriented programming support. C is (in our opinion) not a great place to start out. But it’s certainly not a bad place to visit during your coding journey, and you may (like many programmers) end up deciding it’s where you want to settle down.

**Java**
Many desktop programs are built in Java. So it’s worth knowing, but don’t start with it. For all its usefulness, Java is wordy and complex with a lot of complexities that trip up newcomers. The good news is that there is a great community; the bad news is that you’ll need it.

Java was the primary language for Android app development and has widespread use throughout the tech industry. So it’s certainly a good tool to have in your toolbox.

**HTML & CSS**
HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are the two languages at the heart of websites. You don’t typically use these tools to create programs. That said, both are important side-skills that you should acquire. Alongside learning to program, it’s a good idea to learn how to build websites. You can also combine this with some server skills, by hosting your website using software such as Apache.

**JavaScript**
JavaScript is core to the web and is (according to StackOverflow, insights.stackoverflow.com) the most popular language in use today. While JavaScript is useful for adding power to your website, it’s more limited than Python, Java or C. JavaScript is a useful skill to have alongside your HTML and CSS, so take a look alongside learning web skills.

**Python**
Python is the de-facto standard programming language for learners. And with good reason. Its modern syntax makes code stylish and easy to read and understand; Python is an incredibly powerful language and is used by data scientists and web developers, so it’s not just a stepping stone. It’s used by Spotify, Facebook and YouTube. You can start small: creating scripts and programs to run on your computer. Python is very popular so there’s huge support from websites, courses, and other coders, and it has been adopted throughout the industry.

By using Python, it’s easy to add power to your website, useful skill to have.
Your computer is set up, your IDE is open, and it’s time to learn core coding concepts. The good news is that you don’t need to learn that much to get started! It is very easy to fall down a rabbit hole of understanding though. So be careful to learn what you need and get back on track.

Here is a brief overview of the key concepts you will need to learn, read through the lot to get an overview then start looking to learn more about each area. Each concept has a recommended resource for learning more.

**Syntax**

It’s traditional to start any programming language by getting it to print out “Hello World!”. Open Thonny Python IDE, click New, and enter this line of text the main window:

```
print("Hello, World!")
```

Click Run and Thonny will prompt you to save the file. Call it "hello.py” and it will run. You will see: “Hello, World!” in the Shell.

You’re off to a good start. Now you need to know about syntax. What does the `print` command do? And why is it `print` and not `Print`; why is "Hello, World!" in quotes? And what’s with those brackets? These are conventions that change from language to language (although many are the same in each language).

Head over to the Python tutorial at W3Schools and complete its Syntax section (magpi.cc/pythonsyntax). Any encounter with a new language should start with a fresher course on its syntax.

**Variables and data**

Once you’ve practised typing and running programs, the first thing you need to learn about are variables. These are containers for values: numbers and text strings, and so on.

```
foo = 32
bar = "Jill"
```

When you write this, Python creates two variables: `foo` contains the integer `3`; `bar` contains a string of characters making up the name “Jill”. What’s with `foo` and `bar`? You will see these during tutorials (or sometimes you’ll see `x` and `y`). These variables can be anything, `foo` and `bar` are rumoured to relate to a slang term, but they’re just used as placeholders. Variables can be any word you like, and typically the name relates to the contents, like `first_name` or `age`. Each language has its own conventions, but Python variables should:

- Only contain lower-case characters
- Use underscores instead of spaces

For a good grounding in variables, check out the Raspberry Pi Foundation's Introduction to Python (magpi.cc/pythonintro).
### Data structure

Now that you are making variables and getting to grips with basic syntax, you’ll start thinking of variables as the data that make up your program. You’ll begin bunching data together into different groups. These lists of data collate a bunch of variables into a single space, which is itself a variable.

By far the most common is a straightforward list, which is a bunch of (typically) similar items. So, you might have three strings called Jack, John, and Jill and a list variable called `people`. Python has six different data types, but you only need to learn lists, sets, and dictionaries.

- **LISTS.** A basic sequence of any type of items (strings, integers, etc) that you can add and remove items to at any point. This is known as “mutable”. Lists are common and are easy to spot in Python because they are inside square brackets and items are separated by commas, e.g.:

  ```python
  names = ["Jack", "John", "Jill"]
  ```

- **TUPLES.** Same as lists but once created you don’t add and remove items to them. Tuples are written in Python with round brackets:

  ```python
  names = ("Jack", "John", "Jill")
  ```

- **SETS.** These are a bit rarer and harder to comprehend. Sets store multiple different items into a single unordered list. Each item has to be different, so you couldn’t have two Jills in your list for example. Spotted by the use of curly braces:

  ```python
  names = {"Jack", "John", "Jill"}
  ```

- **DICTIONARIES.** Harder to get your head around yet useful. Dictionaries store values in key:value pairs. Each item has a key and a value. Typically you use this to get the value by providing the keys. Spotted by the use of curly braces and colons separating the key/value pairs. In this example, our names are the key, and the values are the person’s age:

  ```python
  names = {"Jack: 32", "Jill:29", "John:45"}
  ```

Getting your head around the various data structures can be challenging. It is best to learn lists, tuples, and sets and leave dictionaries for later. Learn Python has a good page to help you get started ([magpi.cc/learnpythonlists](https://magpi.cc/learnpythonlists)).

### Flow

After variables and lists, it’s time to start learning about program flow. This is the first big departure for a language like Python from a markup language like HTML. Flow enables you to make decisions in a program using conditional statements (in Python these are *if*, *elif*, and *else*).

You can perform the same action multiple times, known as “looping”, using *for* and *while* statements.

To learn these, we recommend you head to the Raspberry Pi Foundation’s Introduction to Python ([magpi.cc/pythonintro](https://magpi.cc/pythonintro)).
**Functions**

Now that you can make variables and control flow, it’s time to learn about functions. Programming languages come with all kinds of functions, such as `print()` and `type()`. In Python, you can spot them as single words followed by brackets. A list of Python’s built-in functions can be found here: magpi.cc/pythonfunctions.

The real power of functions is to create your own using a piece of code called a “function definition”, which you can spot as `def` in Python code. Function definitions are like copy-and-paste for code. Instead of writing out identical code several times, you create it once, inside a function definition and then use the function in your program.

Functions are a vital part of programming and it’s great to roll your own. W3Schools has a good tutorial on Python Functions magpi.cc/w3pythonfunctions.

**Debugging**

This is one thing you’ll have no shortage of practice in. Every program you write will need debugging at some point. Because it is something you will spend so much time doing, it’s worth learning how to do it better.

Sasha Vodnik has created a great video tutorial on debugging on LinkedIn Learning (magpi.cc/debugging) as part of Programming Foundations: Beyond the Fundamentals. After that, freeCodeCamp has detailed documentation on improving your debugging skills (magpi.cc/debuggingskills).

---

**Tip! Tabs or Spaces**

A key aspect of code is indentation. These are the spaces at the start of a line. In Python, you indent with blocks of four spaces and never use the tab key.

> magpi.cc/pythonindentation

“We created a guide to object-orientated programming in *The MagPi* issue 54”
**Object-orientated programming**

Once you have got to grips with variables and functions, you can start to bring it all together with object-orientated programming (OOP). This is a complex coding concept that groups variables and functions together into “classes” and uses them to create objects. You then access the functions, now referred to as “methods” using dot notation. Confused? You probably will be. At least at first.

Learning OOP is vital, although it’s a big subject and be careful not to fall too deep down a rabbit hole. You only need the basics: how to import modules and use dot notation to access methods. This will enable you to use other folks’ code in your programs, and access important APIs (application programming interfaces).

We created a guide to object-orientated programming in *The MagPi* issue 54 ([magpi.cc/54](magpi.cc/54)). For a more hands-on course, check out the Raspberry Pi Foundation and FutureLearn’s course: ‘Object-orientated Programming in Python: Create Your Own Adventure Game’ ([magpi.cc/futurelearnoop](magpi.cc/futurelearnoop)).

**GUI**

Even after learning all these skills, you will still mainly be working in text documents and the command line. This is a powerful place for a programmer to be, but the command line is not where your average non-coder lives.

To make working programs, you’ll need to implement GUIs (graphical user interfaces). Don’t worry, other people have done all the heavy lifting and there are several APIs for different operating systems. We’ve got your back here with our book *Create Graphical User Interfaces with Python* ([magpi.cc/gui](magpi.cc/gui)).

**File handling**

At some point, you’ll need to address saving data. Python manages all this data automatically, but when the program closes, everything is cleaned up and thrown away.

In order to save a program, and come back to it later, you’ll need to learn about file handling. You’ll need to save data in files. Also, many programs rely on handling data found in other files (text documents and images and so on). W3Schools has a good course on File Handling ([magpi.cc/pythonfilehandling](magpi.cc/pythonfilehandling)).

**TRAINING WEBSITES**

Here are some websites that carry full coding courses:

- FutureLearn
  > magpi.cc/futurelearn

- freeCodeCamp
  > magpi.cc/freecodecamp

- W3Schools
  > magpi.cc/w3schools

- Khan Academy
  > magpi.cc/khan

- MIT OpenSource
  > magpi.cc/mitopen

- Raspberry Pi Projects
  > magpi.cc/projects

- Coursera
  > magpi.cc/coursera

- Codecademy
  > magpi.cc/codecademy

- Udemy
  > magpi.cc/udemy
The best way to learn and improve your coding abilities is to start a project. You can learn a lot from fixing problems in projects and building something from the ground up. Plus, it’s simply more fun to build something you’re personally interested in.

What you pick is up to you. It depends on what you are interested in and there are lots of lots of ideas out there. We advise you to pick up a pen and start writing down ideas, after you get past the first few your later ideas will start to become interesting.

Here are some areas to help you get started…

**Build a full-stack website**

Everybody uses the web and many of us have had some experience of building websites. So, this is a great place to start.

Of course, because you’re now programming as well as web designing, you can take things much further and build a server setup and host your website. The freeCodeCamp course Responsive Web Design is a good place to boost your HTML & CSS skills ([magpi.cc/responsiveweb](magpi.cc/responsiveweb)). Udemy’s Web Dev Bootcamp ([magpi.cc/webdevbootcamp](magpi.cc/webdevbootcamp)) takes things a step further with web applications and database integration.

Build a web server or even a personal web browser. Building a web scraping program is a fun coding project that teaches you how to crawl websites. And you can also create websites that integrate APIs, such as a weather site that outputs the weather in a specific location. freeCodeCamp has a video with six such web projects to help you get started ([magpi.cc/quickpython](magpi.cc/quickpython)).

**Make something physical**

This is one area where we have plenty of suggestions. Every month inside this magazine you’ll find projects and tutorials that combine the electronic hardware of Raspberry Pi with the real world. Raspberry Pi is ideal for making real-world items.

Head to our website and click on Tutorials ([magpi.cc/tutorials](magpi.cc/tutorials)). At this moment, there are guides for building an Android tablet, an intruder alarm, a digital do-not-disturb sign, and a Pomodoro timer. Plus, hundreds more ideas. How about picking up the latest copy of the *Raspberry Pi Handbook 2022* ([magpi.cc/handbook2022](magpi.cc/handbook2022))? It has over 200 pages packed with projects for you to try out at home.

If you want help getting started with electronics, check out Mark Vanstone’s Get Started with Electronics guide ([magpi.cc/electronics](magpi.cc/electronics)), which is packed with information on prototyping circuits with breadboards, and adding batteries, buttons, switches, speakers, and all manner of widgets to your project.
Build an application
Making websites is fun, but many folks learn programming to make standalone applications, either for the desktop or as apps for mobile devices.

Both are eminently possible with Raspberry Pi. So much so, that the real challenge is picking a program project to develop. In the process of learning to code, you’ll build many command-line applications, but integrating a GUI will help you transition to making a full program (magpi.cc/gui). Along the way Create Graphical User Interfaces with Python teaches you how to create a simple paint drawing application and an animated GIF creator.

Beyond the guizero library, you will find many useful Python libraries. Guizero is built on top of Tkinter (magpi.cc/tkinter), and for the 20% you can’t achieve in guizero, turn to Tkinter. Beyond that, there’s PyQT5 (magpi.cc/pyqt5) which provides further bindings for features such as location, NFC, and Bluetooth and extends development up to platforms including mobile apps.

The big thing is picking an idea. Get a pen and paper and keep brainstorming until you find something that appeals. You could create a music player, news content aggregator, expense tracker, to-do list app, imager editor, or just about anything you want.

Create a game
Most computer fans grew up with video games in some form or another, and making a game is a good way to practise coding skills with more freedom of expression. The indie video game market has never been more vibrant, so a good story could catch the public eye.

If you have a memory of 1980s video gaming, then Code the Classics is the place to start (magpi.cc/codetheclassics). It features recreations of classic-style video games, like Pong and Pac-Man, but coded with Pygame Zero (magpi.cc/pygamezero). Our sister title Wireframe (wfmag.cc) has a monthly Source Code section that features video game development.

Learn to Code RPG is a new open-source game that features 600+ computer science quiz questions. If you want to make a similar RPG of your own then take a look at Mark Vanstone’s Make an Adventure Game with Ren’Py (magpi.cc/makeadventure) or KG Orphanides’ Make Your Own Video Games feature in The MagPi magazine issue #73 (magpi.cc/73), which takes you through the process of creating games and distributing them.

“Making a game is a good way to practise coding skills with more freedom of expression”
Retro Gaming with Raspberry Pi shows you how to set up a Raspberry Pi to play classic games. Build your own games console or full-size arcade cabinet, install emulation software and download classic arcade games with our step-by-step guides. Want to make games? Learn how to code your own with Python and Pygame Zero.

- **Set up Raspberry Pi for retro gaming**
- **Emulate classic computers and consoles**
- **Learn to code your own retro-style games**
- **Build a console, handheld, and full-size arcade machine**

BUY ONLINE: magpi.cc/store
Create your own teletext service

Feeling retro? Turn any Raspberry Pi into a teletext broadcast service, make your own pages, and even generate content from the web.

The web? A bit overrated if you ask us. What was wrong with the beautiful teletext pages that came into our homes in the 1980s? The latest news, pop gossip, holiday bargains, and of course Digitiser. Did you think teletext was gone forever? Well, not only have a small group of dedicated archivists been saving and transcribing old teletext signals, but they have produced Raspberry Pi software that can generate the signals required to deliver those pages to your TV. Teletext is back! Here, we’ll show you how to get a teletext service running and even create your own pages.

Choose your operating system

Teletext works by adding encoded data to the top few lines of the PAL video signal, which is why we cannot use HDMI for this project. The software we are going to use creates this encoded information, which the TV will detect as a teletext signal. This is done at the frame buffer level, which means a graphical user interface, such as Raspberry Pi Desktop, is not required. So, it’s up to you. You can install either full Raspberry Pi OS or the Lite version, which is preferable if you’re going to be displaying teletext all the time.

Get your kit together

We’re basing this project on our Raspberry Pi Zero 2 W build from issue 113 (magpi.cc/113) to which we added composite video output last month. The computing requirement of the project is quite low, so a Zero-class device is perfect and cheap too. That said, this tutorial will work with any Raspberry Pi with a composite video signal out. You’ll also need a TV with a built-in teletext decoder. These are hard to find new but are plentiful second hand. You can normally see from the remote control whether it has the capability. Older CRT televisions will also give a truly retro feel to the project.

Preparation

Once your OS is installed, open a command line, or SSH into your Raspberry Pi and before continuing, make sure everything is up-to-date with `sudo apt -y update & sudo apt -y upgrade`. If you are using Bullseye, the latest major release of Raspberry Pi OS, then Raspberry Pi requires a little configuration change. Open the main configuration file as follows:

```
sudo nano /boot/config.txt
```
Top Tip
Get interactive

VBIT2 comes with a powerful interface for updating pages in real-time. See magpi.cc/vbit2ci for more information.

You can still find teletext decoding capabilities even on modern flat-screen TVs.

Near the bottom of the file, look for a line that reads:

dtoverlay=vc4-kms-v3d

Comment it out so it looks like this:

#dtoverlay=vc4-kms-v3d

Now save and exit (CTRL+X followed by Y). Reboot before continuing.

04 Check video output
Make sure your composite output is working from boot. Disconnect any HDMI cable you have been using, and make sure you have a working display. If not, run sudo raspi-config and enable composite output in Display Options > Composite. Also, for the teletext software to work, the TV has to be able to ‘see’ the encoded data at the top of the screen. If you have configured overscan compensation in raspi-config then it will not be able to do so. Double-check Display Options > Underscan is set to ‘No’. You may need to reboot.

05 Install VBIT2
Time for the magic part. The software that generates the teletext signal for us is called VBIT2 by Peter Kwan, with the help of raspi-teletext by Alistair Buxton. Thankfully, the community has made the installation of this collection of software a piece of digital cake. To install VBIT2, raspi-teletext, and all its dependencies, run this from the command line:

curl https://raw.githubusercontent.com/peterkvt80/vbit2/master/getvbit2 | bash

This will also install a configuration utility, similar to raspi-config, that will help you get set up quickly and easily. After downloading everything it needs, it will go straight into the configuration menu.

Create your own teletext service | magpi.cc | 47
VBIT2 comes packaged with a few different services, including Teefax, a community-led collection of pages. Creating ‘blockart’ is a popular pastime amongst teletext fans.

06 Configuration

You should now see a rather lonely ‘Install service’ menu item. That’s because VBIT2 cannot run until it’s got some content to work with. Tap ENTER to select and then review the choices. Out-of-the-box, VBIT2 offers a selection of teletext services. Some are community projects; others are archives of commercial services such as Ceefax, the BBC teletext service. We chose ‘Teefax’. If you get an error at this point, try another option and then select ‘Update services’. When you return to the main menu, a few new items will have appeared. Select ‘Options’ and then both items to make sure you keep up to date and start the service on boot.

07 Start and test

VBIT2 is now ready to start. From the configuration menu, select ‘Start VBIT2’ and then exit the utility. At this point, you should be dropped back to the command prompt. Try pressing the teletext button on your remote control. Hopefully, you’ll get a colourful home page (page 100). If it didn’t work, check whether you...
can see the encoded lines at the top of the screen. If so, you’ll need to change the underscan options in raspi-config (see Step 4) and try again. If all is well, enjoy moving from page to page by entering prompted page numbers.

08 Try a different service
As mentioned earlier, you can switch between services to vary your output. At any time, run `vbit-config` from the command line and change the way VBIT2 behaves. You can also ‘Update services’ which will check for newer collections of pages and updates to existing ones. Some are art collections, while others are collections of popular pages from the past (such as the writings of Mr Biffo and the Bamboozle quiz games). Others are snapshots of an entire service in a moment of time. Some even dynamically update from news feeds. Just ‘Select service’ to change what is being broadcast in real time.

09 Add some music
Giles Booth (magpi.cc/blogmywiki) was looking to recreate the late-night feel of Ceefax pages cycling on TV with groovy background music. If you’ve got sound capability on your Raspberry Pi computer, let’s add some tunes. From the command line, run the following:

```
sudo apt install mpc mpd
mpc add http://icecast.radiofrance.fr/fip-midi.mp3
mpc play 1
```

Now add audio cabling from your Raspberry Pi to your TV’s A/V input for relaxing sounds to accompany your teletext. You can choose any music you like of course. Be warned, the music will restart on boot.

10 Add Twitter!
Tweets as teletext pages. What’s not to like? Mark Pentler has developed an additional service for VBIT2 that will generate pages from your Twitter feed. This Python script will run in the background and generate teletext pages using the TTI format supported by VBIT2. It requires a bit of setup, including creating API keys for your Twitter account. Luckily, it’s all very well detailed on the project’s GitHub page (magpi.cc/teletexttwitter). Mark himself describes the project as ‘useless’, but we think it’s a great bit of fun.

11 Create your own service
So, you’ve trawled Ceefax as it was on the 14 August 1987, played a few games, and found out that the amazing holiday offer has now expired. Never mind, why not start creating your own teletext service? VBIT2 uses the TTI format for storing pages. This is a simple text format, but there’s a lot of commands. To make your own service, you’ll need to create a git repo that VBIT2 can clone and install. Have a look at the ‘~/.teletext-services’ directory and magpi.cc/vbit2 for more information.

12 What’s next?
This is one of those projects which is done because it can be. That said, Teletext’s blocky format can be very eye-catching with its bright primary colours, and it could well serve as an attractive retro display for a shop window or makerspace. The art community is thriving and the ability of VBIT2 to live-update pages via a network interface means it can be hooked up to anything on the internet and turned into a cool display. Plus, there’s always the benefit of upcycling an old, unloved CRT box. So, what will you do with it?

---

### Top Tips

**Check out**
[teletextart.co.uk](http://teletextart.co.uk) **for lots of info on creating teletext art and their regular competitions.**

**PAL Only**
This project only works with TVs that understand the PAL (UK) video signal.
In this series, we are exploring some of the most commonly available sensors and their use cases. Previously, we built a couple of alarms – one for fire and gas safety, the other for detecting intruders – and a basic weather station. In the latter, we made use of an ADC to read the analogue output of a UV sensor. This time we’ll be using an ADC to check the value from a moisture sensor placed in a plant pot. We’ll also add a liquid level sensor to check the water level in a reservoir (saucer) holding the plant pot.

If the soil or water reservoir is dry, the monitor will send us a Pushbullet alert telling us to water the plant to keep it healthy.

Now, with the power to Raspberry Pi turned off, it’s time to connect our ADC – you may already have it set up from the last instalment in this series. Place the MCP3008 in the middle of the breadboard, straddling its central groove. Make sure it’s the correct way around, as shown in the Figure 1 wiring diagram, with the top of writing on top of the ADC nearest Raspberry Pi.

Now connect the jumper wires as in Figure 1. Two go to the ‘+’ breadboard power rail, connected to a 3V3 pin; two others are connected to a GND pin via the ‘−’ rail. The four middle legs of the ADC are connected to the SPI interface on Raspberry Pi: GPIO pins 8 (CE0), 10 (MOSI), 9 (MISO), and 11 (SCLK).

Double–check they are all connected to the correct pins otherwise it won’t work properly and you may even damage the ADC chip or your Raspberry Pi.
Connect the moisture sensor

With the ADC wired up correctly to Raspberry Pi, we can now add our moisture sensor to the setup. We’re using one from the Waveshare Sensors Pack available in the UK from The Pi Hut (magpi.cc/wavesensors). The sensor is available separately, too, and there are also alternative soil moisture sensors you could use.

As in Figure 1, we connect the sensor’s VCC pin to 3.3 V via the breadboard power rail, and its GND pin to GND on Raspberry Pi via the breadboard ground rail.

Since we want to place the sensor in a plant pot, we’re using long female-to-female jumper wires (as supplied in the Waveshare Sensors Pack) to extend the distance from our Raspberry Pi and breadboard.

Finally, we connect the sensor’s AOUT (analogue out) pin to the MCP3008’s channel 0 pin, as shown in Figure 1. You could wire it to any of the eight channels on that side, but we’re using this one in our code.

Moisture level test

Let’s create a simple program to test the sensor. In the moisture_test.py listing, we’re using the GPIO Zero library as it has a handy MCP3008 class, which we import at the top. We assign the moisture variable to the MCP3008’s channel 0 to read the connected sensor’s analogue output.

In a while True: loop, we multiply the digitally converted sensor output (which ranges from 0 to 1) by the 3300 maximum voltage (in microvolts) to get an accurate reading. In our print statement, the %-3.2f format parameter sets each output to a minimum three digits including two decimal places.

Double-check they are all connected to the correct pins otherwise it won’t work properly.
reading. We will then be able to use this to trigger an alert when the soil becomes too dry and the plant needs watering.

Running the `moisture_test.py` program again, we found that the dry soil gave a reading well under 100 mV. When we watered the plant, it rose to over 1400 mV.

**05 Liquid level sensor**

If you have your plant pot placed in a saucer containing a reservoir of water, you can also add a liquid level sensor to check the water level. This is an optional step, and probably a bit of overkill as the moisture sensor should suffice.

Our liquid level sensor is from the Waveshare Sensors Pack and also sold separately. It works in much the same way as the moisture sensor, with the water conducting a current between its metal strips. The higher the water level, the higher the voltage output: from around 0 V at 0 cm to 1.88 V at 4.8 cm. So we can use it to trigger an alert when the water level drops to near zero.

Wire it up as in Figure 1, with the AOUT pin connected to the channel 1 pin of the ADC. Test it out with the `liquid_test.py` code and immerse it at different levels in the water to see the voltage change. Again, be careful not to get water on your electronic connections or Raspberry Pi.

**06 Push notifications**

We will write a program to read the two sensors and trigger an alert when either the moisture or liquid level is too low.

While we could sound an alert with a buzzer, as for the alarms in parts 1 and 2 of this series, we thought it would be more useful to send a push notification to a phone or computer. For this, we’re using Pushbullet, which offers a free service tier.

Go to [pushbullet.com](http://pushbullet.com) and sign up for an account. Then go to Settings > Account on the website and click Create Access Token. Copy this down, as you’ll need to add it into the code.

We will also need to install the Python 3 library for Pushbullet. After making sure your Raspberry Pi is up to date, open a Terminal and enter: `sudo pip3 install pushbullet.py`
Plant monitor code

In our final code, `plant_monitor.py`, we import the `pushbullet` library and assign the `pb` variable to our Pushbullet account – you will need to replace `Your Access Token` with the Pushbullet access token you obtained in the previous step.

You will also need the name of the Pushbullet-connected device that you want to send the push notification to. Your device names can be found either on the Pushbullet website, in Settings > Devices, or by using the line `print(pb.devices)`, as shown near the top of the code.

In the `alert` function, we set the `device` variable to the name of the desired device to send the notification to. We then send a push notification with the text ‘Plant monitor:’ followed by the `message` string determined by the conditional statement in the `while True:` loop. We add a `sleep` of 60 seconds (or more if you want) so that multiple notifications aren’t sent straight after each other.

In the `while True:` loop, we check whether the readings for moisture and liquid level (in mV) and set thresholds of 500 and 1000 respectively (alter them to your preference) under which an alert is triggered with the relevant message.

Taking it further

We now have a working plant monitor that alerts us when it needs watering. You take it a step further by creating a system to water the plant automatically when the soil becomes too dry. For this, you could use a solenoid valve to open and close off the water supply, or a water pump. You’ll also need a relay board to control the solenoid or pump, since it will likely have a higher voltage and current than Raspberry Pi can output.

```python
from gpiozero import MCP3008
from time import sleep
from pushbullet import Pushbullet

pb = Pushbullet("Your Access Token")
print(pb.devices)

message = ""
moisture = MCP3008(0)
liquid = MCP3008(1)

def alert():
    device = pb.get_device('Your Device')
    push = device.push_note("Plant monitor alert: ", message)
    sleep(60)

while True:
    moisture_mv = 3300 * moisture.value
    liquid_mv = 3300 * liquid.value
    print("Moisture: %-3.2f mV   " % moisture_mv,
          "Liquid level: %-3.2f mV   " % liquid_mv, end = "\r")

    if moisture_mv < 500:
        message = "Soil moisture too low! Water me!"
        alert()
    elif liquid_mv < 1000:
        message = "Reservoir level too low! Fill it!"
        alert()
```

Top Tip 🍃

Watch it grow

Connect a Camera Module to your Raspberry Pi and take a photo of the plant every ten minutes, then combine them into a time-lapse video of it growing.
In The MagPi issue 111 (magpi.cc/111), we set up external MIDI interfaces like the Roland MT-32 and SC-55 with Raspberry Pi to play classic game soundtracks and compose music. However, original hardware has become scarce and its interest to collectors has kept prices high. Fortunately, thanks to Dale Whinham’s mt32-pi, you can turn Raspberry Pi 3, 4, or Zero 2 W into an accurate standalone emulated synth. We’re going to use its support for SC-55 quality General MIDI and Roland GS audio, as this doesn’t present the legal complications of MT-32 emulation, which requires that you own ROM images that are not currently available to buy.

**01 Budgeting**

The cheapest version of this DIY MIDI synth project only requires a Raspberry Pi Zero 2 W and its usual set of cables. But you’ll also need one or, more likely, two USB MIDI adapters to connect it. We recommend using branded adapters such as a Roland UM-One Mk 2 (£40), but we’ve also successfully implemented this project with a £7 generic USB MIDI adapter bought on eBay. These are a budget-friendly choice, but we’ve heard reports of such cables not always working as they should. For most setups, you’ll also need at least one female-to-female MIDI cable or adapter, for another £4 or so.

**02 Prepare mt32-pi**

Format a microSD card as FAT32 using Raspberry Pi Imager. Download the latest version of mt32-pi from magpi.cc/mt32releases — at the time of writing, that’s mt32-pi-0.11.0.zip. Unzip the file, and copy the zip file’s contents into the root of your freshly formatted SD card.

If you open config.txt, you’ll find a number of options to help you optimise mt32-pi for different versions of Raspberry Pi. We’ll be leaving all this at the default settings. FluidSynth polyphony has, in the latest version, been adapted to make it run smoothly on the Zero 2 W, but if you experience any performance issues, you can enable an appropriate overclock here.

**03 Basic mode**

Now edit mt32-pi.cfg. The output_device setting should already be pwm, which uses Raspberry Pi’s headphone jack, so all we need to do is configure mt32-pi to use the softsynth. Search for ‘default_synth’ and change the setting to read:

```
default_synth = soundfont
```

If you’re using Raspberry Pi Zero 2 W, which has no 3.5 mm port, set the following option to output sound via an HDMI connection.

```
output_device = hdmi
```

If you connect this to a monitor or TV with built-in speakers, you’ll get high-quality MIDI audio.
You'll Need

- USB MIDI cable(s)
- magpi.cc/umone
- Female-female MIDI adapter
- I2S DAC HAT (optional)
- A HiFiBerry DAC+ ADC in our example magpi.cc/dac2hd
- HDMI audio extractor (optional)
- HDMI audio to 3.5 mm adapter
- mt32-pi magpi.cc/mt32pi

We've built two different mt32-pi devices; this one uses Raspberry Pi 4. You can direct mt32-pi’s audio output via an I2S DAC HAT instead of the default 3.5 mm audio output.

High-quality audio output is also possible without any additional hardware. Here, mt32-pi can output sound via Raspberry Pi Zero 2 W’s HDMI cable, connected to speakers via an audio extractor, amp, or AV receiver.

Note that no visual information will appear. A better option is an AV receiver or HDMI-bearing hi-fi amp with a decent set of speakers connected to it, or an HDMI audio extractor with a 3.5 mm or stereo RCA output.

04 Detect a DAC

mt32-pi uses Raspberry Pi’s headphone port by default, but this isn’t a recipe for optimal audio quality. The software supports DAC HATs, but you’ll need a bit more information to use one. Connect your HAT to Raspberry Pi and pop in a microSD card with a fresh install of Raspberry Pi OS. Boot the system with the DAC connected. We’re going to find its address. In a Terminal:

```
sudo raspi-config
```

Set Interfacing Options > I2C > Yes to enable > Finish.

```
sudo apt install i2c-tools
i2cdetect -y 1
```

Note the position of the UU symbol – ours was in column 4, row 40, giving us the address 4d.

Having got this information, we’re going to add it to mt32-pi.

05 Configure mt32-pi.cfg for your DAC

Edit `mt32-pi.cfg` in your microSD card’s root directory again. We’re going to change a few settings and add support for our DAC. If you’re using a HiFiBerry DAC+ ADC, change the following:

```
default_synth = soundfont
output_device = i2s
i2c_dac_address = 4d
i2c_dac_init = pcm51xx
```

We want to use a General MIDI soundfont via FluidSynth, as we have no MT-32 ROMs. We want to initialise and output via our HAT. And we need the `i2c_dac_address` to match whatever result you got from `i2cdetect` in the last step.

Other settings here allow you to enable and disable features such as network MIDI support, an FTP server for easier version updates, and support for LCD displays and physical controls.
06 Specialist DACs

A list of known-functional DACs for mt32-pi can be found at magpi.cc/mt32wiki

Another option is a custom hardware HAT (magpi.cc/mt32custom) specifically designed for MIDI use, such as clumsyMIDI and mt32-pi-midi-hat. These include features such as MIDI ports, tiny OLED displays and physical buttons. These sometimes turn up on Tindie, but you’re more likely to have to assemble them yourself from open specifications. mt32-pi also supports MIDI ports connected via GPIO, standalone OLED displays, buttons, and rotary encoders.

07 First connections

Save your changes, eject the card, and insert it into Raspberry Pi. Everything’s configured, but we need to provide Raspberry Pi with power and a physical MIDI interface. This is where things get a bit tricky. mt32-pi isn’t a USB MIDI device, so you can’t just plug it into a computer and start blasting MIDI tunes.

To add the MIDI connectivity required by mt32-pi, plug in a USB MIDI adapter such as a Roland UM-One mk2. For sound output, connect Raspberry Pi’s headphone port (or Raspberry Pi Zero 2 W’s HDMI port, or any DAC you may be using) to powered speakers or an AV receiver as appropriate.
In the MIDI tab, select your USB MIDI adapter from the GM Device drop-down. Most Linux systems, including Raspberry Pi, show this as USB MIDI Interface [ALSA]. In the MT-32 tab, select your MIDI interface again and tick Roland GS device.

Leave the Audio tab’s Preferred device as <default> to have Sound Blaster audio through your usual sound card as well as MIDI output from mt32-pi. If you want to send a game’s Ad-Lib or Sound Blaster compatible MIDI audio via mt32-pi, use the Edit Game menu to change this to your USB Midi adapter on a per-game basis. Disabling General MIDI support in the MIDI tab can be used to force MT-32 emulation.

11 **DOSBox-X**
Install DOSBox-X on your PC – follow the instructions at [magpi.cc/dosemulation](http://magpi.cc/dosemulation) to build it on Raspberry Pi. We’re going to assume you’re using a Raspberry Pi or other Linux PC for this. For Windows or macOS installations, see the documentation at [magpi.cc/dosboxx](http://magpi.cc/dosboxx).

Start DOSBox-X and, at the prompt, type: `mixer /listmidi alsa`.

Copy the results listed for your USB MIDI interface – 32:0 in our example. Open the Configuration tool from the Main menu and select MIDI. For `mididevice`, enter `alsa`; for `midiconfig`, enter the address you noted down in the previous step. Click OK, then Save, and Save & Restart.

12 **Synth breakout**
The mt32-pi is also a useful tool for musicians who work with retro synth sounds. Because it’s a standalone external device, it’s easy to just plug in with minimal configuration, and you don’t have to worry about the additional resource overheads of running FluidSynth on the same system that you’re composing on.

The ability to connect digital outputs and high-quality DAC hats is a benefit, as the analogue output quality on 30-year-old original hardware can leave something to be desired.

DAWs such as Qtractor and Rosegarden use JACK to detect your USB interface and either automatically configure themselves to use it, or allow you to use JACK’s MIDI connections interface to attach the DAW’s output to your MIDI interface.
Build a LEGO® remote-controlled car

Use LEGO and the Raspberry Pi Build HAT to build a robot car, then program it so you can control it with a Bluetooth connection from your Android phone. Then add some LEDs to dazzle your friends.

In this tutorial, you will build a wheeled car using LEGO components. Its movement will be determined by two separately driven wheels placed on either side of the car, allowing it to move forwards, backwards, and turn. You can optionally add LEDs to the car to act as brake lights, indicators, or headlights.

You’ll need a Build HAT, two Technic™ motors, and an assortment of LEGO, including two wheels (we used a selection from the LEGO Education SPIKE™ Prime kit)

This tutorial is from the Raspberry Pi Foundation and you can find a more detailed guide online at magpi.cc/legocar.

01 Set up the Build HAT
Before you begin, you’ll need to have set up your Raspberry Pi computer and attached your Build HAT. See our Get Started with Raspberry Pi Build HAT in The MagPi issue 112 (magpi.cc:112).

02 Set up the LEGO SPIKE motors
It is easier to test and develop your program before you build your robot. This reduces the risk of ruining your wonderful model when a motor unexpectedly sends the robot in the wrong direction and it careens off your desk (although, of course, the good thing about using LEGO is that you can always rebuild).

The Raspberry Pi Build HAT and its Python library allow you to control LEGO Technic motors directly from your Raspberry Pi computer. Plug two motors into ports A and B on the Raspberry Pi Build HAT.

Open Thonny on your Raspberry Pi from the Programming menu.

Enter the code from bt_car_test.py (overleaf) to spin both motors at 50% of their maximum speed for 10 seconds. (They’ll run one at a time, not together.)

Run your program and check the motors turn.

Your current program should move the motors in opposite directions because the motors will be mounted on opposite sides of the car’s chassis. So anti-clockwise rotation on the left-hand wheel will move the robot forward, whereas a clockwise rotation is needed on the right-hand side.

Now that you have tested the motors, you can create functions to make the motors stop and drive forward.

Remove the two lines of code (lines 6 and 7) that make the motors run for ten seconds, and add the stop() and forward() functions from bt_car_test2.py.

The forward() function works differently from the run functions of the LEGO motors so that both motors will run together this time.

Type the rest of the lines from bt_car_test2.py. Run the code and test that your motors work.

03 Testing Blue Dot
To remotely pilot your car, you’re going to use the Blue Dot library and Android app. You should only have to pair your Raspberry Pi and mobile device once. After that, they should connect easily each time.

Open a Terminal window. At the prompt, enter:

```
sudo pip3 install bluedot
```

You should see your Terminal return that the latest version of Blue Dot is installed.
Click on the Bluetooth icon in the top right-hand corner of the desktop and make sure that Bluetooth is turned On and that the device is Discoverable.

Depending on the version of Android you are running, the steps to follow on your device may vary slightly but should be close to: Choose Pair new device, and then select your Raspberry Pi device from the devices shown. Then choose Pair from the dialog box. On the Raspberry Pi, you should be prompted to accept the pairing request.

Clicking on OK should show a successful pairing of the Raspberry Pi and the Android device. Sometimes you might be asked to confirm a code before you are allowed to pair the devices.

04 Testing Blue Dot
Create a new Python file on your Raspberry Pi and enter the code from bluedot_test.py. Run the program and then, on your Android device, open the Blue Dot app. The first screen will show you a list of Bluetooth devices that have been paired with your device.

Click on ‘raspberrypi’ from the menu and you should see a big blue dot on your screen. Tap it.

In order for Blue Dot to connect to your Raspberry Pi, a server needs to be running on the Raspberry Pi. This means that a BlueDot object (dot = BlueDot()) must have already been created in your Python program and be waiting for connections.

Make sure that you are running your program before trying to connect with Blue Dot and that it has no errors.

On the Raspberry Pi, you should see that your program has accepted the Bluetooth connection and has successfully responded to you pressing the blue dot.

05 Control your motors with Blue Dot
The Blue Dot app and Python library can be used to control your LEGO Technic motors, from your device.

Our final bt_car.py code listing is based on bt_car_test2.py. Add the import statement to line 2:

```
from bluedot import BlueDot
```

You should also replace the sleep import on line 3 with the following:

```
from signal import pause
```

Remove the for loop at the end of the code and add a function that uses Blue Dot to call the forward function to the bottom of your script.

Add the move function (as shown in bt_car.py). It has a single parameter, which has been called pos. This will be automatically passed to the function, depending on where the Blue Dot is touched.

Add the dot.when_pressed and dot.when_released calls to the bottom of your code. These will make the car move forward and stop. The final pause() call makes sure the program doesn’t just end at the bottom of the script.

Run your code. On the Blue Dot app on your device, press the blue dot near the top and the motors should turn. When you take your finger off the blue dot, the motors should stop.

At the moment, the motors will only turn the wheels in the forward direction. By using the pos parameter, you can make them turn in all directions. Add the dot.when_moved = move function so that the motors will move the car backwards, left, and right. Your final code should look like bt_car.py

Run your code again, and test it with the Blue Dot app. Pressing on the right, left, and bottom of the blue dot should now move the motors in different directions.

You can add a single line to your code so that Blue Dot responds not only to presses, but also to when your finger moves over the blue dot.

You’ll Need

- Raspberry Pi
- Raspberry Pi Build HAT
- 2 × LEGO Technic motors
- Some LEGO elements to build a wheeled vehicle
- An Android mobile phone or tablet
- 5 × AA batteries and a holder pack with a barrel jack

Top Tip

GitHub files

You can view the source code for this project on the Raspberry Pi Foundation’s GitHub page magpi.cc/legocargit.
Run your program and experiment with pressing the blue dot on your Android device, and moving your finger around to different positions. The motors should spin in different directions, and stop when you lift your finger off the blue dot.

To read the full documentation for Blue Dot, go to magpi.cc/bluedotdocs.

06 Assemble your robot

Now you have the motor code working, it is time to construct and test your robot. The basic design needs to have a mounted Raspberry Pi and Build HAT with two motors (and wheels) mounted parallel to each other, and a caster or balance point at the front. There also needs to be a secured battery pack with a barrel connector.

Raspberry Pi and Build HAT can be secured to LEGO Maker Plate using M2 machine screws and nuts. There are lots of ways to connect or mount a Raspberry Pi computer to LEGO elements. The easiest way is to use the Maker Plate that comes with the LEGO Education SPIKE Prime Expansion Set (45680). You can also use the cable clip LEGO elements from a LEGO Education kit, or design and make a laser-cut or 3D-printed adapter.

You can power the Raspberry Pi and Build HAT using a battery connected to a barrel jack. A minimum of five AA batteries or a 9 V battery will be required. Once your robot is assembled, you should test it using Bluetooth with your Android device. Power up your Raspberry Pi, and then run your `bt_car.py` program. Test that your car works when using Bluetooth and the Blue Dot app from your Android device.

You may need to make changes to your code depending on which side of the car, and which way around, your motors are connected.

07 Going headless

Now you need to make Raspberry Pi run headless. This means running your code without needing to have a monitor, keyboard, or mouse connected.

First of all, make sure your Raspberry Pi is connected to a wireless LAN network.

Now, you can use a program called cron to make your Python script run every time the Raspberry Pi is booted.

Open a Terminal by pressing `CTRL+ALT+T` on your keyboard.
from bluedot import BlueDot

dot = BlueDot()

print('Waiting...')

dot.wait_for_press()

print("It worked!")

bluedot_test.py

Language: Python

001. from bluedot import BlueDot
002. dot = BlueDot()
003.
004. print('Waiting...')
005. dot.wait_for_press()
006. print("It worked!")

bt_car.py

Language: Python

001. from buildhat import Motor
002. from bluedot import BlueDot
003. from signal import pause
004. from gpiozero import LED
005.
006. motor_left = Motor('A')
007. motor_right = Motor('B')
008. dot = BlueDot()
009. led_left = LED(20)
010. led_right = LED(21)
011.
012. def stop():
013.     motor_left.stop()
014.     motor_right.stop()
015.     led_right.on()
016.     led_left.on()
017.     led_right.off()
018.     led_left.off()
019.     led_right.blink(0.2)
020.     led_left.blink(0.2)
021.     led_right.off()
022.     led_left.off()
023.     led_right.off()
024.     led_left.off()
025.     led_right.blink(0.2)
026.     led_left.blink(0.2)
027.     led_right.off()
028.     led_left.blink(0.2)
029.     def right():
030.         motor_left.start(-100)
031.         motor_right.start(100)
032.         led_right.blink(0.2)
033.         led_left.blink(0.2)
034.         led_right.off()
035.         led_left.off()
036.         led_right.off()
037.         led_left.off()
038.         led_right.off()
039.         led_left.off()
040.         led_right.off()
041.         led_left.off()
042.         led_right.off()
043.         led_left.off()
044.         forward()
045.         if pos.top:
046.             forward()
047.         elif pos.bottom:
048.             forward()
049.         elif pos.left:
050.             forward()
051.         elif pos.right:
052.             forward()
053.     dot.when_pressed = move
054.     dot.when_released = stop
055.     dot.when_moved = move
056.     pause()
Make your own retro platformer

Code your homage to Rainbow Islands in Python – a vertical scrolling platformer where enemies meet incredibly colourful deaths

AUTHOR
JORDI SANTONJA
Jordi currently works as a software engineer in Norway for a company that develops systems to design infrastructure: roads, railways, utilities… He originally comes from Spain and, in his spare time, codes video games.

DRAW PLATFORMS
Let’s start coding some of the game’s main mechanics. We’ll first test the player’s interactions with platforms: jumping, falling, and walking around. This way, the player can advance upwards by jumping from platform to platform. To achieve this, the platforms consist of a single open polyline from left to right, which doesn’t intersect itself and has no loops.

PLAYER PROTOTYPE
The first lines of Listing02 PrototypePlayer.py contain the platforms, called surfaces in the code, copied from the result of the previous program. The rest of this listing is quite complicated and long, but it contains the main foundation of the full game. After the platforms come some player...
parameters: size, acceleration, jump speed, terminal speed, and lateral speed. You can change them at will to see how they affect the character’s movement.

Next, the Player class is defined. The update function contains its main behaviour, where the collision with the platforms is handled. The collision is computed only when the player is falling down, that is, when \( \text{speedY} \geq 0 \). An object with a speed greater than zero goes down. When the speed is less than zero it goes upwards, and it doesn’t collide with anything. The collision is computed using Shapely’s intersection function between two Shapely geometries: the platform line and the player bounding box. Both are defined as Shapely LineString. The intersection result consists of one or more points. In order to avoid missed intersections when the player is falling down at a high speed, the player’s bounding box size is expanded down by the speed value. The highest point of the intersection is assigned to the player’s position. This way, when the player’s character intersects a platform, it will remain at the top.

After the collision detection, the new position is computed from the old position and speed, limit checks are performed, and finally, input management from the keyboard is done with the three functions: jump, left, and right.

The pseudo-code for this prototype can be described as follows:

```python
class Player:
    def __init__:
        centre = initial position
        lineString = box around centre
    def draw:  
        draw lineString lines
    def update(surfaces):
        if speedY >= 0:
            for all surfaces:
                intersection surface / box
                if intersection:
                    centre.Y = intersection highest point
                    centre.Y += speedY
                    translate(lineString)
        def jump:
            if not jumping:
                speedY = -playerJumpSpeed
                jumping = True
        def left:
            centre.X -= playerLateralSpeed
        def right:
            centre.X += playerLateralSpeed
        player = Player()
    def draw():
        screen.clear()
        for all surfaces:
            draw surface
        player.draw()
    def update():
        if keyboard.left:
            player.left()
        elif keyboard.right:
            player.right()
        if keyboard.up:
            player.jump()
        player.update(surfaces)
```

PROTOTYPES

When you have an idea for a new mechanic, one of the best ways to check its feasibility and fun factor is to build a prototype. A prototype must be fast to code, and needs no fancy graphics: some boxes and lines will do. But it needs to show the mechanic as best as it can be implemented, so if it’s successful, it can be translated directly to the final game with finished graphics. I recommend taking a quick and dirty approach to prototyping – build it in the engine or programming language you’re most comfortable with. If the prototype’s successful, it can be translated into proper code in the final platform.
VERTICAL SCROLLING
To tackle the scrolling, we need some platforms. To make them, we’ll adapt the drawing platforms listing to get Listing03_DrawPlatformsVertical.py, where pressing the arrow keys ‘Up’ and ‘Down’ displaces the whole screen vertically. Then we adapt the player’s prototype code to allow the vertical scrolling in Listing04_PrototypeVerticalMovement.py, which defines the new variable `screenPosition`, which stores the vertical viewing position. All the operations are performed as before, with the intersections done in the world coordinates. In the `draw` functions, `screenPosition` is subtracted from all objects’ vertical position to draw them in the right place on the screen after scrolling. `screenPosition` is updated in `screenPositionUpdate` from the vertical position of the player.

Another tweak has been introduced here. If you’ve played around with the first prototype, you may have noticed that if you press ‘Jump’ after you fall from a platform, your character jumps. To avoid this double jump, a check has been introduced when the character is in mid-air:

```python
if not self.jumping and self.speedY >= 4:
    self.jumping = True
```

This sets the character as jumping when the vertical speed downwards is higher than 4, thus avoiding a double jump. This still allows the jump in mid-air just after falling from a platform. This is a useful technique in platform games called ‘coyote time’. The next step is to rename the ‘4’ to something more meaningful, `coyoteTimeVerticalSpeed`, and test different values.

ENEMIES
Let’s prototype the enemies now. In Listing05_PrototypeEnemies1.py and Listing06_PrototypeEnemies2.py, we code two basic enemy behaviours: back and forth on a platform, and falling down from the border of a platform.

The first enemy moves over a platform without leaving it. That behaviour is achieved by using a line to intersect the platform. The line is placed in front of the enemy to find the vertical position of the platform and place the enemy accordingly. If the line doesn’t intersect the platform, it means the enemy has passed the border, so it must reverse its trajectory to go back to the platform.

The second enemy moves over a platform, but when it reaches the platform’s border, it falls down. The collision box is similar to the player’s character one, so we could reuse the code here.

SHOOT RAINBOWS
Next, prototype the rainbows. We reuse the code from Listing04_PrototypeVerticalMovement.py to create the new file Listing07_PrototypeShootRainbows.py. Here, every time the player presses the `SPACE` bar, a new rainbow is created. As the rainbow is placed in front of the player, a new variable is defined: `directionX`, which can store two values, +1 and −1, facing right and facing left. The new rainbow is then added to the current platforms as an arc, so the player can walk and jump on it. But in the first prototype, we find that the player’s box gets into the arc when walking on top.

To fix this, we must compute a `Polygon/LineString` intersection, that draws a `LineString` completely inside the `Polygon`. Then we get the highest point of the resulting `LineString` to place the player’s character. Now the player correctly rests on top of the arcs without intersecting it. The file Listing08_PrototypeShootRainbows_Polygon.py contains the changes.

Another change is the size of the bounding box to detect intersections. When you jump from a platform to the one above, you’ll suddenly...
appear on top of the platform, which is a jarring behaviour. We can reduce the size of the bounding box to avoid this.

**COLLECTABLES**

We have enemies and rainbows. What happens when an enemy is killed by a rainbow? An item is released that can be collected by the player. 

Listing09_PrototypeShootCollectables.py

prototypes how these items fly from the enemy and land on a platform.

Here, when the player presses the SPACE bar, a new flying item is created at the character’s position. Then it starts to fly in a random direction, computed with a pair of randint – one for horizontal speed and the other for vertical speed – and it flies until it lands on a platform. The collision detection is similar to the player character collision detection. 

The Collectables class contains two lists, collectablesFlying and collectables, that store all the collectables, and defines the function addCollectable to add a new flying collectable. When a flying collectable lands on a platform, it’s deleted from the collectablesFlying list and added to the collectables list.

**DESTROY RAINBOWS**

The Listing10_PrototypeDestroyRainbows.py program prototypes both ways to destroy a rainbow: when it reaches its time limit and when the player jumps over it.

In this code, the rainbows are independent entities, stored in the class AllRainbows. This class keeps two lists: rainbows and fallingRainbows. When the player shoots a rainbow, it’s appended to the rainbows list, and when the player’s character jumps over a rainbow, or its time of life rainbowTimeLife has ended, it’s removed from the rainbows list and appended to the fallingRainbows list.

When the falling rainbow exits the screen, it’s removed from the list. When the player’s character lands on a rainbow, its speed is compared against playerVerticalSpeedToDestroyRainbow and, if bigger, allRainbows.rainbowFall is called to destroy the current rainbow and to add a new falling rainbow.

Another tweak to the player’s behaviour is also introduced here: the variable jump. When the player presses the SPACE bar longer, the character jumps higher, and with a short press, the character jumps shorter. This has been achieved by calling stopJump when the SPACE bar is released.

**GRAPHICS**

Now we have a basic prototype for all the main elements of our platformer: the player’s character, the platform, enemies, collectables, and rainbows as weapons. With some adjustments, it’s possible to start putting it all together to get the final game. It’s also the time where we can start creating the graphics. My preferred software is GIMP (gimp.org), but you can use any other software capable of creating graphic files with transparency – PNGs in our case. Please replace the provided graphics with your own – I’m not an artist!

**PLATFORMS**

With the help of GIMP, we’ve designed a set of platforms to be placed on the screen. The list of platform file names is stored in the file Listing11_PlatformNames.py under the name platformNames. Now we must draw a line over each to define the top line where objects can land. We draw the lines with the program Listing12_DrawPlatformLine.py, and, after pressing the SPACE bar, the result is stored in the file Listing13_PlatformLines.py under the variable platformLines.

Next, we create the level by placing the platforms in their final positions with the program Listing14_PlacePlatforms.py. Here, we use the mouse to move and place every platform, the

> Listing12_DrawPlatformLine.py helps us to define the platform surface where the game objects will rest.

> After creating the map, Listing19_PlaceEnemies.py allows us to place the enemies on the platforms.
COYOTE TIME

If the jumping from platforms is pixel-perfect, sometimes it can feel unfair to miss the platform by a pixel and fall to the void without the possibility to jump. To avoid this, some platformers implement a ‘coyote time’, where the player can still jump after having left the platform and being mid-air, in a similar way the cartoons realise they are in mid-air and start falling.

![Can you reach the top of the stage? And more importantly, can you create a more challenging map?](image)

arrow keys to select platforms and to scroll the screen, and the SPACE bar to print on the Python console the final distribution of the platforms. Every entry of this list contains the index of the platform and the global position on the screen. This list is copied into the variable platforms in the file Listing15_Platforms.py.

The code in Listing16_TestPlatforms.py combines the player and the rainbows from Listing10_PrototypeDestroyRainbows.py and the newly created platforms, stored in the new class AllPlatforms. Here, we discover that the previous jump height is not enough to reach the next platform, so we increase playerJumpSpeed from 12 to 14.

At the moment, our background is plain black. As we’ve drawn some clouds as platforms, it feels appropriate to have the background fade from black to blue as the player goes up. The colour is computed in the function backgroundColourUpdate and stored in backgroundColour. This code also draws the platform collision lines to check that everything works as intended.

PLAYER

Now it’s time to test the player and rainbow graphics with the program Listing17_TestPlayerGraphics.py. Here, we’ve defined the variable drawLines to enable or disable drawing the collision lines. In the final game it must be False, but for debugging and testing, it’s convenient to activate it. The player can shoot up to three rainbows at a time. To test it, the variable numberOfRainbows has been introduced, and set to 3. Later, it must start with 1 and be incremented every time a specific item is collected. To allow a small delay in the creation of the rainbows, each rainbow is created with a creationTime, and it becomes active and visible when the time arrives.

All the player image names are stored in the variable playerImageNames, and then loaded into actors at the list actors in the class Player. Then, in the draw function, the right image is selected from the current state of the player.

ENEMIES

Let’s place the enemies on the level map. I’ve created three types of enemies: one that crawls on the same platform, another that crawls on platforms but falls from its borders, and a third that flies. The image file names are stored in the file Listing18_EnemyNames. It’s a list of lists: every element of enemyNames is an enemy, containing the list of all the graphic files that define that enemy. The first element in every list is the enemy facing right, the second one is facing left; and if an enemy has more graphics, all odd positions face right and all even positions face left.

To place the enemies on the map, the program Listing19_PlaceEnemies.py is used. With the arrow keys ‘Left’ and ‘Right’, we can select the enemy type and its direction: facing left or right. With ‘Up’ and ‘Down’, we move the map, and with the SPACE bar, the list of enemies and positions is printed on the Python console. We copy this result on the file Listing20_Enemies.py, in the variable named enemies. Every element of this list stores four values: the enemy index, its direction +1 or -1, and the two screen coordinates X and Y.

FULL GAME

Now it’s time to put all these elements together. Listing21_FullGame.py gathers all the previous pieces of code to connect them and create a full level. We now have a huge chunk of code where it can be difficult to find anything. To fix that, we need to split the code into manageable chunks.

Code files named Listing22 to Listing29 have been extracted from Listing21_FullGame.py, and called from Listing30_FinalGame.py. Some adjustments have been made to pass around parameters and objects, as the global variables defined in Listing21_FullGame.py aren’t accessible from the extracted modules. Listing23_IntersectionRectangles.py is a
simple rectangle intersection calculation. Its function RectanglesIntersect returns True or False when the two input rectangles overlap (or don’t). A rectangle is defined by its central point and its half size. This function performs a faster intersection than with LineStrings of Polygons.

Listing 24_Rainbows.py has all the rainbow-related operations. The class AllRainbows keeps two lists: rainbows and fallingRainbows. A rainbow will only intersect an enemy at the exact time of its creation, when timeFromCreation == 0. A rainbow won’t destroy anything at any other time, but enemies or the player will be affected if they interact with it. Elements in fallingRainbows can, however, destroy enemies if they intersect — this is computed with the RectanglesIntersect function.

The code in Listing 26_Collectables.py manages the collectable items with two main lists in the Collectables class: collectablesFlying and collectables. When an enemy dies, it releases a collectable by adding a new element to the collectablesFlying list at the same enemy position but with a random speed. When the flying collectable lands on a platform, an element from the collectableNames list is selected randomly. One of the elements is a small rainbow — when collected by the player, this adds to the number of rainbows they can cast at once.

Listing 27_Player.py contains all the player’s character stuff. When an enemy collides with the player, the number of lives is decremented, the player’s moved to its starting position, the number of rainbows restarts at 1, and all the enemies are restored. When the number of lives reaches 0, the game ends.

Listing 30_FinalGame.py imports all the previous modules, creates the game classes, and calls them at draw and update. It manages the keyboard input, and also manages the levelClear variable: when the player character reaches the top of the platform map, it’s set to True and the level ends.

NEXT STEPS
In this guide, we’ve learned how to code a platform game from scratch, starting with prototypes to test the mechanics in a complete level. The work is far from done, though.

From here, you can develop the project further: add more enemies, draw new graphics, add sounds and music, more levels, bosses, and so on. Or, even better, prototype your own type of platformer featuring a completely new game mechanic. It could be fantastic!

FINITE-STATE MACHINES
If your player system has more than two states, you must consider using a finite-state machine to control all the states and transitions between them. This way, it’s much easier to know what to do when the user presses ‘Jump’ and the protagonist is in the air falling after a hit by an enemy, for example.
If you have a Raspberry Pi, you’ve already taken a step into making. While programming is a great way to have fun and learn with Raspberry Pi, you can also use it to control things in real life.

You’ll need to get some tools and materials for the job though. Not everything is suitable for all age groups though, which can cause a little brain wracking when you’re trying to come up with things to do.

Fear not, we have the solution for you. Over the next several pages, we’ll lay out the kind of tools safe for younger and older makers alike. Get ready to build up a storm.
Where should you be building?

**YOUR MAKER SPACE**

**Find your space**

Different projects demand different kinds of spaces. Not everyone has access to a shed or garage, but you might have a desk or table. Get cutting mats and wooden boards to work on if you don’t want to damage the surface you’re using, and make sure anything else you don’t want to damage is moved away.

**Safety-conscious**

You need to make sure that any space you choose is safe to use, for both you and everything else. You’ll likely be working with tools that can get hot or are sharp, so keep flammable objects away, wear goggles where necessary, and wear thick gloves if you need to. Cut away from yourself as well, and make sure any tools are out of reach of kids and furry kids.

**Storage thoughts**

This can trip up a few new makers – where will you keep your tools or projects in progress? Plastic storage drawers are cheap and sturdy for tools and such, or you could put them in a toolbox. For smaller projects, you can easily fit them into a drawer, but with larger ones you need to consider how they’ll affect everyone else near your maker space.

**Warning! Soldering**

Soldering irons get very hot, and stay hot for a long time after they’re unplugged. Read up on soldering before connecting the wires.

**Warning! Power tools**

Power tools require some guidance and training in proper use, as well as safety equipment.

**Warning! Blades**

You can easily cut yourself on sharp blades. Cut away from yourself and store them safely.
Stripping wire with scissors

Scissors can be used to cut wire but did you know you can also strip some of the plastic off them as well? Cut wire needs to have the metal filament inside exposed so you can insert it into a breadboard - just position the scissors about a centimetre or two up the wire, apply gentle pressure, and turn the wire around. Pull on the end of the plastic to try and remove it – if it’s not budging, try with the scissors again but apply a bit more pressure.
MAKING WITH CARDBOARD

Tips for perfect builds

01 Measuring and drawing
Although cardboard is a great material to use, and you may have a lot of it, using paper to help plan out your projects can also be good. Use rulers and protractors, and any other drawing tools you have, to mark out what you’d like to do, or just trace the paper you’ve experimented with. We like to use marker pens to get a very visible line, but a pencil can also be good to start with to get it just right.

02 Scoring and cutting
For folding your cardboard, you’ll need to do some scoring; this is when you use one scissor blade to run across a fold line without cutting through it to make the cardboard easier to bend. Metal rulers are great to keep these lines straight. When cutting, cut a little outside of the lines you drew. You can always trim them down, but it’s harder to add cardboard back on.

03 Construction tips
Tabs that slot into other parts of the cardboard are a good way to make construction easier and sturdier. You can also use masking tape and then switch to a stronger tape, use Blu Tack for temporary adhesion, or long and thin bent pieces for gluing. Check out more tips from our friends at Adafruit: magpi.cc/cardboardtips.
With some maturity comes some responsibility. Here are the kind of things an adolescent could use to make some incredible things and put them on the path of being a master builder.

**Model kits**

You can learn a lot by building model robots or planes, from engineering of joints to how to clean up cuts with sandpaper. You can even do custom paint jobs with them. Your nearest hobby store should have some, but we also really like the Bandai Gunpla kits.

**Tools for Teens**

- Wirecutter
- Soldering iron
- Craft knife
- 3D printer
**MATERIALS**

**SOLDER**
- Take your builds to the next level by soldering permanent circuits using a soldering iron and some solder.

**ELECTRICAL WIRE**
- This stranded wire is more flexible but doesn’t hold its shape like prototyping wire – this is useful for soldering anything that might move around a bit more than a breadboard.

**PLA**
- Print 3D designs using a 3D printer. PLA is just one plastic you can use, although they all vary in price and use.

---

**3D MODELLING TO PRINT**

**Tips for perfect models**

**01 Blender**
A free 3D modelling program, Blender ([blender.org](http://blender.org)) is great for creating your own custom models to 3D print. It can be a bit tricky for newcomers though, which is why we recommend the excellent Blender learning courses from the Raspberry Pi Foundation: [magpi.cc/blenderprojects](http://magpi.cc/blenderprojects).

Finished files can be output as an .stl file to work with 3D printers.

**02 TinkerCAD**
While a little more basic than Blender, TinkerCAD allows you to create simpler 3D-printable files much more quickly than Blender – especially if you’re new to 3D modelling. You can find it at [tinkercad.com](http://tinkercad.com) and it’s all done in your browser. It even lets you import image files and such as well.

**03 Combining models**
A little trick we’ve learnt while messing around with miniature figures and other things – you can combine two or more models together in Blender for further modification. This is great if you find something on Thingiverse you like, such as a model car, and want to add a bigger spoiler or a character to the print.
While you can definitely stick with the tools and skills from the last few pages for as long as you’d like, here are some excellent tools that will really get your creative juices going.

Carpentry resources
Steve Ramsey on YouTube has excellent videos with great carpentry tips and tricks, as well as an online course for people really wanting to get stuck in with carpentry: magpi.cc/carpentrysteve.
Learn to weld

Welding can be a really useful tool when working with metal, however it can also be complicated and very dangerous. While there are some YouTube videos you can watch, we recommend finding an online or local course you can do that will teach you how to weld safely.

Local maker spaces

The expense of tools and materials can really ramp up, and you may not have the space or money for them. This is where a local maker space can be great – these community spaces tend to be filled with all the equipment you could need, as well as folks with experience to get advice from. You could also make a few new friends there.

There’s no dedicated website listing maker spaces, so you’ll have to use your favourite web search engine for it.
THE Official
RASPBERRY PI
HANDBOOK
2022

Start incredible projects
with Raspberry Pi

FROM THE MAKERS OF THE OFFICIAL RASPBERRY PI MAGAZINE
THE Official RASPBERRY PI HANDBOOK 2022

Buy online: magpi.cc/store

200 PAGES OF RASPBERRY PI

- QuickStart guide to setting up your Raspberry Pi computer
- Updated with Raspberry Pi Pico and all the latest kit
- The very best projects built by your Raspberry Pi community
- Discover incredible kit and tutorials for your projects
angle.js is a project by Gordon Williams that aims to place hackable smartwatches on the wrists of makers. Bangle.js 2 is the second iteration of the device, and it swaps the circular screen and buttons of the original for a square touchscreen.

Smartwatches are increasingly part of our digital lives, and as Gordon says: “even if you are an experienced hardware designer, it’s difficult to make a watch that is reliable and useful long-term, let alone waterproof and affordable.”

The hardware is therefore a cheap off-the-shelf unit, built on top of Shenzhen Smart Care Technology’s SMA Q3 (magpi.cc/smaq3). These units, bought en masse by Gordon, are then reverse-engineered and the stock firmware replaced with an open-source alternative. So, you get a relatively high-quality watch that is both open-source and programmable.

On the rear sits a four-pin Serial Wire Debug (SWD) port that can be used for connection and charging. You upload apps and code via Bluetooth, and SWD enables direct connection to replace the whole firmware if you wish.

The hardware is pleasant to use. Bangle.js 2 has a 1.3-inch touchscreen display with a single push-button on the right-hand side. The specifications are a none too shabby Cortex-M4 processor, 256kB RAM, and 1 MB on-chip flash and 8 MB external.

There is a choice of three colours: black, blue, and pink. We got the pink model in for testing and it’s a nice-looking watch that your reviewer is happy to wear all day.

Don’t be fooled: the comparison isn’t between Bangle.js 2 and a big brand watch. The default interface is clunky, the text can be tiny, and the bare-bones experience is Spartan. However, a recent 2v11 firmware update has boosted the default text size of the interface and new features are being added all the time.

The real fun begins when you connect Bangle.js 2 to Raspberry Pi and open the Espruino IDE and app store.

Roll your own
Development takes place in Chrome, and because Raspberry Pi OS ships with Chromium, you’ll need to adjust a few settings to get things working. We enabled Experimental Web Platform Features and Web Bluetooth (magpi.cc/espruinobt).

Apps are loaded via a website (banglejs.com/apps) that acts as an app store and installation platform,
with the bonus that code for the apps is available for inspection on GitHub.

Apps are developed by a vibrant community using JavaScript code. And you can develop your apps using Chromium in Raspberry Pi to access the Espruino web-based IDE (espruino.com/ide).

Espruino is a JavaScript interpreter for microcontrollers, which turns out to be a good fit for the Bangle. Bangle.js 2 comes with a range of documentation to help you on your development journey (magpi.cc/bangle2docs). The tutorial section (magpi.cc/bangle2tuts) has a range of development tutorials, including making an app. Some of these were created with the Bangle.js 1 in mind (which had three buttons) and we needed to swap out the button references to get the starter code working. There are a few tutorials specifically designed for Bangle.js 2, and a great community you can reach out to for support (magpi.cc/bangleforum). On the whole, we are having a lot of fun with Bangle.

The real fun begins when you connect Bangle.js 2 to Raspberry Pi and open the Espruino IDE.

Verdict

There isn’t anything else like Bangle on the market, and it’s a great addition to your Raspberry Pi setup. Bangle is a fun way to boost your JavaScript skills, and a great way to integrate wrist-based feedback and interaction into projects.
Maker HAT Base for Raspberry Pi 400

A whole lot more than a simple GPIO header extender. By Phil King

As well as extending the pins of Raspberry Pi 400’s GPIO header, Cytron’s Maker HAT Base has a fair few tricks up its sleeve.

Using a standard HAT with Raspberry Pi 400’s GPIO header isn’t ideal due to the latter’s position at the rear of the keyboard unit, which means that a display or LED matrix HAT faces backwards. Supplied with a short ribbon cable to connect to the GPIO header, the Maker HAT Base makes it easier as you can mount the HAT vertically on the board’s GPIO header, just as you would on any other Raspberry Pi model.

The Maker HAT Base is equipped with quite a bit more functionality, including on-board push-buttons, buzzer, and mini breadboard, plus a female breakout header.

Base for electronics
The sticky-back mini breadboard fits into a space in the middle of the board. Note that if you’re using a HAT, it will cover the breadboard – and there’s not a lot of clearance. Still, it’s a nice addition.

We really appreciate the labelling of the GPIO pins, too, which makes it much easier to find the pin you need rather than having to count along the rows. Even better, there’s a full female GPIO breakout header that you can use even with a HAT attached to the pins, so you can connect jumper wires to electronic circuits on a separate breadboard. A bonus is the inclusion of four on-board push-buttons and a buzzer (with a switch), pre-wired to certain GPIO pins.

As well as being fully labelled, the female header has a status LED for each pin, which tells you whether it’s currently pulled high or low. This should prove useful, especially for beginners, when debugging your own circuits and programs.

Verdict
Far more than a GPIO header extender, this feature-packed electronics breakout board represents excellent value for money.

9/10

 Specs
Connector:
40-pin female-to-female 10 cm ribbon cable with plastic clip

Features:
GPIO header, 4 × push-buttons, piezo buzzer, female breakout header with status LEDs, 3 × Grove connectors (GPIO, UART, I2C)

Compatibility:
Works with any Raspberry Pi model

£9 / $10

The board also acts as a mini electronics lab, complete with breadboard, buzzer, buttons, and breakout header.

There’s a full female GPIO breakout header that you can use even with a HAT attached to the pins.

With the Maker HAT Base connected to Raspberry Pi 400, this example setup is controlling a stepper motor.
HackSpace
TECHNOLOGY IN YOUR HANDS

THE MAGAZINE FOR THE MODERN MAKER

SUBSCRIBE AND SAVE UP TO 35% on the cover price

ISSUE #51
OUT NOW

hsmag.cc
One of the best features of Raspberry Pi and Raspberry Pi Pico is that they’re very small, which means you can slip them into projects or hide them around your house. Here are some of the best smaller projects for you to check out.

**OctoCam**

Zero camera kit

This fun little Raspberry Pi Zero kit will help you attach a camera to a window so that you can track wildlife, or the weather, or whatever you wish.

[octocam](https://magpi.cc/octocam) | £43

**Digital Orrery**

Solar system clock

This little clock not only tells you the time, but also depicts the exact arrangement of planets in the Solar System at the same time! You can also move it forwards or backwards to see the planets at that time.

[picosolar](https://magpi.cc/picosolar)

**Smart doorbell**

See who’s there

Want to make your own Nest-style doorbell? This project will help you do so, without having to go through Google.

[smartdoorbell](https://magpi.cc/smartdoorbell)

**Media centre**

Watch and play

It’s very easy to turn a Raspberry Pi Zero of any kind into a media centre using something like LibreELEC. Just get a nice case and you’re sorted.

[libreelec.tv](https://libreelec.tv)
PiGRRL Zero

Handheld gaming
This very tiny version of the PiGRRL that Adafruit made is about as big as a Raspberry Pi Zero. It’s a fun, if not delicate, project to recreate.

magpi.cc/pigrrlzero

Raspberry Beret

Wearable song reference
Alan McCullagh’s light-up beret also has music and a camera, along with a nice 3D-printed Prince.. symbol? Logo? It’s very cool either way.

magpi.cc/raspberryberet

PiE-Ink Name Badge

Smart identification
This reusable name badge can be customised to display whatever you’d like! Don’t abuse this power.

magpi.cc/smartbadge

Raspberry Pi Smart Watch

Wrist-bound computing
You could get an Apple Watch, or you could make your own smartwatch with Raspberry Pi and some 3D-printed parts.

magpi.cc/smartwatch

Laptop

Palm-sized computing
This clamshell laptop is very small thanks to a Zero – you could upgrade it to a Zero 2 W now as well, for even faster computing in your pocket.

magpi.cc/74

Autonomous Robotics Platform for Pico

Tiny Pico robot
This little robotics platform makes full use of Pico, not only in terms of programming but also size.

magpi.cc/picorobotics | £41
Learn AI with Raspberry Pi

Expand your knowledge of artificial intelligence with these resources. By Phil King

Make Your Own Neural Networks

Artificial intelligence (AI) research dates back to the 1950s, when scientists started creating biologically inspired electronic circuits that could function in a similar manner to the human brain using a ‘neural network’ of connections.

Things have come a long way since those very basic first machines, but the concept remains the same. Aimed at beginners (no specialist knowledge is required beyond school maths), this easy-to-follow book comprises three main parts. The first teaches you how neural networks work. The second is more practical, guiding you through the process of using Python to program your own neural network that can be trained to recognise human handwritten characters – a task that is very difficult using traditional approaches to computing.

The third and final part takes it further as you test out your neural network and try to improve its performance. An appendix shows how to get it all working on a Raspberry Pi using IPython and Jupyter notebooks.

Machine Learning

Machine learning (ML) is the science of getting computers to act without being explicitly programmed. It has led to major advances in AI, resulting in improved speech recognition, computer vision, and self-driving cars, to name but a few.

In this free Coursera course from Stanford University, Andrew Ng takes you through the fundamentals of machine learning in an easy-to-understand manner, via videos and written materials. You will learn about the most effective ML techniques and how to implement them yourself, so it’s a combination of theory and practical know-how.

Areas covered include data mining and statistical pattern recognition, with topics ranging from supervised and unsupervised learning to best practices in ML. You’ll also discover how learning algorithms are applied in fields such as smart robots and medical informatics.
Teach Your Raspberry Pi

Created by Arm, which makes the Cortex CPUs for Raspberry Pi computers, this is an excellent web resource for artificial intelligence newcomers. While some AI projects require the use of an add-on hardware accelerator such as Google’s Coral USB Accelerator, this one will work on any Raspberry Pi computer, including the Zero.

The tutorial guides you through the process of installing TensorFlow and the necessary libraries before training the AI. This involves setting up a Camera Module and environment, recording some data, and then training a new neural network based on it. The finished model will be able to tell whether the person on camera is cheering, sitting, or exhibiting random behaviour. Episode two deals with multi-gesture recognition.

Books to read

BEGINNING ARTIFICIAL INTELLIGENCE WITH THE RASPBERRY PI
Covering concepts such as neural networks, fuzzy logic, and deep learning, it also provides practical, fun projects to code and build, culminating in an AI robot vehicle.
magpi.cc/beginaibook

THE HUNDRED-PAGE MACHINE LEARNING BOOK
Learn all about machine learning in… it’s actually 160 pages, but still an impressively concise introduction to the field, covering the fundamentals in an easy-to-understand fashion.
themibook.com

HELLO WORLD
If you’re interested in the impact AI is having on every aspect of our modern lives, and where it might be heading in the future, Hannah Fry’s book is a fascinating read.
magpi.cc/hwbook

Online courses

INTRODUCTION TO MACHINE LEARNING AND AI
Created by the Raspberry Pi Foundation, this four-week FutureLearn course explores different types of machine learning and shows you how to train your own AI using free online tools.
magpi.cc/futurelearnml

LEARN WITH GOOGLE AI
Google’s education site is a great place to start your journey into exploring AI. The 37 resources here include guides and full courses – start with the Machine Learning Crash Course.
ai.google/education

FUNDAMENTALS OF DEEP LEARNING FOR COMPUTER VISION
Computer vision is a subdivision of AI, and this hands-on course from Nvidia shows you how to implement common deep learning workflows such as image classification and object detection.
magpi.cc/nvidiacv
Alex Glow

AI, robotics, PCB design, 3D printing, art, and more. It’s Alex Glow

If you’re involved at all with the online maker community, you’ve definitely seen Alex Glow’s face somewhere. Whether doing video tutorials, unboxing, interviewing, or just generally showing off a very cool project, she’s everywhere.

“When I was a kid, I’d take apart anything”, Alex says about her maker history. “Keyboards, dolls, and so on. I was raised by many people with their own creative endeavours: my bio-dad showed me how to solder, my foster parents taught me to crochet, and my adoptive parents made amazing Halloween costumes, foods, and home improvements. As a pre-teen, I got really into fountain pens and creating illuminated manuscripts; then, costumes, cut paper, and 2D wire drawings. I joined the FIRST Robotics team in high school, where I learned how to solder and program a little (beyond what I could do on the TI-83+). I dug deeply into human languages – Spanish, Russian, and Mandarin – and holography. Something about that art, steeped in the dark, catching ghosts in glass plates, is so magical to me.

I helped start the Ann Arbor maker space (All Hands Active / AHA) and worked at a screen printing shop. I started a weekly blog about my projects in 2009, making juggling balls, notebooks, a crappy amplifier for my mandolin, and more. I worked tech support for years while doing hardware in my free time.

Light writing using long exposure times

Name Alex Glow | Occupation Video host
Community role Maker leader | Website @glowascii
Lately, I’ve been embracing machine-centric arts… AxiDraw pen plotting, 3D printing, laser cutting, PCB design, and so on. But I still love to create with just my hands.”

**How did you learn about Raspberry Pi?**
It must have been through friends at AHA… gotta be eleven years ago at this point! I do also have a sheaf of Raspberry Pi ad cards I got at a long-ago Maker Faire, advertising things like Sonic Pi tutorials. No lie, I’ve probably been carting those around for over a decade now. I’m sure I’ll get to them soon.

**What led you to join Hackster?**
I was there at the beginning! Our founders, Ben and Adam, had just gotten together and decided to merge their visions into one venture. Ben and I had a mutual friend, Cedric Honnet, who thought I could help them build out a community, reaching out to hardware companies and potential members, plus writing some tutorials to populate the platform. At the time, I thought it would just be “another Instructables”, but I soon saw the difference, with the focus on electronics and connecting creators directly with hardware manufacturers.

**What is your favourite thing you’ve made?**
Off the top of my head, it’s gotta be one of the companion bots! Either Archimedes or F3NR1R – Archie because he was my first, and because I’ve gotten so much joy out of connecting with people at Maker Faires and beyond through him… and F3N was only finished [over the last two years], but he has a cute speaking voice and a really fun design. But on another note, I’ve also been able to build a long-time dream: a stereoscopic light-painting camera! Using the StereoPi – a dual camera setup for the Compute Module – and a long-exposure raspistill command, I’m able to draw things in the air with an LED, then view the whole thing as a 3D image! Magical. I still need to polish that one and write it up more fully. ☐

---

**First Raspberry Pi build**

“The first time I blinked an LED with a Raspberry Pi is documented on Hackster. It was a Raspberry Pi 2 Model B, and I was supposed to demo Samsung’s SAMI control center for IoT. I was delighted to be able to control GPIO pins from Mathematica! It was so strange to connect the ‘imaginary’ terminal with a physical LED.”

---

Alex’s first Raspberry Pi project

Archimedes is a robot owl, which is often seen on Alex’s shoulder
MagPi Monday

Amazing projects direct from our Twitter!

Every Monday we ask the question: have you made something with a Pi over the weekend? Every Monday, our followers send us amazing photos and videos of the things they’ve made.

Here’s a selection of some of the awesome things we got sent this month, and we got quite a lot! Remember to follow along at the hashtag #MagPiMonday!

01. We’ve always loved the idea of 3D-printing parts for your 3D printer
02. A very classic doorbell updated for the modern age
03. Multi-core programming is a little tricky but the results are great
04. The Apple Watch GUI is very neat, so we like this experiment
05. The audio bit is a nice touch – hopefully the radiators are up to the rest of the heating task!
06. We like that these nodes live in classic computer shells
07. One day we would also like to (safely) create an IoT lawnmower
08. Looks great, and is now a lot lighter without the CRT display inside
09. A fun visualisation of the steps in using flight tracking hardware
10. A great idea for a maker space’s wall clock!
11. A fun dice program using LEDs and simple bash scripts
I've been learning about multi-core programming in MicroPython on the Pico, using it for some MIDI note visualisation.

diyelectronicmusic
@diyelectronicmusic

We're minimising stove use due to air pollution. I'm breaking working on a simulator to reproduce the look and sound. Uses a @plimerowilicornatm minimi in a box with translucent cover running adaptation of the forest fire program, while also playing audio. Surprisingly realistic at night!

Mike Foll
@MrBeaver

Here's my weekend project - Setting up Chatburn-ctrl.com on a number of Raspberry Pi Nocs. I started with a batch of RPi's and am now moving over to a bunch of Raspberry Pi Zeros.

Made Mi Liower
@oligol)

Saw this on #YT last weekend. #Upcycling an old lawn mowing robot with a Raspberry Pi and the PiMowBot software. Nice #IoT upgrade to reuse your old or broken lawn power tools again. #DIY #MagPiMonday #MagPiTuesday

diyelectronicmusic.magepi.cc/2022/01/08/1A...
12. The colouring on this being based on Raspberry Pi accessories is great.

13. Stewart Watkiss from last issue’s interview is a great follow for useful updates like this. Stewart Watkiss from last issue’s interview is a great follow for useful updates like this.

14. This robot is fun for all ages.

15. The only Christmas project we got over the last month, but it’s a cool one.
Best of the rest!

Amazing projects that we were contacted about this month

HEART RATE DETECTION

magpi.cc/GtyfhE

This exceptionally clever piece of tech uses Eulerian Magnification which lets you analyse video frames to determine a heart beat.

BETTER SAFE THAN SORRY

magpi.cc/XSqc1V

This funky robot has sensors that let it know if there is nowhere for its foot/leg to go, announces this, and then moves backwards. Clever!

OPENAUTO PRO

magpi.cc/k8QZR4

OpenAuto is software that acts like a modern car computer system, and this one is running on Raspberry Pi.

Crowdfund this!

Raspberry Pi projects you can crowdfund this month

GPS HAT

“The extremely compact and powerful GNSS module GPS HAT for Pi is a concurrent receiver module that integrates GPS with the BeiDou system. It’s pin-to-pin compatible with Quectel’s GNSS module L76 and can receive both GPS and BeiDou open service L1 signals at the same time.”

▶ kck.st/3qz4wb2

Audio Codec HAT

“Audio Codec HAT is a low-power, high-quality stereo CODEC designed for portable digital audio applications which is based on WM8960 IC that have the advantages like low leakage, excellent PSRR and pop/click suppression mechanisms which allow direct battery connection to the speaker supply.”

▶ kck.st/34ugxqZ
Your Letters

Free gift check

When will my Raspberry Pi Zero 2 W gift arrive after I subscribe to the magazine for twelve months in print?

_ibs_ via Twitter

Depending on where you subscribed from, it may take a little while since we are based in the UK. If you are concerned, you can contact magpi@subscriptionhelpline.co.uk for any and all subscription enquiries, including where your free gift might be.

#MagPiMonday email edition

I am writing because I have made a project which I would tag with #MagPiMonday, but I’m too young for social media. I have some light-up tinsel that I put on my mini Christmas tree. It’s either solid on or solid off. I found on the battery-box a 2-pin JST connector from the display in the shop.

I did not have a 2-pin JST connector, however, so I just connected it to my Raspberry Pi Pico. I now can control my tree with PWM.

_Leo_ via Email

Yes, we sometimes forget that not everyone is old enough for social media! You can always email us any projects you make, or get a parent or guardian to send it to us on social media. This is a cool little hack, but remember never to do this with anything with high voltage!
Cameras and Bullseye

I’m still having trouble figuring out the best way to use cameras now Raspberry Pi OS Bullseye is out. I’m happy to learn new ways to use Camera Modules, but I have some stuff that already works with raspistill. Any tips?

Tan via Facebook

Since release, Bullseye has had some updates that allow it to use raspistill, which is great if you have an existing bit of software that uses it. Otherwise, you can still easily install Raspberry Pi OS ‘Legacy’, which is the previous Buster version of Raspberry Pi OS. This has some updates, but not all the security stuff that Bullseye does. For the long term, you should consider using Bullseye.

You can activate legacy camera options from raspi-config in the Terminal.
THE BEST-SELLING MAG FOR PC HARDWARE, OVERCLOCKING, GAMING & MODDING

THE MAGAZINE FOR

PC HARDWARE ENTHUSIASTS

ISSUE 222 OUT NOW

VISIT CUSTOMPC.CO.UK TO LEARN MORE
WIN

A TONYPI
IN ASSOCIATION
WITH HIWONDER

TonyPi is an incredible humanoid Raspberry Pi robot which we reviewed way back in issue 111. It’s 37 cm tall and specialises in object recognition, and is easily remote-controlled to move as well. We have one to give away.

Head here to enter: magpi.cc/win  |  Learn more: hiwonder.hk

Terms & Conditions
Competition opens on 26 January 2022 and closes on 24 February 2022. Prize is offered to participants worldwide aged 13 or over, except employees of the Raspberry Pi Foundation, the prize supplier, their families, or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from The MagPi magazine. We don’t like spam: participants’ details will remain strictly confidential and won’t be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram or Facebook.
Join us as we lift the lid on video games

Visit wfmag.cc to learn more
People get excited about old games. It’s not exactly a secret, given the ongoing boom in remakes, remasters, and repackaged classics. But there’s something special about trying to come as close as you can to the original gaming experience.

Perhaps the most surprising thing I’ve found over years of building emulation systems is how broad their appeal is. Kids who have never known a time without HD screens and realistic 3D take to pixelated platformers and EGA edutainment software without a blink.

Even the difficulty spikes of old-school arcade classics, designed to separate players from their money as efficiently as possible, become artful punctuation in the language of play, thanks to the save states and infinite credits of an emulator-based arcade cabinet.

With greater access to development tools, approachable languages, and supportive communities, game creation is also being radically democratised. And just as there are many people who love to play retro games, there are plenty of individuals and development teams who have set out to create their own.

This can go as far as full-scale releases on physical cartridges and major digital platforms, particularly when it comes to the rich world of Sega Mega Drive and other classic consoles, but both the aesthetic and the inherent technical limitations of developing for older hardware have appeal.

Technical limitations in resolution, memory size, and audio voices can make your game simpler to program. This has directly led to the rise of Fantasy Consoles, virtual 8-bit machines that run on your computer. Priced at $15 (£11), Pico-8 (magpi.cc/pico8) is the best-known of these, with official support for Raspberry Pi and a great community.

Pixel art is still going through a years-long renaissance. With many artists deliberately adopting 256-, 16-, and even four-colour palettes. You’ll see these choices reflected in games as disparate as Unpacking (through its EGA camera filter) and The Eternal Castle, a ‘remake’ of a game that never existed. If you want to start making pixel art on Raspberry Pi, Aseprite (magpi.cc/aseprite) can be compiled, and an unofficial ARM build of Pixelorama (magpi.cc/pixelorama) is now available.

More modern retro aesthetics can be found in the low-rez horror games exemplified by the Haunted PS1 Demo Disc series (magpi.cc/hauntedps1).

Although Unity and Unreal Engine don’t run on Raspberry Pi, the impressive Godot engine does (magpi.cc/godot), albeit with a few limitations.

We’ve got you covered if you want to begin your journey into retro game programming right here with PyGame Zero (magpi.cc/pygamezero).

Meanwhile, for those who want to get a feel for techniques from the past and present, BBC BASIC (magpi.cc/bbcbasic) is available for Raspberry Pi, while developing with using C++ and SDL is far more fun that it has any right to be.

K.G. Orphanides

K.G. Orphanides variously makes weird games, DIY emulation consoles, music, documentation, lost software archives and, quixotic builds.

@KGOrphanides
American Raspberry Pi Shop

- Displays
- HATs
- Sensors
- Cases
- Arcade
- Swag
- Project Kits
- Cameras
- Power Options
- Add-on Boards
- Cables and Connectors
- GPIO and Prototyping

Partner and official reseller for top Pi brands:

- adafruit
- sparkfun electronics
- PIMORONI
- HiFiBerry

Price, service, design, and logistics support for VOLUME PROJECTS

USA PiShop.us

Canada PiShop.ca

Raspberry Pi APPROVED RESELLER
HIGHPI PRO

The new case from the HiPi.io team

- Rapid tool-free assembly and disassembly
- Large internal volume for HATs
- Compatible with Pi 2/3/4
- Multiple lid options
- Passive & Active Cooling options
- Secure microSD card cover
- VESA mount support
- Molding-configurable output ports customizable for volume orders
- Printed logo for your branding

Available at these great Pi stores:

- PiShop.us
- PiShop.ca
- PiShop.mx
- ThePiHut
- Pi-SHOP.ch
- Welectron.

Contact your favorite Pi store if it’s not listed here