Practical Electronics

The UK's premier electronics, computing and maker magazine

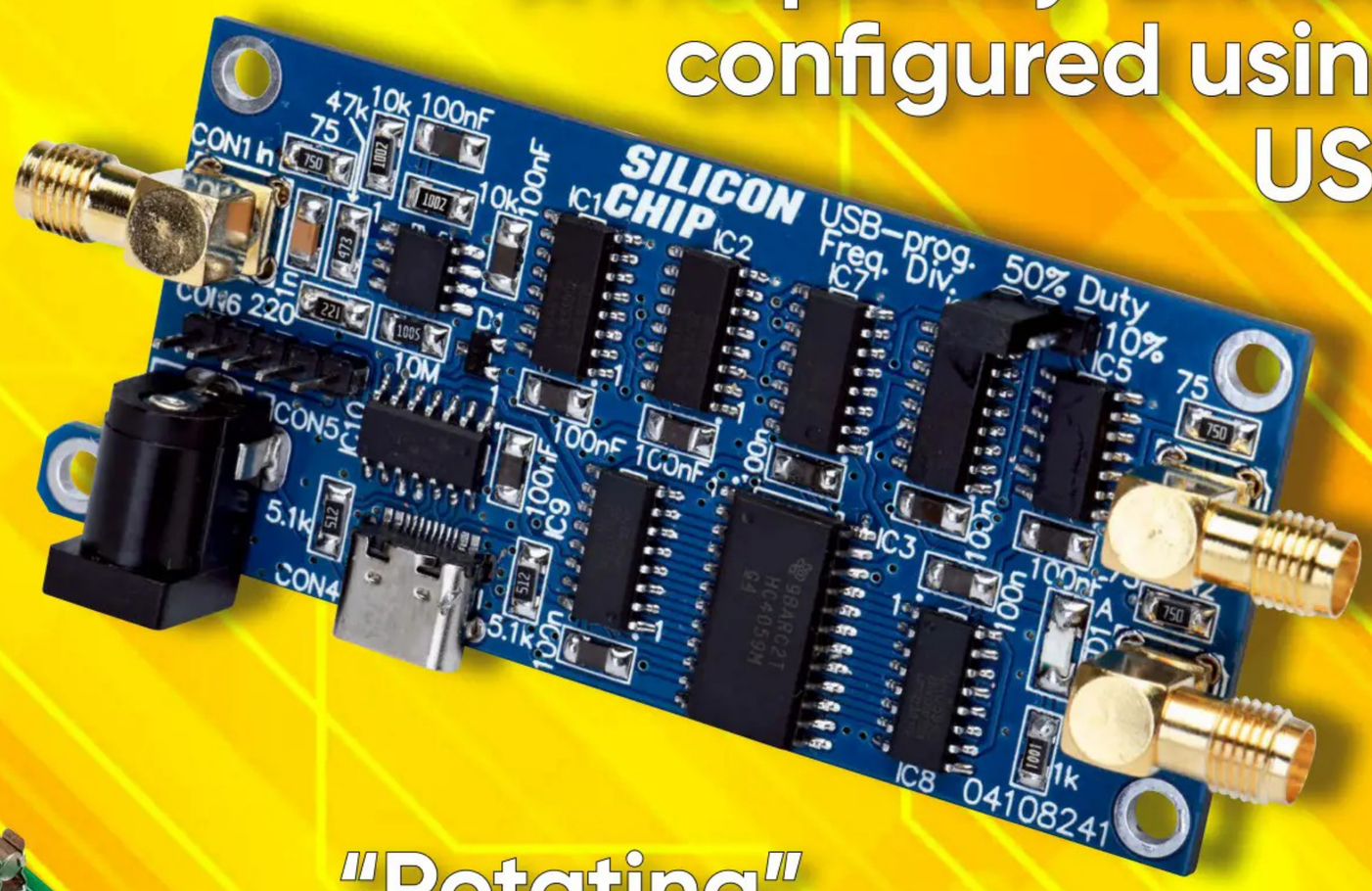
Max's Cool Beans
Updating the display after button presses

Teach-In 12
Part 5: all about radio antennas

Audio Out
Silent stomp switch circuit

Circuit Surgery
Op amp capacitive loading and stability

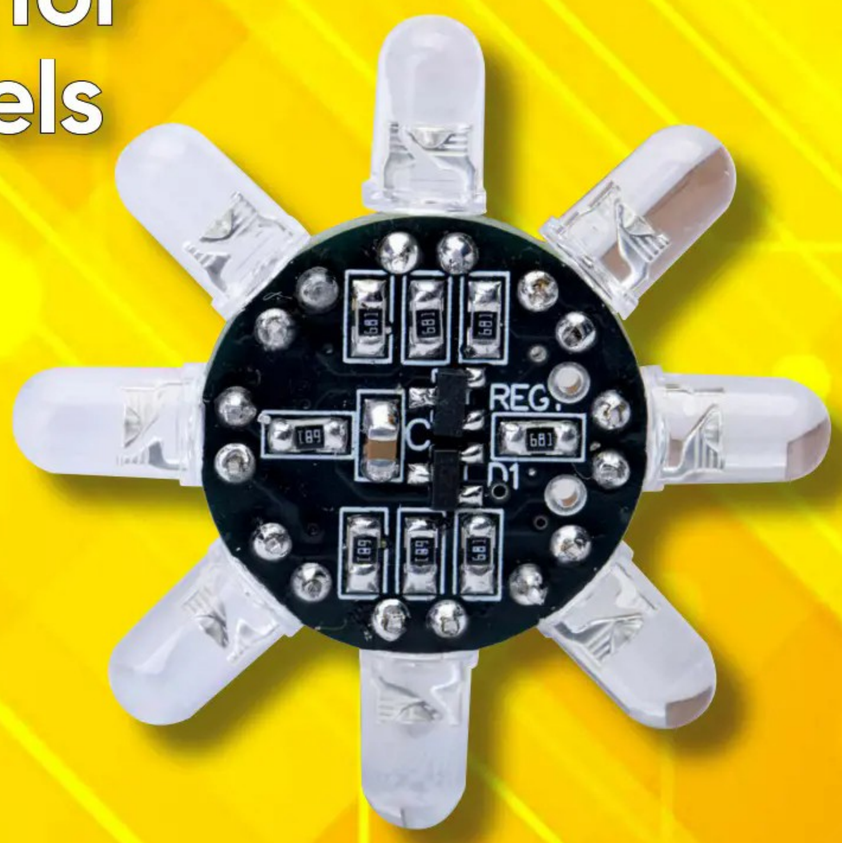
A Frequency Divider configured using USB



Finishing the Power LCR Meter



"Rotating" Light for Models



PLUS!

Techno Talk – physics lessons with a smartphone

Data Centres – what happens behind the scenes

The Fox Report – Windows network file sharing tips and hints

Mar 2026 £5.99



PCBWay

Beyond Standard —

Discover PCBWay's Advanced PCB Lineup

About PCBWay:

With dedicated workshops for advanced PCBs, PCBWay offers full-spec, high-precision, and large-scale production. We ensure customers are supplied with PCBs that meet stringent quality standards at the lowest cost.

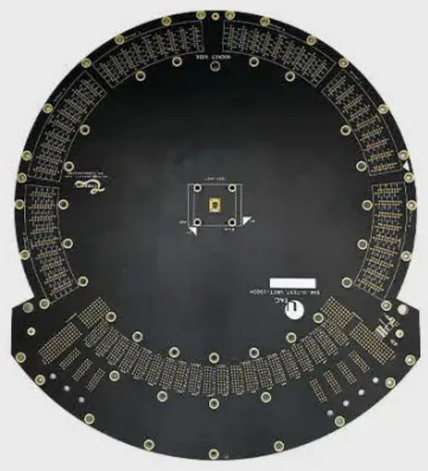


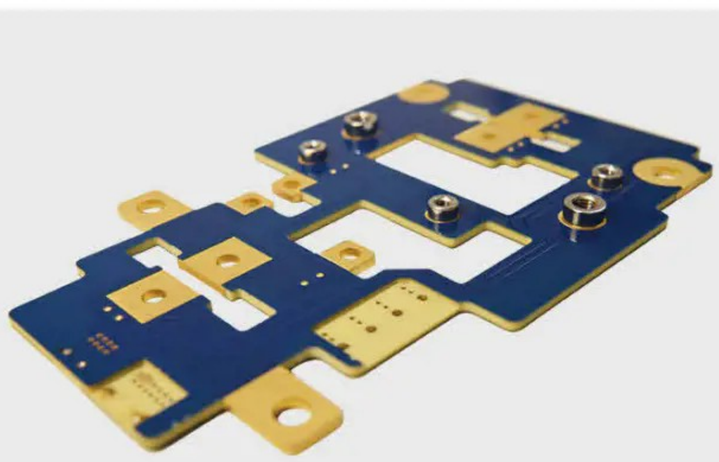
HDI PCB

- 4-60 layers
- FR-4, High Tg, Halogen-free, PTFE
- Finished Copper: 1/3OZ–8OZ
- Min Track/Spacing (inner): 2/2mil (H/HOZ base copper)
- Min BGA Pitch: 0.35mm

High Frequency PCB

- RO4003C, RO4350B, Ro3003, Ro3010, RT5880
- Min. Track/Spacing: 2mil
- Min. Hole Size: 0.15mm
- Finished Cooper: 1-2oz
- Board Thickness: 0.2–3.2mm





Other PCBs with Special Process

- Embedded Copper PCB
- Backplane PCB
- Cavity PCB
- White/Partially Hybrid PCB
- Black Core PCB (After-dark)

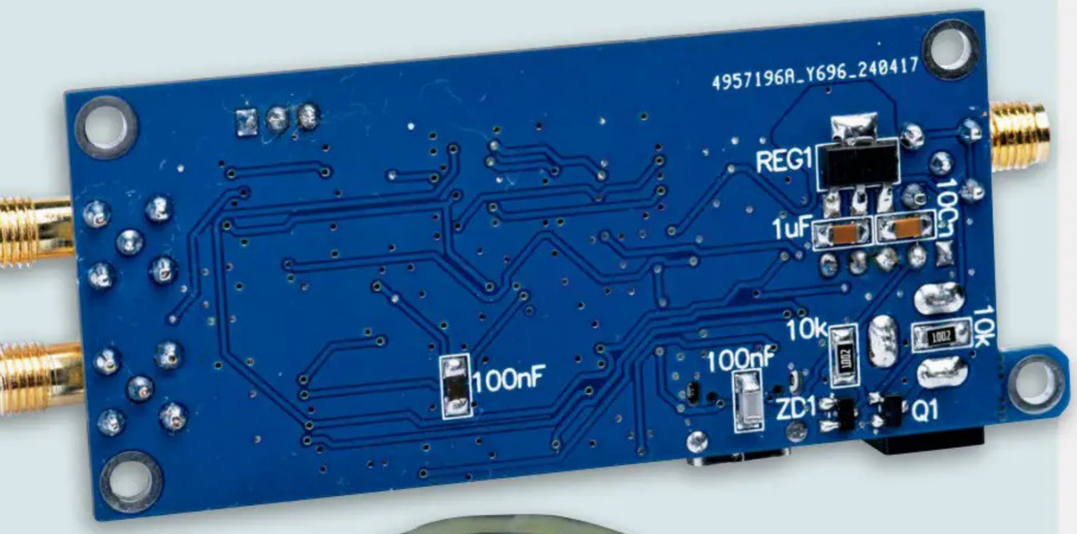


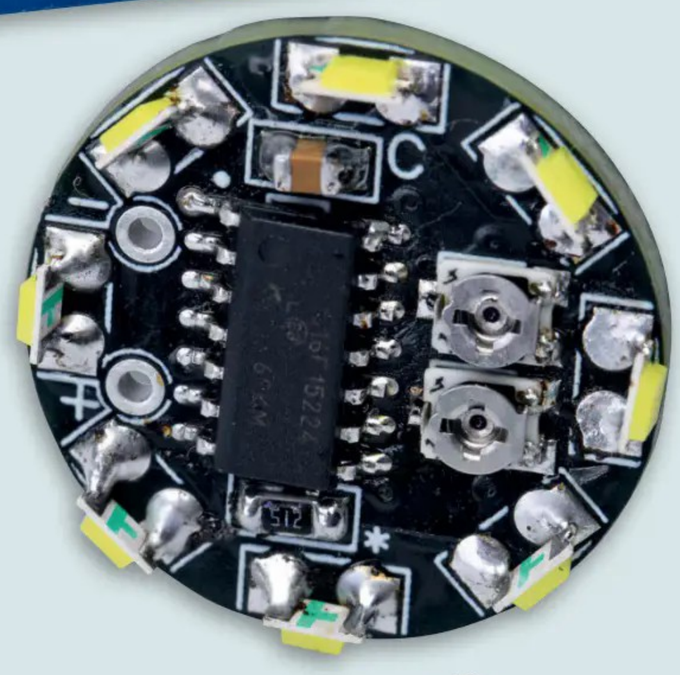
www.pcbway.com service@pcbway.com

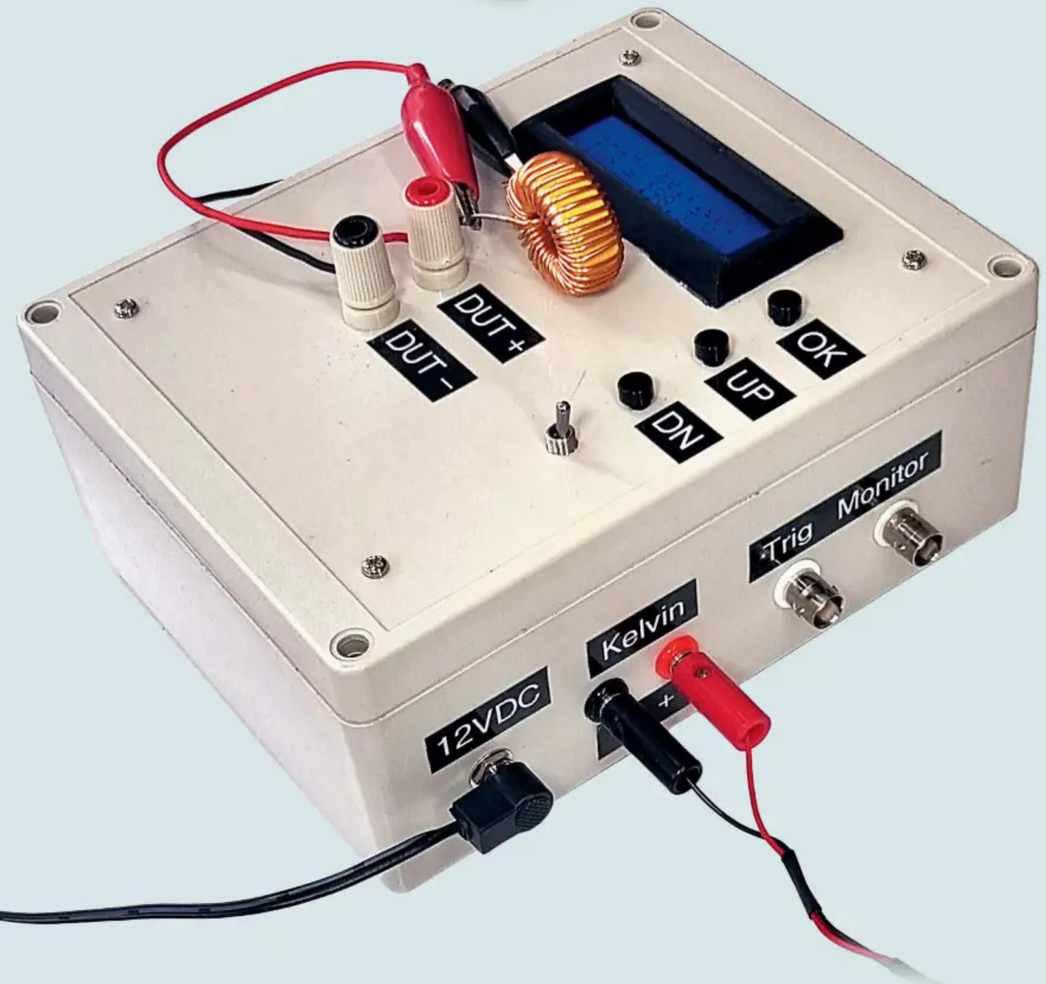
Practical Electronics

Volume 55. No. 03
March 2026
ISSN 2632 573X

Contents







Projects and Circuits

USB-Programmable Frequency Divider/Counter by Nicholas Vlnen **12**
This small PCB doesn't cost a lot but can reduce the frequency of an incoming signal or pulse train by a factor of between 3 and 21,327,000 that's easily configured over a USB serial port. It can also provide basic frequency measurements.

Rotating Light for Models by Nicholas Vinen **42**
Here's a simple circuit that has various applications, such as for a model lighthouse, or as a siren on top of a model emergency vehicle. It sequences eight LEDs, using PWM brightness control, to form a convincing imitation of a rotating light.

Power LCR Meter, part two by Phil Prosser **70**
This isn't just another LC meter; it can deliver up to 30A to determine how an inductor behaves as its core starts to saturate. It can also measure very high capacitances and very low resistances. This article covers assembly, testing and use.

Series, Features and Columns

Audio Out by Jake Rothman **4**
Continuing on the subject of musical instrument pedals and effects, here is a stomp switch that's as quiet as a ninja wearing sneakers.

The Fox Report by Barry Fox **11**
Sharing files over a network in Windows 11 isn't easy, so I made a step-by-step guide.

Teach-In 12.5: the World of Wireless by Mike Tooley **20**
This month, we look in detail at those all-important radio parts, antennas. There's a lot more to an antenna than just a random length of wire. We look at many different types, their advantages and disadvantages, and how to make them.

Circuit Surgery by Ian Bell **32**
Measuring the frequency response of a circuit using a PC sound card, part 6: op amp capacitive loading. The amount of capacitance at the output of an op amp impacts stability and phase margin. We explain how to figure out if it will be stable.

Max's Cool Beans by Max The Magnificent **48**
Weird & Wonderful Arduino Projects, part 15: making cartridges. We are now making great strides in getting the Games Console ready to play games. That includes updating the LED display based on what buttons we press. We're so close, I can taste it!

Techno Talk by Max the Magnificent **56**
The dark arts of technology. From physics experiments to smartphone sensors, reverse engineering, obfuscation and generative AI—an exploration of how abstraction explains, empowers and occasionally conceals modern technology.

Inside Data Centres by Dr David Maddison **58**
This month we write about an important piece of mostly invisible internet infrastructure: data centres. They are rarely seen and little is known of them by the general public.

Regulars and Services

Editorial **2**
Quantity kinds, tagging and units
Reader services – Editorial and Advertising Departments **2**
Subscribe to Practical Electronics and save money **3**
Electronics Teach-In Bundles **40**
Practical Electronics Back Issue DOWNLOADS – 2024 now available! **41**
Practical Electronics PCB Service **69, 78**
PCBs for Practical Electronics projects
Classified ads and Advertiser index **79**
Next month! – highlights of our next issue of *Practical Electronics* **80**



Made in Australia & the UK.

Written in Britain, Australia, the USA and Ireland.

Read everywhere.

© Silicon Chip Publications Pty Ltd 2026

Copyright in all drawings, photographs, articles, technical designs, software and intellectual property published in *Practical Electronics* is fully protected, and reproduction or imitation in whole or in part are expressly forbidden.

The April 2026 issue of *Practical Electronics* will be published on Thursday, 5th of March, 2026 (see p80).

Renewals paid by PayPal, Visa and Mastercard are now handled via our website: www.electronpublishing.com

Practical Electronics

Volume 55. No. 03
March 2026
ISSN 2632 573X

Editorial

Editorial offices
Electron Publishing (Australia)

Tel +61 2 9939 3295
Email pe@pemag.au
Web www.electronpublishing.com

Address mail to:
Electron Publishing (Australia)
PO Box 194, Matraville NSW 2036
Australia

Advertising enquiries
+61 2 9939 3295 pe@pemag.au

Editor	Nicholas Vinen	
Publisher	Nicholas Vinen	
Digital subscriptions	Stewart Kearn	**Tel 07918 614662**
Online Editor	Alan Winstanley	
Web Systems	Kris Thain	
Production	Bao Smith	
Technical staff	Tim Blythman, John Clarke	

Print subscriptions
Practical Electronics Subscriptions
PO Box 6337
Bournemouth BH1 9EH **Tel** 01202 087631
United Kingdom **Email** pesubs@selectps.com

Technical enquiries
We regret that technical enquiries cannot be answered over the telephone. We are unable to offer any advice on the use, purchase, repair or modification of commercial equipment or the incorporation or modification of designs published in the magazine.

Questions about articles or projects should be sent to the editor by email: pe@pemag.au

Projects and circuits
All reasonable precautions are taken to ensure that the advice and data given to readers is reliable. We cannot, however, guarantee it and we cannot accept legal responsibility for it.

Some projects and circuits published in *Practical Electronics* employ voltages that can be lethal. Do not build, test, modify or fix any mains-powered equipment unless you fully understand the safety aspects involved and you use an RCD (GFCI) adaptor.

Component supplies
Silicon Chip Publications may offer kits or other parts for making our projects, but not in all cases. When kits are not available, readers will need to find and source parts themselves. We advise readers to check that all parts are still available before commencing any project in a back-dated issue.

Advertisements
Although the proprietors and staff of *Practical Electronics* take reasonable precautions to protect the interests of readers by ensuring as far as practicable that advertisements are bona fide, the magazine and its publishers cannot give any undertakings in respect of statements or claims made by advertisers, whether these advertisements are printed as part of the magazine, or in inserts. The Publishers regret that under no circumstances will the magazine accept liability for non-receipt of goods ordered, or for late delivery, or for faults in manufacture.

Transmitters/bugs/telephone equipment
We advise readers that certain items of radio transmitting and telephone equipment which may be advertised in our pages cannot be legally used in the UK. Readers should check the law before buying any transmitting or telephone equipment, as a fine, confiscation of equipment and/or imprisonment can result from illegal use or ownership. The laws vary from country to country; readers should check local laws.

Quantity kinds, tagging and units

I've previously mentioned the importance of units and why it's a bad idea to leave them off numbers. For example, if I'm calculating the power in a resistor, I'd write 5V × 2A = 10W, not 5 × 2 = 10W, because you don't get a value in watts by multiplying two plain numbers. The result is in watts because you multiplied a voltage by a current.

This concept is well-established across the sciences. However, something that's talked about much less is the idea of quantity kinds – that is, the type of thing a number represents, even when it has no physical units. Take pH, a measure of acidity or alkalinity. It's a dimensionless quantity, because it's derived from a ratio of ion concentrations. So we might say that pure water has a pH of 7, but not "7 pH" because pH isn't a unit.

Similarly, angles are technically dimensionless (although radians are treated as an SI unit), and decibels are a logarithmic ratio without units. Other examples of unitless quantities include refractive index, relative humidity and coefficients of friction.

Despite all these being unitless, it would be nonsense to add 90° to the pH of water, or to add the number of decibels of a CD's dynamic range to the refractive index of the laser lens that reads it, and then to the coefficient of friction of the spindle motor. These values are all unitless numbers but they represent very different things.

One of the great advantages of having units is that they prevent this sort of mistake. You can't blindly compute 9.81m/s + 25°C ÷ 3kg without realising something's off – unless you have a meaningful way to convert or transform the units involved (you probably don't). But with unitless numbers, there's no such safety net.

This is where quantity kinds become useful – a way to tag a number with what it represents, even when there's no physical unit. It makes me wonder whether scientists should adopt quasi-units or semantic tags for every distinct kind of dimensionless quantity, to make it clearer how they can (or can't) be combined.

We already do something like this in electronics. Consider decibels: we often write dBV for a measurement relative to 1V, dBu for 775mV RMS, dBm for 1mW into a reference load, dB SPL for sound pressure relative to 20µPa, and dBi for antenna gain relative to an isotropic radiator. These aren't units in the strict sense, but they tag the number with its meaning and help prevent confusion.

I think this is a good idea, and it might be worth extending it to all dimensionless quantities more broadly. I'm not entirely sure how that could be implemented, but it seems like a fruitful idea.

Nicholas Vinen, Electron Publishing (Australia)*
* a division of Silicon Chip Publications Pty Ltd.

Practical Electronics

UK readers SAVE £1 on every issue

SUBSCRIBE NOW!



01202 087631

Take out a one-year subscription and save more than £10 over the year.
Even better – save over £1 per issue if you subscribe for two years – a total saving of £26.

Overseas rates represent exceptional value

You also:
- **Avoid any cover price increase for the duration of your subscription**
- **Get your magazine delivered to your door each month**
- **Guarantee a copy, even if your local newsagent sells out**

Order by:
- **Phone or post with a cheque, postal order or credit card**
- **Or order online at: www.electronpublishing.com**

Practical Electronics

SUBSCRIPTION PRICES

Subscriptions for delivery direct to any address in:

UK: 6-months £33.99, 12-months £59.99, 24-months £114.99

Europe Airmail: 6-months £38.99, 12-months £69.99, 24-months £129.99

Rest Of The World Airmail: 6-months £46.99, 12-months £84.99, 24-months £164.99

Cheques (in **£ sterling only**) payable to ***Select Publisher Services Ltd*** and sent to: Practical Electronics Subscriptions, PO Box 6337, Bournemouth BH1 9EH, United Kingdom

Tel: 01202 087631
Email: pesubs@selectps.com

Also via our online shop at**: www.electronpublishing.com**

Subscriptions start with the next available issue. We accept MasterCard or Visa.
(For past issues see the Back Issues page.)

DIGITAL SUBSCRIPTIONS

Electronic subscriptions are available from £29.99 for 12 months, more details at: **www.electronpublishing.com**

SUBSCRIPTION ORDER FORM

☐ **6 Months:** UK £33.99, Europe £38.99 (Airmail), Rest of the World £46.99 (Airmail)

☐ **1 Year:** UK £59.99, Europe £69.99 (Airmail), Rest of the World £84.99 (Airmail)

☐ **2 Years:** UK £114.99, Europe £129.99 (Airmail), Rest of the World £164.99 (Airmail)

To: Practical Electronics Subscriptions
PO Box 6337
Bournemouth BH1 9EH
United Kingdom

Tel: 01202 087631

Email: pesubs@selectps.com

I enclose payment of £ .....
(cheque/PO in £ sterling only)
payable to ***Select Publisher Services Ltd***

☐ Please charge my Visa/Mastercard

My card number is: .....
Please print clearly, and check that you have the number correct

Card Security Code ..... Valid From Date.....
(The last 3 digits on or just under the signature strip)

Card Ex. Date .....

Name .....

Address .....

Post code ..... Tel. .....

Email .....

Subscriptions start with the next available issue.

AUDIO OUT

By Jake Rothman

A silent stomp switch

All the previous guitar pedals I've described over the last few months have had a large mechanical latching stomp switch with three double-pole/changeover sections, like the one shown in Photo 1. That setup is particularly neat and I want to do something similar for my future pedals.

In case it isn't obvious, it's called a stomp switch because you press it with your foot, and depending on how passionately you are playing, you might be pressing it rather vigorously! Hence, such switches must be robust.

This is an expensive and elusive component, which is prone to failure, especially regarding the latching mechanism.

The way around this, used by volume pedal manufacturers such as Boss, is electronic switching using a cheap single-pole momentary switch.

It usually works out cheaper as well, but alas not for the home constructor who cannot benefit from quantity discounts and automated assembly.

The Boss circuit used in the FZ-2 pedal forms the basis of the design given here. It can be retrofitted into existing designs if required, space permitting, as shown in Photo 2.

Silence is golden

All mechanical latching switches, especially stomp switches in metal boxes, produce a loud acoustic crack when pressed. Some guitarists find this noise reassuring, although it should be obvious if the effect has been engaged, especially if it's fuzz.

These switch cracks can spoil a vocal recording if somebody is singing and playing at the same time. Grindle, the board designer and pedal maker, also informs me that there is a type of bedroom musician who just wears socks! They like the feel of a "soft" switch.

[Editor's note – I attended a performance by top UK musician Steven Wilson recently (who plays many instruments, including electric guitar and bass) with his touring band, and it was interesting to note that he was barefoot the entire time on stage! I bet he prefers a soft stomp switch…]

Switching transients

Switching audio waveforms usually produces audible artifacts. Mechanical switches 'slice' the waveform which, along with switch bounce, produces high-frequency clicks. Electronic switches tend to make a 'soft LF thump' with a bit of delay as they 'fade' in/out. This is because most use JFET switching elements with a ramped control voltage. This design is no exception.

Distortion and level

Electronic switches have more distortion, and can only handle signal swings up to a couple of volts below the power rail. The low-order

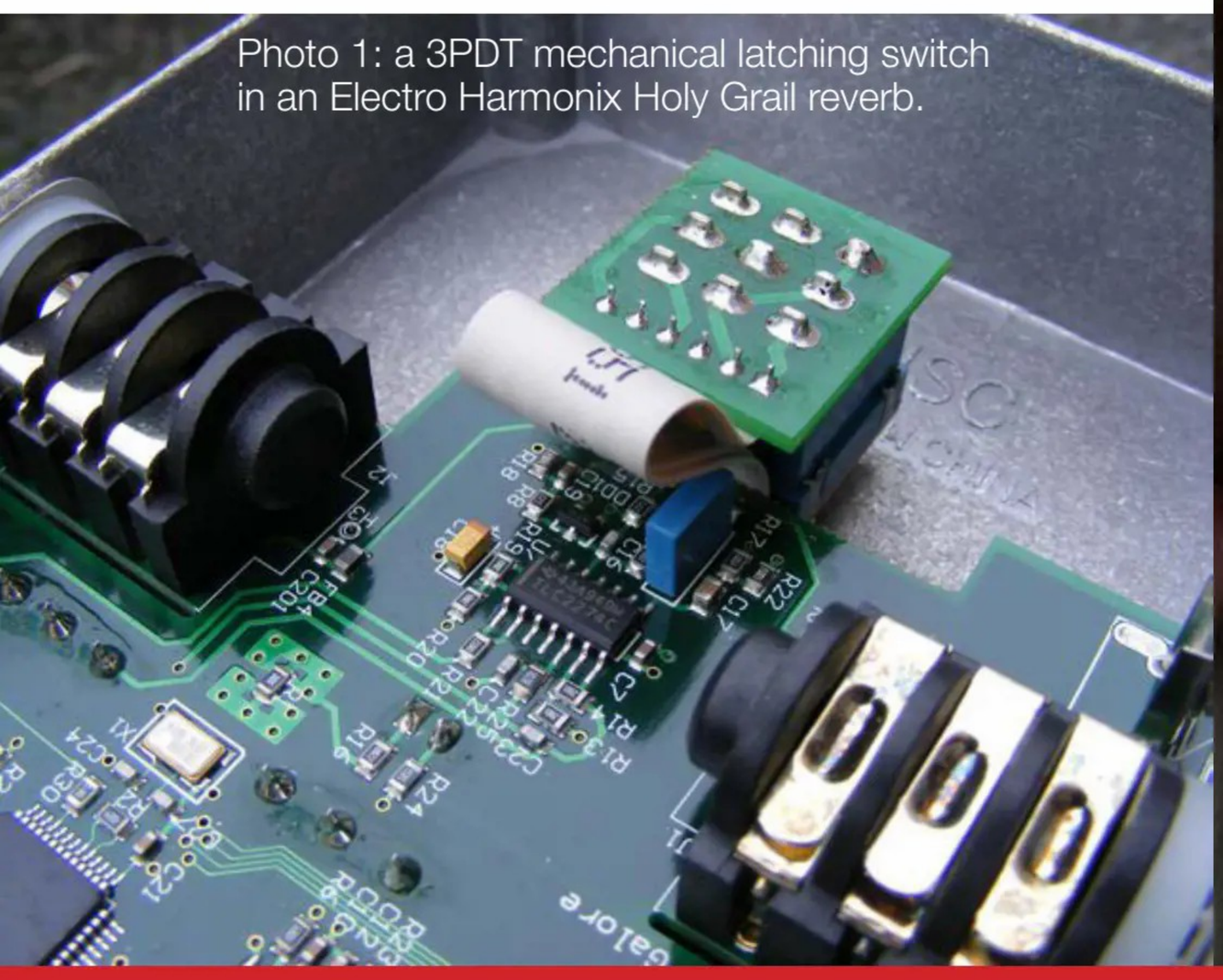


Photo 1: a 3PDT mechanical latching switch in an Electro Harmonix Holy Grail reverb.

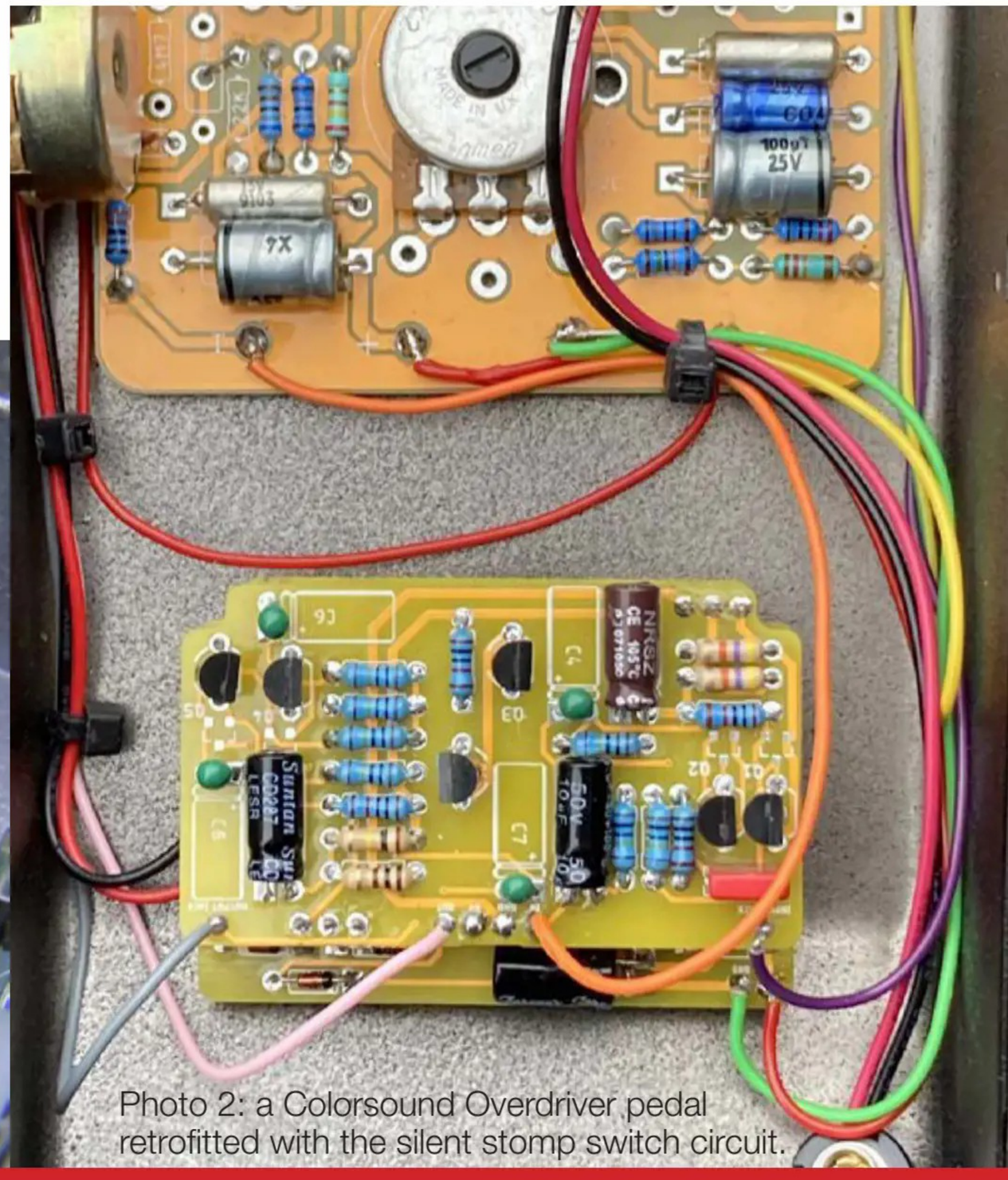


Photo 2: a Colorsound Overdriver pedal retrofitted with the silent stomp switch circuit.

harmonic distortion of around 0.1% generated by the JFETs is no problem with guitars, and is possibly desirable, depending on the effect the musician is going for.

Hard-bypass switching

Mechanical switches can offer the advantage of not stopping the sound if the power goes off, if wired with an extra input switch section, called hard-bypass. Electronic switches need power to pass a signal regardless of their state. Hard-bypass avoids frantic unplugging of a faulty pedal while playing if the battery dies.

Since JFETs are conducting when biased at 0V, it may be possible to design an electronic switch with hard-bypass. The trouble is that buffers are needed as well, and they won't work without power.

The system

It still shocks me how much electronics is needed to replace a simple analog mechanical switch. Even a guitar pedal stomp switch needs eight transistors, or a few chips, to do the job properly. This is more than a typical fuzz box. The thing to remember is that commodity 'jelly bean' components are very cheap, especially in surface-mount packages.

This circuit also has a JFET input buffer and an output buffer, which will enhance the performance of most pedals. This was the case with the Colorsound Overdriver, which has an input impedance of 50kΩ and an output impedance of 25kΩ, both non-optimum values for guitar use. These are improved to 1MΩ for the input impedance and 200Ω of output impedance with this circuit.

The block diagram is shown in Fig.1, with the system being split into two PCBs, one for the switch latching circuit and the other for the audio circuit. This partitioning allows each board to be used in its own right in other systems. It also enables the boards to be stacked on top of each other, with the switch underneath, for compactness.

Switch latching circuit

The circuit for this board is shown in Fig.2. The switch latching part of the circuit is based on the classic two-transistor discrete flip-flop or monostable. This instantly recognisable configuration, using cross-coupled feedback paths from one transistor's collector to the other transistor's base, formed the basis of many late-fifties and early-sixties computers.

My grandad used to get me scrap computer boards full of these flip-flops from Metrovicks back in 1971. They held lots of Mullard OC42 germanium transistors with long leads, worth money today.

If resistors R17 and R20 are replaced with capacitors, the physics teacher's favourite circuit results: the flashing light multi-vibrator. One problem with this symmetrical circuit is that its initial state upon switch-on is undefined. Due to transistor variations, such as differences in turn-on voltage (V_{be}), one side will usually become active before the other, but you won't know which.

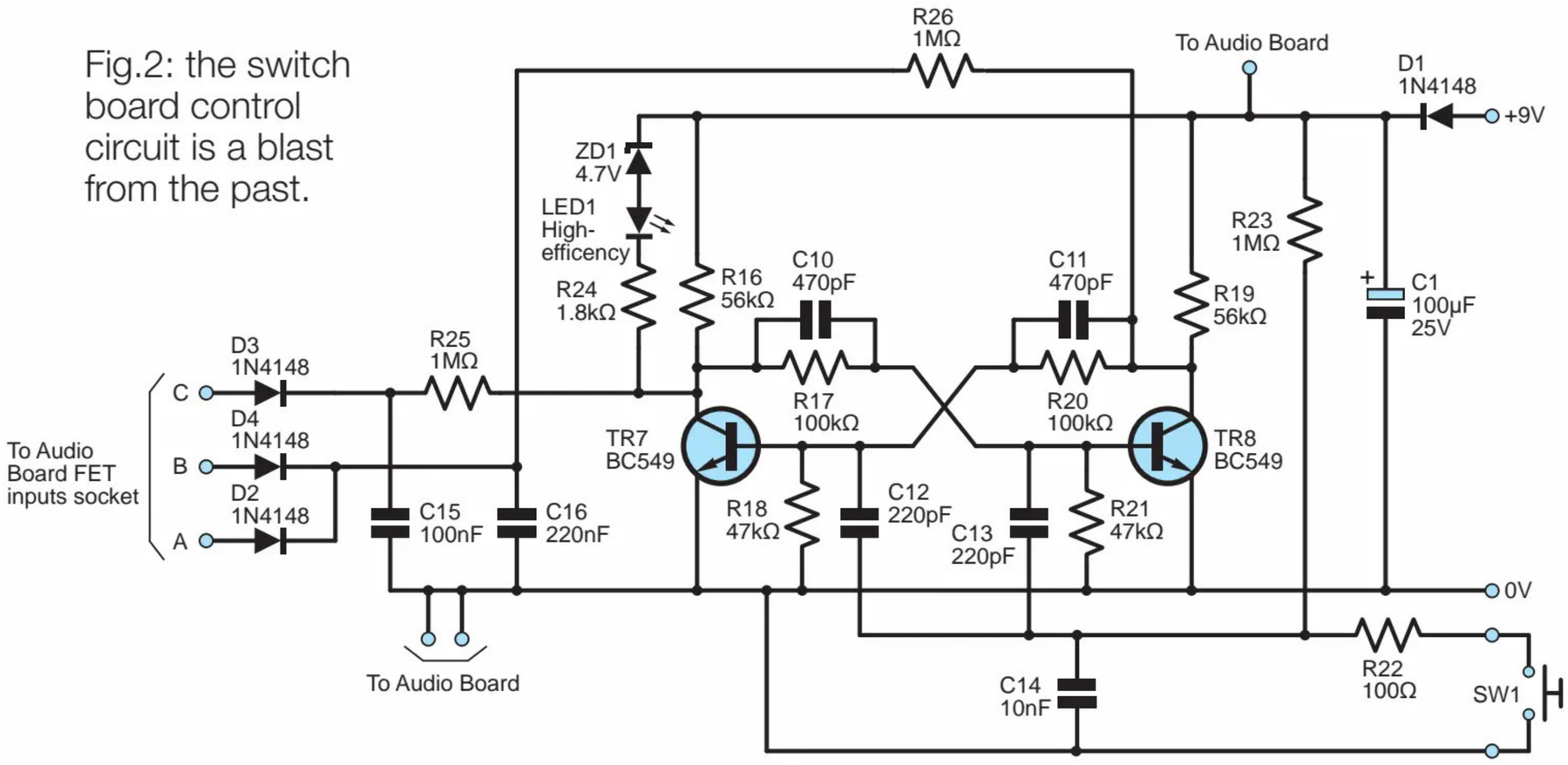
One way to define it is to add a deliberate imbalance or a start-up circuit. In this case, because one side (TR7) drives the LED, and it has 10× more collector current, it needs more voltage to switch on, resulting in the other side (TR8) switching on first. It's theoretically not 100% guaranteed, though.

Guitarists generally prefer the circuit to start up with no effect engaged (with the LED off) until the switch is pressed, as happens here. If you have trouble, the circuit can be deliberately unbalanced by altering the values of the trigger capacitors C12 and C13.

C10 and C11 are speed-up capacitors, which make the triggering more definite. Going to the switch part of the circuit, C14 is charged up by R23. When pressed, the switch quickly discharges C14 to trigger the circuit. R22 is needed to limit the current through the switch contacts.

R25, R26, C15 and C16 form ramp networks to smooth the control voltage to the

Fig.2: the switch board control circuit is a blast from the past.



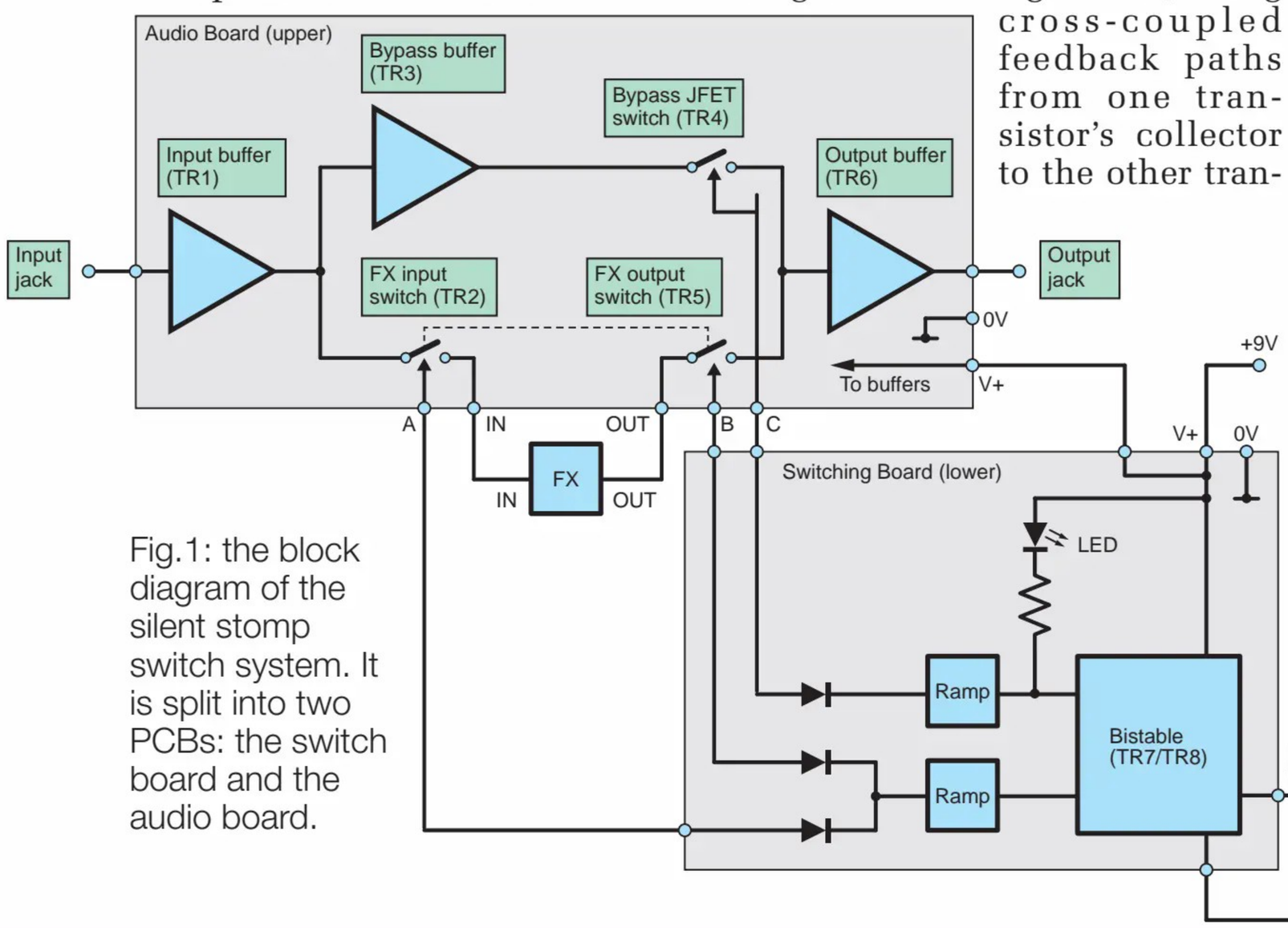


Fig.1: the block diagram of the silent stomp switch system. It is split into two PCBs: the switch board and the audio board.

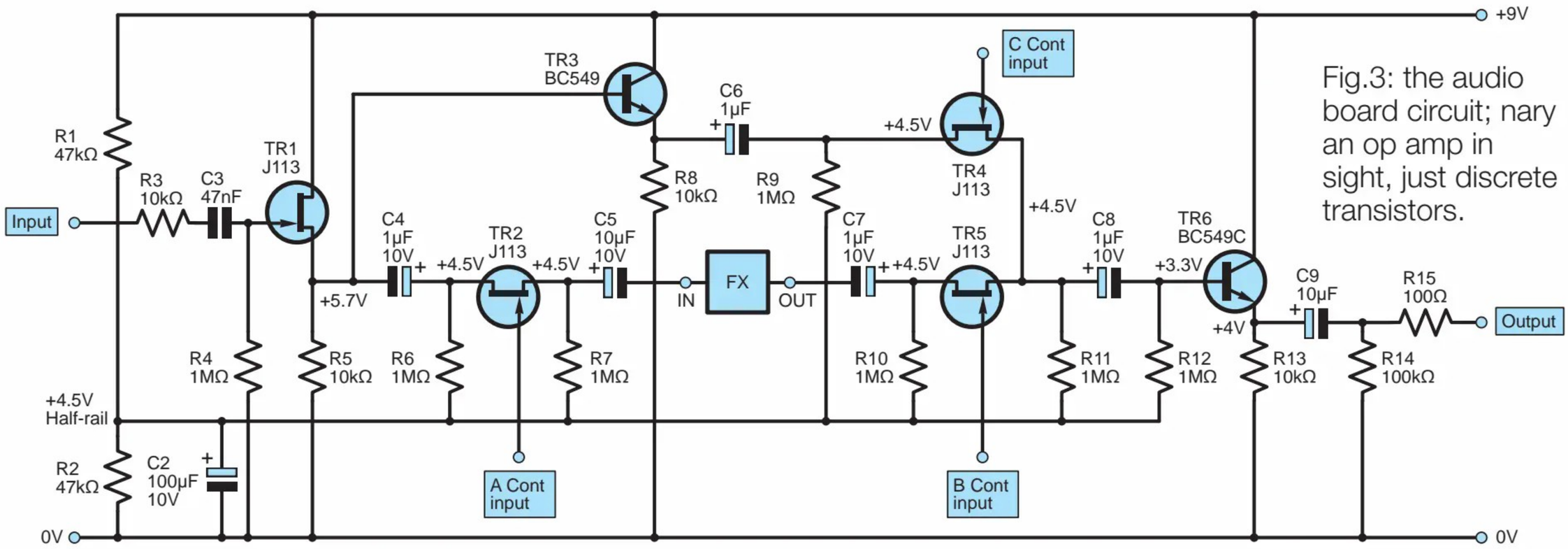


Fig.3: the audio board circuit; nary an op amp in sight, just discrete transistors.

switching JFETs in the audio board, preventing clicks. Interestingly, having two different ramp times for the bypass JFET (TR4) and the effect JFETs (TR2 and TR3) gave minimum thump.

Diodes D2, D3 and D4 protect the JFET gates from possibly being destructively forward-biased under fault conditions. They also seem to reduce distortion, especially if used with a boot-strapping resistor from drain to gate, but I've not tried this in a single-rail circuit (such as this) yet.

Power supply

The circuit has the basic power supply housekeeping, consisting of the main decoupling capacitor C1, which is also used to feed the audio board. Reverse polarity protection is provided by diode D1.

This is not really necessary, but I've known guitarists stick all sorts of power supplies into pedals, including a 33V Hewlett Packard printer supply of reverse polarity, causing the decoupling capacitor to explode!

Audio circuit

This board contains the half-rail bias generator, consisting of the divider formed by resistors R1 and R2, decoupled by capacitor C2. This part of the circuit is shown in Fig.3. Since the circuit uses a relatively low voltage (+9V) rather than the more normal audio ±15V (30V total), we have to use a lower voltage pinch-off (V_p) JFET for the audio switching.

The normal J111 and J112 used in professional audio have too high a maximum V_p to work. So, we use the J113 here, which is very cheap and is cut off at a maximum of -3V (relative to the half-supply rail). This allows the circuit to work down to 6V, where the battery is getting a bit flat.

It is an inferior switching device, having an on resistance (Ron) of around 100Ω. That's three times more than the J111, which means three times more distortion, but this is not a Hi-Fi application, so it's no problem.

The original Boss and Roland pedals used the 2SK30, which is now rare and expensive. A strange choice, since its main attribute is a high V_{ds} rating of 50V, but the V_p is also low at -0.5 to -2V, while Ron is high at 300Ω.

JFET series switches give acceptable results when driven from a low impedance and loaded by a high impedance. Buffer transistors TR1, TR3 and TR6 provide these conditions. TR1 uses a JFET to provide a better match for the guitar input. TR6 should be a high-h_{FE} (>500) device, such as a BC549C, to maximise the input impedance and minimise the voltage drop on R12.

It would offer a slight improvement to use another JFET, but they cost ten times as much. It may look as if the buffer (TR3) on the bypass path is superfluous, but it is necessary to provide isolation to prevent positive feedback occurring around high-gain non-inverting effects, such as Big Muff fuzz units, during switching.

With some unity-gain effects, such as chorus, TR3 can be linked out and the emitter resistor R8 omitted. Another bit that is optional in some effects is the input switch TR2. It does little damage if the input to, say, a phaser is left connected during bypass. It is a different matter with a fuzz unit, though, because the harsh square wave can couple through into the clean bypass.

Further simplification

Boss did many variations of their switching circuit. The one used in some chorus pedals like the CE-2 is shown in Fig.4; it is greatly simplified, using only two JFETs and an op amp. It was previously shown in Audio Switching Part 4, September 2024, in Fig.81 on page 50.

The circuit still needs the switch board to provide the JFET control voltages, however. It will be worth getting an alternative audio board together for this in a later pedal.

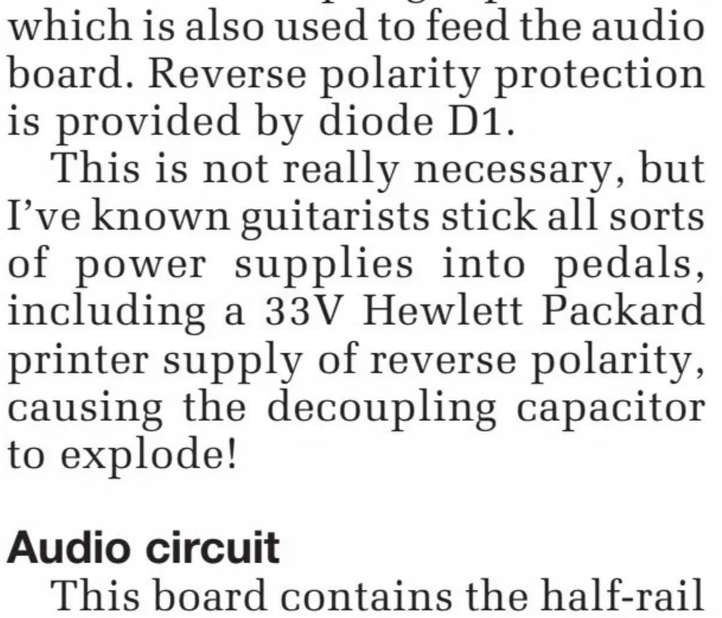


Photo 3: this cheap momentary push button from Rapid saves almost £2.50 compared to the 3PDT switch.



Photo 4: this Tayda switch is rugged and looks good. The switch board has a special hole to accommodate it.

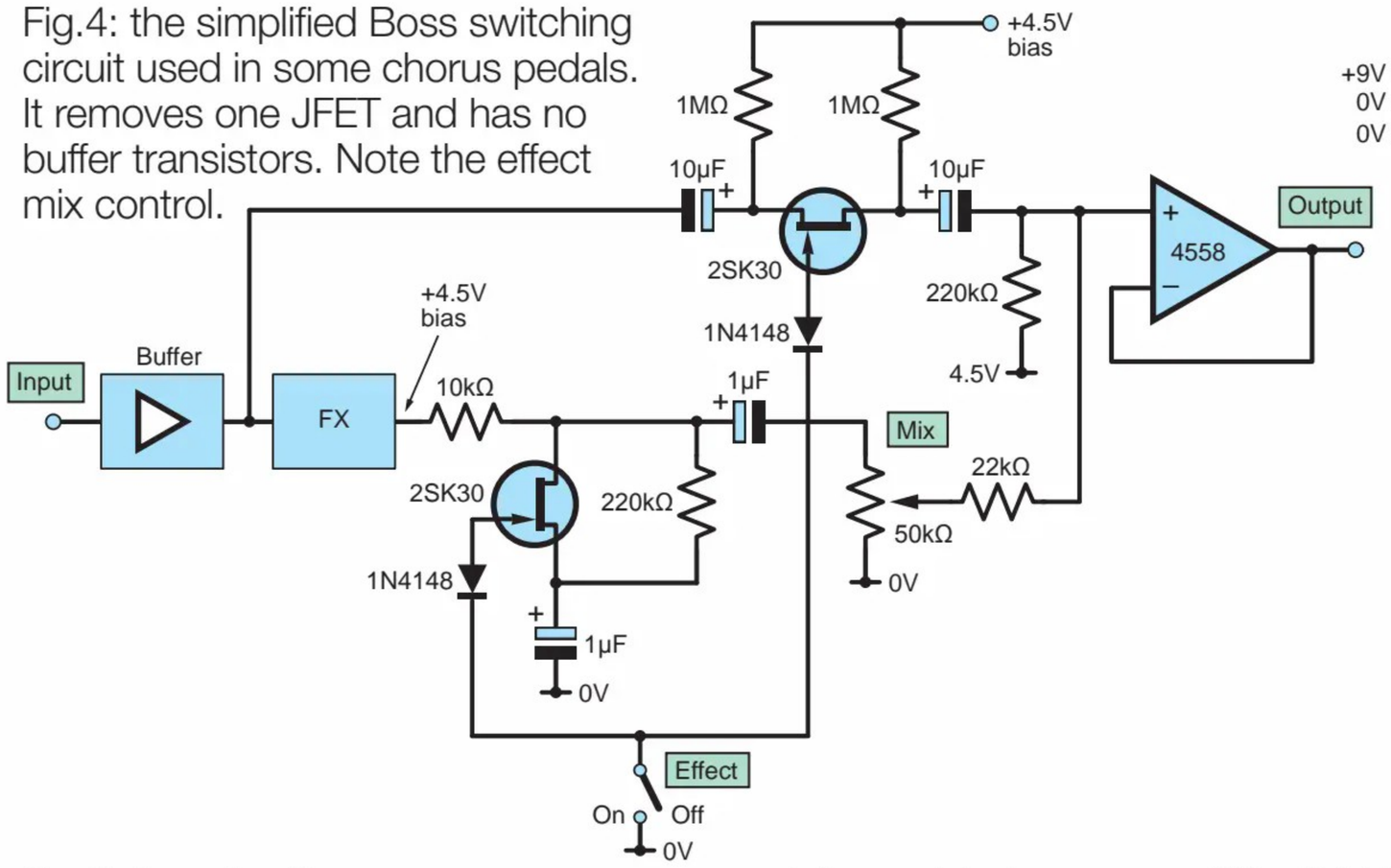


Fig.4: the simplified Boss switching circuit used in some chorus pedals. It removes one JFET and has no buffer transistors. Note the effect mix control.

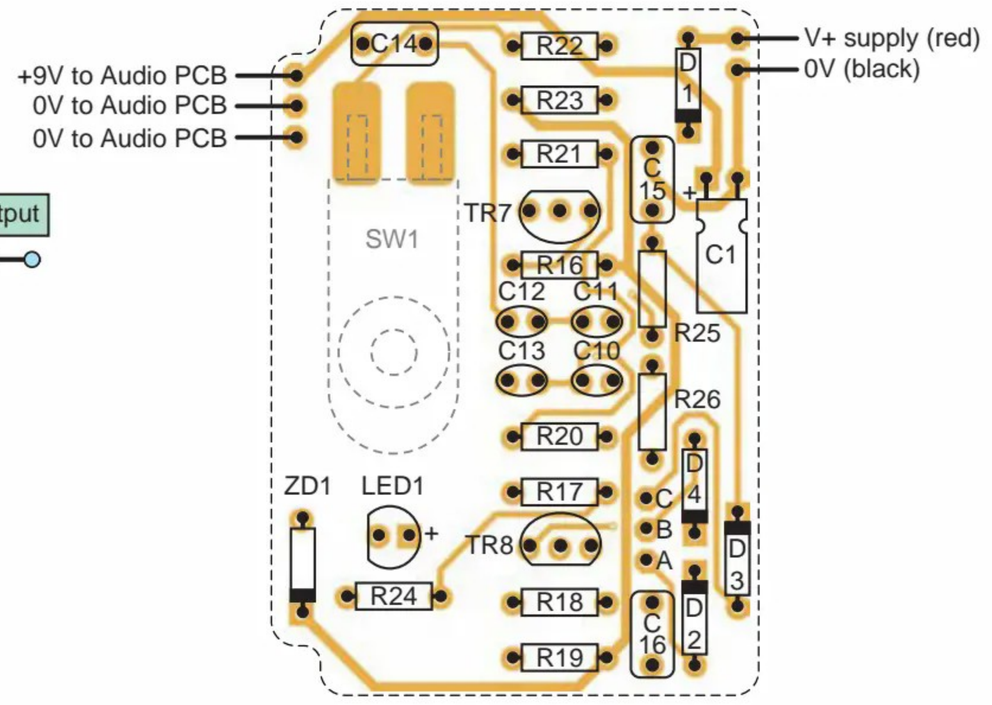


Fig.5: the switch board overlay. The odd-shaped hole took some work.

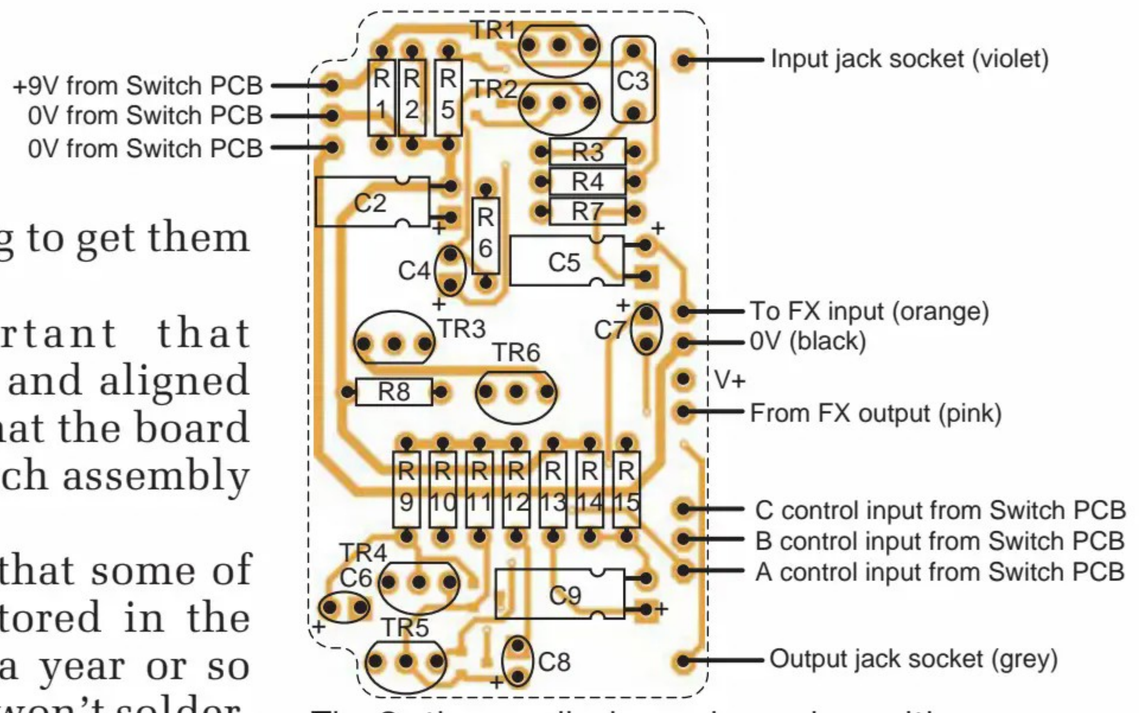


Fig.6: the audio board overlay with optional SMT pads for the JFETs.

Switch selection

One has to be careful in selecting the switch. I bought some Alpha momentary foot switches from Rapid (78-0743) that made a loud click when pressed, defeating the whole point.

It is possible to save money by using large generic momentary push buttons, such as the SCI R13-507MA (Rapid 78-0183), shown in Photo 3, which cost £0.52. They will need an extra nut to make them tall enough for foot operation, though.

The best one is the A-1091 from Tayda, which looks like a proper stomp switch, shown in Photo 4. Sadly, it's relatively expensive for a momentary switch, at around £1.97.

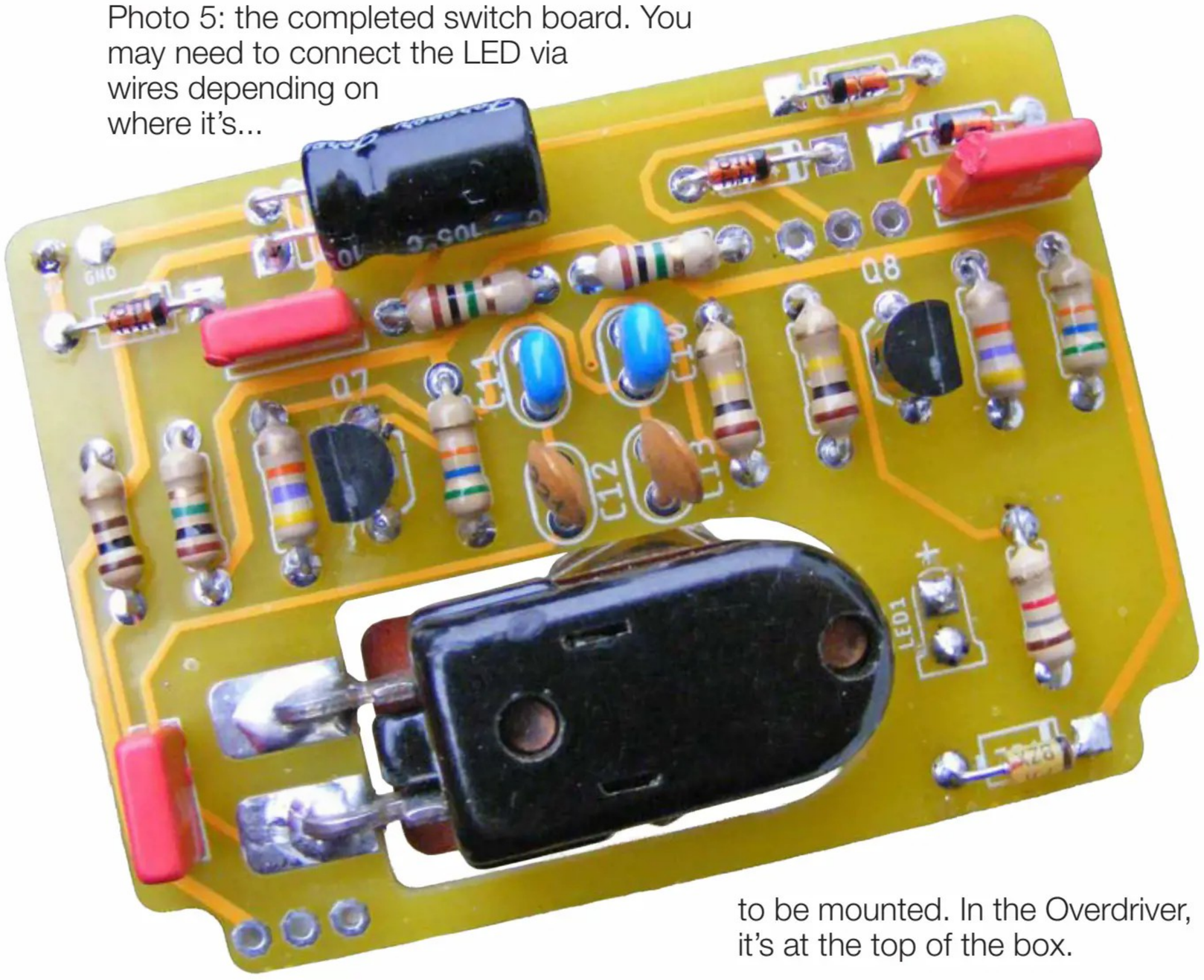
Soldering the switch is quite tricky, with its tags fitting into slots on the board; the PCB may need a bit of filing to get them to slide in.

Also, it's important that everything is straight and aligned before soldering, so that the board is level when the switch assembly is bolted in.

Finally, I've found that some of these switches I've stored in the damp British air for a year or so need cleaning or they won't solder. Remember that the thermal capacity of the tags is much greater than the PCB, so hold the iron's bit to the tag first and let the tag's heat do the soldering to the board.

Because of the heat retention, the solder will take a long time to solidify, so don't let it move while it's cooling. The finished switch PCB is shown in Photo 5.

Photo 5: the completed switch board. You may need to connect the LED via wires depending on where it's...



to be mounted. In the Overdriver, it's at the top of the box.

Remote control

It is a simple matter to wire a transistor across the switch contacts to obtain a control input. This can be very useful if a particular configuration of effects is required to be auto-switched. There also seems to be a recent convergence of Eurorack synthesists and guitarists who would like the idea of gated fuzz.

Assembly and components

Fig.5 shows the overlay for the switch board and Fig.6 is the audio board. They are both fairly compact by guitar pedal standards, so require more care in construction.

Note that the square pads on electrolytic capacitors do not necessarily denote the positive leads; check the + symbols in Figs.5 & 6 rather than trusting the pad shapes.

JFET selection

Provision is made on the audio board for the use of SMT JFETs, which are cheaper than the leaded types. A common problem with JFETs is the orientation of the gate. On most packages, it is not the

middle pin as one would hope. The other pins, the drain and source, can be interchanged because JFETs are symmetrical.

The finished audio PCB is shown in Photo 6. Note the bent-over electrolytic mounting to obtain a low profile. Grindle and I use this technique in many of our commercial designs. It also allows an upgrade to tantalum bead capacitors if needed (or desired).

Testing

Get the switch PCB working first before attaching to the audio section. It is essential to check the DC voltages, the power rail (especially if using a battery), the 4.5V bias (V_b) and the others marked on Fig.3.

After that, check the current consumption, which should be reasonably close to 1.7mA while off and 3mA while on, for the whole unit.

If desired, the unit can be powered with an 18V supply, but the current then rises to 3.2mA and 9.8mA, a bit high for battery use.

If there are problems with the audio transmission, it is usually the JFETs being incorrectly wired. I have become dependent on using the Peak DAC75 analyser to identify pinouts. The pinout diagrams in data sheets/books are often wrong or unclear as to whether is a top or bottom (pin) view.

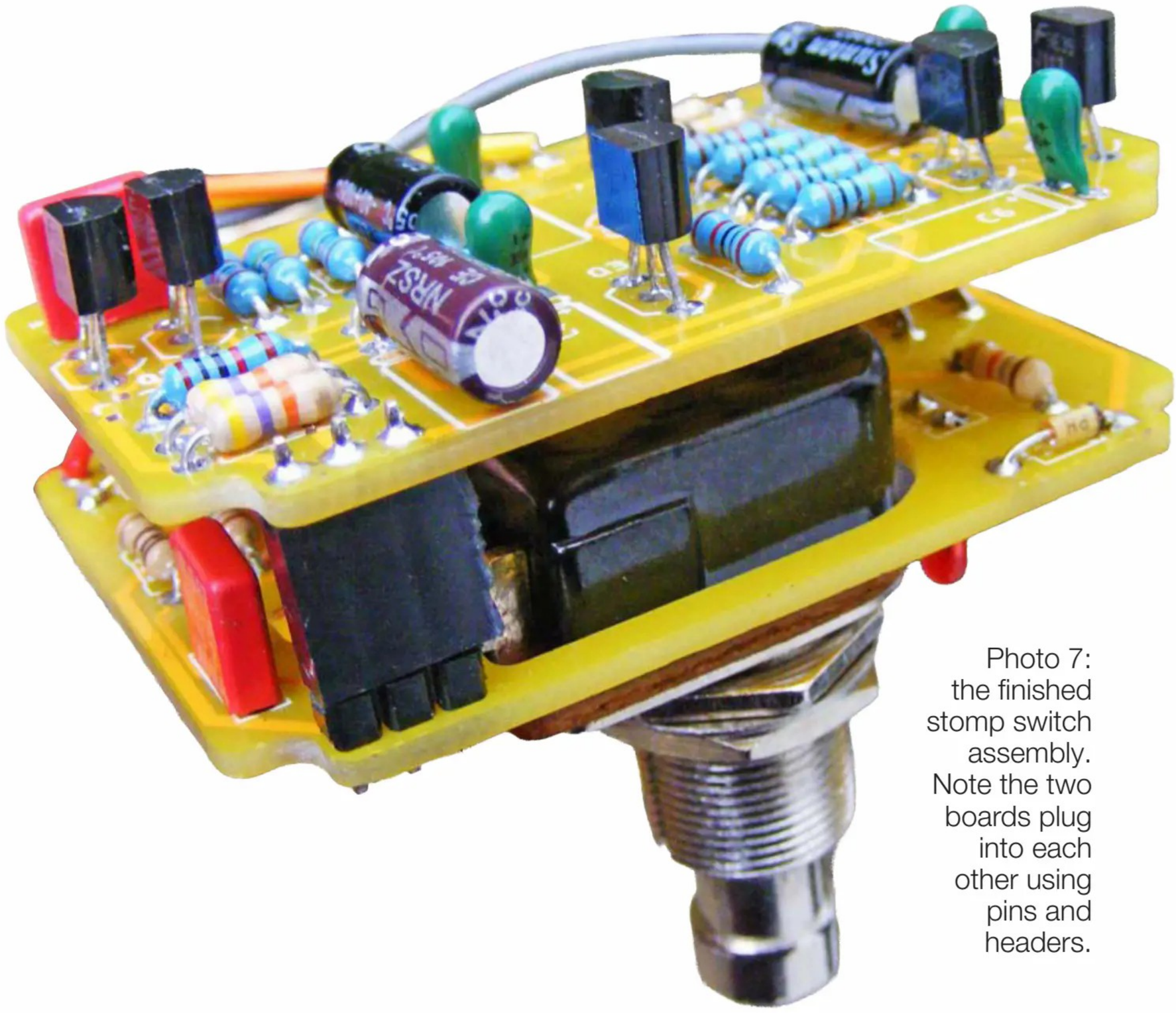
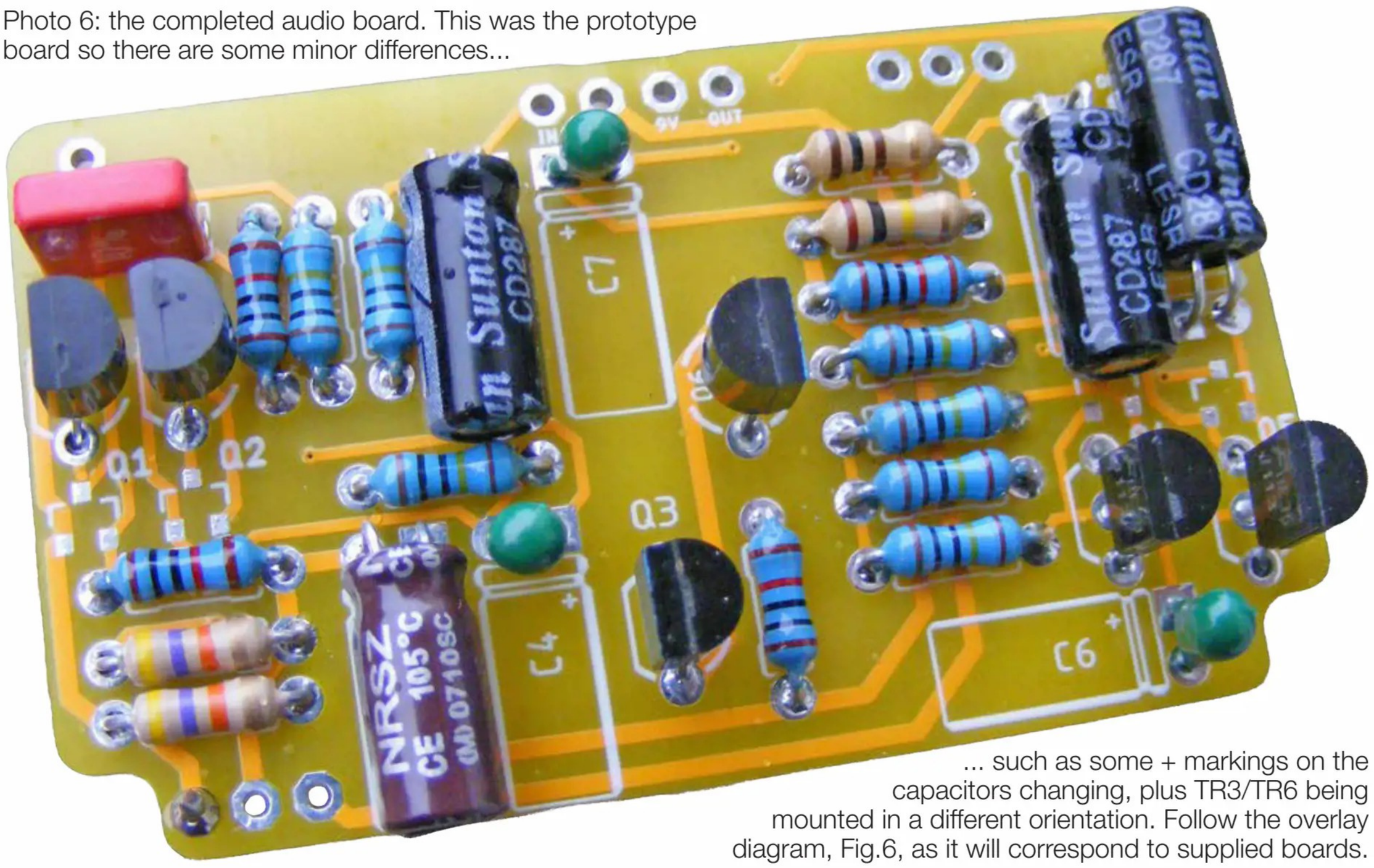


Photo 7: the finished stomp switch assembly. Note the two boards plug into each other using pins and headers.

Installing the unit

Photo 7 shows the 'sandwich' construction, with the audio board plugging into the switch board. Later boards may have some mechanical mods to stop the audio board

Photo 6: the completed audio board. This was the prototype board so there are some minor differences...



... such as some + markings on the capacitors changing, plus TR3/TR6 being mounted in a different orientation. Follow the overlay diagram, Fig.6, as it will correspond to supplied boards.

falling off. If the board-to-board connection were hard-soldered rather than connected via a plug and socket, that would not be a problem, but then service and testing would be.

An extra switch-to-board mounting plate to improve rigidity might be a future mod.

One thing that worries all pedal makers is that the main switch nut becomes loose, and the user keeps stomping on it, breaking the PCB underneath.

The way to avoid this is to use the supplied lock washer between the switch body and the case. Always tighten with a box spanner rather than pliers to get sufficient torque and to avoid scratching the box. Pliers are not the right tool for the job as the jaws do not open parallel.

Putting it to bed

So there you have it. I've replaced over 20 mechanical parts in the switch with 50 extra electronic bits to avoid a click. Does this count as technological progress? I must make a silent pull-switch for our bathroom light so I can get some sleep. PE

Parts List – Silent Stomp Switch

1 double-sided switch PCB coded AO-MAR26-1, 40 × 56mm
1 double-sided audio PCB coded AO-MAR26-2, 32 × 56mm
1 momentary 'soft' foot switch **[eg, Tayda A-1091]**
2 3-way 0.1-inch square-pin PCB plug and header pairs

Semiconductors
4 J113 N-channel switching JFETs (TR1, TR2, TR4, TR5)
4 BC549C NPN high-gain bipolar signal transistors (TR3, TR6, TR7, TR8)
4 1N4148 or similar small-signal diodes (D1-D4)
1 BCY88C4V7 4.7V 400mW Zener diode (ZD1)
1 high-brightness 3mm red LED (LED1)

Capacitors (all ±20% tolerance or better)
1 100µF 25V radial electrolytic (C1)
1 100µF 10V radial electrolytic (C2)
2 10µF 10V radial electrolytics (C5, C9)
4 1µF 10V tantalum bead or radial electrolytics, 2.5mm pitch (C4, C6-C8)
1 220nF polyester film, 5mm pitch (C16)
1 100nF polyester film, 5mm pitch (C15)
1 47nF polyester film, 5mm pitch (C3)
1 10nF polyester film, 5mm pitch (C14)
2 470pF 50V ceramic, 5mm pitch (C10, C11)
2 220pF 50V ceramic, 5mm pitch (C12, C13)

Resistors (all ¼W ±5% carbon film or better)
10 1MΩ (R4, R6-R7, R9-R12, R23, R25-R26)
3 100kΩ (R14, R17, R20)
2 56kΩ (R16, R19)
4 47kΩ (R1, R2, R18, R21)
4 10kΩ (R3, R5, R8, R13)
1 1.8kΩ (R24)
2 100Ω (R15, R22)





AOShop

The home for specialist audio, analogue and historic components – provided by Jake Rothman, *PE's Audio Out* columnist. The AOShop is your best bet for classic analogue 'NOS' (new old stock) components, including all parts for *Audio Out* projects and designs.

Ge semiconductors

Small-signal PNP transistors
NKT214F, OC57, OC59, 2N1377, 2N525 **£1.00**

Low-noise PNP transistors
GET106 **£2.50**

Small power PNP transistors
AC153, AC153K, AC188 **£1.50**

Small power NPN transistors
AC176, AC141K **£2.00**

High-voltage PNP transistors
OC77, CV7001 **£1.50**

TO3 PNP power transistors
OC22, CV7054 (OC23), OC25, OC35, OC36, AD143, AD149, AD161, AD162 **£2.00**
AD140 **£3.50**
AD149, AD161/2 matched pair **£5.00**

RF PNP transistors
OC41, OC42, 2SA12, 2SA53,AF124, AF178, GET872A **£1.50**

NPN transistors
OC139, OC140, ASY73 **£2.00**

Diodes
CV7049 (OA10), CG92 (OA91) **£0.50**

Si semiconductors

Diodes
ZC5800 RF Schottky **£0.20**

Low-noise silicon transistors
BFW16A, 2SC3071, 2SC3068, 2SA1016K, 2SC2362K, 2SA970BL **£1.50**
2SC2204, 2SD655, BC550C **£0.50**
ZTX651 **£0.30**
ZTX751 **£0.50**

RF transistors (suitable for Theremin)
BF199 **£0.50**

Audio power MOSFETs
Exicon 10N20, 10P20 **£6.50**
Hitachi 2SJ99, 2SK343 **£3.50**
Hitachi 2SJ56, 2SK176 **£8.50**

JFETs
BFW11, BFW10, TIS73L, J177, J113, U1994, U1898, 2SJ176, J201 **£1.00**
J175, J176, J112, J111, 2N3820, 2N5457, BF244, 2N5460, J230 **£0.60**

Small power output/driver transistors
2SB649A, D669A, 2SA1208, 2SC2910, MJE253G, MJE243G, 2SA1725, 2SC4511 **£1.20**
BD139,BD140, BD135, BD136, BD435, BD436, MPSA63, BCV46 **£0.50**

MOSFETs
ZVP2106A **£0.30**
Dual-gate 3SK45, BFS28 **£1.50**

Metal-cased transistors
BCY71 **£0.30**

Amplifiers
LM318 high-speed op amp **£0.35**
µA709 metal-cased op amp **£2.00**
5534H metal-cased op amp **£3.00**
LM384, TDA2030A, TDA2050V **£1.50**
TAA435 (Mullard power amp driver for AD161/AD162) **£1.20**

Synthesiser ICs

That/dbx 2180 VCA/VCF **£6.00**
CA3080 VCA/VCF (vocoder) **£2.80**
CA3280 VCA/VCF **£4.00**
LM13600/ LM13700 VCA/VCF **£2.00**
CA3086 transistor array **£1.00**
PT2399 echo/delay **£1.50**

Dual transistors

2N2639, 2N2223, 2N2910 (NPN) **£4.00**
E401 (JFET Moog) **£4.00**
2SK2145-Y dual JFET **£0.80**
2N5564 JFET **£8.00**
HN3C51F, HN3A51F **£1.00**
DMMT3904/6, HN1A01F, HN1C01F **£0.50**

Loudspeakers

PE Mini-Monitor
Volt PE165 6.5-inch woofer (each) **£85**
Morel MDT29 tweeter (each) **£25**
Kit *pair* of PE165/MDT29 plus Volt crossover parts and PCBs **£299**
Monacor DT-28N tweeter (each) **£35**
Vifa 19mm BC20SC15-04 tweeter (each) **£15**

Volt crossover inductors
1.2mH, 1.5mH, 2mH, 2.7mH, 0.5mH (tapped at 0.3mH) (each) **£5.85**

Fully assembled and tested high-quality speaker prototypes – ask for details
LS3/5As and other similar speaker systems (pair) **£200-£350**

Fully tested reclaimed speakers
Vifa BC14 5-inch woofer (each) **£10**
Vifa TC26 1-inch tweeter (each) **£10**

Low-price speakers
Philips 4-inch 4070 **£2.00**
EMI 10x6-inch, 30Ω Alnico **£7**
64mm 64Ω neodymium **£1.20**

Capacitors

Note '10/63' denotes '10µF 63V'.

Polyester
3.3/100, 4.7/250, 4.7/63 **£1.00**
5.6/63, 8.2/63, 10/63 **£2.00**
Mullard 'Mustard' C296 0.22/400 **£2.00**

Polycarbonate
Axial 2.2/63 1%, 4.7/160, 6.8/63 **£1.00**

Radial
6.8/160V, 10/63 **£2.00**
Reclaimed 22/63 **£2.00**

Polystyrene
Philips 1% 4.7nF/160, 6.2nF/500, 12nF/63, 22nF/63, 110nF/63, 24nF, 2nF **£1.00**
RIFA 1% 100nF/100, Suflex 90.9nF 0.5% **£2.00**
Suflex 2.5% 10nF/63 (rad. or ax.) **£0.50**

Radiation resistant
Siemens cellulose acetate MKL 2.2/25 **£0.80**

Electrolytic – Mullard blue 017 series
10/25, 22/25, 100/10 **£0.50**
150/40, 470/40, 1000/40 **£1.00**

Tantalum – axial metal cased
22/50, 47/35, 68/25, 100/20, 120/10, 150/16, 220/10, 330/6 **£2.00**
22/35, 33/35, 47/20, 68/15, 100/10, 150/6 **£1.25**
4.7/50, 6.8/35, 10/25, 10/35, 22/15 **£1.00**

Axial moulded-case tantalum
Kemet axial 6.8/10 **£0.30**
Kemet radial 33/10 **£0.40**
STC radial 100/20 **£1.50**

Tantalum bead
22/50 **£1.00**
470/3 **£2.00**
680/6.3 **£3.00**

Wet tantalum
220/25 axial **£3.20**
Castanet button 140/30, 470/3 **£3.20**
Hughes 540/10 **£3.20**

Bipolar
Hermetic bipolar tantalum 16/35 **£3.20**
Elcap axial 10/50 **£0.50**
Generic radial 100/16, 470/35, 100/ 63, 22/35, 4.7/35, 220/16 **£0.50**

Philips solid-aluminium (axial) 121/123
47/16 **£1.00**
330/6.3 **£2.00**
100/35 **£4.00**

Philips Pearl 122 series (radial)
10/16 **£0.30**

Silvered mica (radial)
1nF/500 1% **£1.00**

Trimmer capacitors
Vishay plastic-film 4-40pF **£0.80**
Vishay plastic-film 5.5-45pF **£1.00**
Vishay plastic-film 5-80pF **£1.20**
Vishay plastic-film 10-250pF **£2.00**
Mica 1-12pF, 2-40pF **£1.00**

Audio transformers and inductors

Eagle transformers
LT44, LT722 driver, LT700, LT723 500Ω output **£2.50**
LT30 500mW output **£3.50**
5:1 interstage (fuzz box) **£1.50**

Repanco
T/T3 splitter transformer **£4.00**
CH2 5mH RFC **£2.00**

Balanced output transformer
Vigortronix 600Ω VTX-101-007 **£10**
Vigortronix 600Ω VTX-101-3001 **£10**
Vigortronix 600Ω VTX-101-3002 **£15**
Gardners 150Ω **£10**
Reclaimed BBC LL74/MPC nickel core 600Ω **£12**
Low-level 1:1 600Ω (modem) **£2**

Reclaimed mic input transformer **£15**

Inductors
82µH, 4.7mH, 100µH, 270µH, 10µH, 14µH (low Z) **£0.50**
4-inch ferrite rod with MW and LW windings **£3.00**

Special resistors

Bourns wire-wound trimmer 5kΩ 3059 JM panel-mount **£2.00**
Thermistor RA53, R13 **£4.00**
A13 **£2.00**
Thermistor CZ1, CZ6 **£1.50**
Holco H2 2.2MΩ 1W, 1% **£1.00**
Welwyn 1GΩ 2W **£1.00**
5k Bourns 3321H cermet trimmer **£0.50**

Potentiometers

Bourns 81
25kΩ lin cermet **£2.00**
25kΩ lin, 5kΩ lin cond. plastic **£3.00**

Bourns 91
10k dual-gang lin or log **£10.00**

Plessey moulded-track
5kΩ log with switch **£3.00**
50kΩ A/log **£3.00**
Mil 250kΩ lin dual **£5.00**

Alpha
16mm 4.7kΩ A/log **£0.80**
220kΩ A/log **£0.80**
10kΩ lin centre-detent dual-gang **£1.50**

Allen Bradley J series/Honeywell
10kΩ lin 1 million cycles **£5.00**
Blore Edwards AB 45 dual 5kΩ A/log with switch **£3.50**
Alps RK9 dual-gang 5kΩ RD law **£4.00**
Alps 50k log tapped motor. stereo **£5.00**
BI P260 500kΩ log conductive plastic 1 million cycles **£2.00**
Colvern wire-wound 100kΩ or 50kΩ dual-gang 3W **£5.00**
Mil Spec hermetic 10Ω **£8.00**

Miscellaneous

PCBs
Silent Stomp Switch (pair) **£4.00**
Penta-fuzz Bender (*PE*, Feb26) **£3.00**
Pre-driver (*PE*, Jan26) **£3.00**
Pre-driver jack bd (*PE*, Jan26) **£1.50**
Fuzz Box (*PE*, Nov25) **£3.00**
Fuzz Box jack bd (*PE*, Dec25) **£1.50**
Dual JFET Buffer (*PE*, Oct25) **£2.00**
Mosfet Buffer (*PE*, Oct 2025) **£2.00**
Dual Mosfet Buffer (*PE*, Oct25) **£2.00**
Cascode Amp (*PE*, Sep 2025) **£2.00**
Cascode PS (*PE*, Sep 2025) **£2.00**
Pocket Theremin (*EPE*, 1996) **£2.00**
Elysian Theremin (*EPE*, 1996) **£6.00**
Synth VCF, VCO (*EPE*, 2017) **£3.00**
48V PSU (*EPE*, 2019) **£3.00**
SMT dual transistor adapter **£0.40**

Relays
STC hermetic 4190 24V DPDT **£4.00**

Switches Blore Edwards/Plessey
MH1 4-pole 3-way **£3.00**
Elma 01 2-pole 6-way **£6.00**
3PDT guitar pedal stomp switch **£2.50**

Contact
Jake Rothman
Manchester House
Market Street, Craven Arms
Shropshire SY7 9NN
(visit by appointment)
+44 (0)1597 829102
jrothman1962@gmail.com

Minimum order £5.00 inc post

Payment
PayPal, bank transfer, cheques (payable to "J Rothman", UK pounds only). No VAT payable

Postage
Small Jiffy bag £2.99
Small package £4.99
Big boxes and overseas at cost – ask for a quote

The Fox Report

Barry Fox's technology column

Networking not working?

I have been fighting computers since the early days of Sinclair dead-hand-feel keyboards, the ahead-of-its-time Tandy 100 warhorse laptop, not-quite-DOS Apricots, IBM Wintels and endless versions of Windows.

One common thread of misery has been the difficulty of networking devices, first with coaxial cable and then on into the Ethernet (Cat5 etc) and Wi-Fi world. Interestingly, the new MoCA, Multimedia over Coaxial Alliance, uses old TV coax cabling as a way to carry Ethernet signals.

Build a network and it will 'find' devices but then 'lose' them again for no logical or discernible reason. New PCs may come with Network Settings set by default to block networking, eg, with Network Discovery off. Is all this to create paid work for IT support engineers?

The Internet is awash with network fix-it advice, often garbled. Recently, I found a tutorial on YouTube that is quite helpful: **https://youtu.be/N-R44Clys9A**

From this, a mish-mash of other posts and a lot of personal trial and error, I've put together a simple-as-possible guide:

Left-click to open File Explorer (the yellow square icon in the horizontal taskbar that runs along the bottom of a Windows screen). Right-click on the blue icon This PC in the vertical panel that runs down the left side of the screen. Left-click Properties (the Spanner icon). Note: you may need to click Show More Options to find it.

This takes you to System – About. Left-click Rename this PC to give the PC a friendly name that is easy to recognise when networked. Note: you need to use hyphens rather than spaces. Then reboot the PC.

Now (again) left-click to Open File Explorer. Right-click on This PC and left-click Properties. This takes you (again) to System – About, with the computer name now changed to your more friendly choice. You should see a group of related links, including the option to change the workgroup name.

But – thanks a lot, Microsoft – you may not see it. If not, try a refresh or reboot, or use the search tool (magnifying glass in the taskbar at the bottom of the screen) to search for Change Workgroup Name, then Open.

Here, you can add a friendly description of the PC. You can also change the name of your network, but it's easier just to leave it at the default WORKGROUP. Click OK.

Now use the taskbar search tool to find and open the Control Panel. Click on Network and Internet, then Network and Sharing Centre. Now click on Change Advanced Sharing Settings (in the left-side vertical column).

In the Private Networks section, set all settings on. In the Public Networks section, set all settings off. In the All Networks sections, leave Sharing off. To keep things simple, for the time being at least, set Password Protected off.

Close the page and now right-click on any blank part of the Windows desktop and create a new folder, perhaps named "Share". Right-click on this new folder, choose Properties and then Sharing. Click on the Share box, then on the down arrow and select Everyone. Click Add.

Click on the down arrow next to Everyone and select Read/Write, click Share (at the bottom), then Done and Close.

Next, right-click on the same folder and go again to Properties/Sharing and click on Advanced Sharing. Click on Share This Folder, then click on Permission and Allow Full Control for Everyone. So everything in this folder is now shared with everyone with full permissions on a private network. Phew!

Now click OK to close it and repeat the whole tedious process with any other computer you want on the network – with appropriate variations in naming, of course.

Next, on each computer, left-click to open File Explorer in the taskbar. Look for, and left-click on, the blue icon Network in the vertical column down the left side of the screen. On each computer, you should see the other computer. But it may need a refresh (right-click on empty desktop space, and perhaps also click Show Options) or reboot.

If no joy (thanks again, Microsoft), try this trick. Type run in the taskbar search box and, in the Open box, type "\\My-Laptop" or whatever name you have given the other computer. Click OK and you should now, at last, see the files on the other computer.

If you have forgotten the name of a computer, you can find it on that computer by typing "settings" into the taskbar search box, then open it, and the Settings home page will show the computer basics, including its name.

This insanely complicated process should give you a network of computers that share the folders you created and shared. You can now use the same basic steps and tricks to make other folders and files shareable; and perhaps add password protection.

If you hit inexplicable roadblocks, the culprit might be a firewall or anti-virus software silently blocking the connection. To check, turn any such protection off for a couple of minutes and see if that fixes the problem.

In a sensible world, your firewall and/or anti-virus would have a big red button marked "Press here to kill me for a test; press again to revive me". More likely, you will have to dig into the software settings of whatever software your PC uses.

Another way to help networked computers 'find' each other is to create desktop shortcuts to networked content. Right-click on empty desktop space and click New → Shortcut, then type "\\My-Laptop" (or whatever name you used), then Next/Finish. This creates a shortcut on the desktop that leads straight to the networked computer.

Tip: Whatever the hazy, largely unwritten rules of networking may say, I have found it easier not to mix Ethernet and Wi-Fi connections. If your laptop does not have an Ethernet socket, spend a tenner on a USB Ethernet adaptor. If nothing else, it removes one of the many variables in the ridiculous game Microsoft has crafted.

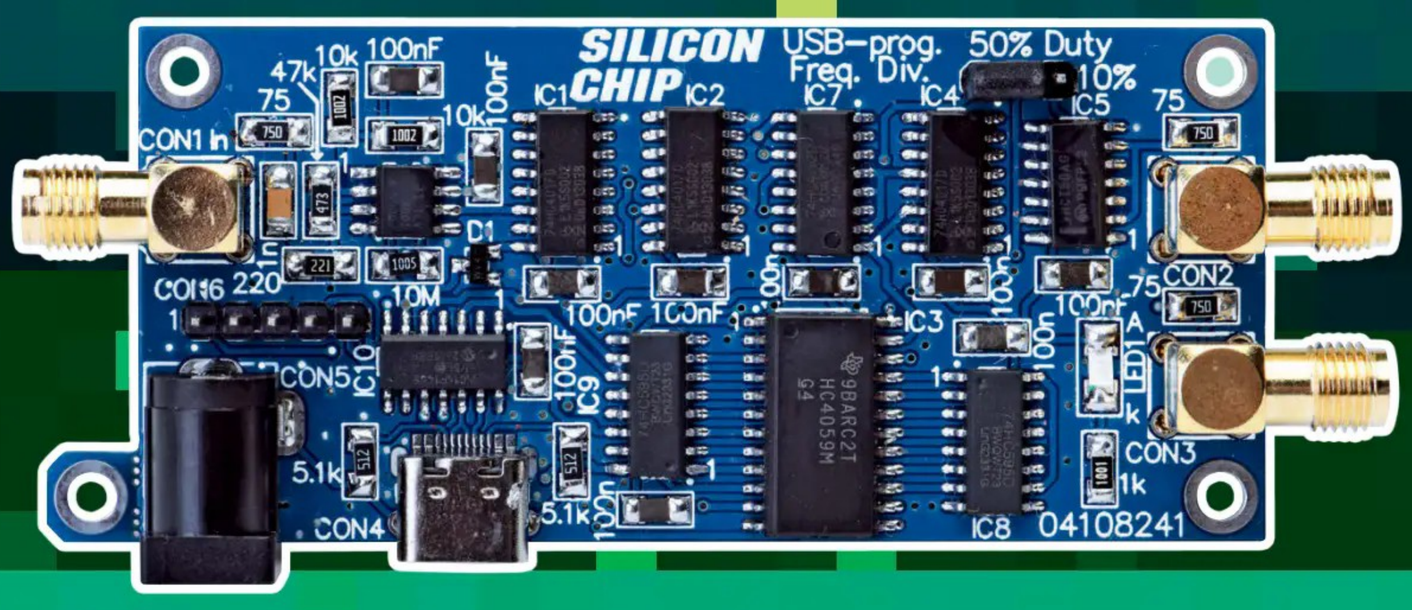
Internet posts suggest changing the Service settings. Try to avoid this, because you may create new faults. But if you feel desperate and/or brave, try these settings (making a note of anything you change so that it is easy to change back).

To do this, type services in the taskbar search. Open it and set the following entries to Automatic:

- DNS Client
- Function Discovery Provider Host
- Function Discovery Resource Publication
- Server
- SSDP Discovery
- TCP/IP NetBIOS Helper
- UPnP Device Host
- Workstation

If and when I recover my mental strength and good humour, I may tackle the mysteries of networking Windows computers with Mac and Android devices next. PE

Programmable Frequency DIVIDER COUNTER



This small PCB doesn't cost a lot to build but provides several useful features. It can reduce the frequency of an incoming signal or pulse train by a factor of between 3 and 21,327,000 that's easily configured over a USB serial port. It can also provide basic frequency measurements.

By Nicholas Vinen

- Input frequency range: 300Hz to 77MHz (typical)
- Over 85,000 possible division ratios:
  - 3–21,327 (with short output pulses)
  - 30–213,270 in steps of 10 (50% duty cycle output)
  - 213,300–2,132,700 in steps of 100 (50% duty cycle output)
  - 2,133,000–21,327,000 in steps of 1000 (50% duty cycle output)
- Configuration: via USB serial interface; default power-up ratio can be selected
- Can measure the input signal frequency to ±0.25% (prototype was within ±0.02% @ 20MHz)
- Output duty cycle: 10%/50% (if division ratio is multiple of 10; otherwise < 34%)
- Input signal level: 20mV to 3.2V RMS (100mV+ or more at low frequencies)
- Output jitter: estimated at 0.1ns
- Propagation delay: approximately 110ns
- Input/output impedances: 50Ω or 75Ω
- High noise immunity with 23.5mV hysteresis
- Outputs are in phase with inputs
- No output signal toggling in the absence of an input signal
- Power supply: 5-12V DC @ 20mA
- Power connectors: USB Type-C, 2.1mm/2.5mm ID barrel socket

My Compact Frequency Divider design, published in the March 2025 issue, was a purposefully simple design. However, the hardware used was capable of doing a lot more than just dividing a signal frequency by a fixed ratio.

I could have added jumpers to allow the ratio to be changed, but at least 19 jumpers would be required and figuring out how to set them would be very complicated.

I instead decided to add a low-cost microcontroller with a USB interface that would allow you to type any desired division ratio. It could quickly reconfigure the divider chips to achieve the desired ratio. The cost and size of the extra parts are minimal, so the result is still quite compact and affordable.

It can easily be configured with a wide range of possible division ratios; the USB cable used to configure it can also power it. For maximum flexibility, you can also save the ratio you set into flash memory and then power it from just about any low-voltage DC source away from a computer.

I also realised it was pretty easy to add an onboard frequency counter that would be reasonably accurate, so I did that as well.

You just type 'm' (for "measure") and press Enter and within a second or so, it measures and reports the input frequency. So it's a fairly useful little device that's small and not too expensive to build.

The signal chain is very similar to the March 2025 design mentioned earlier, with four digital logic ICs doing most of the work. There are three 74HC4017s, each dividing their inputs by a factor of 10, plus a 74HC4059 programmable divider chip. Cascading those four stages gives many possible division ratios.

Like that earlier design, a high-speed push/pull comparator is used to boost and square up the input signal, while a hex buffer IC provides the drive strength for 50Ω or 75Ω termination. Besides the new parts to control the division ratio (a microcontroller and three more digital logic chips), the remainder of the circuit is a linear power supply with reverse polarity protection and a power indicator LED.

The unusual 74HC4059

While updating this design, I considered why the way the 74HC4059

Programmable Frequency Divider

divider works is so strange, and I think I have the answer.

The 4059 was designed in a time before microcontrollers. If you wanted to build a digitally controlled, PLL-based (phase-locked loop) radio tuner, you would need to have some sort of 'register' chip or chips to store the currently selected frequency.

There would be a way to increase or decrease the value stored in that register (using buttons, or a knob, or something like that) and its output would be a binary number that would be fed to the PLL to control the tuning.

You would also need a frequency divider as part of that PLL. A PLL essentially works by using negative feedback via a divider to multiply a frequency. So it appears that the 4059 was designed to be that part of the circuit, with its 16-bit digital inputs fed from the register.

That explains why it has a programmable prescaler that effectively multiplies the 16-bit ('J') value by a figure of 2, 4, 5, 8 or 10. That is the way that you would select the size of the steps caused by each increment or decrement of the J-value.

For our purposes, we can use a fixed prescaler value of 8 since we don't need configurable steps; that value gives us the widest possible range of division ratio (3 to 21,327) and the microcontroller can figure out what values to feed to the 16 J inputs to achieve the desired ratio.

To make the circuit more flexible, rather than just controlling the configuration of the 74HC4059, I have also added the ability to bypass one, two, or all three of the 74HC4017 divide-by-ten stages. That is done using a single 74HC4052 dual 1-to-4 analog multiplexer chip, as described in more detail later. So we can divide by a factor of 1, 10, 100 or 1000 in addition to the configurable ratio of the '4059.

Ideally, we want the final divide-by-10 stage in the circuit for two reasons. One is that the output pulses from the 74HC4059 are very narrow (only the length of the period of its input signal) and that last divide-by-10 stage acts as a clock stretcher. The other is that it also gives us the 10% or 50% duty cycle choice.

So while ratios that are not multiples of 10 can be set (up to 21,327), the output pulses will have a duty cycle of 1/r, where r is the ratio. For example, a ratio of 321 will result in output pulses with a duty cycle of 1/321 or 0.31%. That's fine for feeding to most frequency counters, using as a trigger pulse for an oscilloscope and so on. But there may be applications that such short pulses are not suited to.

Since we have a microcontroller in the circuit, we can feed the output of the divider to it and have it count the pulses over a fixed time to give us an onboard frequency counter capability. The micro has no crystal but it synchronises its clock to the USB host, which likely does, so it should be reasonably accurate.

So that you can use this device without it being tied to a computer, once the ratio is set, it can be stored in internal flash memory. In that case, it will be automatically restored each time it is powered up after that until it's changed again. So you can plug it into a computer, set the ratio, unplug it and take it somewhere else to use it. It just needs a 5-12V DC supply to operate away from a computer.

Circuit details

The full circuit of the new Programmable Frequency Divider is shown in Fig.1. If you compare it to the one from March 2025, you will see that the power supply and circuitry around IC6, the comparator that acts as an input signal amplifier, is identical. The one difference in the power supply is that I didn't have room to easily fit a 2-pin header as an alternative to the barrel socket (CON5) on this PCB.

The arrangement of IC1-IC5 is also essentially the same, except that IC3's configuration inputs are now driven by microcontroller IC10 and two 74HC595 serial-to-parallel shift registers (IC8 & IC9).

Also, rather than a fixed signal chain of CON1 → IC6 → IC1 → IC2 → IC3 → IC4 → IC5 → CON3, we now have IC7b selecting whether IC3's input clock comes from IC6, IC1 or IC2. The other half of that device, IC7a, also selects whether the output signal fed to IC5d/e/f comes from the output of IC3 or IC4. IC7 itself is controlled by microcontroller IC10 via its S0 & S1 inputs.

With S0 & S1 both low, Ya0 and Yb0 are selected, so all three 74HC4017s are bypassed and only the 74HC4059

Converting a division ratio into an appropriate configuration

As mentioned in the text, we only use a prescaler mode of 8 for the 74HC4059 chip because it gives the largest possible range of division ratios and has no disadvantages in this configuration. As the prescaler can be preset, that doesn't restrict us to division ratios that are multiples of eight.

Given a desired ratio, first we check if it is within the possible range of 3-21,327,000. If it is below 3, we make it 3, and if it is above 21,327,000, we clamp it to that value. We then check whether we need one, two, or three divide-by-ten stages based on the value. If the ratio is a multiple of ten and at least 30, we force it to have at least one divide-by-ten stage so we can get a 50% duty cycle output.

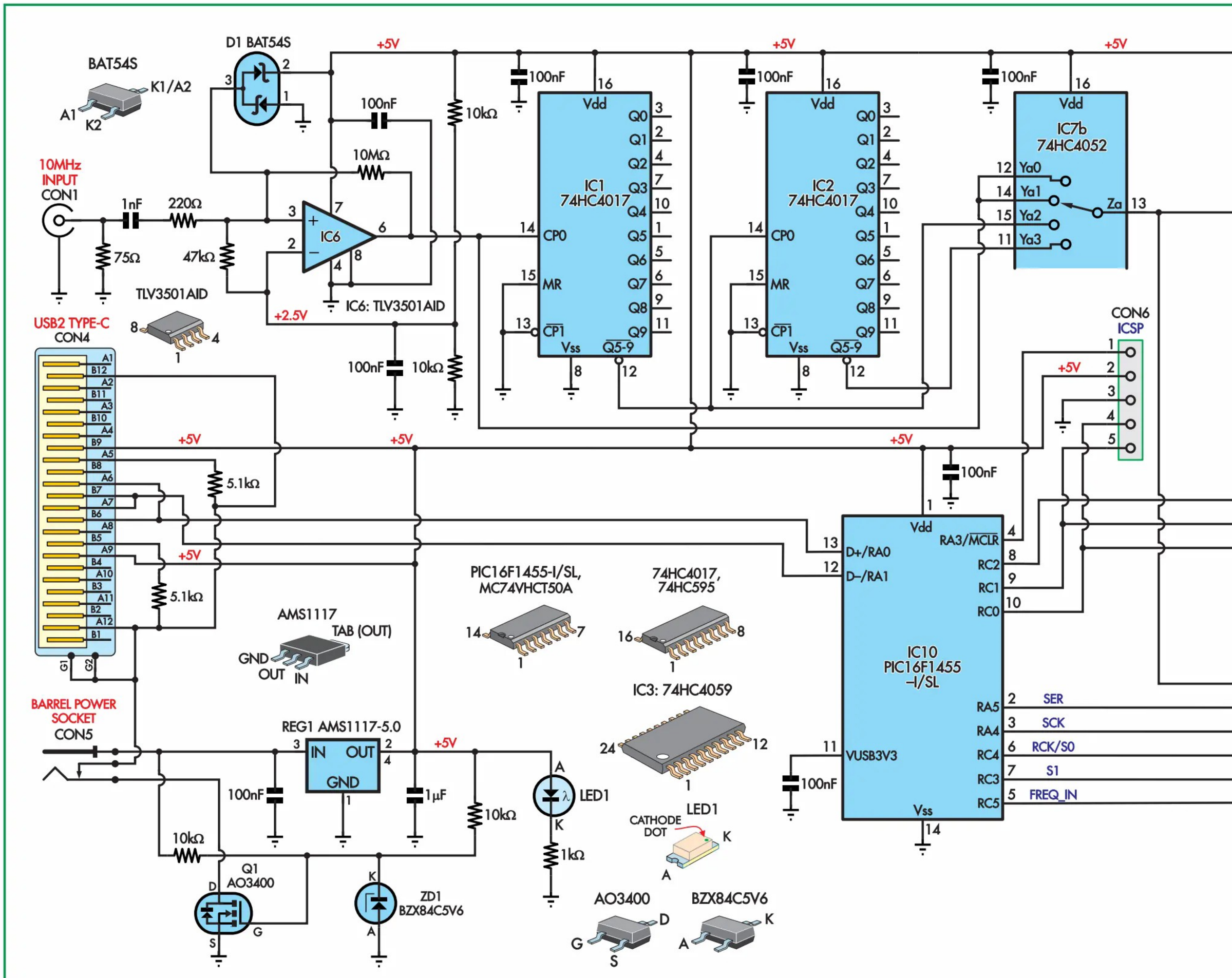
We can then divide the desired ratio by either 1, 10, 100 or 1000 to determine the division factor necessary for the 74HC4059, which will be a value between 3 and 21,327. The final step is to figure out the 16-bit J value that produces that ratio and load it into the shift registers.

J1-J3 are the prescaler preset value, between 0 and 7, calculated simply as the desired ratio modulus eight (ie, what is the remainder after dividing the ratio by 8).

We then divide the ratio by eight and, if the result is 1000 or more, bit J4 must be 1. If J4 = 1, we subtract 1000 from the remainder. We can then determine the value for bits J13-J16 by dividing the remainder by 100 again. We take the remainder from that and divide it by 10 to get the J9-J12 value, then the remainder is the value for J5-J8.

There's just one remaining trick, which is that when we do the divisions by 100 and 10, we might get a value of more than 15, which is not possible to fit into four bits. In that case, we simply use the value of 15 but add the excess back to the remainder so that it flows into the later calculations.

That's necessary because of the odd 'extended range' trick needed to use the full set of possible division values of the 74HC4059. It's a result of it being a BCD device but having 4-bit registers that can be programmed to any value between 0 and 15, not just 0 to 9 like a regular BCD device.



provides frequency division. This is for division ratios from 3 to 21,327 but not multiples of 10 (except for 10 and 20) because, as explained earlier, we ideally want IC4 to be involved so the 10% and 50% duty cycle outputs have the expected duty cycles.

For the ratios 3-29 or 31-21,327 that are not multiples of 10, IC4 must remain out of the circuit. JP1 must be in the 50% position but the output duty cycle will actually be much lower (1/r, as mentioned earlier). If you put JP1 in the 10% position, you will get an output but its frequency will be divided by 10 compared to the other position.

With S0 high but S1 low, Ya1 and Yb1 are selected. IC1 and IC2 are still bypassed (ie, the signal goes straight from IC6 to IC3) but now the output comes from IC4 instead of IC3. This is for ratios from 30 to 213,270 that are multiples of 10. Above 21,327, only multiples of 10 are available.

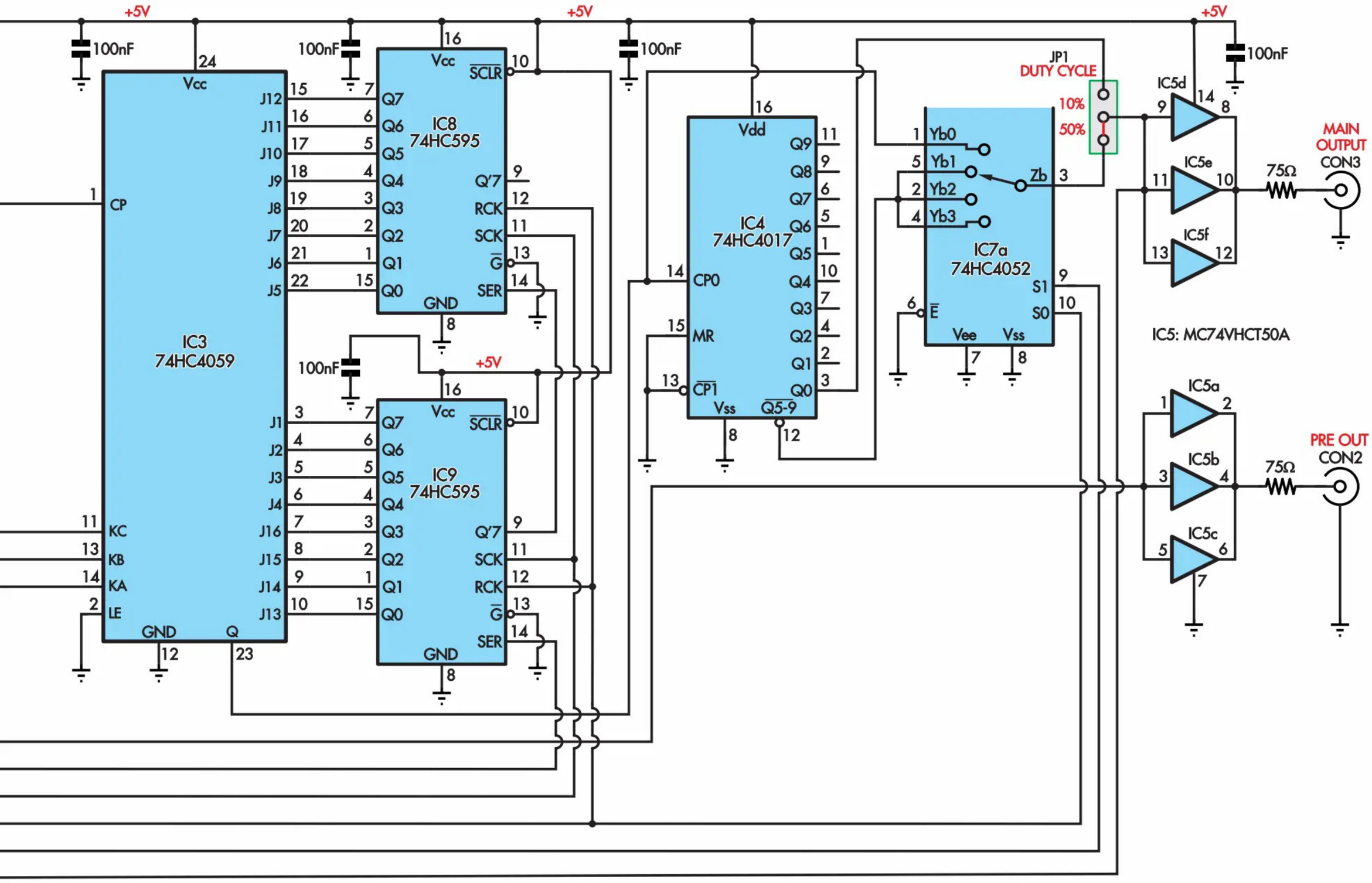
With S0 low and S1 high, Ya2 and Yb2 are selected. Now only IC2 is bypassed and the range of available ratios is 300 to 2,132,700. With both S0 and S1 high, the signal passes through all four divider chips and the range of available ratios is 3000 to 21,327,000.

Microcontroller IC10

The firmware running on IC10 does a few things:

- USB serial communications. It receives and echoes characters typed by the user. When enter is pressed, it parses the command and, if it's valid, changes the division ratio, saves it to flash or performs a frequency measurement via its RC5 (pin 5) digital input.
- Configuring IC3. When a new division ratio is sent to IC10, it calculates the closest possible ratio that is actually achievable and figures out what values of J (16 bits) and K (3 bits) are required to achieve it. It then spits out the 16 bits of J one at a time via the SPI-like bus formed by the following digital outputs: RA5 (pin 2, serial data); RA4 (pin 3, serial clock) and RC4 (pin 6, latch enable). The latch-enable function is shared with the S0 control for IC7 to save a pin, since the S0 state doesn't matter while the ratio is being reprogrammed.
- Performing frequency measurements. Pin 5 is also the 8-bit Timer 0 clock input (T0CKI), allowing IC10 to easily measure the output pulses. 16-bit Timer 1 counts IC10's internal 12MHz clock pulses simultaneously. The currently set division ratio (which IC10 steps through to find the best one) along with the ratio of the two counts allows it to calculate the input frequency. For more on how it does this,

Programmable Frequency Divider



SC ©2024 PROGRAMMABLE FREQUENCY DIVIDER / COUNTER

Fig.1: the signal from CON1 is amplified/squared up by IC6, then is divided by 10 by IC1, and again by IC2. IC7b determines which of the original or divided signals arrives at the CP input of IC3. Its division ratio is set using serial-to-parallel registers IC8 & IC9 along with signal direct from micro IC10. The output is optionally divided by 10 again by IC4 and then fed to the output buffers.

see the panel at the end of the article.

In-circuit serial programming header CON6 allows IC10 to be reprogrammed while on the board. This was very useful during development and you may need it if you plan to program the chip yourself. If you purchase a programmed chip from the Silicon Chip Online Shop (possibly as part of a kit), you could leave CON6 off.

Input signal conditioning

We skipped over this in the description above, partly because it was already covered in the March 2025 article. Here's a quick rundown.

The incoming signal is terminated by a 75Ω resistor (it could be 50Ω depending on your preference) and then AC-coupled to the non-inverting input of high-speed comparator IC6 via a 220Ω resistor. This resistor limits the current in case the input signal level is too high, in which case the voltage at the non-inverting input of IC6 is clamped to the +5V/0V rails by dual schottky diode D1.

The inverting input of IC6 is held at 2.5V, ie, half of the 5V supply due to a pair of 10kΩ resistors across the 5V supply and a 100nF capacitor to stabilise it and keep the source impedance low. Because both inputs are DC biased to close to the same voltage, only a small signal is required at the CON1 input to cause IC6's output to toggle and swing between +5V and 0V.

Without hysteresis, this would have a tendency to oscillate, as even a bit of noise would be enough to cause that toggling. However, the 10MΩ resistor from IC6's pin 6 output to its pin 3 non-inverting input means that the input signal needs to exceed 23.5mV peak-to-peak before the output level will switch. It also means there is no output from IC6 if CON1 is left disconnected.

There will be a small voltage across the 47kΩ resistor due to IC6's input bias current, which places an upper limit on the practical value of that resistor. It will form a divider with the 220Ω resistor but it only reduces the incoming signal level by about 0.5% so it doesn't really affect the operation.

Outputs

The two outputs are each driven by three parallel stages of IC5, the MC74VHCT50A hex buffer. This is similar to a 74HC04 hex inverter except that the outputs follow the inputs rather than being inverted, so there is no phase inversion.

We need three in parallel for each output in case the constructor chooses 50Ω termination, in which case the buffers could be driving a load

as low as 100Ω. In that case, the output current requirement at 5V is 50mA, right at the limit of two stages @ 25mA each.

The output load will be higher at higher frequencies due to the characteristic impedance of the output cable, approaching 150Ω (50Ω source + 50Ω cable + 50Ω termination). Still, at low frequencies, we have to assume that the cable impedance is close to 0Ω.

Power supply

If a USB-C cable is connected via CON4, it powers the circuit directly. 5.1kΩ pull-down resistors are provided on the A5 and B5 pins to ask the host to supply 5V. If a plug is inserted in barrel socket CON5, USB power is disconnected from the circuit to avoid it feeding back into the computer.

With 5-12V applied to CON5 (ideally at least 6V), power is fed to low-dropout 5V regulator REG1, which supplies the rest of the circuit. Regardless of the source of 5V, LED1 lights up.

Mosfet Q1 is provided in case a power supply is connected to CON5 with reversed polarity. With the correct polarity, Q1's gate is pulled positive via one or both of the 10kΩ resistors. Q1 is switched on and it connects the barrel socket ground to circuit ground (its body diode conducts before it switches on fully).

Alternatively, if CON5 has the wrong polarity, Q1's gate is pulled negative, holding it off, and its body diode is reverse-biased, so no current can flow between circuit ground and CON5's outer barrel contact. Thus, no damage can occur. Zener diode ZD1 protects Q1's gate from high applied voltages; it is safe up to -30V, at which point Q1's channel could begin to break down.

Firmware

The firmware is based on a CDC USB/serial implementation that provides the serial console used to configure the device. Added to that is code to set up the inputs and outputs and program the dividers in response to commands sent from a computer via USB. It also incorporates a frequency measurement routine that lets the board act as a basic frequency counter from 300Hz up to about 70MHz.

The software code is converted to a HEX file using Microchip's XC8 compiler. There is a free version of that compiler that lacks size optimisation (-Os). There is also a free 60-day trial of the Pro version that includes that feature. Because the PIC16F1455 only has 8k words of flash, we had to use the -Os option to fit all the features we wanted.

Since we supply a compiled HEX file, you don't need the compiler. You can either purchase a pre-programmed PIC or load the HEX file yourself. But if you want to modify the code with the full feature set, you will need the Pro version of the compiler.

Another option is to remove the line "#define XC8_PRO" near the top of the file "main.c". That will remove the L, H and P commands (see below) but the code will then compile with the free version of XC8 v2.5.0 (using -O2 optimisation) and still fit in the available flash. Those commands are not critical to the device's operation.

Construction

Assembly of the device is similar to the March 2025 version, except that there are a few extra chips onboard. The layout of the PCB is also similar, although last time, comparator IC6 and its associated components were on the underside and most of the other parts were on the top.

This time, IC6 has been moved to the top, keeping all the signal components in a row from left to right, with just the regulator and a few related power supply components on the underside. That makes assembly a bit easier overall. As before, top and bottom ground planes have been used with plenty of vias, and signal tracks have been kept short to avoid too much distortion of high-frequency signals.

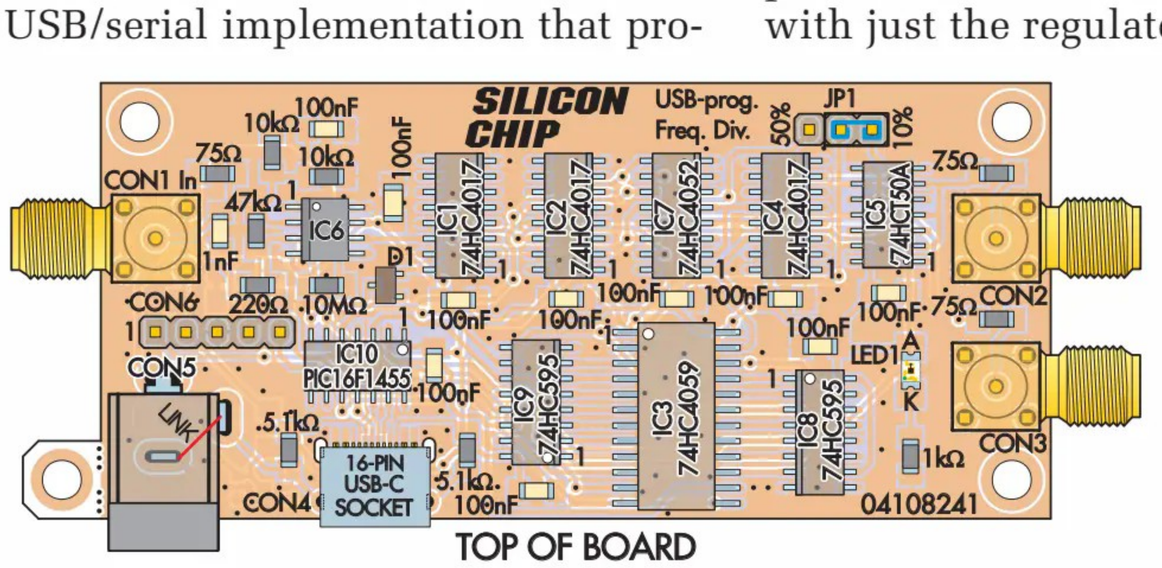
The device is built on a double-sided PCB coded 04108241 that measures 84 × 35.5mm. Fig.2 shows the components on the top side, while Fig.3 shows those on the underside. We suggest fitting all the SMD parts to one side of the board, followed by the other, then the through-hole parts. It's best to start with the top, as more parts are on that side.

The passives are all M3216/1206 size at 3.2 × 1.6mm and all the ICs are SOIC/SOP types with a relatively large 1.27mm lead spacing. The only slightly tricky part is the USB socket, as it has a row of fairly closely spaced pins. It has two mounting pins that go into through-holes, so insert it first and push it down fully, then tack-solder one of the small end pins.

Check that all the pins are lined up over their pads; if not, remelt that solder and gently nudge it one way or the other until they are all lined up. Then spread a thin layer of good-quality flux paste over all the pins, clean the soldering iron tip, add some fresh solder and drag it over the row of pins. You may need to add more solder if you run out partway.

If a few joints get too much solder, resulting in a bridge to adjacent pins, remove the bridges using a bit more flux paste and an application of clean solder wick, sliding it away from the pins when it starts to draw up the solder.

Clean off the flux residue using a solvent and use a magnifier and good light to check that the solder joints on all the pins look good and there are no



Figs.2 & 3: most of the components are SMDs (on the large side) and virtually all mount on the top of the PCB. Those on the underside are basically just the power supply. You could use vertical SMA connectors if you wanted to; CON2 is not strictly required, as CON3 is the main output. CON2 usually provides a squared-up version of the input signal.

more bridges. If so, solder the mounting posts into their holes to give it mechanical strengths.

Now move onto the ICs, soldering them similarly. But first, a word of warning. I was very proud of how neatly I soldered the ICs onto the board, until I realised that I had put them all in with pin 1 at upper left – that's not how the board is designed! Many of the ICs have their pin 1 towards the bottom of the board, so I had to use a hot air station to remove and then resolder them. Don't make that mistake!

In each case, place the IC, check its orientation (!), tack-solder one pin, check the alignment of all the pins and adjust if necessary. Then add flux paste, solder the remaining pins and remove any bridges. Once all the ICs are in place, clean off the flux residue, as it's much easier to do it in stages. Chemtools' Kleanium Deflux-It G2 Flux Remover is our preferred solvent but pure alcohol will also work.

With all the ICs in place, solder diode D1 next, then all the passives on the top side of the board, referring to Fig.2. There is just one 1nF capacitor on the top of the board, next to CON1; all the other capacitors on this side are 100nF types. The resistors will be printed with codes indicating their values; see the parts list if you are unsure about that.

Next, use a DMM on diode test mode to carefully probe the ends of the SMD LED until it lights up. The red probe will be on the anode (A) and black on the cathode (K), so use that information and Fig.2 to orientate it correctly before soldering it in place. Give the PCB another clean to remove flux residue, then flip it over.

Underside parts

Solder REG1 as shown in Fig.3, then ZD1 and Q1, being careful not to get them mixed up as they will look similar (if you do, check the markings and compare them to the codes in the parts list). There is one 1µF capacitor on this side; the rest are 100nF. Both resistors are 10kΩ types.

Clean off the flux residue on the underside, then finally move on to the through-hole parts. Those are CON1-CON3, CON5 and JP1. You may also want to fit pin header CON6 if you need to program IC10 in-circuit. Note that you could leave SMA connector CON2 off if you don't need or want

Parts List – Programmable Frequency Divider

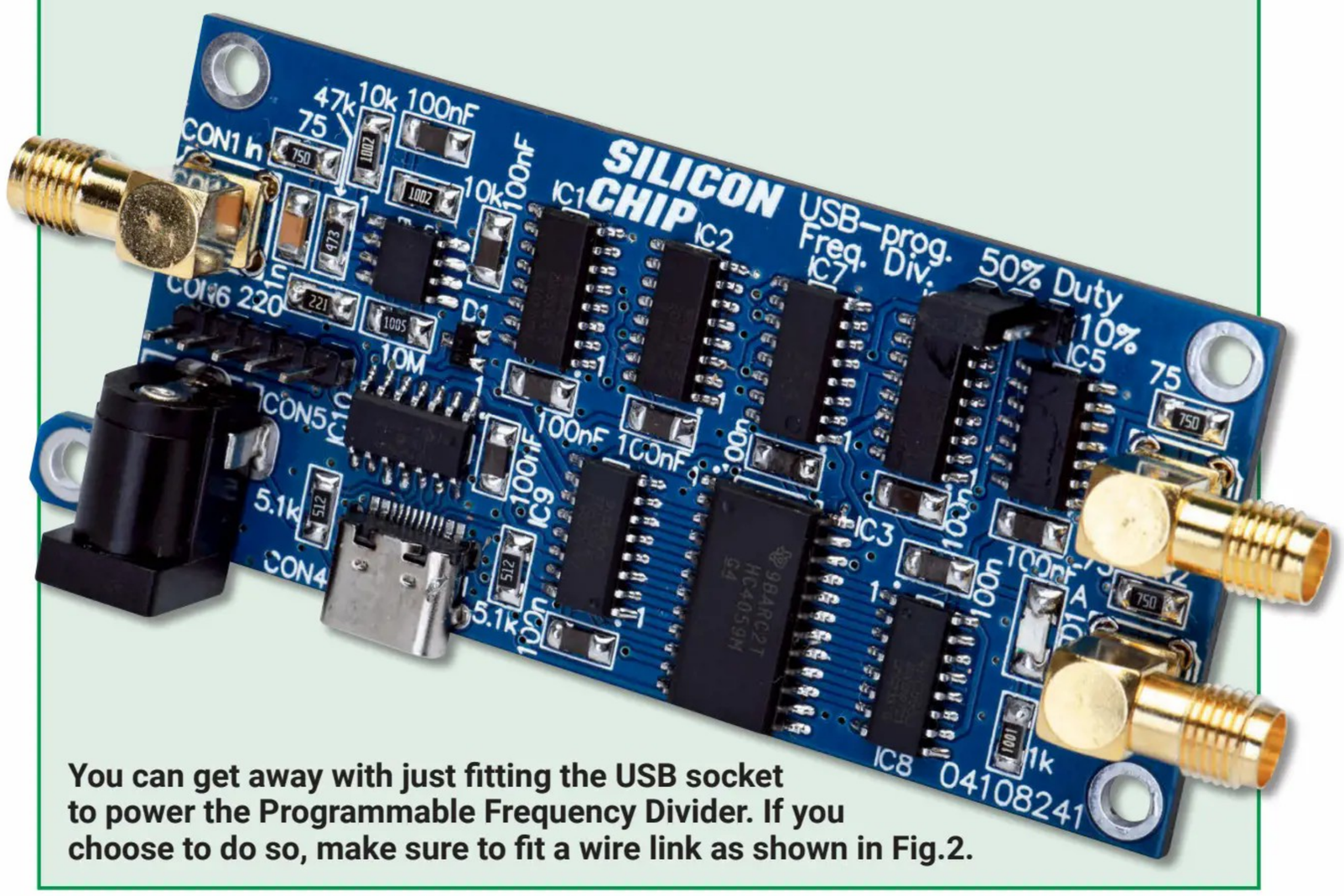
1 double-sided PCB coded 04108241, 84 × 35.5mm
3 right-angle or vertical through-hole SMA connectors (CON1-CON3)
1 SMD USB Type-C power plus USB 2.0 data socket (CON4) **[GCT USB4105]**
1 PCB-mount DC barrel socket (CON5; **optional**)
1 5-pin header, 2.54mm pitch (CON6; **optional**, programming IC10 in-circuit)
1 3-pin header, 2.54mm pitch (JP1)
1 jumper shunt (JP1)
1 6-pin stackable header (only needed for programming IC10 in-circuit)

Semiconductors
3 (CD)74HC4017(M96) CMOS Johnson decade counters, SOIC-16 (IC1, IC2, IC4)
1 (CD)74HC4059 high-speed CMOS programmable divide-by-N counter, SOIC-24 (IC3)
1 MC74VHCT50A hex CMOS non-inverting buffer, SOIC-14 (IC5)
1 TLV3501AID rail-to-rail high-speed comparator, SOIC-8 (IC6)
1 74HC4052 dual CMOS 4-to-1 analog multiplexer, SOIC-16 (IC7)
2 74HC595 8-bit serial-to-parallel shift registers, SOIC-16 (IC8, IC9)
1 PIC16F1455-I/SL 8-bit microcontroller programmed with 0410824A.HEX, SOIC-14 (IC10)
1 AMS1117-5.0 or compatible 5V 1A low-dropout regulator, SOT-223 (REG1)
1 AO3400 30V 5.8A N-channel logic-level Mosfet or equivalent, SOT-23 (code XORB) (Q1)
1 SMD LED, SMA/M3216/1206 size, any colour (LED1)
1 BZX84C5V6 5.6V 1% tolerance zener diode, SOT-23 (code YX) (ZD1)
1 BAT54S dual series schottky diode, SOT-23 (code KL4 or L44) (D1)

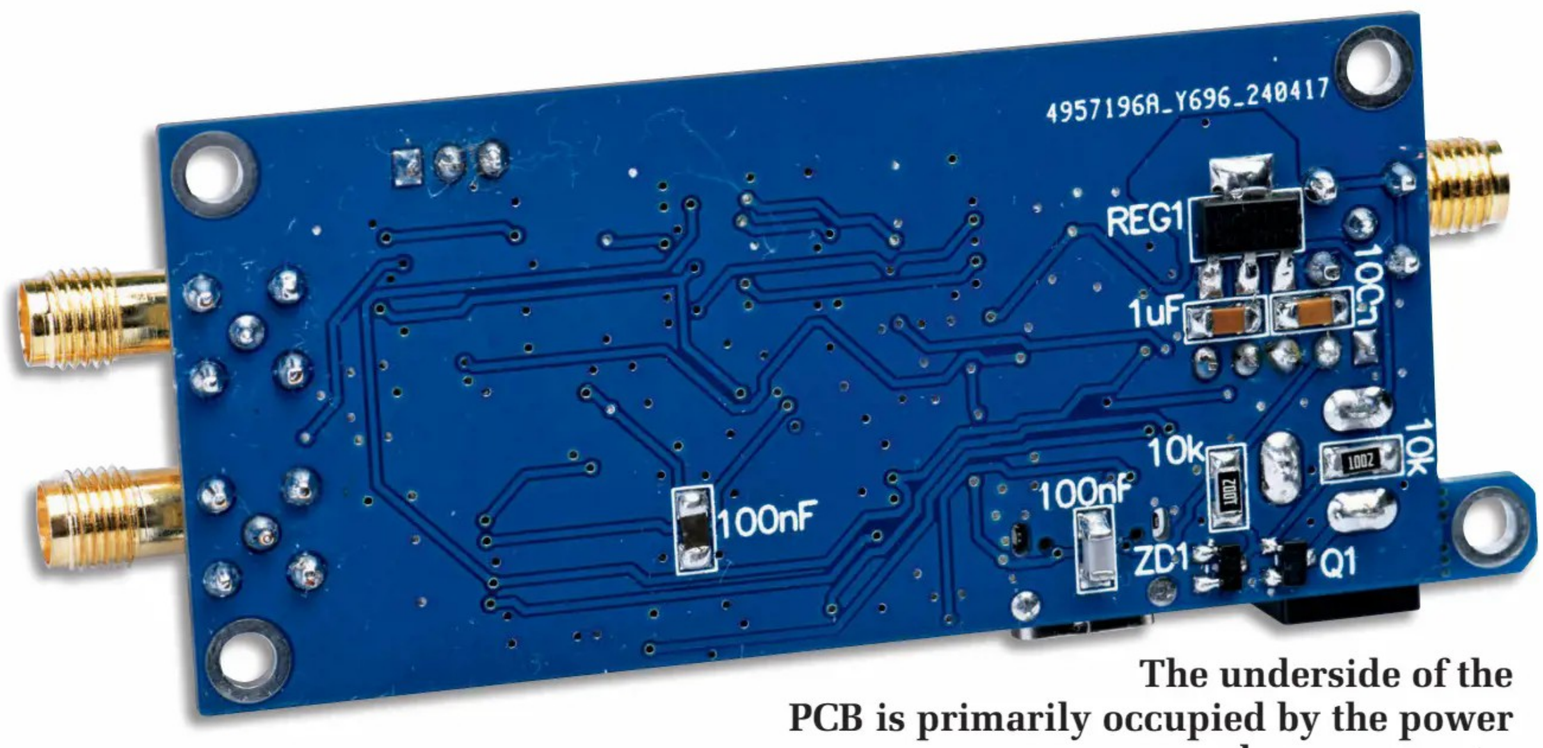
Capacitors *(all SMD M3216/1206 size 50V X7R unless noted)*
1 1µF
13 100nF
1 1nF

Resistors *(all SMD M3216/1206 size 1%)*
1 10MΩ (code 106 or 1005)
1 47kΩ (code 473 or 4702)
4 10kΩ (code 103 or 1002)
2 5.1kΩ (code 512 or 5101)
1 1kΩ (code 102 or 1001)
1 220Ω (code 220 or 220R)
3 49.9Ω (code 49R9) or 75Ω (code 75R or 75R0) (to suit impedances)

Silicon Chip can supply a kit (SC6959, ~£36 + P&P). It includes all components listed in the parts list except the programming header.



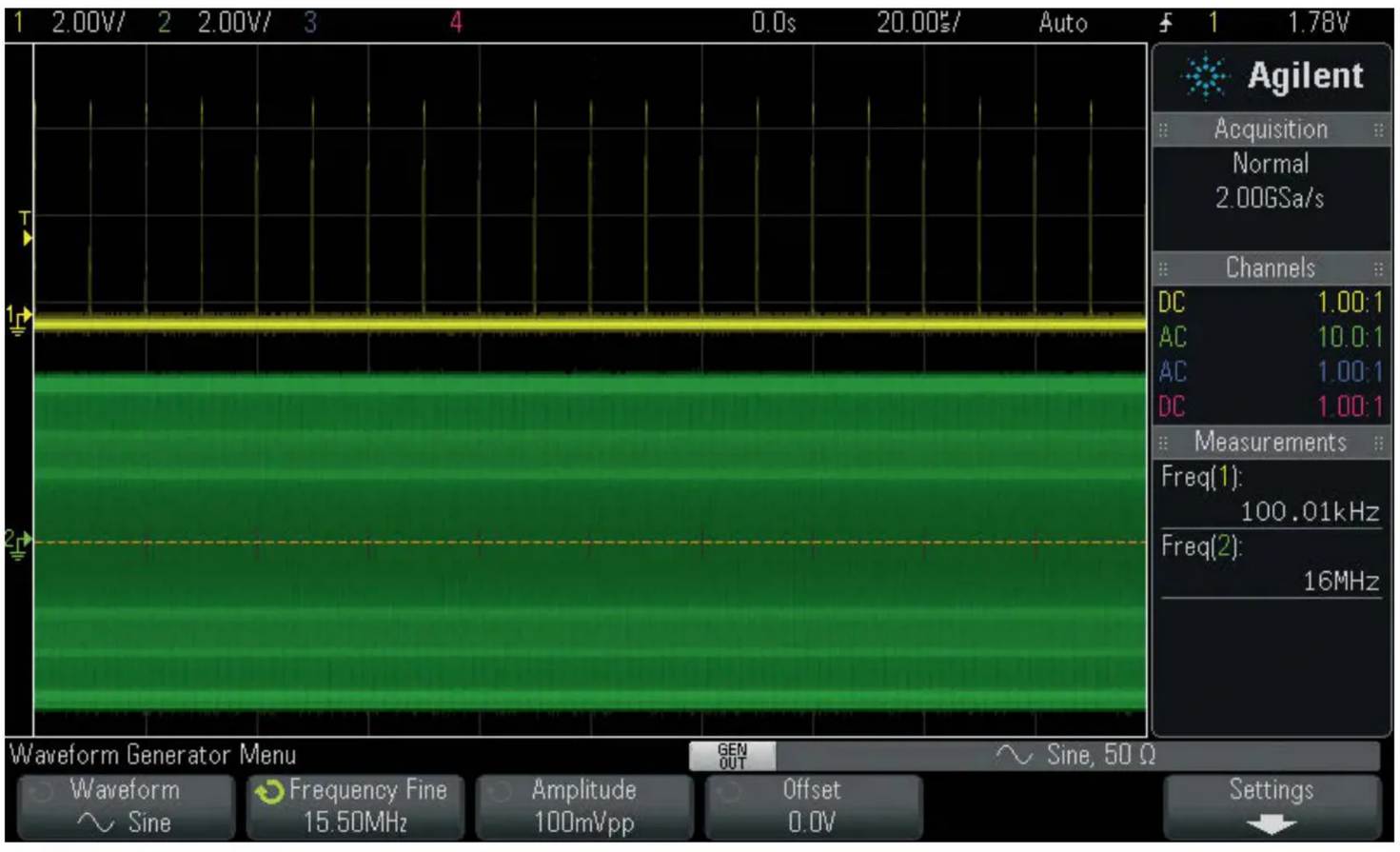
You can get away with just fitting the USB socket to power the Programmable Frequency Divider. If you choose to do so, make sure to fit a wire link as shown in Fig.2.



The underside of the PCB is primarily occupied by the power supply components.



Scope 1: the CON2 (green) and CON3 50% duty cycle (yellow) outputs for a 20MHz 100mV sinewave input and with the default 1000:1 division ratio.



Scope 2: the CON2 (green) and CON3 '50% duty cycle' (yellow) outputs for a 15.5MHz 100mV sinewave input and with a 155:1 division ratio. Because it isn't a multiple of 10, the output pulses from CON3 are not 50% duty cycle but instead are short (1/155 or 0.65% duty). Despite that, the 'scope has no trouble measuring the frequency.

the extra output that usually provides a buffered, squared-up version of the input signal (or with its frequency divided by 10 or 100 for higher programmed division ratios).

If you only want to power it using the USB socket and you will leave CON5 off, you need to solder a wire link as shown in red on Fig.2 or there will be no ground connection for the USB socket. If fitting CON5, do not fit that wire link.

Testing

Assuming IC10 is already programmed, the easiest way to test the board is to connect it to your computer using a USB-C cable. If your computer detects a new USB serial port and LED1 lights, things are looking good. If not, unplug it quickly and check for faults like bad solder joins, especially on the USB socket and IC10.

If the board isn't behaving, common problems to look for are solder bridges, pins where the solder hasn't adhered to the PCB pad below, or incorrectly orientated ICs (we did warn you!).

If the serial port does appear, use a program like Tera Term pro to connect to it (the baud rate is unimportant). Type "s" and press Enter/Return and you should get a similar status report:

```
J=292 K=4 P=10 ratio=1000
```

The default division ratio is 1000 but you can change it with the 'r' and 'w' commands.

If you have an oscilloscope with a built-in waveform generator (or separate signal generator), or a signal generator and frequency counter, hook them up to CON1 & CON3 and verify that the output frequency from CON3 is 1/1000th that of the input at CON1. If there's no output signal, carefully check all the parts on the board for correctness and good soldering.

If there's an output but the frequency is wrong, that suggests a problem with the soldering of either IC3, IC8, IC9 or possibly IC7/IC10. If the ratio is correct, hook it up to a computer via USB and try changing the ratio to values like 3, 30, 30,000 and 300,000 by typing "r", then the ratio, then pressing enter. The new ratio should be applied immediately.

Scope 1 shows both outputs for a 20MHz input and 1000:1 division ratio. You can see that the 50% duty cycle output is nice and square and very accurate, while the CON2 output

Programmable Frequency Divider

has frequency matching the input signal.

Scope 2 shows the '50% duty cycle' output for a 15.5MHz input and division ratio of 155:1. Each positive pulse is the length of one input cycle. So in this mode, the higher the division ratio, the lower the duty cycle.

Usage

You can get help on the available commands by typing "h" and then pressing enter (see below). Besides the ratio-setting command "r" mentioned immediately above, the most useful commands are "w" <Enter> (which saves the current ratio as the default at power-up) and "m" <Enter> which measures and then displays the input frequency. Low-frequency measurements may take a second or so.

Listed just below are 16 measurements we made in a row of a 20MHz source:

- 19996544
- 19999168
- 19999472
- 19997136
- 19999040
- 19997824
- 20005056
- 20001968
- 19998656
- 19998048
- 19997424
- 19999568
- 20001184
- 19998960
- 19996192
- 19999360

You will see that they vary a little above and below the exact frequency but they are surprisingly close given that the board has no crystal and the measurement method is fairly basic. The mean of those measurements is 19,999,100Hz (-0.011%), while the standard deviation is 2137 (0.015%).

The "p" command lets you set the J, K & prescaler (1/10/100/1000) values directly but you generally shouldn't need to use that. There are some combinations that you can set with that but not the "r" command but we don't think they are that useful; we just included it for completeness. More details on these (and other) commands follows. PE

Typing 'h' (or 'H' or '?') and pressing Enter should display a help message only that's an abbreviation of this list. We have also added a short description for each command. ►

How IC10 measures the input signal frequency accurately

To measure the input frequency, we use the programmable divider to divide it by a factor r, then measure the ratio of the result (F_{OUT}) to the 48MHz USB-derived clock (F_{USB}). We do that by counting the number of F_{OUT} pulses and F_{USB} pulses over an identical period. The actual period is not relevant, except that it determines the precision of the measurement.

Let's say the pulse counts are P_{OUT} and P_{USB}. We can then calculate the input frequency as $F_{IN} = P_{OUT} \times r \div P_{USB}$. However, as the PIC16F1455 is a low-end device with relatively little flash and RAM, we don't have space for floating-point calculations, so it's a bit trickier than just performing a multiplication and division.

8-bit Timer 0 counts pulses from the divider, while 16-bit Timer 1 counts the 48MHz USB clock pulses. Timer 0 is initiated at 255 and Timer 1 is initialised at 0 but is inactive. When the first pulse comes from the divider, Timer 0 rolls over to zero and triggers an interrupt that enables Timer 1. Both timers then count until Timer 0 rolls over again, at which point they are both paused.

There is a slight delay between the positive edge of the divider output being received and Timer 1 starting/stopping, but the code is designed for the delay to be the same in both cases, so it cancels out.

The number of times Timer 1 rolls over while the timer is active are counted, effectively extending Timer 1 from 16 bits to 24 bits, required to get an accurate result. The resulting 24-bit value is the number of 48MHz clock pulses that occurred between the 1st and 257th pulse from the divider.

We can then calculate the division ratio times 48 million (the number of USB clock pulses per second), divide it by the number of actual USB clock pulses counted, then multiply by 256.

Just before starting the counting, the division ratio is set to 80 as that is the highest multiple of ten that avoids a 32-bit integer overflow in these calculations.

If we find the 24-bit Timer 1 counter overflows during this measurement (after about 350ms), that means the frequency is below 58.6kHz. In that case, we drop the divider ratio to 3 and restart the measurement. That allows us to measure down to 2.2kHz.

If it still overflows, we change the number of Timer 0 clock pulses to measure over to one and try again. That would theoretically let us measure down to 9Hz, although measurements get pretty inaccurate below 300Hz due to hardware limitations.

We perform 16 measurements and average them to get a more stable reading, although the number of measurements is reduced to four or one if the frequency is determined to be on the low side (eg, close to one second per measurement). Otherwise, making that many measurements would take too long.

```
L             load ratio from flash memory
this is automatically done at power-up, but if you have changed the
ratio and wish to reset to the default, you can run this command
M             measure frequency
it's most accurate over the range of about 1kHz to 50MHz but will work
from 300Hz up to the upper limit, which is typically around 77-80MHz at
room temperature
P Jx Ky Pz    set J/K/P to x/y/z
valid ranges for values are: J[0-65535] K[0-7] P[1,10,100,1000]
the J and K numbers are fed directly to IC3, while the P (pre/
postscaler) value determines the configuration of the 74HC4052 and thus
which 74HC4017s (if any) are bypassed
Rx            set ratio to x
valid range for values is: R[3-21327000]
if the exact ratio given is not available, the closest possible ratio
is used. Remember that ratios that are not multiples of 10, or ratios
of 10 or 20, will give shorter output pulses and the 10% duty cycle
output ratio will not be correct
S             show status
this reports the J, K, P values and ratio. These are also reported
after changing or loading the ratio
W             write ratio to flash memory
the current ratio is stored and will be used at power-up in the future
(the initial default is 1000). High-endurance flash is used, so it
should not wear out after even 100,000 writes
```

Teach-In 2026



by Mike Tooley

World of Wireless – An Introduction to Radio and Wireless Technology

Series 12, part 5: antennas

Last month, we showed how digital systems can control analog oscillators, with a particular emphasis on phase-locked loops (PLL). We also introduced digital frequency synthesis and software-defined radio (SDR).

We examined different SDR solutions and show how a cost-effective SDR paired with powerful software enables you to receive a wide range of radio signals at frequencies extending from HF to UHF. Our *Hands-On* practical project involved the construction of a modular amplitude-modulated (AM) signal source.

This month, our focus shifts to antennas, covering how several popular types are built, how they work, and how well different types of antenna perform.

We'll demonstrate easy ways to construct several basic antennas and discuss why matching them properly with your equipment matters.

This part also explores antenna feeders and the voltage standing wave ratio (VSWR). Our *Hands-On* project is a handy reference dipole antenna.

Serving both transmission and reception, antennas are essential components in all radio and wireless systems. A transmitting antenna turns electrical energy into electromagnetic waves, while a receiving antenna does the reverse. According to the law of reciprocity, an antenna's gain and directional properties are the same whether transmitting or receiving.

The isotropic radiator is a theoretical antenna that serves as a reference for comparing and calculating the gain and directional characteristics of real antennas. Fig.5.1 shows how the radiation from a real antenna compares with that from an isotropic radiator.

The isotropic radiator in Fig.5.1(a) radiates uniformly in all directions. In other words, when placed at the centre of a sphere, the antenna would uniformly illuminate the internal surface. While this antenna can't be realised in practice, it's used as a reference for antenna analysis.

The real antenna in Fig.5.1(b) has directional properties and a radiation pattern that illuminates a much smaller area of the spherical region. Concentrated energy radiated in a specific direction implies an apparent gain. The energy supplied to an isotropic source would have to be correspondingly greater

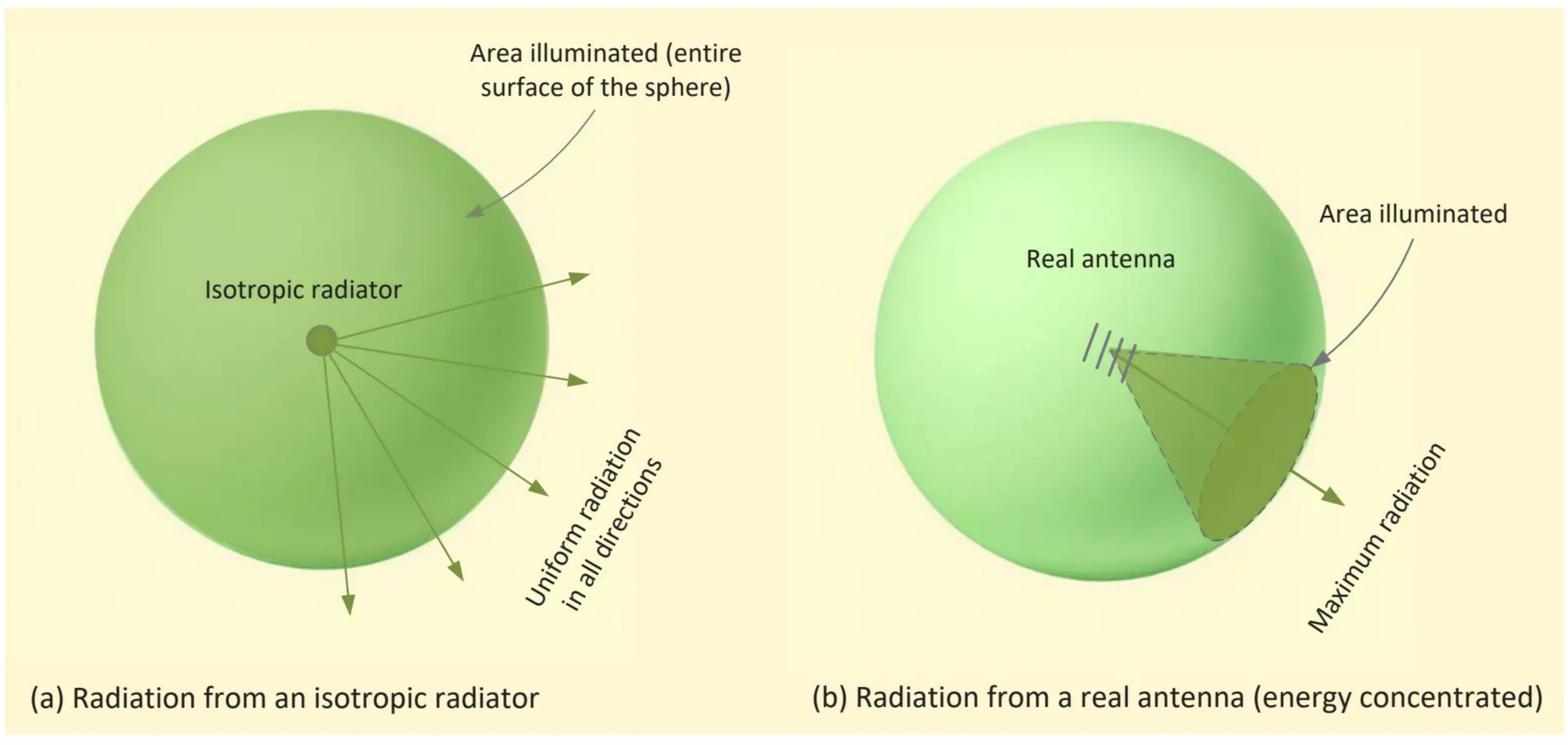


Fig.5.1: a comparison of an isotropic radiator (a) with a real antenna (b).

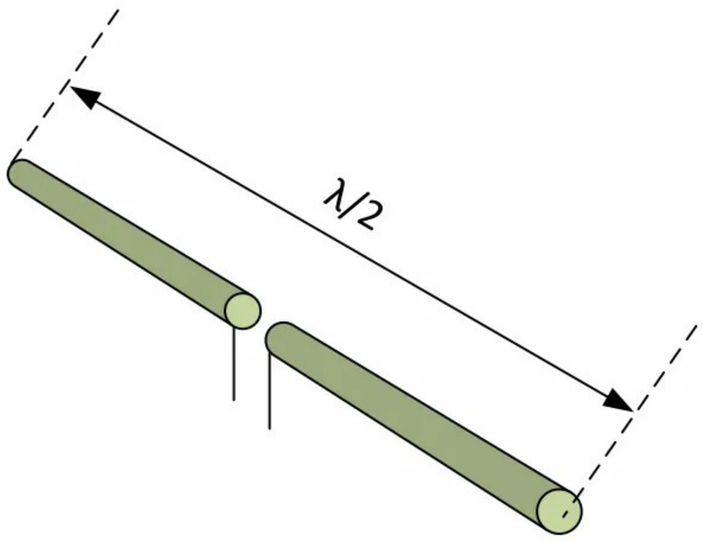


Fig.5.2: a half-wave dipole antenna.

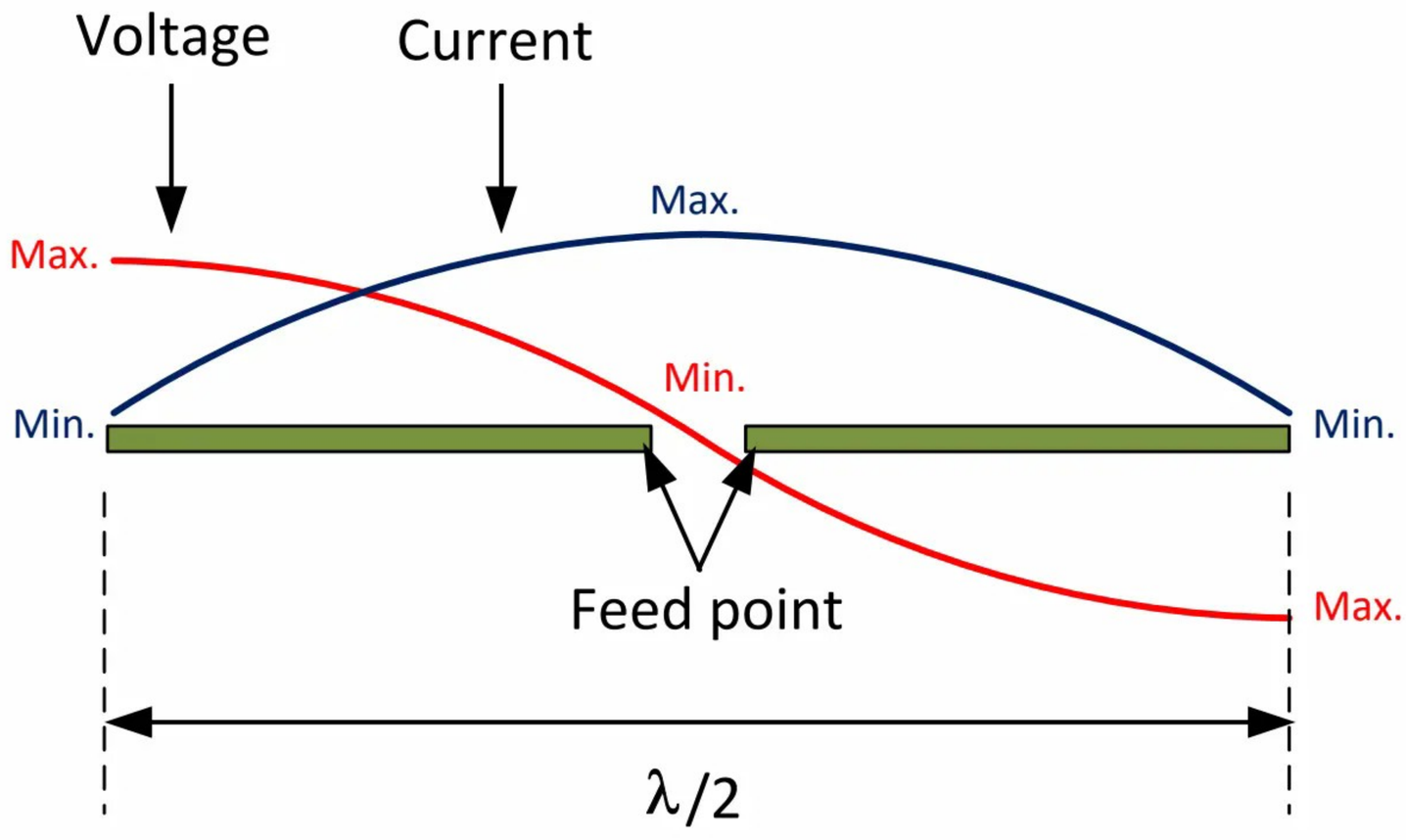


Fig.5.3: the voltage and current distribution in a half-wave dipole antenna.

to maintain the same field strength at a particular point.

To understand how antennas work (and why they sometimes don't work as expected), it's important to have a basic grasp of antenna theory. Many complex antenna types are merely refinements of one or another basic form of antenna. We shall start by describing one of the most fundamental types: the half-wave dipole.

Half-wave dipole antenna

The basic half-wave dipole antenna (Fig.5.2) consists of a single conductor split in the centre, having an overall length equal to one-half of the signal wavelength. The split at the centre is used for the feeder connection, which may be coaxial cable or twin-feeder (explained later).

An approximation of antenna length in metres can be found by dividing 150 by the frequency (in MHz) of the wave being transmitted. The relationship between the overall length of a half-wave dipole, l, (in metres) and its resonant frequency, f, (in MHz) is $l = (150 \div f)$m.

In practice, because of the capacitance effects between the ends of the antenna and the ground plane, the antenna is invariably cut a little shorter than a half wavelength, meaning the physical length of the antenna is less than its electrical length.

When determining the optimum dimensions, a correction factor, k, is applied. End effects, or capacitance at the ends of the antenna, require that we reduce the actual length of the antenna by about 4-6%, so k usually ranges from 0.96 to 0.94.

At frequencies below 3MHz, k is often close to 0.96, whereas in the range of 3-30MHz, k falls to about 0.95. Above 30MHz, k is usually 0.94, or less. The formula then becomes $l = (150k \div f)$m.

For VHF and UHF, where wavelengths are very short, it is more practical to work in centimetres, rather than in fractions of a metre. The length of a half-wave dipole antenna can then be calculated as $l = (15000k \div f)$cm.

Current and voltage distribution

Fig.5.3 shows the distribution of current and voltage along the length of a half-wave dipole antenna. The current is at a maximum in the centre and zero at the ends. Conversely, the voltage is zero at the centre and maximum at the ends.

The dipole antenna in Fig.5.3 has the directional properties illustrated in Fig.5.4. Fig.5.4(a) shows the radiation pattern of the antenna in the plane of the antenna's electric field (parallel to the ground), while Fig.5.4(b) shows the radiation pattern in the plane of the antenna's magnetic field (vertical).

In Fig.5.4(a), minimum radiation occurs along the axis of the antenna, while the two regions of maximum radiation are at 90° (perpendicular) to the two dipole elements. In the other (orthogonal) plane, Fig.5.4(b), the antenna radiates uniformly in all directions.

The important conclusion from Fig.5.4 is that a horizontal dipole will have a bidirectional characteristic, whereas a vertical dipole will have an omnidirectional radiation pattern. In three–dimensional space, the combined effect of these two patterns will be a doughnut shape,

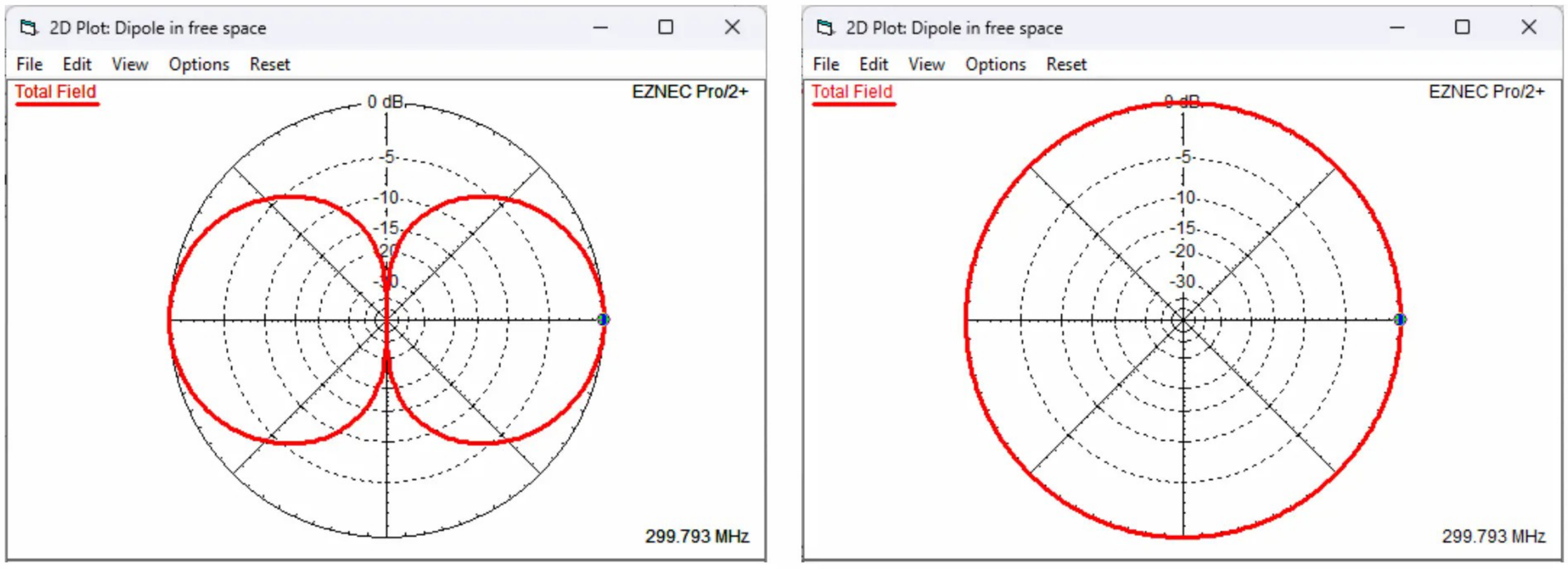


Fig.5.4: horizontal/azimuth (left) and vertical/elevation (right) plots for a vertical half-wave dipole in 'free space'. These idealised plots were made using a popular antenna modelling package. If it isn't obvious how this relates to the antenna, refer to Fig.5.5.

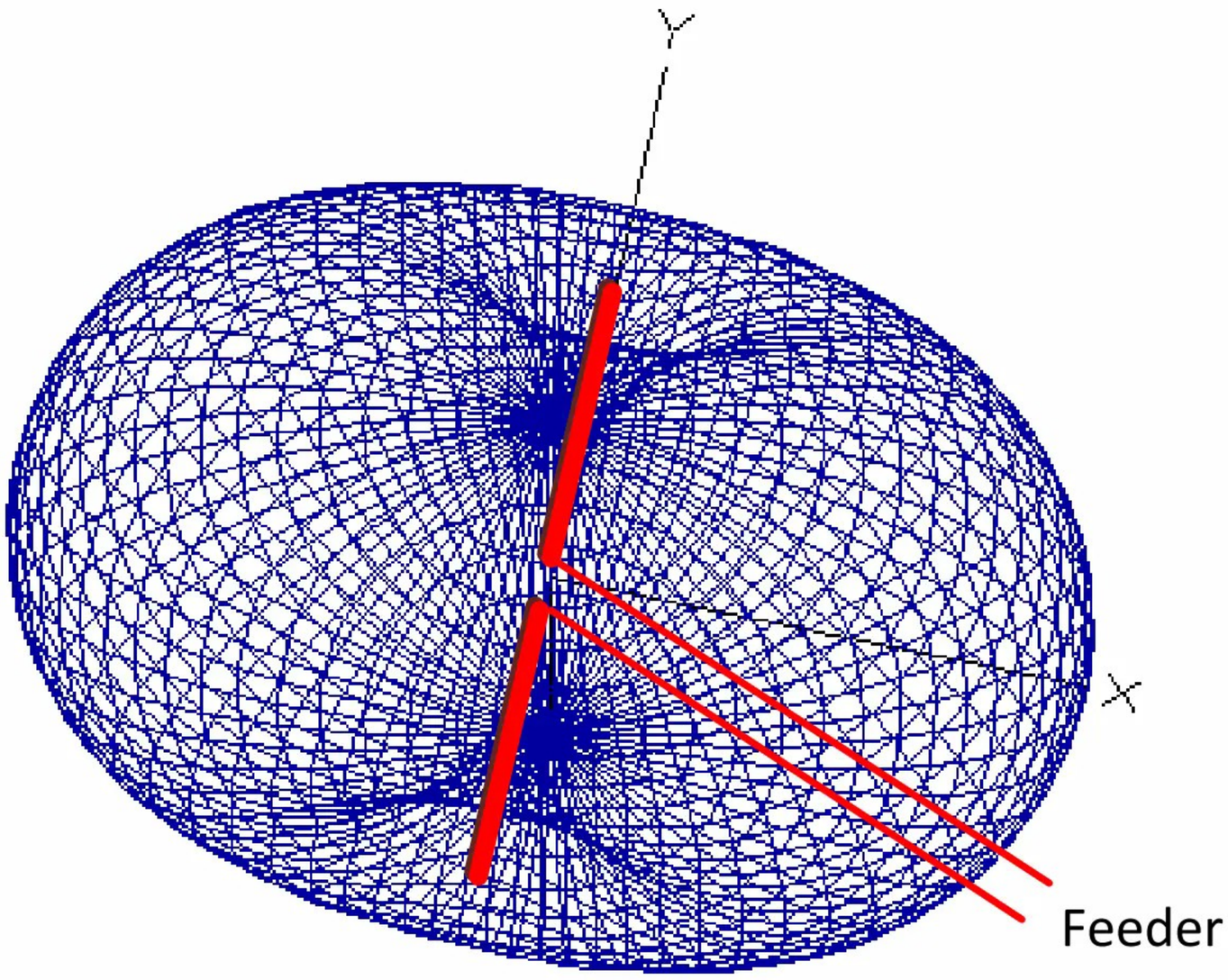


Fig.5.5: 3D radiation pattern produced by popular antenna modelling software.

illustrated by modelling software in Fig.5.5.

You will be able to verify this from this month's *Hands-On* project! Note that a horizontal dipole will radiate horizontally polarised waves, whereas the radiation from a vertical dipole will be vertically polarised (recall from Part 1 that electromagnetic waves are said to be polarised in the direction of the electric field).

Monopole antennas

A vertically mounted dipole antenna providing omnidirectional radiation characteristics is desirable in scenarios where uniform signal coverage is required. Unfortunately, mounting a dipole in the vertical plane can sometimes have problems due to the proximity of a conductive mounting structure.

An alternative solution is a vertical quarter-wave antenna in conjunction with a reflecting ground plane, effectively cutting the antenna in half, as shown in Fig.5.6. Such antennas produce an omnidirectional radiation pattern in the horizontal plane, and radiate vertically polarised signals.

Practical quarter-wave antennas can be easily produced for VHF and UHF applications, but their height can become prohibitive at frequencies below 30MHz.

To produce a reasonably flat radiation pattern, it is essential to incorporate an effective ground plane. At HF, the ground plane can be the Earth itself. To minimise Earth resistance and increase the efficiency of the antenna, it is often necessary to incorporate some buried Earth radial conductors emanating from the grounded base of the antenna.

At VHF, quarter-wave radial elements can instead be mounted at the feed point. These can be either straight or drooping, as shown in Fig.5.7. The sloping arrangement in Fig.5.7(b) produces a slightly flatter radiation pattern. A typical quarter-wave vertical antenna with a sloping ground plane is shown in Fig.5.8. This antenna is ideal for aircraft reception in the VHF band.

Another popular alternative is shown in Fig.5.7(c). This $^{5}/_{8}{}^{th}$ wave vertical offers some gain over a basic quarter-wave antenna and behaves electrically as a three-quarter-wave antenna with the incorporation of an inductor in the radiating element.

Antenna gain

Since different antenna types produce different field strength values for the same applied RF power level, we attribute a power gain to them. To be meaningful, this power gain is specified in relation to a reference antenna, specified in decibels (dB). Reference antennas are usually either the isotropic radiator that we met previously, or a standard half-wave dipole.

Since the former type exists only in theory, the latter is usually quoted. When quoting antenna gain, subscripts *i* (isotropic) and *d* (dipole) distinguish the two types. Due to its directivity (see Fig.5.1), the half-wave dipole provides a gain of about 2.15dB when compared with that of the fundamental isotropic reference. We can thus infer that $0dB_d = 2.15dB_i$.

As an example, an antenna having

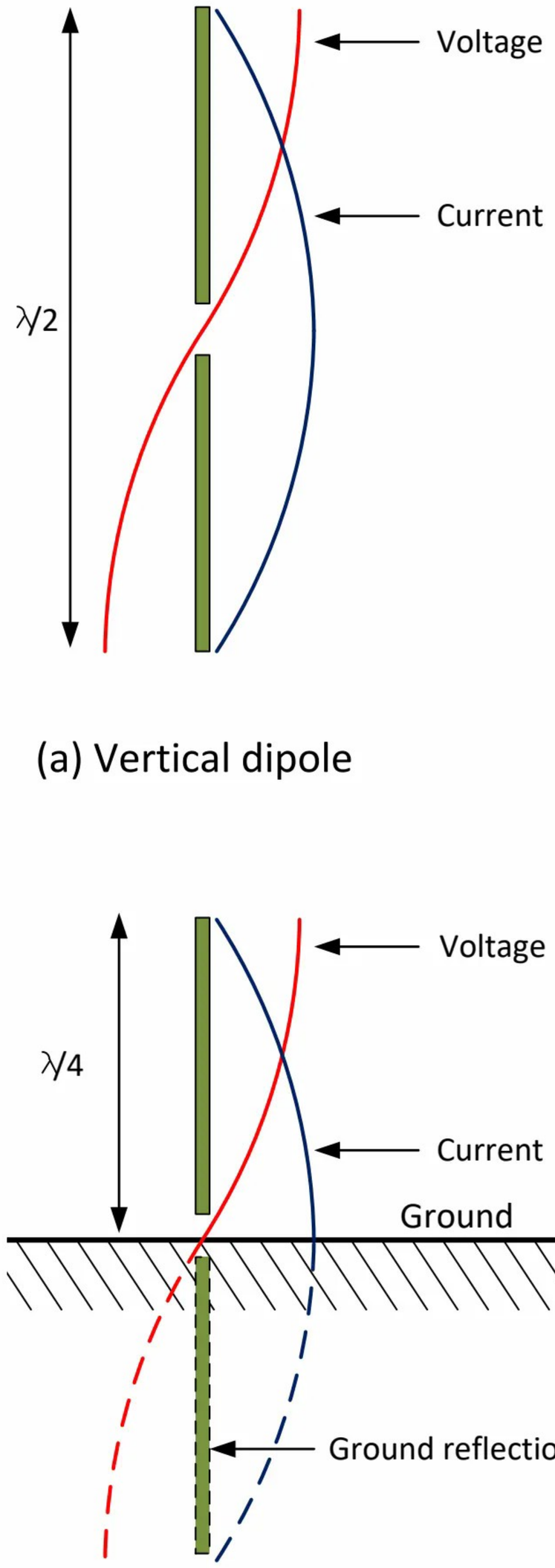
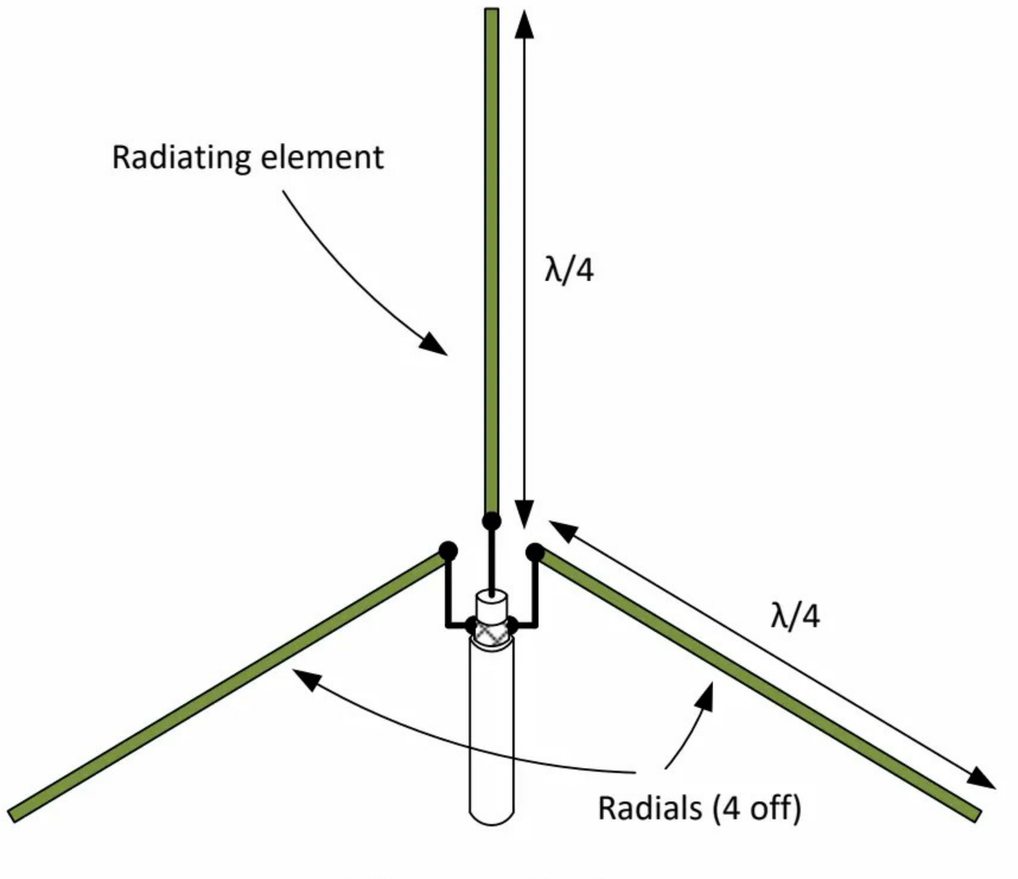


Fig.5.6: vertical dipole and vertical monopole antennas.

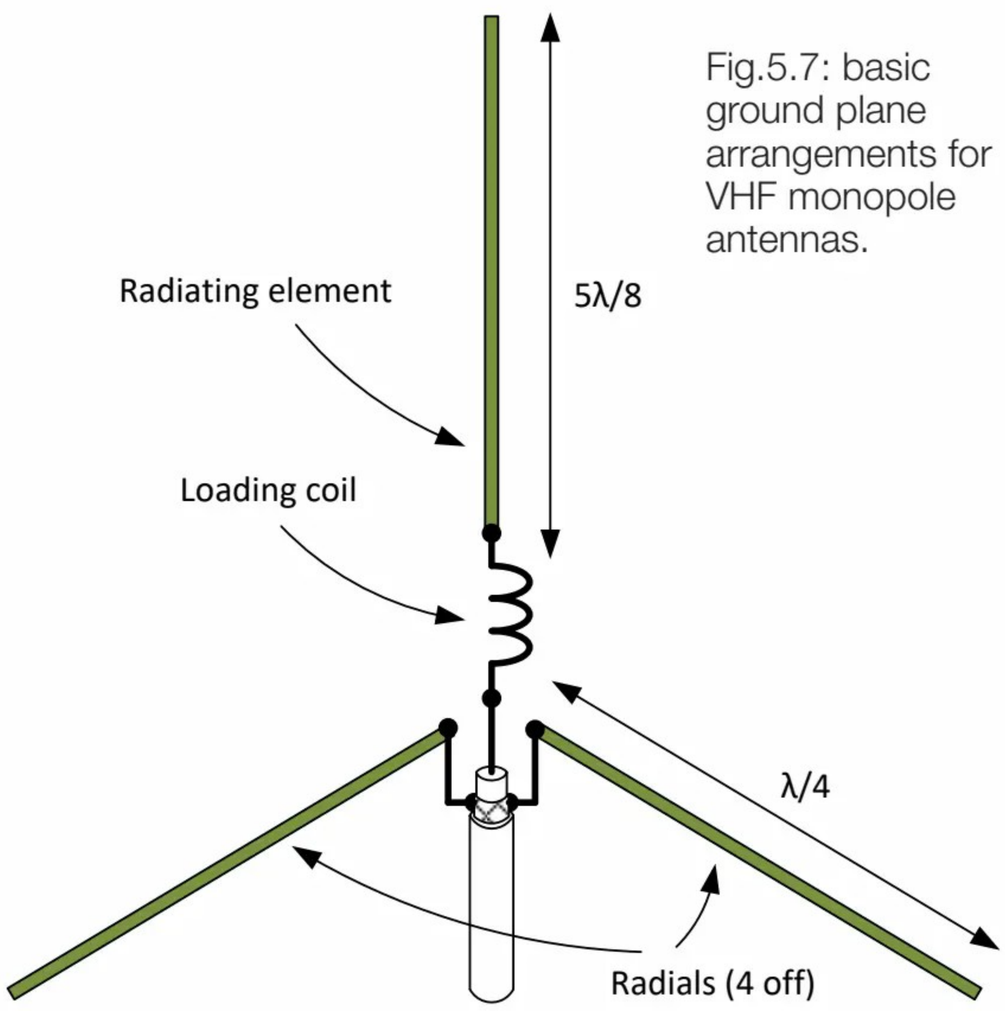
a quoted gain of $10dB_d$ is equivalent to $12.15dB_i$. Furthermore, since 10dB is a power ratio of 10:1, we can conclude that you would have to deliver ten times more power to a half-wave dipole antenna to maintain the same field strength at a given point as this example antenna.

It should come as no surprise that an antenna located on the top of a tall building will usually perform significantly better than one located at ground level. To achieve the same signal level, a higher gain antenna will be required if it must be mounted at a relatively low elevation. Note that 'height gain' only applies to an in-situ antenna and not to a particular antenna type.

(a) Quarter wave vertical with ground plane



(b) Quarter wave vertical with drooping ground plane



(c) 5/8 wave vertical with drooping ground plane

Fig.5.7: basic ground plane arrangements for VHF monopole antennas.

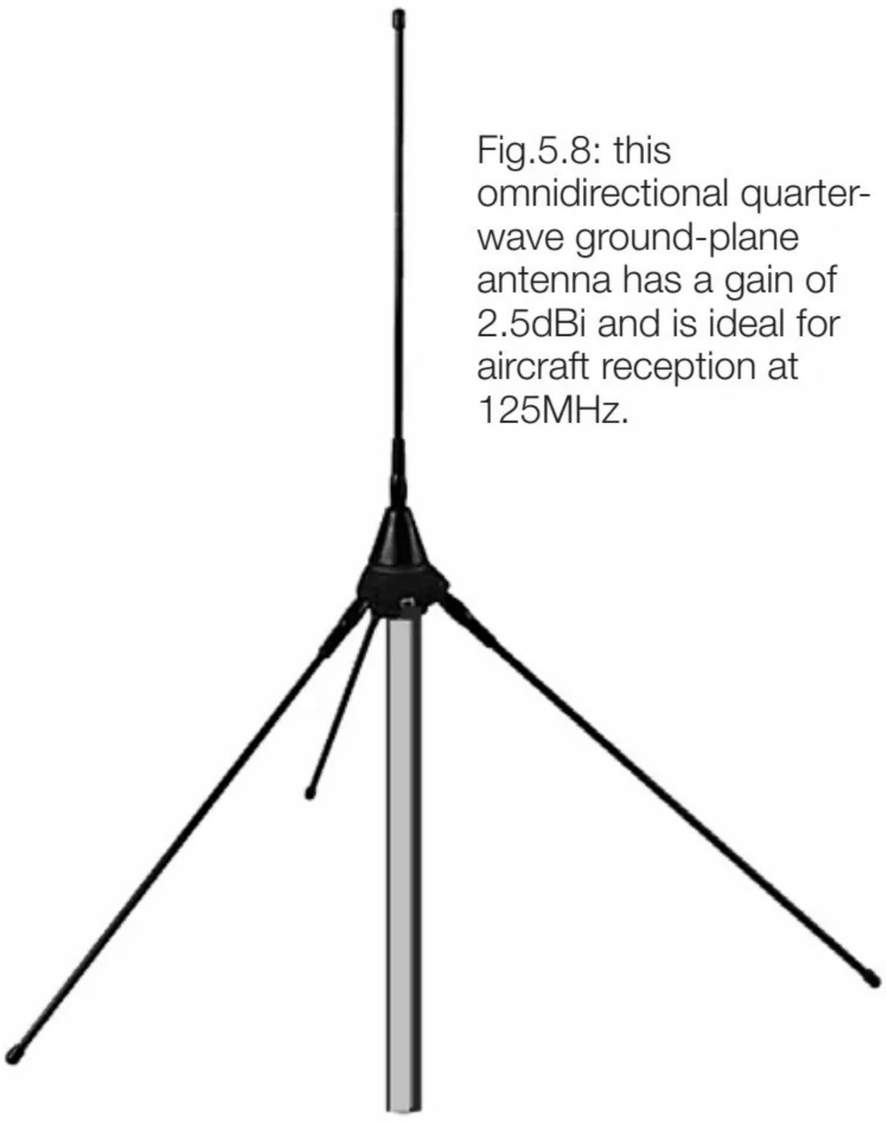


Fig.5.8: this omnidirectional quarter-wave ground-plane antenna has a gain of 2.5dBi and is ideal for aircraft reception at 125MHz.

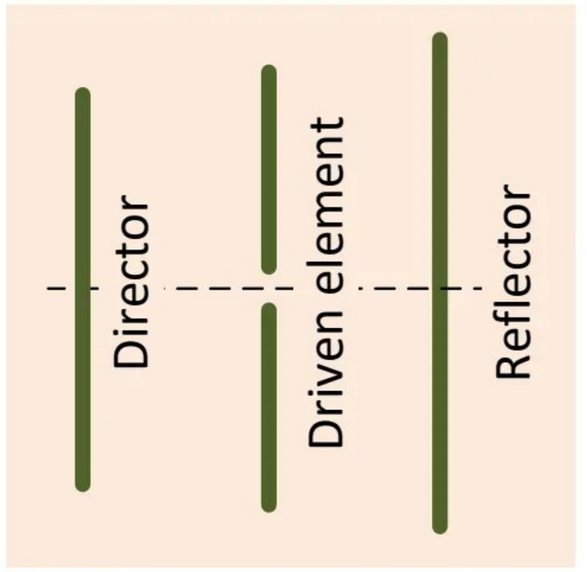
Multi-element antennas

Invented by Japanese scientists Yagi and Uda, the Yagi beam array has remained extremely popular for use in applications such as terrestrial TV, FM radio and long-range Wi-Fi.

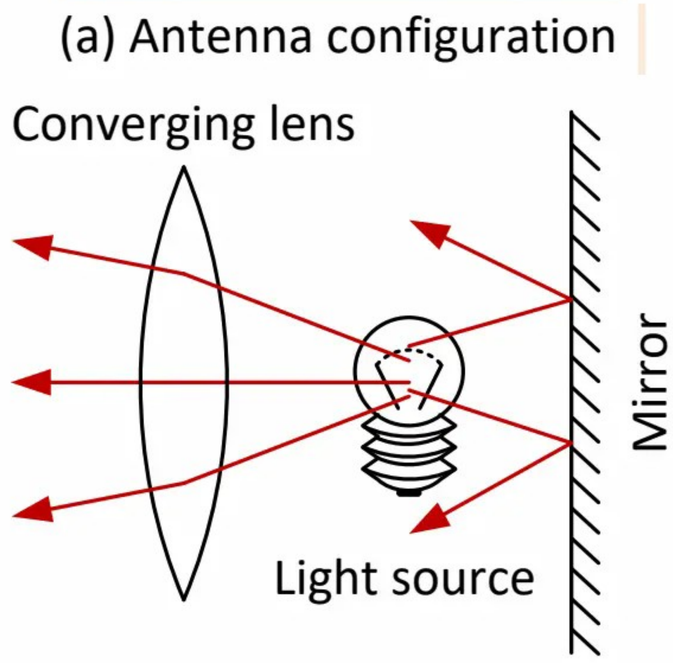
To explain in simple terms how the Yagi antenna works, we shall use the analogy shown in Fig.5.9, where an ordinary filament lamp radiates electromagnetic energy in the form of visible light. Just like an antenna, the filament lamp converts electrical energy into electromagnetic energy. The main difference is that we can see the energy that it produces!

To concentrate the light radiated from the filament lamp (which would otherwise radiate in all directions), we can place a reflective mirror surface behind it. Here, the radiation is reflected while undergoing a 180° phase change, reinforcing the light on one side of the filament lamp.

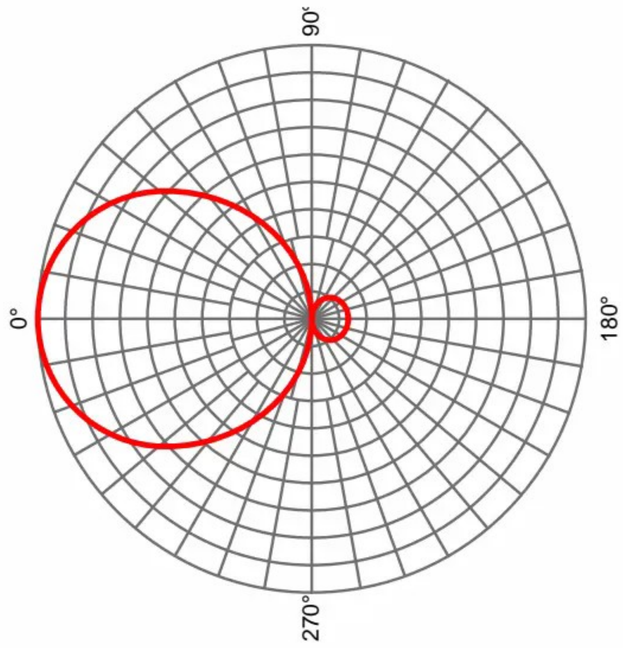
To achieve the same effect in our antenna system, we need to place a conducting element about one quarter of a wavelength from the dipole element. This element is referred to as a reflector,



(a) Antenna configuration



(b) Light analogy



(c) Directional pattern

Fig.5.9: how a Yagi antenna works by analogy with focusing the output of a light bulb (light is also a form of electromagnetic radiation).

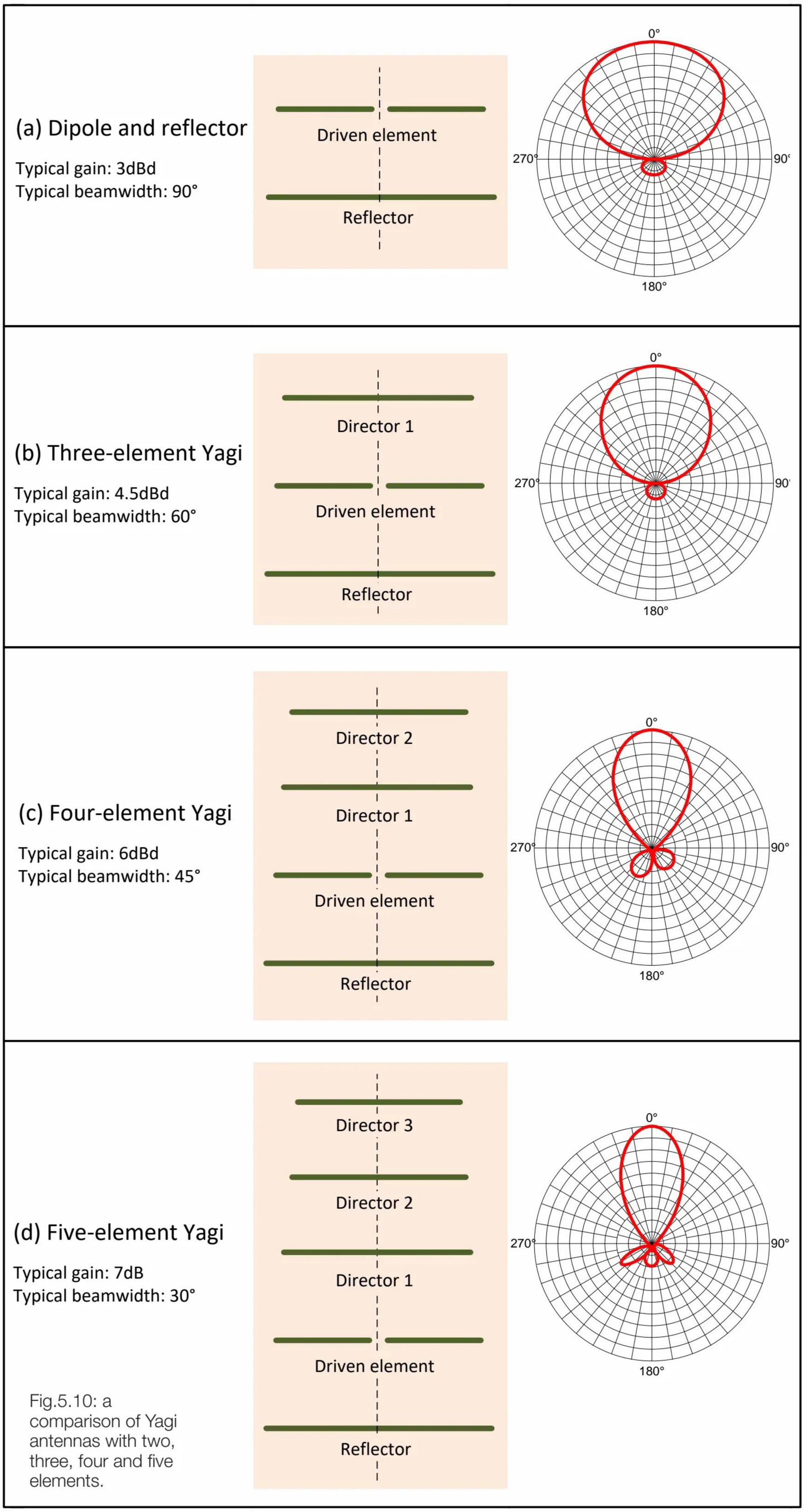


Fig.5.10: a comparison of Yagi antennas with two, three, four and five elements.

and it is parasitic (ie, not actually connected to the feeder). The reflector needs to be made slightly longer than the driven dipole element. The resulting directional pattern now has only one major lobe.

To further concentrate the light energy into a narrow beam, we can add a lens to the optical system. To ensure that the light emerging from the optical system is bent towards the normal line, the filament lamp must be positioned at the focal point of the lens.

To achieve the same effect in our antenna system, we need to place a conducting element on the other side of the dipole and about one quarter of a wavelength from it. Once again, this element is parasitic but, in this case, it needs to be cut slightly shorter than the driven dipole element. This element is called a director.

The resulting directional pattern will now have a narrower major lobe as the energy becomes further concentrated at right angles to the dipole elements.

To improve the directional characteristics of our optical system still further, we can simply add more lenses, each time bending the light beam further towards the normal axis. The result is a parallel beam of phase-coherent light. In the same way, we can add further directors to our antenna system so that the energy is concentrated into a narrow parallel beam.

Fig.5.10 shows how the gain is increased and beam width reduced for Yagi antennas with two, three, four and five elements. As a rule of thumb, an increase in gain of 3dB will result each time the number of elements is



Fig.5.11: the twin monopole quarter-wave VHF antennas on a Piper Saratoga 2 for navigation and VHF comms.



Fig.5.12: the author's home-built magnetic loop antenna. Used with its companion unit, it provides continuous HF coverage over 10-30MHz.

doubled. Thus, a two-element antenna will offer a gain of about $3dB_d$, a four-element array will produce $6dB_d$, an eight-element Yagi will realise $9dB_d$ etc.

When more elements are added, additional side lobes begin to appear (see Fig.5.10).

Front-to-back ratio

The front-to-back ratio is defined as the ratio (expressed in dB) between the radiation from an antenna in the wanted direction to that in the opposite direction. Because the two main lobes are equal in size, a horizontal half-wave dipole antenna (Fig.5.4) has a 0dB front-to-back ratio.

The five-element Yagi array shown in Fig.5.10(d) has a much larger front-to-back ratio of around 10dB. The larger the front-to-back ratio, the better an antenna is at rejecting signals and interference arriving from directions other than that of the wanted signal source.

Other antenna types

Many other types of antenna are in common use. Here are just a few of those you might encounter.

Helical antennas are well-suited for use in portable and handheld VHF and UHF equipment. They are inexpensive, robust and reasonably small. Gains of around $-1.5dB_d$ are common.

Ferrite antennas are commonly used in portable domestic MW and LW radio receivers. They comprise one or more inductive windings on a high-permeability ferrite rod or slab. The inductive winding is brought into resonance at the signal frequency by a parallel-connected tuning capacitor. Such an arrangement has directional properties – the direction of maximum sensitivity is at right angles to the axis of the ferrite rod.

Printed antennas are fabricated as part of a larger printed circuit board, avoiding the need for off-board connectors and external antennas. Printed antennas are restricted to low-power applications at frequencies above 1GHz. They are found in IoT and Wi-Fi devices, mobile phones, plus Bluetooth equipment.

Loop antennas (sometimes referred to as 'magnetic loops') are useful at frequencies below 30MHz. Their inherently low impedance necessitates the use of a matching and tuning unit. The gain and directivity of a properly matched loop can often approach that of a full-size half-wave dipole antenna.

Frame antennas were commonly used in domestic receivers before the advent of the ferrite rod. Instead of an inductive winding on a ferrite rod, the coil was wound on a large (invariably rectangular) former running round the inside of the case of the receiver. As with the ferrite rod aerial, the inductive winding formed part of a tuned circuit brought to resonance at the signal frequency. Again, the aerial has directional properties.

Foil, plate & telescopic antennas are used in domestic portable VHF and other receivers and, while this type of antenna may be adequate for local station reception in strong signal areas, its performance cannot rival that of an external half-wave dipole or small Yagi. The directional characteristics of such aerials are hard to predict.

Corner reflectors are an alternative to the Yagi antenna that consist of two perpendicular conductive reflecting surfaces (which may be solid or perforated to reduce wind resistance). The reflecting surfaces are mounted behind a driven element (usually a half-wave dipole). Corner reflectors provide gains and beamwidths of around $10dB_d$ and 40°, respectively.

Parabolic reflectors meet the need for very high gain coupled with a directional response at UHF and microwave frequencies. A parabolic reflector is used in conjunction with a radiating element positioned at the feed-point of the reflecting surface. To provide high gain, the diameter of a parabolic reflecting surface must be large in comparison with the wavelength of the signal. Gains of up to $46dB_i$ and beam widths as low as 5° are not unusual.

Log-periodic arrays are broadband directional antennas comprising a series of elements that gradually vary in length and spacing. The elements are arranged along the supporting boom in such a way that the electrical length and spacing between elements varies logarithmically, allowing the antenna to maintain consistent performance over a wide range of frequencies. Typical applications are wideband reception and spectrum monitoring.

Horns, like parabolic reflectors, are commonly used at microwave frequencies. Horn antennas can be used alone or as a means of illuminating a parabolic (or other) reflecting surface. They are ideal for use with waveguide feeds; the transition from the waveguide to the free space aperture is accomplished over several wavelengths, the waveguide being gradually flared out in both planes.

Figs.5.11 to 5.14 show various antennas used in different RF applications.

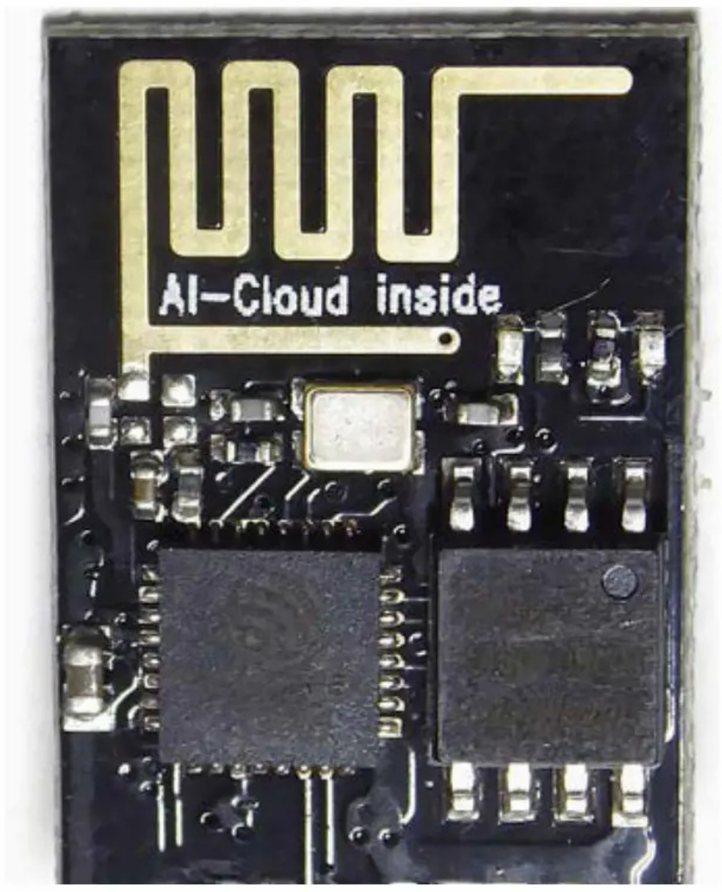


Fig.5.13: the compact printed antenna on a 2.4GHz ESP8266 Wi-Fi module has an indoor range up to about 15m.



Fig.5.14: this large horn-fed parabolic reflector antenna supports a microwave telecommunications link between remote communities in southern Greenland.

Antenna impedance and losses

Because voltage and current appear in an antenna (albeit in tiny quantities in the case of most receiving antennas), they will exhibit impedance. It's worth remembering that impedance is the combined effect of resistance, R, and reactance, X (both of which are measured in ohms [Ω]). R remains constant while X varies with frequency.

The impedance of a half-wave dipole at its design frequency is usually between 70Ω and 75Ω. At this frequency, the antenna will appear purely resistive (ie, its reactive component is zero). Fig.5.15 shows the three series-connected components that together make up the impedance of an antenna:

- the DC loss resistance, R_{DC}
- the radiation resistance, R_R
- the reactance, X (this may be inductive or capacitive)

When an antenna is operated at a frequency that lies in the centre of its passband, the off-tune reactance will be zero. We are then left with just two elements: the loss resistance and the antenna's radiation resistance. The important point to note from this is that for minimal wasted power, the loss resistance must be very much less than the radiation resistance.

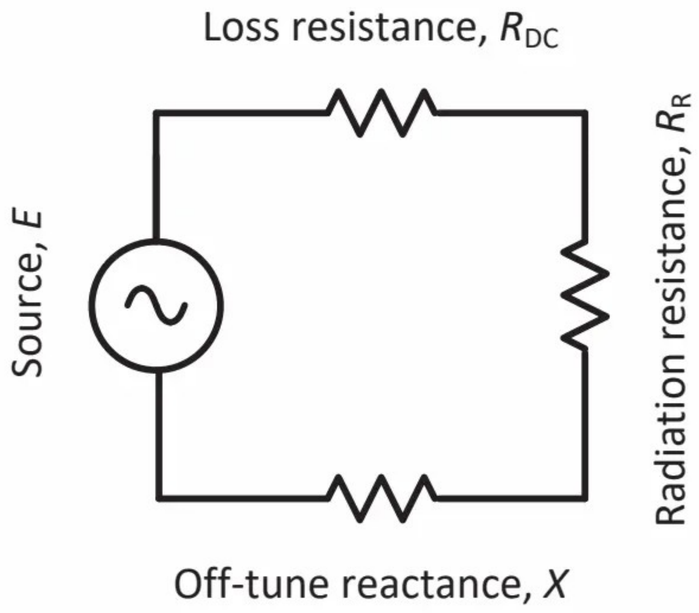


Fig.5.15: an antenna-equivalent circuit.

Feeders and cables

The purpose of a feeder line is to deliver maximum power from the transmitter to the antenna. Ideally, a feeder would have no losses (no power would be wasted in it), and it would present a perfect match between the impedance of the output stage of a transmitter and the impedance of the antenna to which it is connected.

The impedance of a feeder (called its 'characteristic impedance') is the impedance that would be seen looking into an infinite length of the feeder at its working frequency. Fig.5.16 illustrates this point.

In a feeder considered loss-free, this impedance, Z_0, is determined by the inductance, L, and capacitance, C, present in the feeder and given by $Z_0 = \sqrt{L \div C}\Omega$.

The values of L and C are referred to as the 'primary constants' of the feeder. In this respect, L is the loop inductance per unit length, while C is the shunt capacitance per unit length, as shown in Fig.5.17. In practice, there will be a small amount of loss resistance in the feeder,

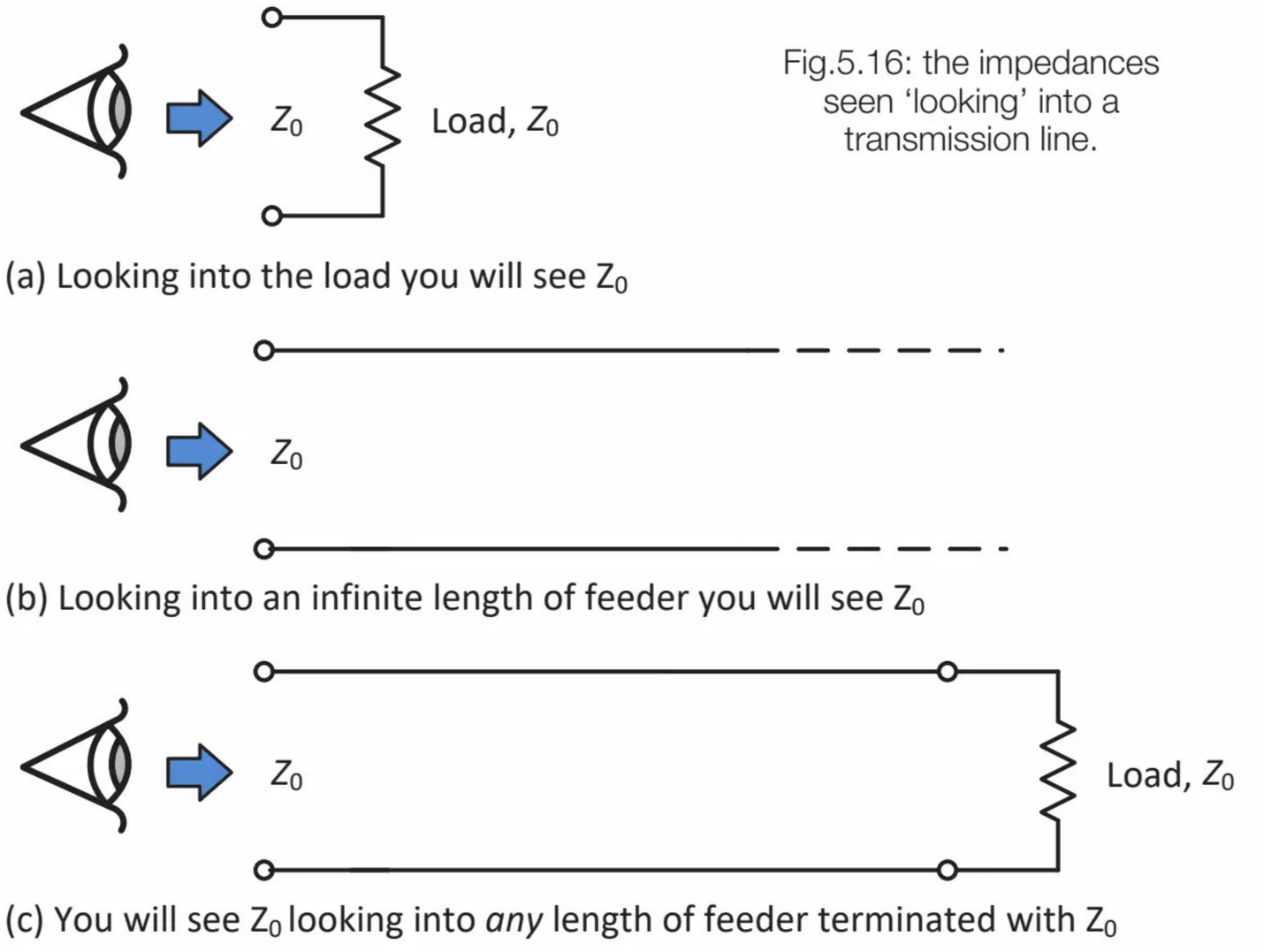
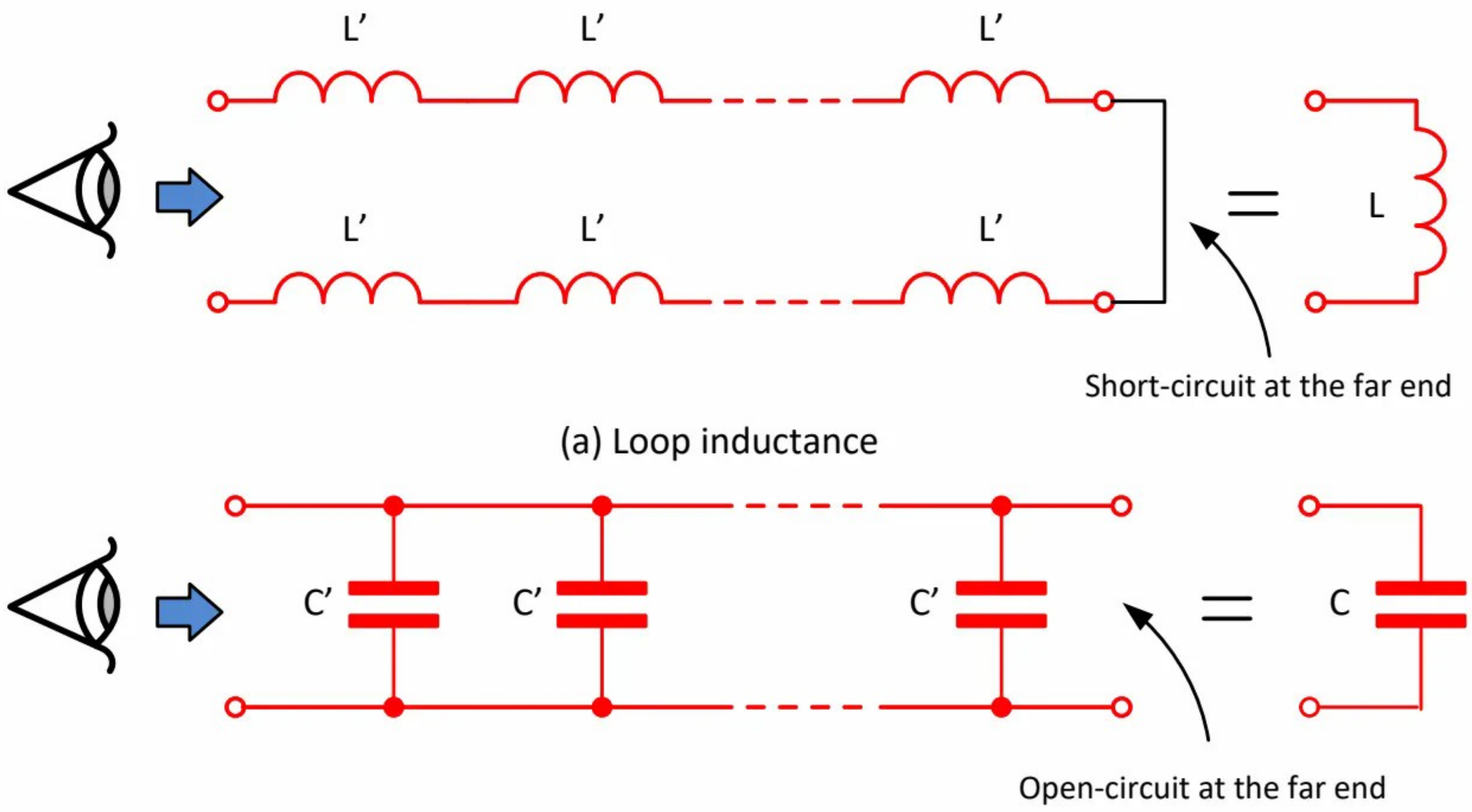


Fig.5.16: the impedances seen 'looking' into a transmission line.



(b) Loop capacitance

Fig.5.17: the primary constants of a feeder.

but this can usually be ignored.

The characteristic impedance of a cable can be easily calculated knowing its primary constants. For example, a cable with a loop inductance of 20nH and a loop capacitance of 100pF can be determined as follows (20nH = 20 × 10^{-9}H and 100pF = 100 × 10^{-12}F):

$$Z_0 = \sqrt{\frac{180\times10^{-9}}{100\times10^{-12}}} = \sqrt{1.8\times10^{3}} = 42\Omega$$

Coaxial cable

Fully screened coaxial cables are used for feeders in VHF and UHF (as well as many HF) applications. Coaxial cables have a centre conductor (either solid or stranded wire) surrounded by an outer conductor that completely shields the inner conductor, as shown in Fig.5.18.

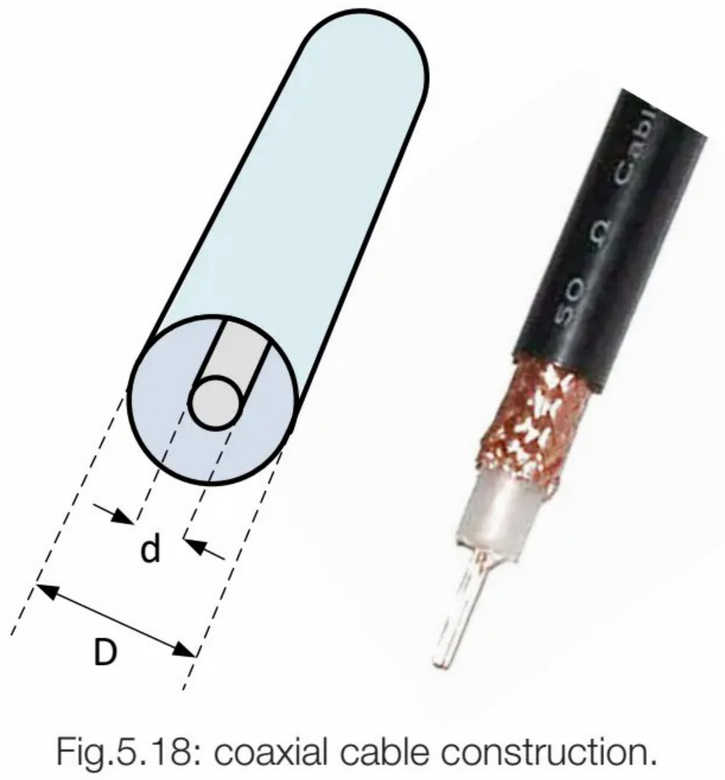


Fig.5.18: coaxial cable construction.

The two conductors are concentric and separated by an insulating dielectric that is usually air or some form of polythene. The characteristic impedance, Z_0, of such a cable is given by $Z_0 = 138\log_{10}(D \div d)\Omega$.

Here, D is the inside diameter of the outside conductor, and d is the outside diameter of the inside conductor, both in millimetres.

Thus, the characteristic impedance of a coaxial cable can be calculated if you know its dimensions. The characteristic impedance of a cable with an inside conductor diameter of 2mm and an outside conductor diameter of 10mm is calculated as:

$$Z_0 = 138\log_{10}\left(\frac{10}{2}\right) = 138\log_{10}(5) = 97\Omega$$

Table 5.1: commonly-used RF coaxial cables

Type	Characteristic Impedance	Loss (db/100m) @ 100MHz	Velocity Factor	Diameter	Typical application
LDF4-50	50Ω	~2.2dB	0.88	12.7mm	High-performance uses up to 2GHz
LMR-400	50Ω	~3.9dB	0.85	10.3mm	General purpose up to 2GHz
RG-6	75Ω	~5.6dB	0.82	6.9mm	Cable & satellite TV
RG-58	50Ω	~11dB	0.66	5.0mm	General purpose up to 500MHz
RG-59	75Ω	~7.4dB	0.66	6.2mm	Video and CCTV
RG-174	50Ω	~22dB	0.66	2.8mm	Jumper leads, short cable runs
RG-213	50Ω	~6.7dB	0.66	10.3mm	High-power HF/ VHF transmission

Velocity factor

The velocity of a wave travelling in a cable or transmission line will be less than that of the wave in free space. The ratio of the two (actual velocity and velocity in free space) is known as the *velocity factor*.

Depending on the cable construction and dielectric material, the velocity factor varies from about 0.6 to 0.95. This reduction in velocity has an impact on the wavelength in the cable. Table 5.1 provides details of commonly used RF coaxial cables.

Voltage standing wave ratio (VSWR)

Where the impedance of a feeder is perfectly matched to the load, all the energy delivered by the source will be transferred to the load. In this ideal state, the voltage will be the same at all points along the feeder. If the match is less than perfect, a proportion of the energy will be reflected, as shown in Fig.5.19.

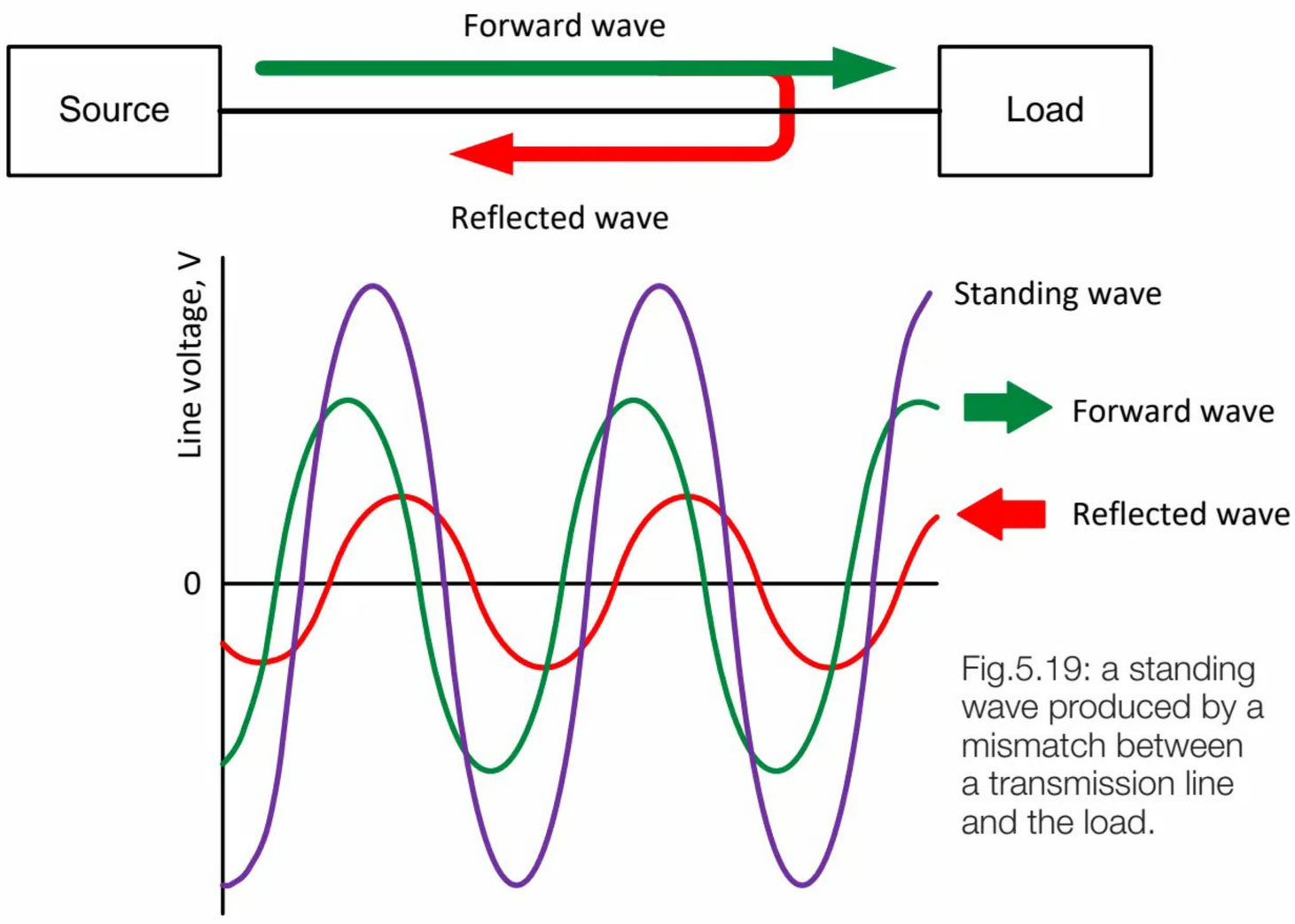


Fig.5.19: a standing wave produced by a mismatch between a transmission line and the load.

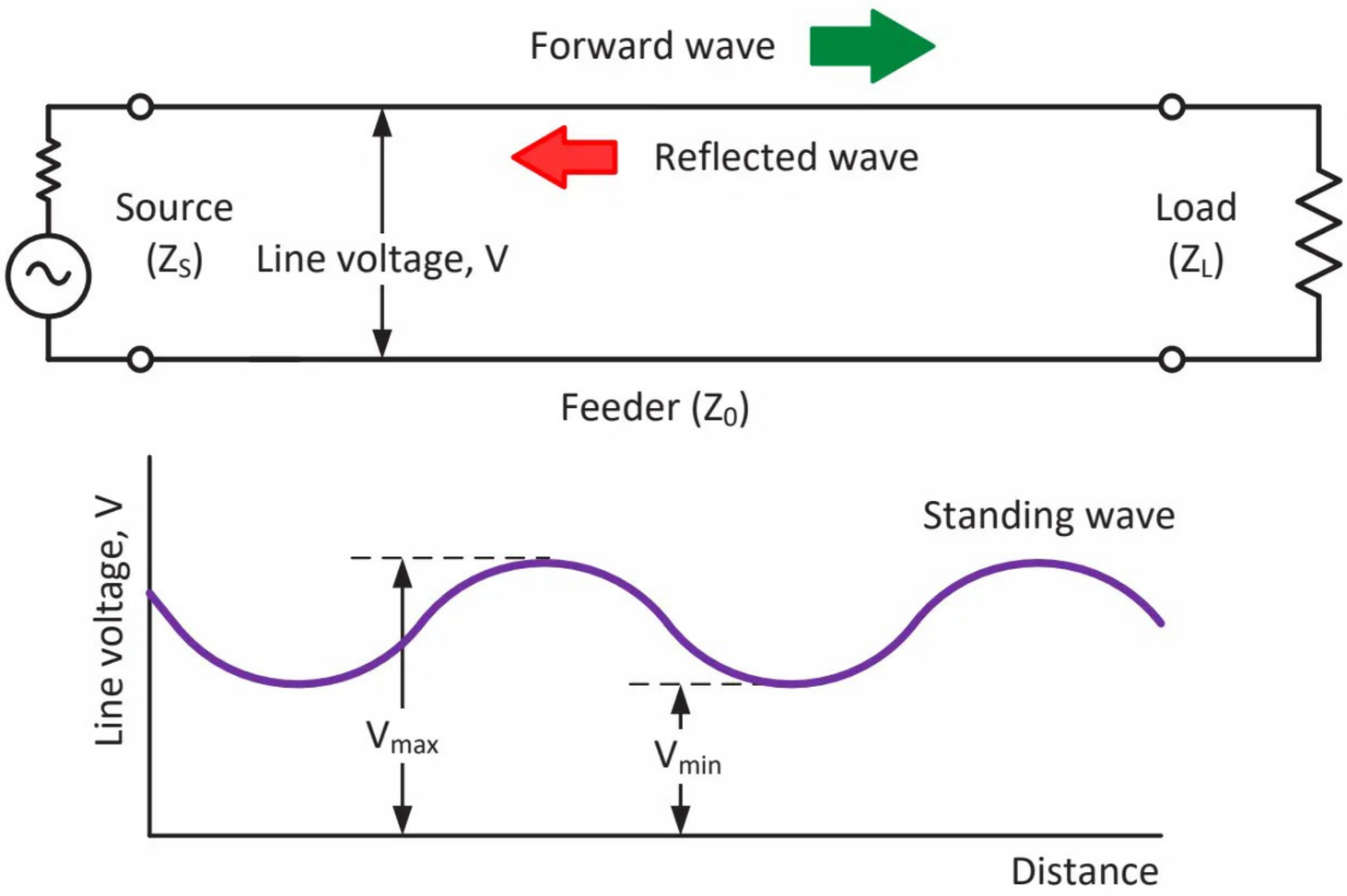


Fig.5.20: a standing wave and the resulting voltage along the length of the feeder.

In this non-ideal state, the reflected wave (shown in red) will interact with the forward wave (shown in green) to produce a standing wave along the line. The line voltage will then no longer be constant at all points, but will vary with maximum and minimum values determined by the degree of mismatch.

While both the forward and reflected waves are moving, the standing wave remains. The current distribution along the feeder will have a similar pattern, but the voltage maxima will coincide with the current minima, and vice versa.

Fig.5.20 shows a source (transmitter) connected to a load (antenna) in an unmatched system, where measurements of the line voltage reveal the presence of the standing wave in the feeder.

To understand what's happening, it can be useful to contrast the ideal and worst-case conditions with those shown in Fig.5.20. If the system is perfectly matched to the load, only the forward wave will be present, and the feeder voltage will be constant at all points. In this condition, $V = V_{max} = V_{min}$.

In the worst-case condition, when the line is either short-circuited or open-circuited at the load, all the energy will be reflected, and the forward and reflected waves will be identical. In this condition, the largest possible standing wave will be present in the feeder where $V_{min} = 0$ and $V_{max} = 2V$.

The degree of mismatch is expressed by the ratio of maximum to minimum voltage along the feeder. This important quantity ratio is referred to as the voltage standing wave ratio (VSWR). Thus, $VSWR = V_{max} \div V_{min}$.

Values for VSWR can range from 1 (perfectly matched) to infinity (worst-case). In practice, values between 1 and 1.5 are usually considered acceptable, while values above 2 usually merit investigation and indicate a need for improvement.

Table 5.2 will give you some idea of the percentage of power that's reflected for different values of VSWR.

Table 5.2: VSWR vs reflected power

VSWR	Reflected power	VSWR	Reflected power
1.0	0%	2.2	14.6%
1.1	0.2%	2.5	18.4%
1.2	0.8%	2.7	21.1%
1.3	1.7%	3	25.0%
1.4	2.8%	4	36.0%
1.5	4.0%	5	44.4%
1.6	5.3%	6	51.0%
1.7	6.7%	7	56.3%
1.8	8.2%	8	60.5%
1.9	9.6%	9	64.0%
2.0	11.1%	10	66.9%

The mismatch indicated previously in Fig.5.19 corresponds to a VSWR of about 3.

Baluns

A balanced-to-unbalanced transformer (or 'balun') can be used to match a balanced antenna (such as the half-wave dipole that we met earlier) to an unbalanced coaxial feeder. These handy passive devices can be easily constructed using ferrite cores or purchased ready-made. They are usually fitted close to the antenna feed point, but they can also be integrated into the antenna.

This month's *Hands-On* project uses a simple balun constructed using a small toroidal ferrite core.

Fig.5.21(a) shows a dipole antenna fed using a ribbon cable. Since both the feeder and the dipole are balanced, no balun is needed. In Fig.5.21(b), a dipole antenna is fed with a coaxial cable.

Although this may be expedient, the mismatch between the balanced antenna and the unbalanced feeder can cause current to flow in the shield of the coaxial cable, resulting in unwanted radiation from the cable as well as the antenna. This can also distort the antenna's radiation pattern and on receive may introduce unwanted noise and interference from nearby sources.

It may also introduce unwanted noise and impact on overall performance. Fig.5.21(c) shows a balun correctly connected at the feed point of a dipole.

Hands-On: Dipole reference antenna

This month's *Hands-On* project is a dipole antenna suitable for use across much of the VHF spectrum. It serves as a valuable reference for testing and measurement and can also be used for broadcast listening,

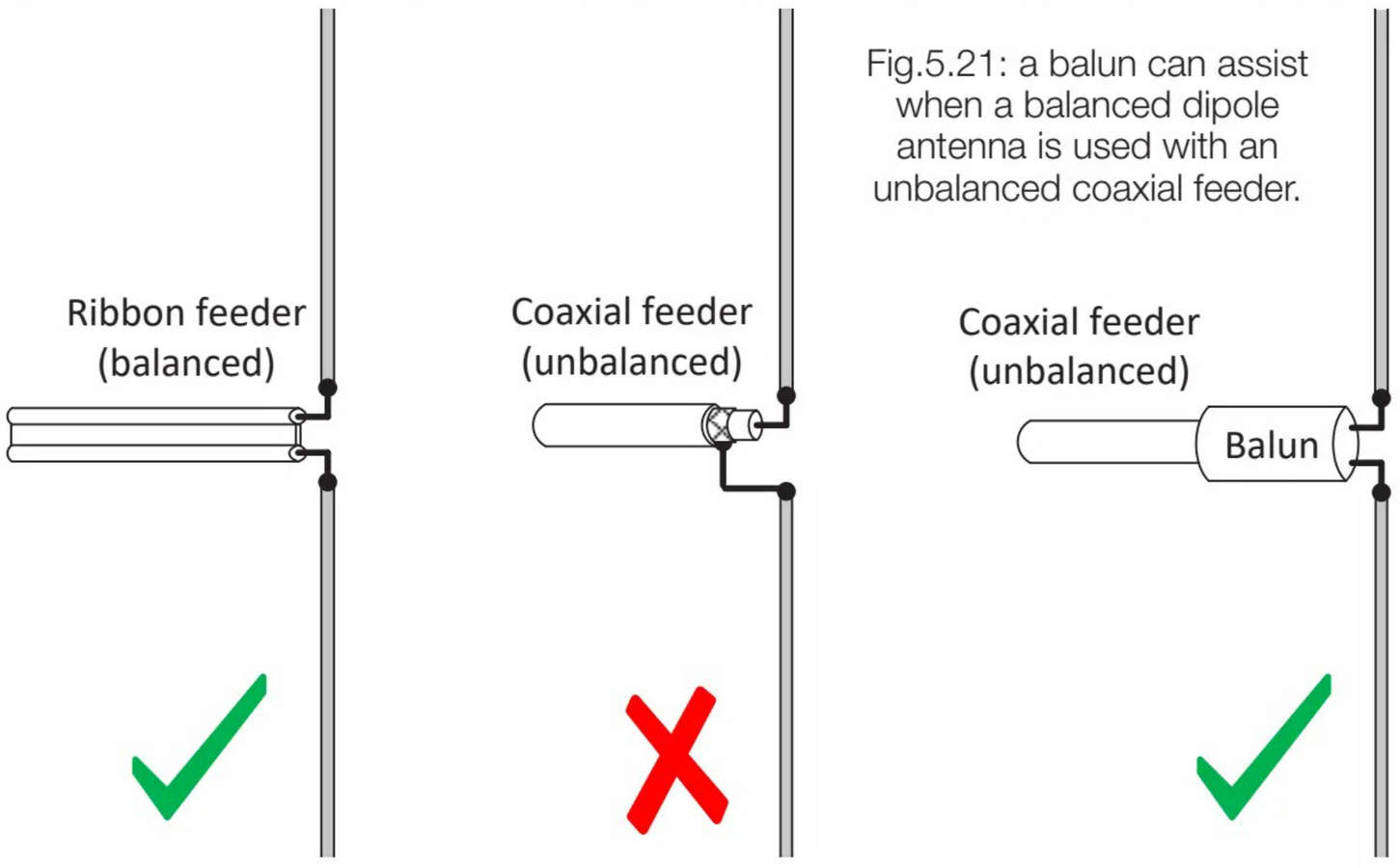


Fig.5.21: a balun can assist when a balanced dipole antenna is used with an unbalanced coaxial feeder.

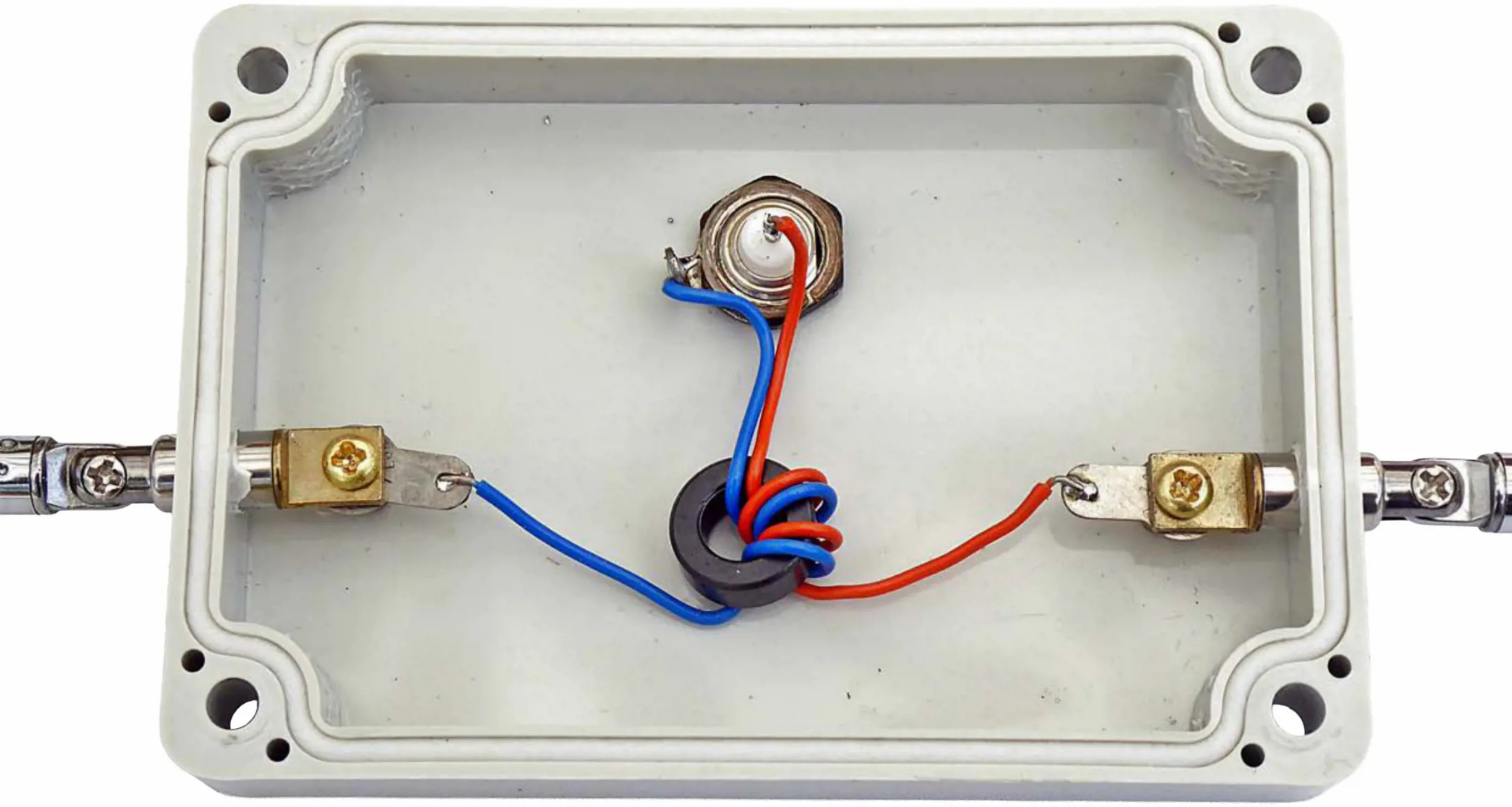


Fig.5.22: the internal layout of the dipole reference antenna (underside view of the lid).

aircraft, marine and amateur radio in the 4m, 2m, and 1.5m bands.

The antenna comprises two telescopic elements mounted on the opposite faces of a small ABS enclosure. The symmetrical internal arrangement is shown in Fig.5.22. To permit connection of the coaxial feeder, a female chassis-mounting BNC connector is attached to the lid of the enclosure, and the two telescopic antennas are secured using two M3-tapped hexagonal pillars.

The balun is constructed by winding two complete turns of solid hook-up wire over the ferrite toroidal core, as shown in Fig.5.23. Start by cutting two 9cm lengths of differently coloured hookup wire before winding them together in a bifilar arrangement (wound together side-by-side) around the core; it's important to keep the windings symmetrical and closely interleaved. The wires can then be trimmed to length ready for soldering into the enclosure.

The ferrite core should be a good quality, low-loss ferrite material suitable for use at VHF. The Amidon Fair-Rite FT50-43 is ideal for this application; we used a 4.9mm-thick core with inside and outside diameters of 7mm and 12.7mm, respectively. These are available from several internet suppliers.

Once again, to ensure balance, it's important to keep the layout symmetrical and wiring lengths identical.

If you don't have a suitable toroidal ferrite core, the antenna can be built without the balun following the layout shown in Fig.5.24 overleaf. However, if you use this arrangement, the results will be less predictable, and you may find that the antenna's directional properties will be affected by the proximity of the feeder cable.

The reference dipole antenna should be connected to the receiver or other equipment using a short length of good-quality coaxial cable fitted with BNC male connectors. A 2m length of RG-58 coaxial cable should be adequate for most applications, but longer cable lengths can also be used where necessary.

Fig.5.25 (overleaf) shows the prototype antenna mounted on a camera tripod during testing in the author's garden.

Testing

The dipole antenna can be easily tested using a VHF receiver (the SDR described last month is ideal). Set up the antenna in a clear space about 1.5-2m above the floor and well away from other electronic devices and mains wiring. The antenna should ideally be mounted on a tripod, but a timber stool or table is a viable alternative.

Connect the coaxial feeder from the antenna to the receiver, ensuring that the feeder is led away from the

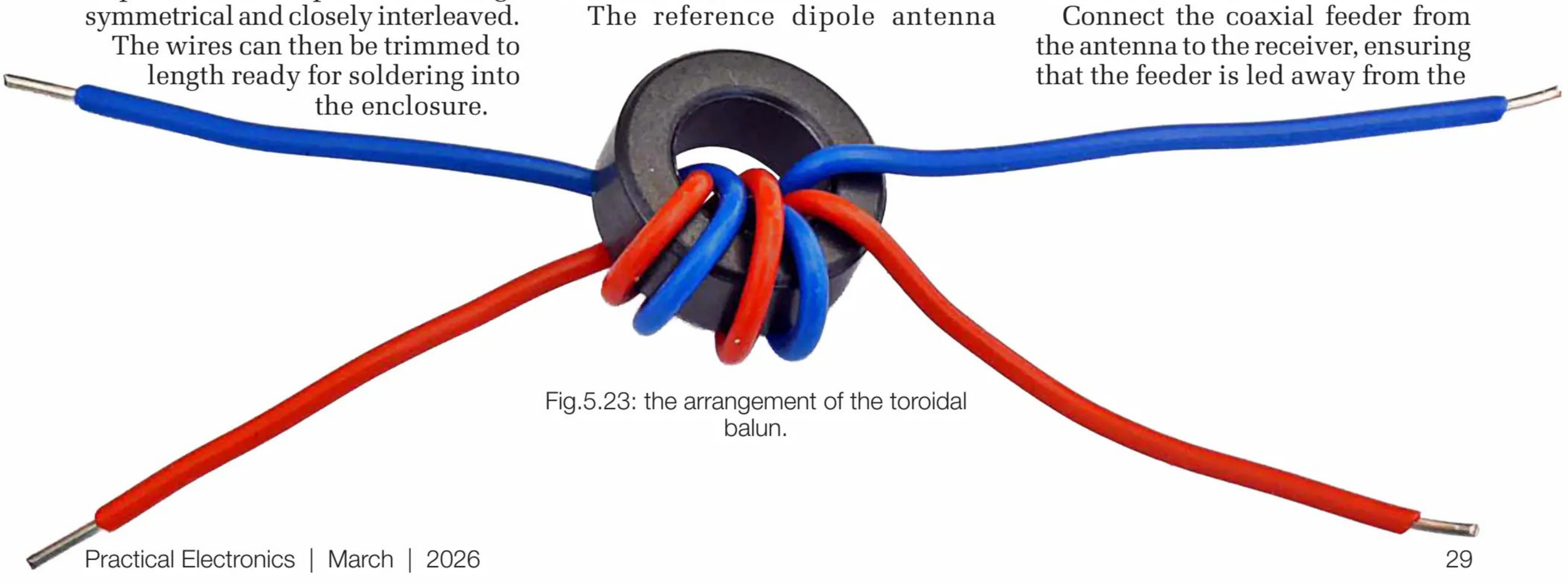


Fig.5.23: the arrangement of the toroidal balun.

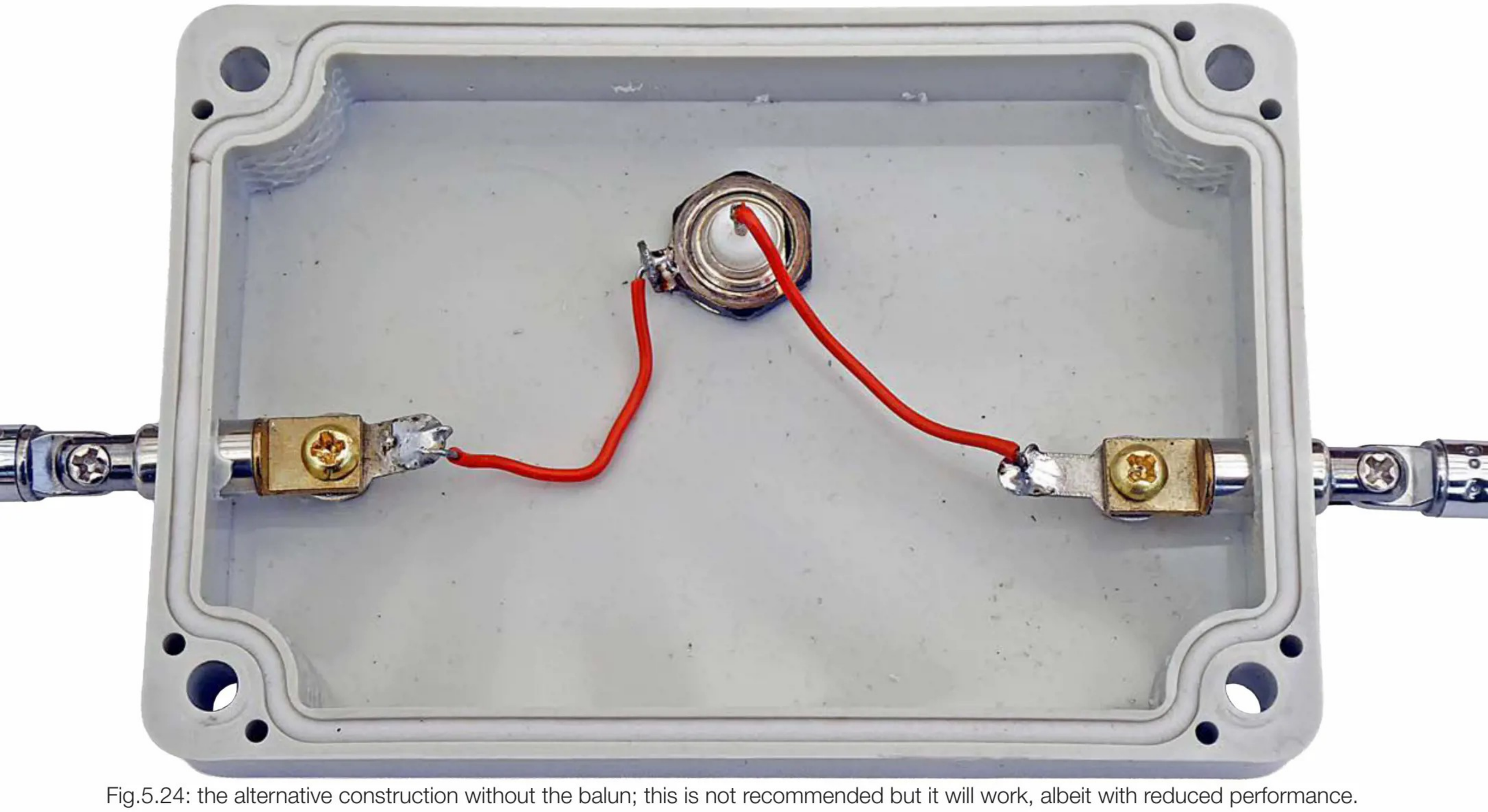


Fig.5.24: the alternative construction without the balun; this is not recommended but it will work, albeit with reduced performance.

antenna at 90° before being dropped vertically to the floor. You should also avoid running the feeder close to the two telescopic elements, as this may impact the antenna's directional characteristics.

With the aid of a tape measure, adjust the telescopic elements so they have an overall length of 1.3m. Tune the receiver to a local broadcast FM signal, rotate the antenna to obtain the strongest signal and then rotate the antenna through a full angle of 360°, observing the effect on the received signal strength.

You should find two directions of maximum response and two nulls at right angles to them (see Fig.5.27).



Fig.5.25: testing the Hands-On dipole antenna in my garden.

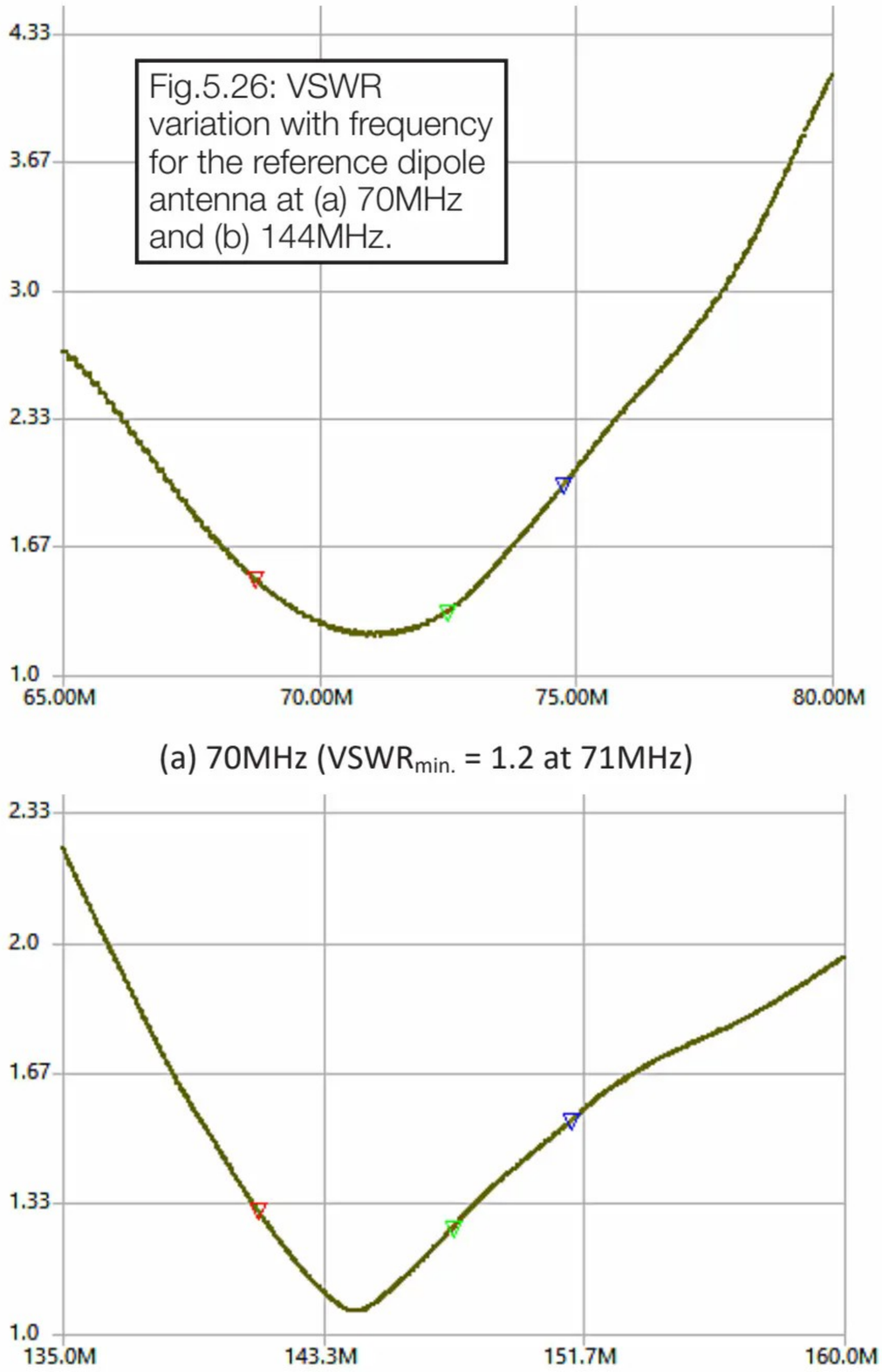


Fig.5.26: VSWR variation with frequency for the reference dipole antenna at (a) 70MHz and (b) 144MHz.

(a) 70MHz ($VSWR_{min.}$ = 1.2 at 71MHz)

(b) 144MHz ($VSWR_{min.}$ = 1.05 at 144.3MHz)

The prototype antenna performed well over a frequency range extending from 66MHz to 232MHz (see Table 5.3). To extend the frequency coverage above 232MHz, the antenna can be configured as a 1.5-wave dipole where required. Note how the VSWR increases above 300MHz.

The reference dipole antenna can be used for low-power (typically less than 10W) operation in the VHF amateur bands. Table 5.4 gives the overall antenna lengths required, but some further adjustment may be required to obtain the lowest VSWR at your desired operating frequency.

The variation of antenna VSWR with frequency for the UK 4m and 2m amateur bands is shown in Fig.5.26. This indicates minimum VSWR values of less than 1.2 on both bands, indicating an excellent match. Finally, Fig.5.27 shows the antenna's polar response measured at 125MHz. The two nulls are down by approximately 14dB on the antenna's maximum response.

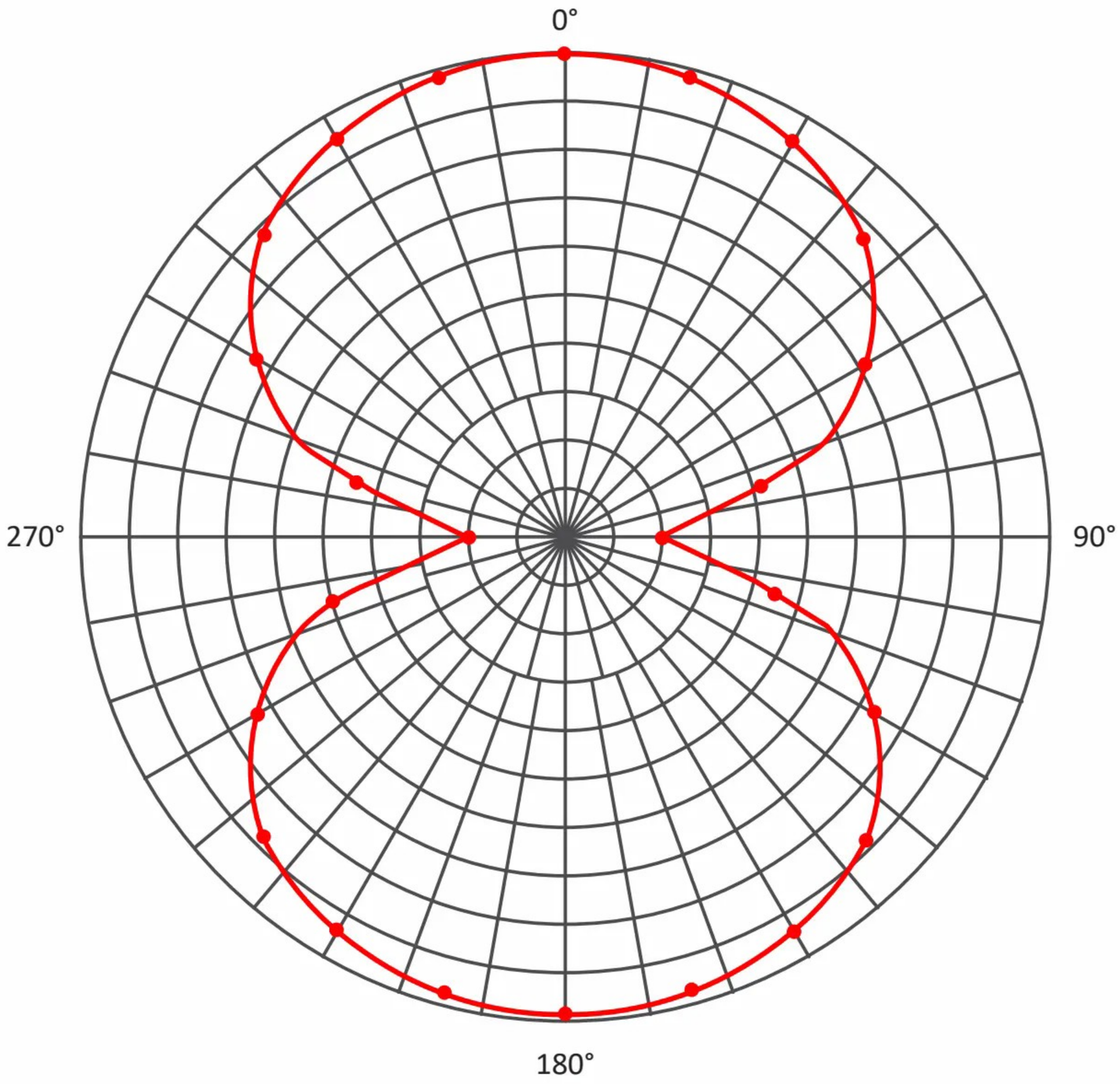


Fig.5.27: a polar response plot of the reference dipole at 125MHz.

Table 5.3: overall antenna length, operating frequency and VSWR

Overall length	Telescope sections	Half-wave dipole		1.5-wave dipole	
		Frequency	VSWR	Frequency	VSWR
48cm	1	232MHz	1.6	–	–
77cm	2	165MHz	1.3	–	–
104cm	3	124MHz	1.1	370MHz	1.3
132cm	4	100MHz	1.1	300MHz	1.2
158cm	5	85MHz	1.1	255MHz	1.1
186cm	6	76MHz	1.1	225MHz	1.1
212cm	7	66MHz	1.2	197MHz	1.1

Table 5.4: antenna length for VHF hams

Amateur band	Frequency	Overall length	Typical VSWR
1.5m	223MHz	51cm	1.5
2m	145MHz	89cm	1.1
4m	70.2MHz	202cm	1.2

Coming up!

Next month, *Teach-In* will cover key RF testing and measurement. Our *Hands-On* project will be a probe for in-circuit RF voltage measurements viewable on a standard multimeter. Join us then! PE

Parts List – VHF dipole reference antenna

1 ABS or polycarbonate IP67 sealed enclosure, 115 × 65 × 40mm
2 7-segment 130cm+ telescopic antennas with screw terminals [eg, https://www.aliexpress.com/item/1005010403940280.html]
1 BNC panel-mount socket
1 Amidon Fair-Rite FT50-43 toroidal core, 12.7 × 7 × 4.9mm
2 10cm lengths of medium-duty hookup wire (red & blue)
2 M3 × 9mm tapped spacers
1 2m-long RG-58 cable terminated with BNC plugs
2 3mm solder lugs
4 M3 × 6mm panhead machine screws

HAMMOND®

1591 ABS flame-retardant enclosures

Learn more:
hammondmfg.com/1591

uksales@hammfg.com • 01256 812812



Circuit Surgery

Regular clinic by Ian Bell

Measuring the frequency response of a circuit using a PC sound card, part 6: op amp capacitive loading

In recent articles, we have been looking at the measurement of frequency response. This was motivated by the need to measure the response of a digital filter (from our series on DSP) without using advanced lab test equipment.

We covered the general principles of frequency response measurement and use of the sound card (audio interface) of a PC or laptop together with free software called Room EQ Wizard (REW).

Recently, we have been looking at the signal conditioning that is needed to interface a circuit to the sound card's line inputs and outputs. Op amp circuits are the key building blocks of the circuitry. Last month, we covered concepts relating to op amp amplifier feedback, frequency response and stability.

This month, we will consider conditions that may cause op amps to become unstable, in particular, capacitive loading of the output and how to simulate these circuits and measure phase margin. This is relevant to the signal conditioning circuit because of potentially high capacitive loads presented by circuits under test, and any long audio cables which may be used.

However, our investigations are also more widely applicable to op amp circuit design for other applications.

Feedback and stability recap

Op amps have very high open-loop gain; too high to use directly. Therefore, negative feedback is applied, in which a fraction (β) of the output is subtracted from the input signal. The use of negative feedback in op amp amplifiers provides significant advantages in circuit performance and ease of design (op amps have always been intended to be used with negative feedback).

There are three different gain values that can be used when discussing an amplifier with feedback. The op amp on its own (without feedback) has an open-loop gain (A_O). Amplifier circuits with feedback have a closed-loop gain (A_C) value. The value $-\beta A_O$ is the loop gain, which is the gain around the closed feedback loop.

The signal inversion that makes the feedback negative is equivalent to a phase shift of 180°, which is by convention taken as positive (±180° phase shifts are the same for a steady sinewave). As the frequency increases, the phase shift from the delay around the loop becomes more negative, so the effective phase shift reduces from 180° towards (and then past) 0°.

If the total effective phase shift from inversion and delay is 0° or an integer multiple of ±360°, the feedback will be positive rather than negative. If the gain at the frequency where this occurs is unity or more, the circuit can enter sustained oscillations and the amplifier is deemed to be unstable.

In an ideal amplifier, the oscillation amplitude could continuously increase, but in real circuits the amplitude is limited by non-linearities (eg, the limited supply rails).

Amplifiers that are close to being unstable will not oscillate continuously, but may produce undesirable outputs, such as large overshoot and severe ringing (decaying oscillations) when input transients occur. The term 'unstable' is often used in such cases anyway.

Changes in conditions such as temperature, supply voltage or component ageing may shift a marginally stable circuit into continuous oscillation. To avoid this, we can measure how close a circuit is to being unstable using phase margin or gain margin. Phase margin is the difference between the phase shift of the loop gain due to delay and -180° (see Fig.1).

Typically, 45° is regarded as acceptable, but this is a rule of thumb and not universally used. Applying more feedback (a larger β value) results in a greater

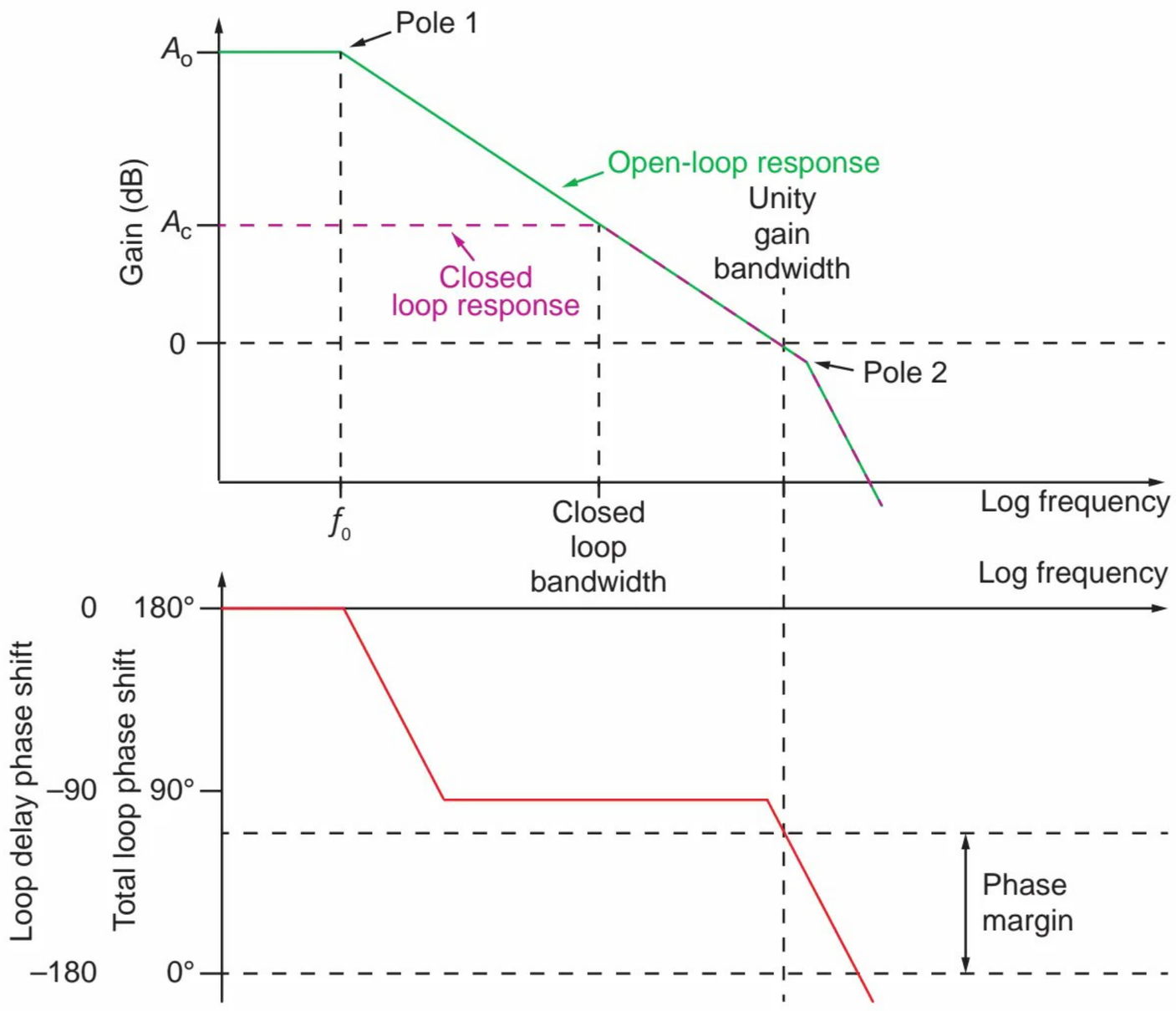


Fig.1: determining the phase margin of an op amp from its frequency response and phase plot.

likelihood of instability, so low-gain op amp amplifiers are more likely to oscillate.

The open-loop gain of most op amps is deliberately reduced at high frequencies so unity-gain amplifiers are stable. This is commonly achieved using dominant pole compensation, which creates a low frequency (typically 1-10Hz) pole in the open-loop response (Pole 1 in Fig.1 is typical of an op amp dominant pole).

Above this frequency, the open-loop gain falls off (from A_O) at 20dB per decade until another pole or zero occurs.

The closed-loop gain (A_C) remains flat until frequencies at which it approaches the open-loop gain (see Fig.1), which is at a high frequency because A_O is very large for most op amps. If we multiply the closed-loop gain by the closed-loop bandwidth, we get the gain-bandwidth product (GBP or GBW). A feature of dominant pole compensated frequency responses is that the GBP is constant.

Capacitance in the feedback loop

Last month, we briefly discussed the fact that capacitance in the feedback loop of an op amp will modify the frequency response of the feedback loop (the loop gain). Any change to the loop gain's response will change the stability of the op amp; it may get better or worse, depending on the specific change in the loop response.

We also showed an example of the impact of capacitance at the op amp's input, which introduced an additional pole in the loop response.

The key locations for possible capacitance in an op amp amplifier's feedback loop are from the op amp's input to ground, across the feedback capacitor, and at the op amp's output. Capacitance at the input and output are most likely to cause instability, whereas capacitors are quite commonly added across the feedback resistor to improve stability.

Parasitic capacitance

All the loop-gain influencing capacitances can be present as parasitic or stray capacitance – the unwanted capacitance inherent in the components and interconnection (eg, wires and PCB tracks), and the op amp's internal capacitances. Fig.2 shows an example amplifier circuit with the parasitic capacitance from the wiring, components (C_{p1} to C_{p3}) and the op amp's input capacitance(C_{in}).

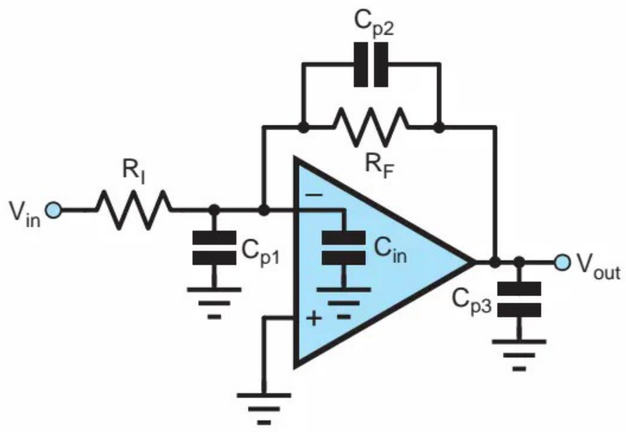


Fig.2: parasitic capacitances in an op amp feedback loop.

Parasitic capacitance values will depend on the layout of the constructed circuit, but will typically be in the low picofarad range. For example, in 2014, Dave Jones from EEVblog measured the capacitance between rows on a solderless breadboard because there was no definitive value published that he could find. The result was 2-3pF range (see **https://youtu.be/6GIscUsnlM0**).

Generally, keeping parasitics as small as possible will reduce problems. There is not much you can do about it when using solderless breadboard, which limits its use in high-frequency circuits. For PCB layouts, make sure tracks on the inverting input are short, for example by placing both gain setting resistors as close as possible to the inverting input pin.

The capacitance at the input pin can also be reduced by removing any ground plane in the area under the connections to the pin, although this could increase interference pickup.

The interaction between all the stray capacitances in different parts of the feedback loop and the op amp's frequency response is complex and may be difficult to predict, as the exact value of the parasitics may not be known. However, if you want to get into the details, there are software tools available that will calculate PCB trace capacitances.

Different construction techniques and different layouts for the same circuit will have different parasitics. This means that if two copies of the same circuit are constructed differently, one may be OK, while the other is unstable.

I remember someone building an op amp circuit that was fine on solderless breadboard, but which oscillated when built as a neat-looking (but maybe not so well designed) PCB. I do not remember the details, but it serves to illustrate that stability can be a tricky problem.

Op amp input capacitance also contributes to the capacitance in the loop (see Fig.2). This is often not specified on device datasheets and is difficult to measure, so again, this leads to uncertainty about the exact situation you have in a particular circuit. Typically, op amp input capacitances are in the range of 2-20pF, depending on the type of op amp.

Analog Devices provides some example values in an article on measuring op amp input capacitance (**https://pemag.au/link/acal**).

Capacitive loading

In an op-amp-based amplifier, the capacitance at the op amp's inverting input is likely to be just the stray capacitance of the local connections between the gain resistors and the op amp pins, plus the op amp's input capacitance. So, in most cases, it will be relatively small.

On the other hand, capacitors may be specifically connected to the output, and the capacitance from the next stage inputs plus the potentially much longer interconnects may be significantly higher than that seen at the input.

Unity-gain op amps are often used to buffer DC voltages in circuits such as mid-supply references or voltage reference IC outputs. Often, relatively large capacitors are used on the outputs of these circuits to stabilise the reference voltage during transients.

In our sound card interface signal conditioning circuit, we used a split supply, so did not use a mid-supply reference. However, this approach could be used in a version of the interface, if required.

Reference buffers and mid-supply references can suffer instability problems due to the large output capacitances used. It may not be obvious, as these circuits are used for DC purposes; however, the op amp buffer is still an AC amplifier and could be triggered into temporary or sustained oscillations.

In audio or other AC contexts, the capacitance on the output (load) is typically from the input capacitance of the next stage and from cables used to connect systems together. For example, RCA (phono) audio cables typically have capacitances of 40-150pF/m. Some lower and significantly higher values may be found in certain cables.

Fig.3 shows an op amp non-inverting amplifier with a capacitive and resistive load (C_L and R_L, respectively). We have labelled the op amp's internal output impedance (Z_o) as an impedance for generality, but it may often be considered simply to be a resistance (R_O).

The op amp's output impedance will interact with the load capacitance and is therefore important when considering the impact of capacitive loading on the op amp.

Circuit analysis

We will analyse a simple version of the circuit in Fig.3, where R_L is not present (this implies a very high load resistance) and we are using a unity gain buffer, so

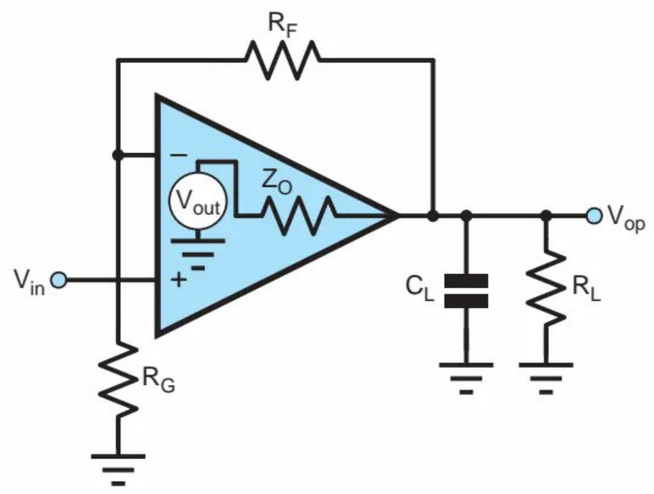


Fig.3: an op-amp-based amplifier circuit showing output impedance and load.

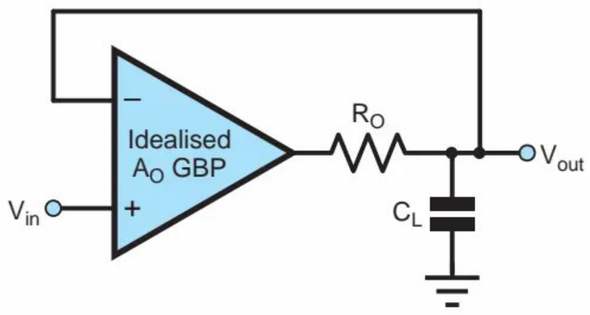


Fig.4: a simple model of the effect of capacitive load on a unity-gain buffer.

R_F=0 and R_G is not present. It is a good idea to use the case of a unity-gain buffer because, as mentioned above, this circuit is most likely to be unstable. We will assume that the op amp's output impedance is a resistance (R_O).

Consider an idealised op amp that has specific open-loop gain (A_O) and GBP, with dominant pole compensation, but no other poles, zero output impedance and infinite input impedance. It could drive a capacitive load with no impact on its behaviour due to its zero output impedance. Therefore, as noted above, we need to include the op amp's output resistance to model the effect of load capacitance.

This leads to a simple model with an RC circuit formed by R_O and C_L in the feedback loop (see Fig.4).

To analyse the circuit, we will use the *s*-domain component values for the RC circuit, as discussed in recent articles. In Fig.4, we have a potential divider formed by R_O and C_L, which will contribute to the overall loop gain. Without the RC circuit, the loop gain is -βA_O, where for the idealised unity gain amplifier β=1 and is not frequency-dependent.

A_O has a single low-frequency pole (like pole 1 in Fig.1) due to the op amp compensation, so we write $A_O(s)$ to remind us that it is frequency-dependent. With the capacitive loading included, the loop gain becomes:

$$\frac{-1/sC_L}{R_O+1/sC_L}\beta A_0(s)$$

This is obtained by multiplying the existing loop gain by the potential divider equation for the RC circuit (recall that capacitive reactance in the *s*-domain is $1/sC$). As we did in the January article, we rearrange the RC part to get polynomials in *s* in the numerator and denominator:

$$-\frac{1}{1+sR_OC_L}\beta A_0(s)$$

We see that the output resistance and load capacitance combined contribute a pole at $1+sR_OC_L=0$ (where the *s*-domain gain is infinite), which is the same as for an RC circuit on its own. As in previous examples, we substitute $s = j\omega = j2\pi f$ and find the magnitude to get a pole frequency of $f = 1 \div (2\pi RC)$.

This is very likely to be at a much higher frequency than the dominant pole from the op amp's compensation, so it will correspond to pole 2 in Fig.1. Adding the capacitive load does not change the pole in $A_O(s)$, but if the two poles happened to be very close in frequency (unlikely in typical op amp circuits), they would not show as two obviously distinct features in the response as in Fig.1.

In this idealised scenario, the circuit must be stable without the capacitive load because the single pole $A_O(s)$ only provides -90° of phase shift. With the additional pole provided by R_O and C_L, the phase shift can reach 180° and potentially cause instability. We can simulate some examples to illustrate this.

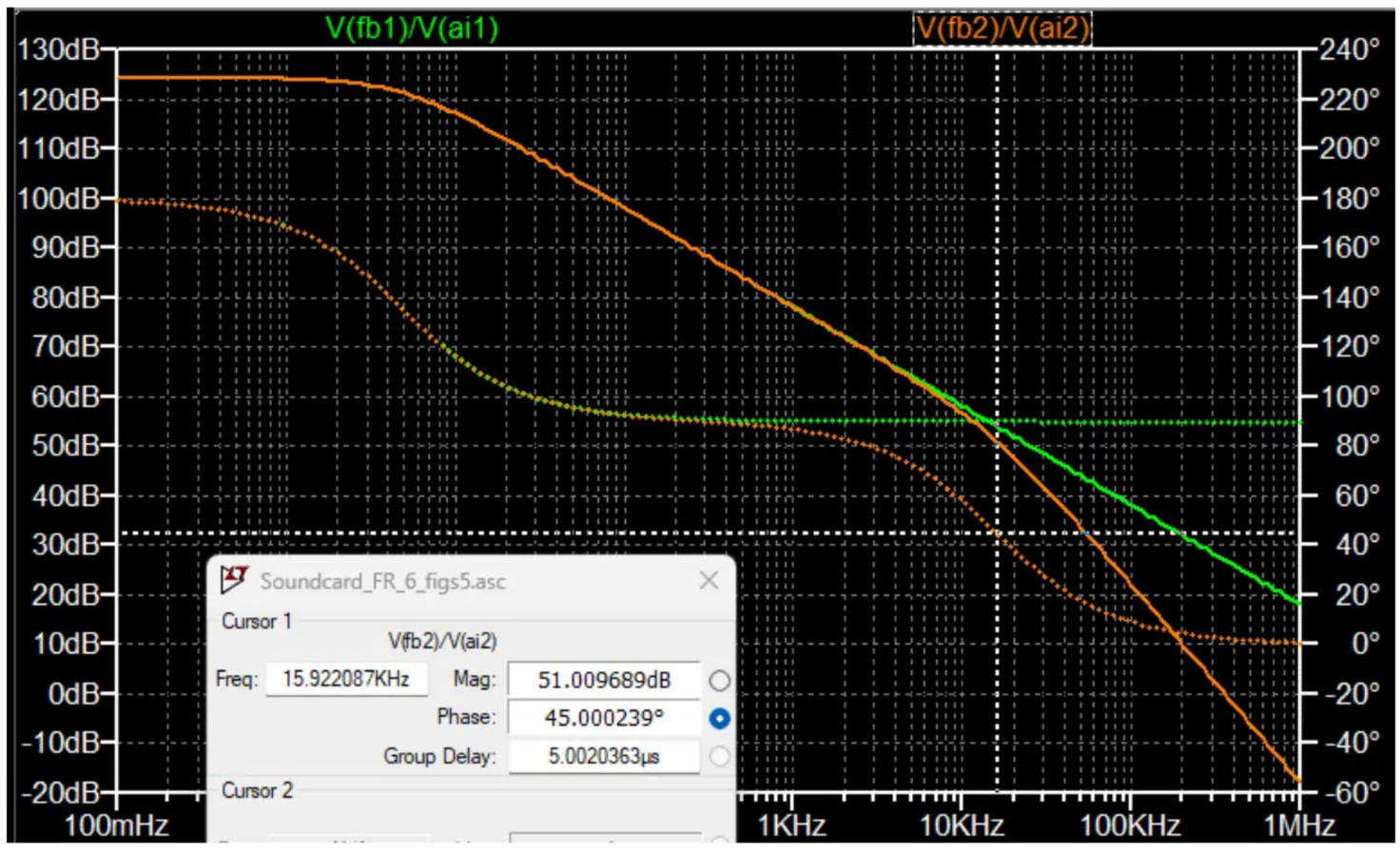


Fig.6: the simulation results for the circuit in Fig.5.

If we have a complex situation (eg, the load resistance R_L is present), we proceed with the analysis in the same way as above, but the algebra will have more terms.

Capacitive load simulations

The Fig.5 schematic contains two versions of the op amp loop gain simulation circuit discussed last month. The circuits are configured for unity gain by setting the feedback resistors to zero and the grounded resistor to a very high value (10GΩ) so it is more-or-less open circuit. There are two copies of the circuit, one with and one without a capacitive load, to facilitate comparison.

The op amps in the circuits in Fig.5 are idealised generic devices using LTspice's UniversalOpAmp4 model. This model was chosen because it has an output resistance that can be set via the device's parameters (right-click the symbol), which we need for this simulation. Other lower-numbered (simpler) UniveralOpAmp models do not provide this.

UniveralOpAmp4 has two poles in its open-loop gain. Like last month, I used $A_O = 1.778 \times 10^6$ (125dB) and GBP = 8MHz to provide a reasonable match to the OPA2134 used in the sound card interface.

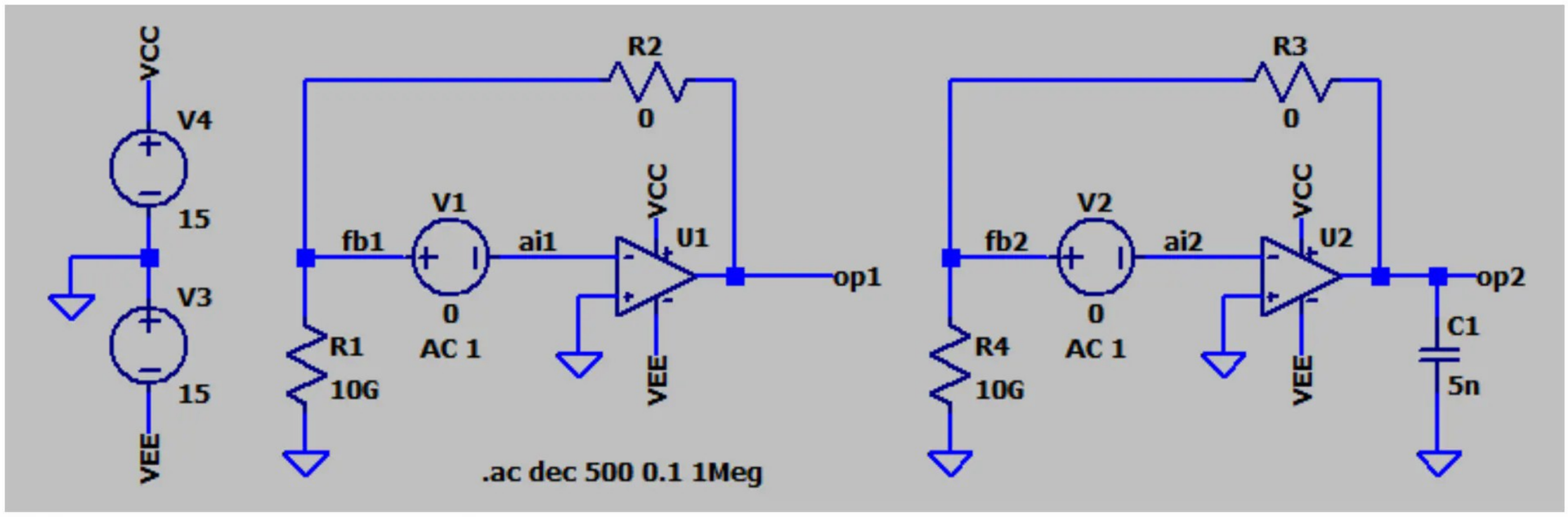


Fig.5: an LTspice schematic for simulating the loop gain of ideal op amp with capacitive loads.

For this simulation, we ideally want a single-pole op amp to match the preceding discussion, so the second pole of the op amps in this example was pushed to a high frequency by setting a high phase margin (130°).

We are just looking at the general form of the response, so can choose arbitrary values for the op amp output resistance and load capacitance. To obtain a clear graph, we need a pole frequency well above the op amp's low frequency pole, but also away from the high frequency second pole.

The UniveralOpAmp4's default output resistance (**Ro**) is 1kΩ, but I increased it to 2kΩ for this example, which together with the 5nF load capacitor should produce a pole in the loop gain at $1 \div (2\pi RC) = 1 \div (2\pi \times 2000 \times 5 \times 10^{-9}) = 15.9\text{kHz}$.

The UniveralOpAmp4 model has input capacitance, both common mode (**Ccm**) and differential (**Cdiff**), which I changed to zero so that only the load capacitor and the op amp poles were influencing the frequency response, in line with the analysis above.

The results from the simulation in Fig.5 are shown in Fig.6. As explained last month, we plot $v_{fb} \div v_{ai}$ to show the loop gain of the amplifier. In Fig.6, we see that the circuit without the capacitive load (green traces) has a single low-frequency pole from the op amp's internal compensation. The op amp's high-frequency pole is beyond the range of the graph, so we just see a simple single-pole response.

The circuit with the load capacitor (orange traces) has an additional pole at 15.9kHz, in line with the analysis above. Fig.6 shows the pole frequency measured by placing a cursor at the point where it causes -45° of phase shift relative to much lower frequencies.

To find the pole frequency this way, we need any adjacent poles to be well separated. The phase shift due to the low-frequency poles has levelled out at 90° before any significant shift due to the second pole, so the condition is met here.

When using the cursors in AC analysis, they are attached to the gain curve by default. To measure on the phase curve, click the radio button to the right of the phase measurement to select it in the cursor readout display, as shown in Fig.6.

The phase margin of the loop gain in the circuit with the capacitive load is about 2.4° - this is the amount of phase above 0° on the graph at the point where the gain falls to unity (0dB). The -177.6° from the delay in the loop is close to cancelling the 180° from the negative feedback signal inversion. 2.4° is a very poor phase margin, indicating potential instability.

Simulation with the OPAx134

This example (and the analysis above) is idealised and only involves two poles (ignoring the UniveralOpAmp4 high frequency pole, which is not visible on Fig.6), so the maximum phase shift due to delay is -180°. The results produce straightforward frequency responses that match closely with the simple example analysed above. This is useful for understanding the fundamentals, but real circuits will exhibit more complex behaviour.

We will look at some examples using the OPAx134 op amp from the sound card interface. Fig.7 shows three circuits similar to those in Fig.5 but with the real op amp model. The three circuits have different capacitive loads of zero (for U1), 1nF (for U2) and 2nF (for U3) so we can compare their responses.

Creating multiple copies of a circuit (as in Figs.5 & 7) is straightforward using the duplicate tool from the LTspice toolbar. The components are automatically given new names, but user-named nets (wires) are not renamed, so you must remember to do this manually where needed. Otherwise, the simulation will give erroneous results or may fail with an error.

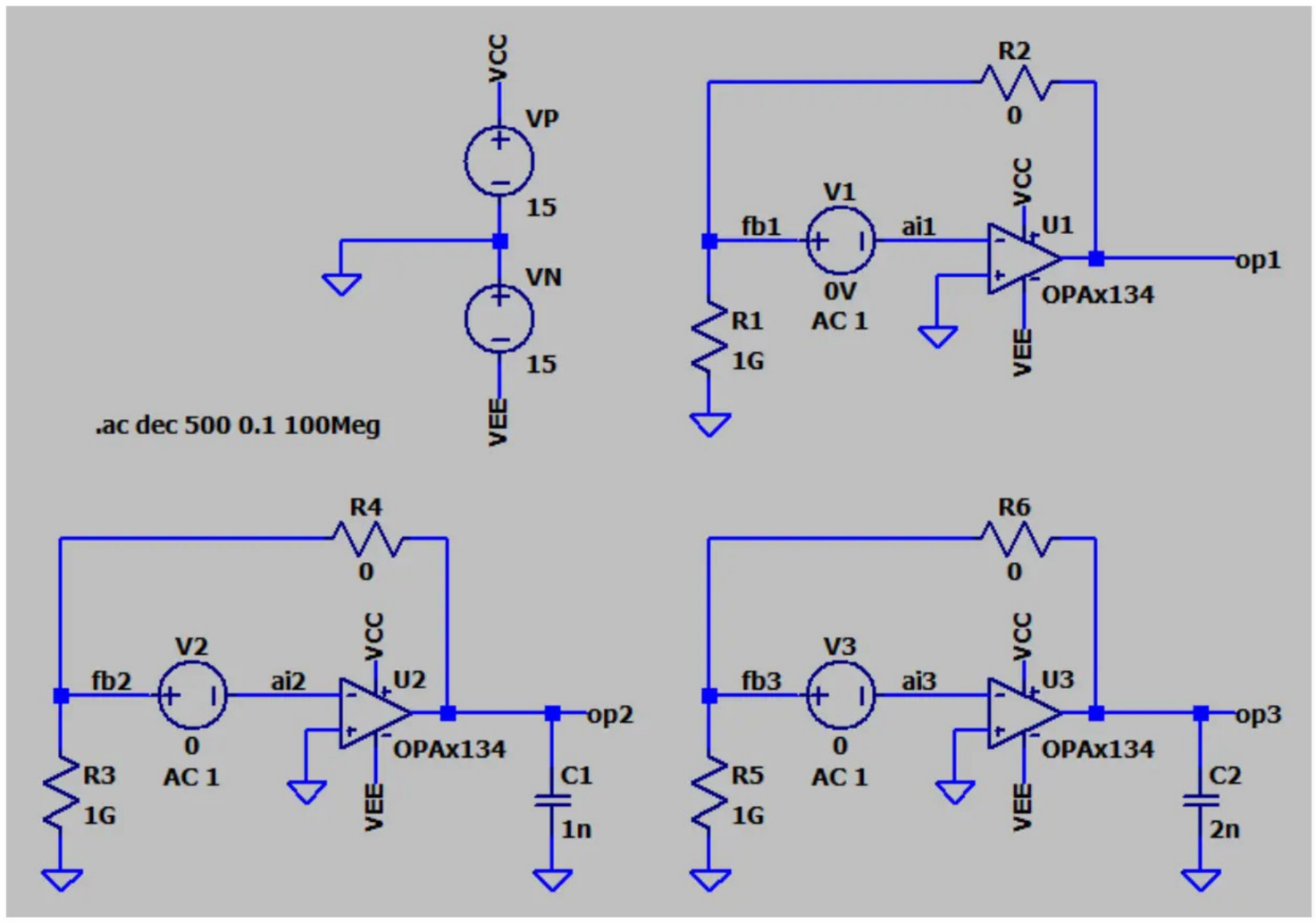


Fig.7: simulating the loop gain of an op amp with various capacitive loads.

For example, in this circuit, the supplies (nets **VCC** and **VEE**) do not need to change as they are the same for each circuit, but the signal nets **fbx**, **aix** and **opx** must be named differently in each copy.

Nets that have not been given names manually will be automatically given new unique names, but should not be used for plotting. It is a good idea to name signals that you want plot results for; otherwise, it will not be clear from the name on the plot what the signal is.

Fig.8 shows the results of simulating the circuits in Fig.7. We can see that the circuits with the capacitive loads (orange and red traces) have an additional pole in the low megahertz range. The gain decreases more rapidly above this frequency, and the phase shift at frequencies above about 1MHz is significantly higher for the loaded circuits.

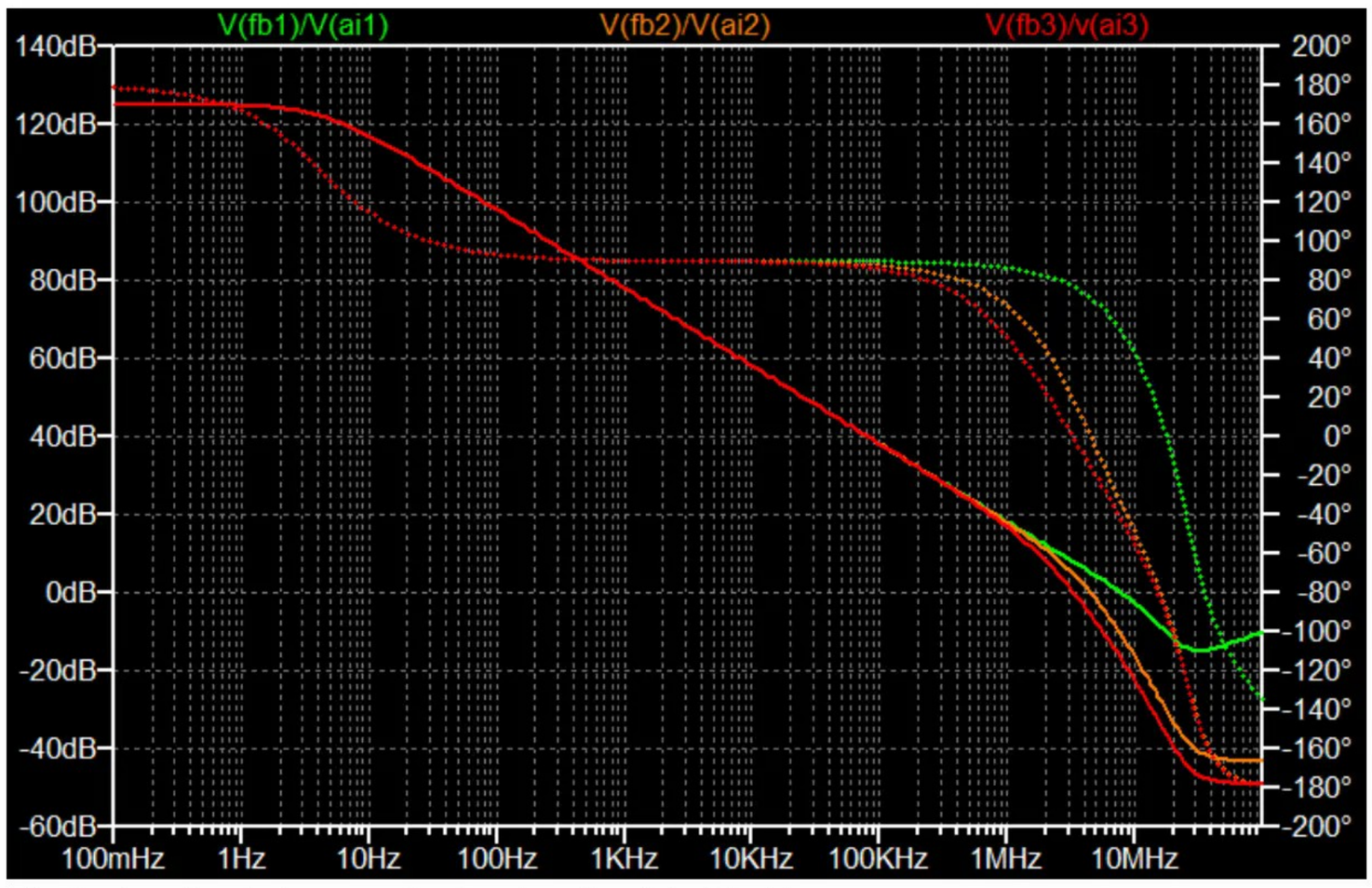


Fig.8: the simulation results for the circuit in Fig.7.

Simulation files

Most *Circuit Surgery* columns feature the use of the free circuit simulation software LTSpice. It is used to support descriptions and analysis in *Circuit Surgery*.

The examples and files for this issue are available for download: https://pemag.au/link/ac9b

Fig.9 shows a zoom-in on Fig.8 at the unity gain (0dB) point of the three curves, with the phase shift graph moved so that 0° phase shift aligns with 0dB and scaled so that an appropriate range fits on the plot. This makes reading the phase margin straightforward. The relative position of the gain (left axis) and phase (right axis) graphs is arbitrary and can be chosen to help present the data effectively.

From Fig.9, we can see that the phase margin for the unloaded circuit is about 54°. For the 2nF load, the phase margin is very slightly negative (implying instability).

Annotation and LTspice FRA

Last month, we briefly discussed LTspice's transient Frequency Response Analysis (FRA) (`.fra` command), which uses a stepped frequency sinewave to plot the loop gain of a circuit. We will look at using this with the capacitive load in a moment.

First, it is worth noting that using FRA automatically annotates the frequency response graph with the phase margin, which is useful as it saves trying to measure it manually from the graph. However, this annotation is not exclusive to the FRA.

You can add the gain and phase margin annotations to an AC analysis result (as shown in Fig.10) using "Notes & Annotations" from the right-click menu of the graph background. Select the "Annotate Phase Margin" (or gain margin) from the menu.

By default, this shows the data for just the first trace, but if you attach a cursor to one of the other traces, the data relevant to that trace will be added to the plot. To attach a cursor, click the trace name at the top of the plot.

The annotation does not state which trace it relates to, but you can change the text colour to match the trace (as in Fig.10) or edit the text (right-click the text to do this). You may also need to move the annotation to a more convenient location using the Move tool from the Toolbar.

Fig.11 shows an LTspice schematic for running an FRA for a unity-gain buffer with capacitive load. The results (with a 1nF load) are shown in Fig.12. This shows the circuit is marginally stable,

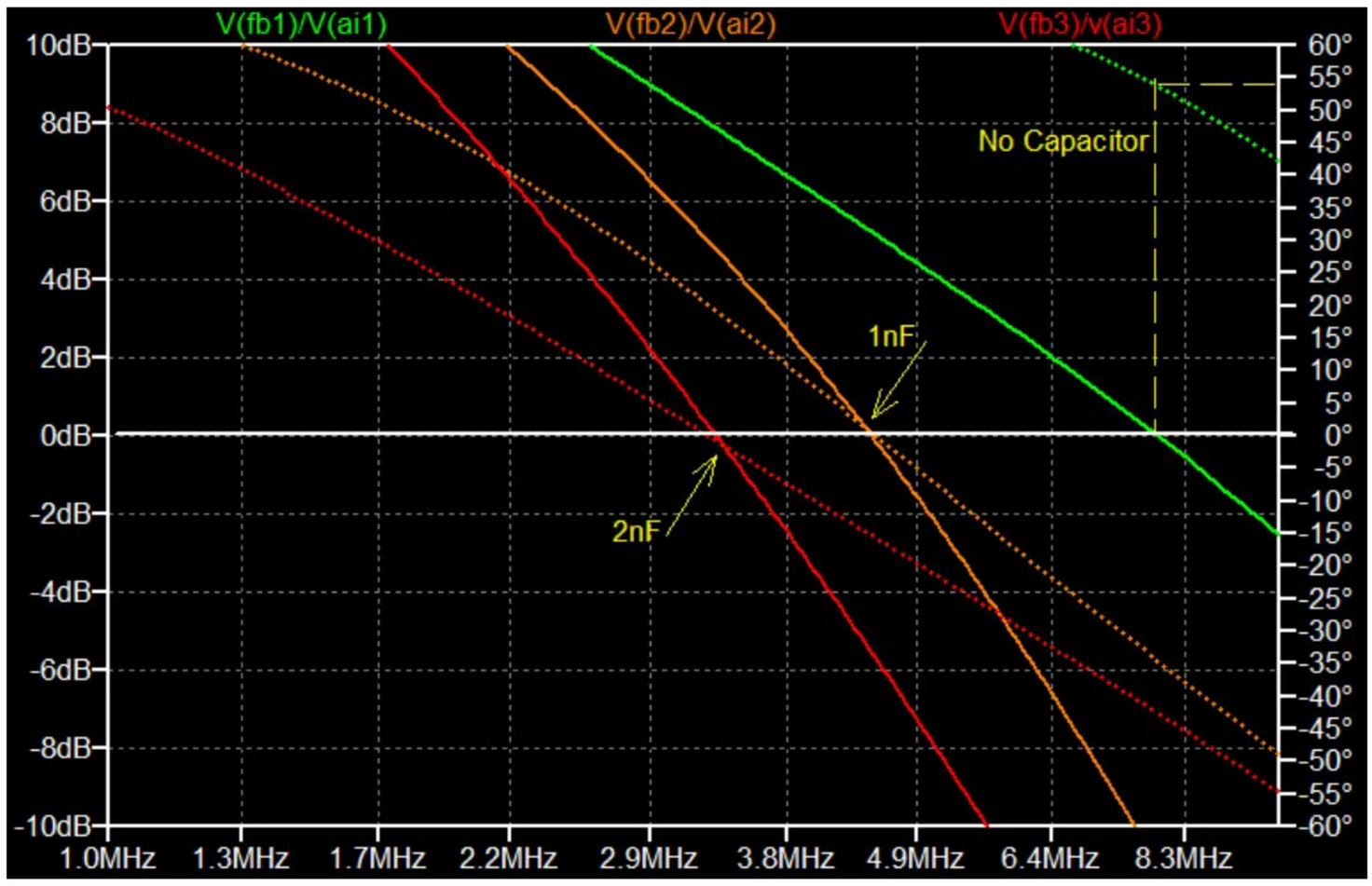


Fig.9: a zoom-in on simulation results for the circuit in Fig.8.

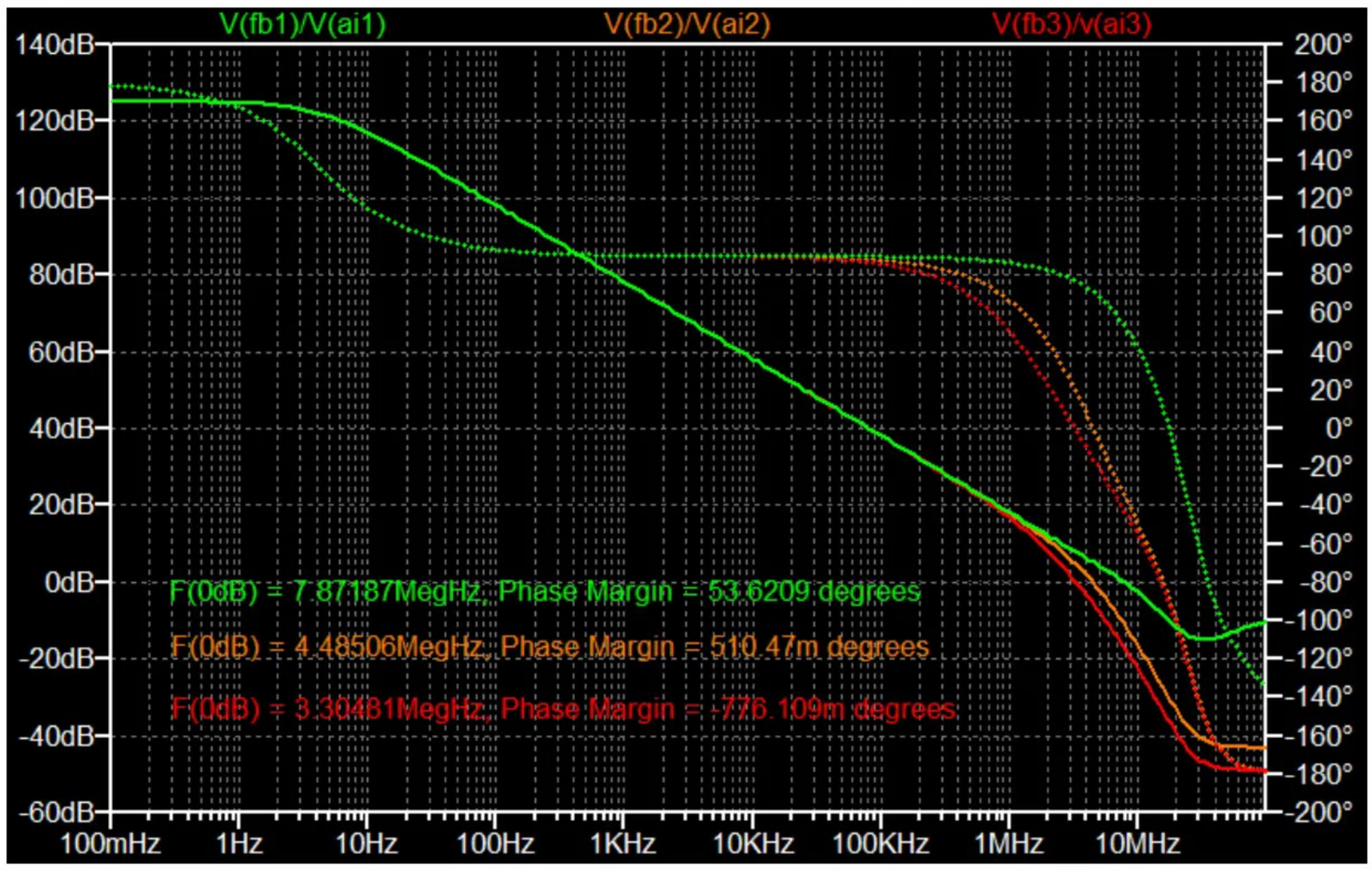


Fig.10: phase and gain margin annotations added to an AC analysis.

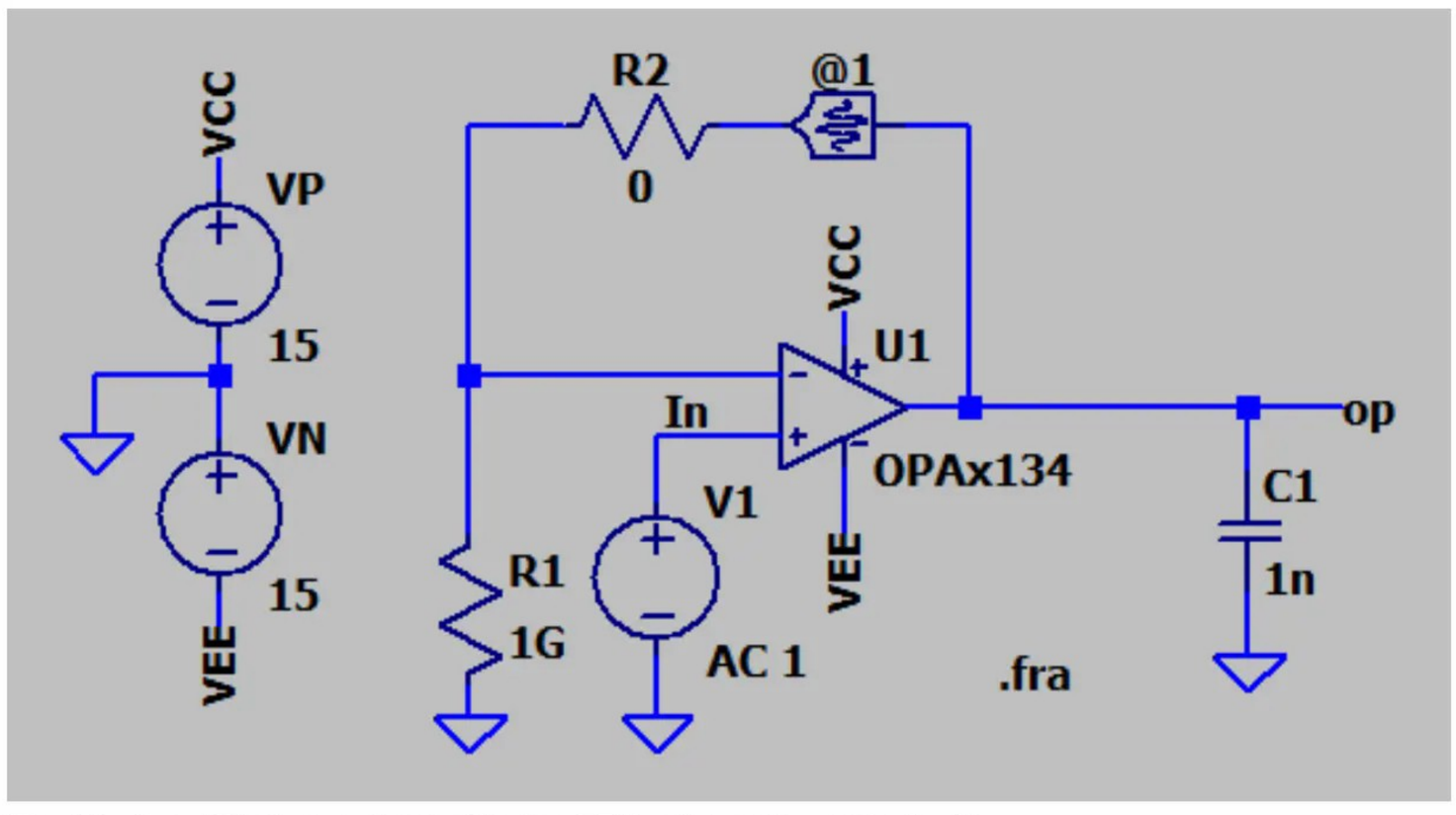


Fig.11: the LTspice schematic for FRA of a unity-gain buffer.

with a phase margin of 5.7°, which is not exactly the same as the value from the AC analysis (0.51° in Fig.10).

Running the FRA without a capacitive load indicated a phase margin of 53.5° (graph not shown), very close to the 53.6° from the AC analysis.

The FRA uses transient analysis with the full model of the op amp, whereas the AC analysis uses a linearised model. The FRA should be more accurate, but if oscillations occur in the FRA transient analysis, the results will not be valid. Oscillations will

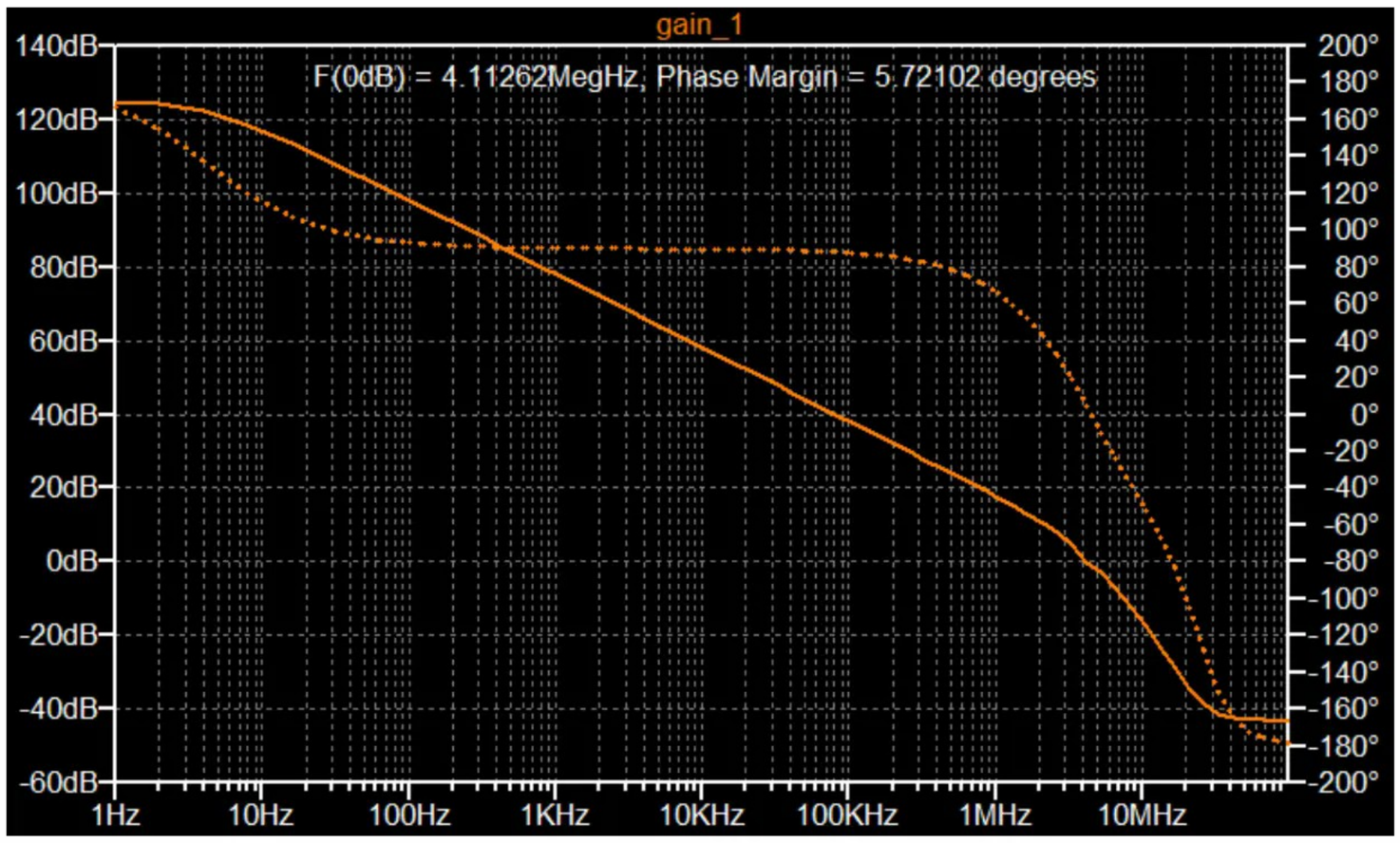


Fig.12: the FRA results from the circuit in Fig.11 with a 1nF load.

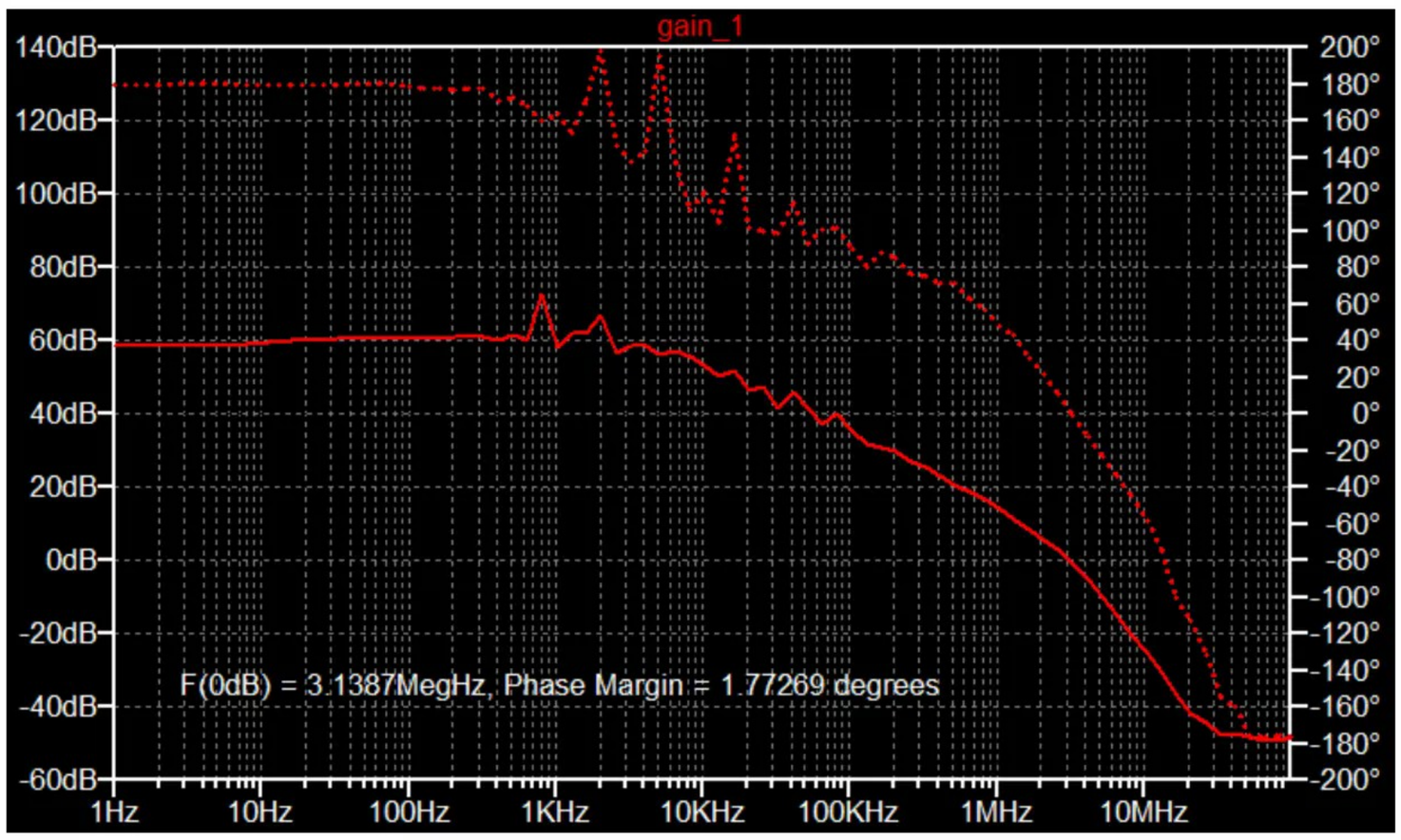


Fig.13: the FRA results from the circuit in Fig.11 with a 2nF load.

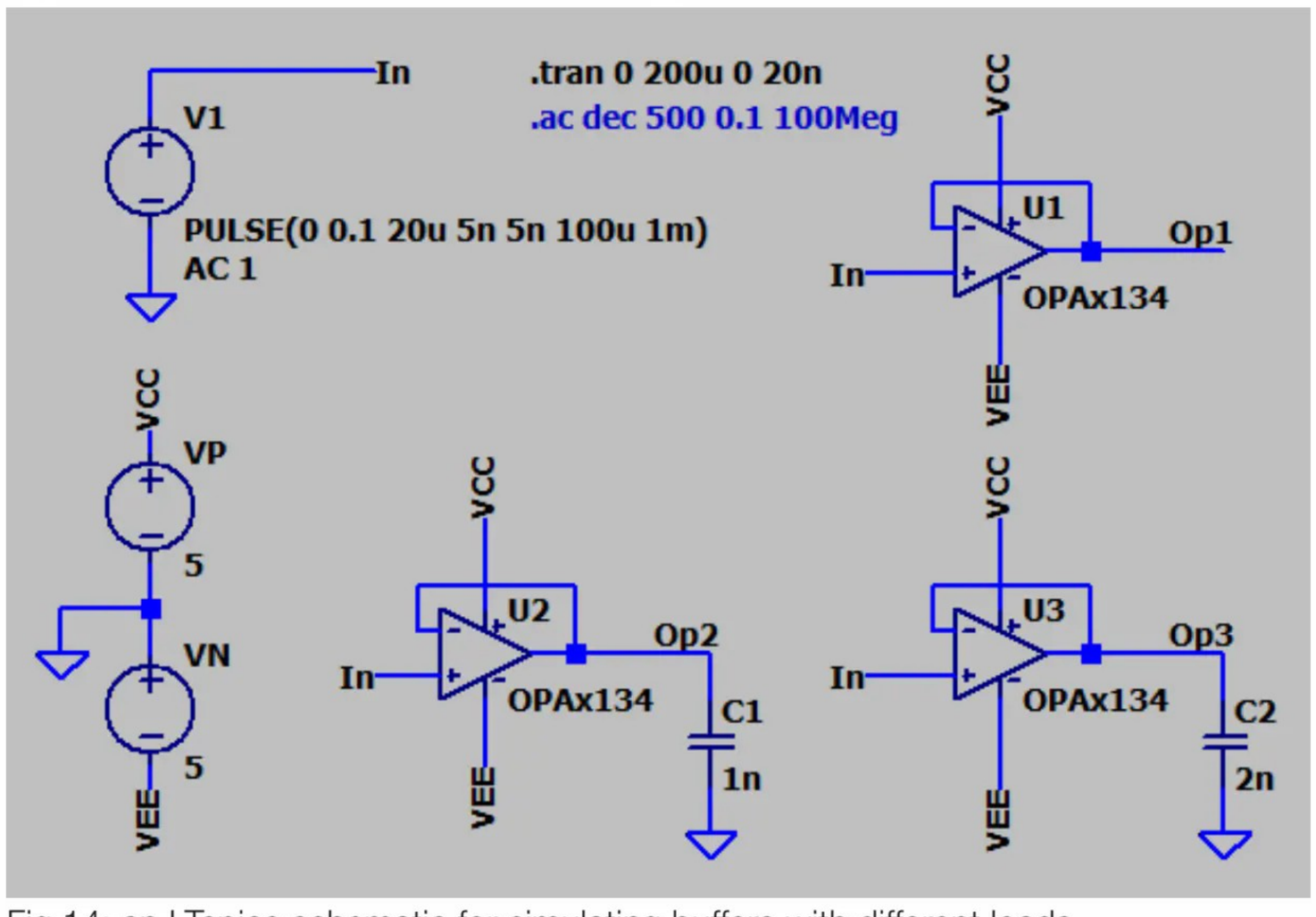


Fig.14: an LTspice schematic for simulating buffers with different loads.

not occur in the AC analysis, which is not using the time domain; it is just calculating the gain and phase shift at specific frequencies.

With a 1nF capacitive load, the unity-gain buffer is very close to being unstable, but should not oscillate continuously, particularly in a simulation where there is no noise or other disturbance to trigger it, and in which we do not apply any abruptly changing inputs (the FRA uses smoothly changing sinewaves).

5-year collections

2019-2023

£49.95

2018-2022

£49.95

2017-2021

£49.95

2016-2020

£44.95

Purchase and download at:
www.electronpublishing.com

The results in Figs.9 & 10 indicate that the circuit is unstable with a 2nF load. If we run the FRA with a 2nF load capacitor, we get the results shown in Fig.13. This is clearly erroneous, because the loop gain at low frequencies, which should be the same in all cases (as seen in Fig.10), is very different (60dB in Fig.13 vs. 125dB in Fig.10).

There are also significant irregularities in the curves in Fig.13.

For circuits that oscillate in the FRA, you have to use AC analysis to get the response curves. Of course, the AC analysis results for an unstable circuit are fictitious in a way – the circuit does not actually work as an amplifier with the loop gain or circuit gain indicated. However, it is useful as an analysis tool to indicate how much the curves need to shift to obtain stability.

Closed-loop response

Above, we have described special techniques to obtain the loop gain of the circuits, from which we can directly find phase margin. Or course, we can also run standard transient and AC analysis of the closed-loop circuits to observe the circuit's behaviour, including oscillations due to instability.

In a similar approach to the AC analysis discussed above, Fig.14 shows a schematic for the transient analysis of three versions of an OPAx134-based unity-gain buffer with different capacitive loads of zero (for U1), 1nF (for U2) and 2nF (for U3). The simulation is configured to provide a 100mV step input to the op amp (a pulse waveform).

The results are shown in Fig.15. The circuit with no capacitive load produces a more-or-less clean reproduction of the input step, with just a small overshoot (top plot pane, green trace). With the 1nF load, a circuit we know is very marginally stable, there is significant overshoot and a decaying oscillation – this is a case of very severe ringing (middle pane, orange trace).

For the 2nF load, the unstable case, there is sustained oscillation, which initially ramps up in amplitude after the step input (bottom pane, red trace).

Fig.16 shows a zoom-in of the waveforms around the step transition time. We can see that the unloaded circuit just overshoots and settles quickly; there is effectively only about one cycle of oscillation.

The cursors have been placed on the ringing from the 1nF loaded circuit and show an oscillation at around 4.5MHz, which is close to the 4.485MHz 0dB (unity gain) frequency shown in Fig.10. From the theory discussed last month, we would expect the oscillation to occur at the loop gain 0dB frequency.

We can also apply a sinewave isngal using the same schematic as in Fig.14, with the V1 source configured appropriately. Using `SINE(0 0.1 1k)` produces a 100mV peak 1kHz sinewave. The results are shown in Fig.17, which uses the same order and colours for the signal as Figs.15 & 16.

The unloaded circuit and the circuit with the 1nF load both output a clean sinewave. There is no indication of problems with the 1nF load here, unlike the severe ringing with the step input.

The smooth nature of a sinewave means it is less likely to trigger decaying oscillations than a step input, but this does not mean it will never happen in a physical circuit, particularly as other factors (noise, interference, supply glitches etc) could act as a trigger. The circuit with the 2nF load is unstable and therefore oscillates with the sinewave input.

Fig.18 shows the results of a closed-loop AC analysis for the OPAx134 unity gain buffer. This was obtained using the schematic in Fig.14 but with C1 = 250pF (U2, op2, cyan trace) and C1 = 1nF (U3, op3, orange trace). U1 has no capacitive load, as in the previous examples (op1, green trace).

These capacitor values avoid the unstable circuit with the 2nF load (although the AC analysis will run OK with a 2nF load, a physical circuit would not produce useful results).

The results shown in Fig.18 indicate that the amplifier tends to have a peak in the closed-loop gain around the 0dB point in the loop gain. Gain peaking is measured in dB relative to the flat (low-frequency part) of the closed-loop gain curve, as shown for op2 in Fig18. Increased capacitive load (and

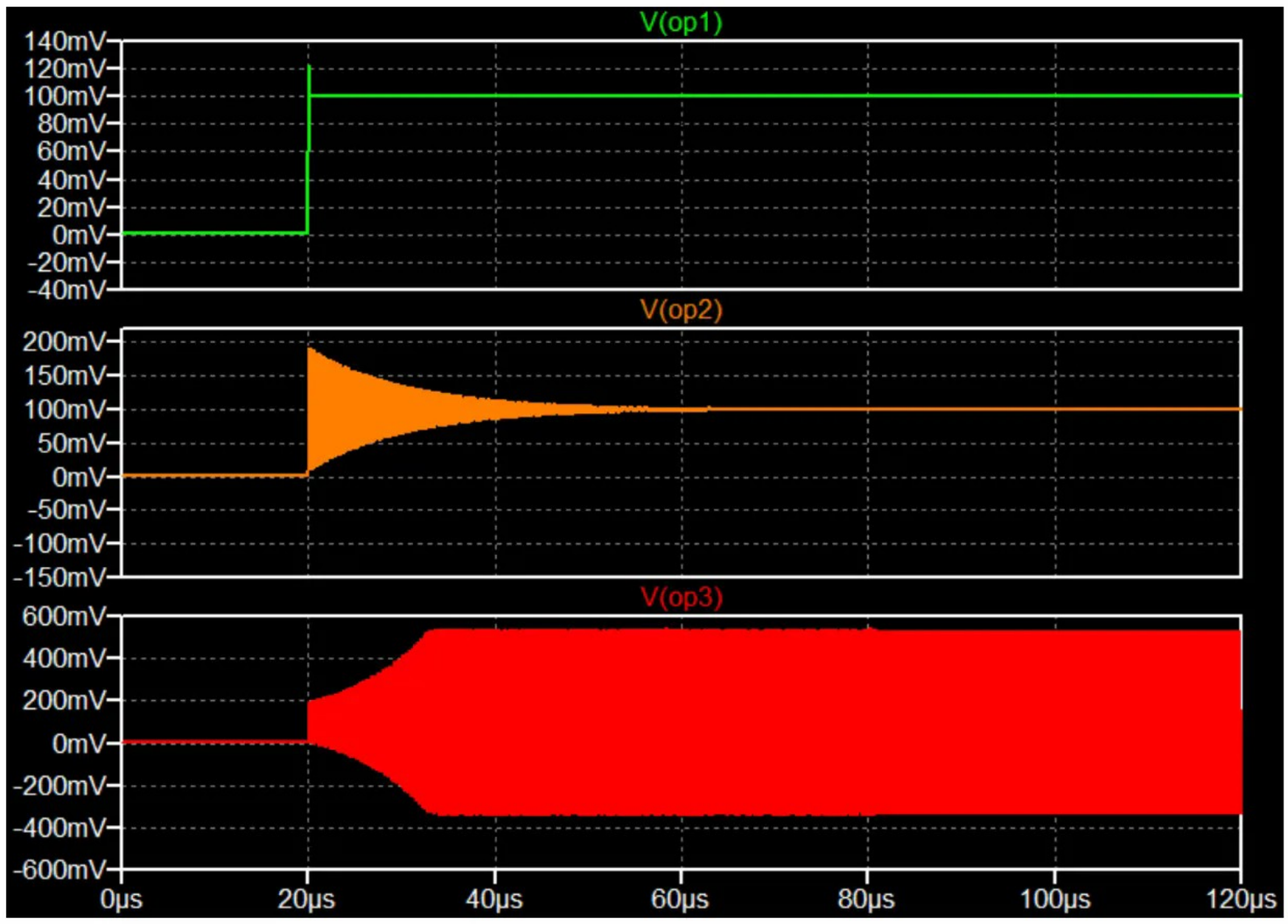


Fig.15: the results from simulating Fig.14.

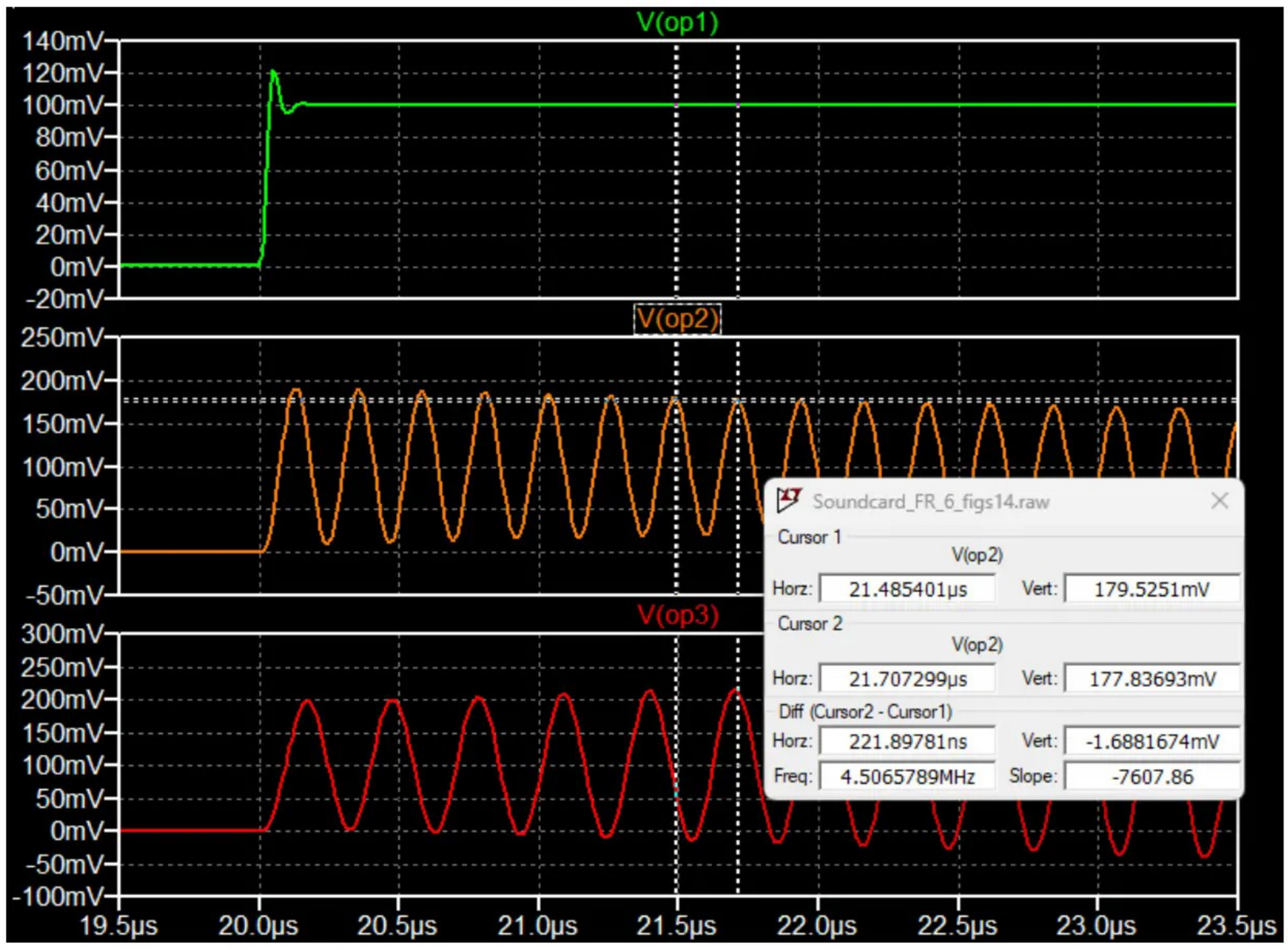


Fig.16: a zoom-in on the step response in Fig.15.

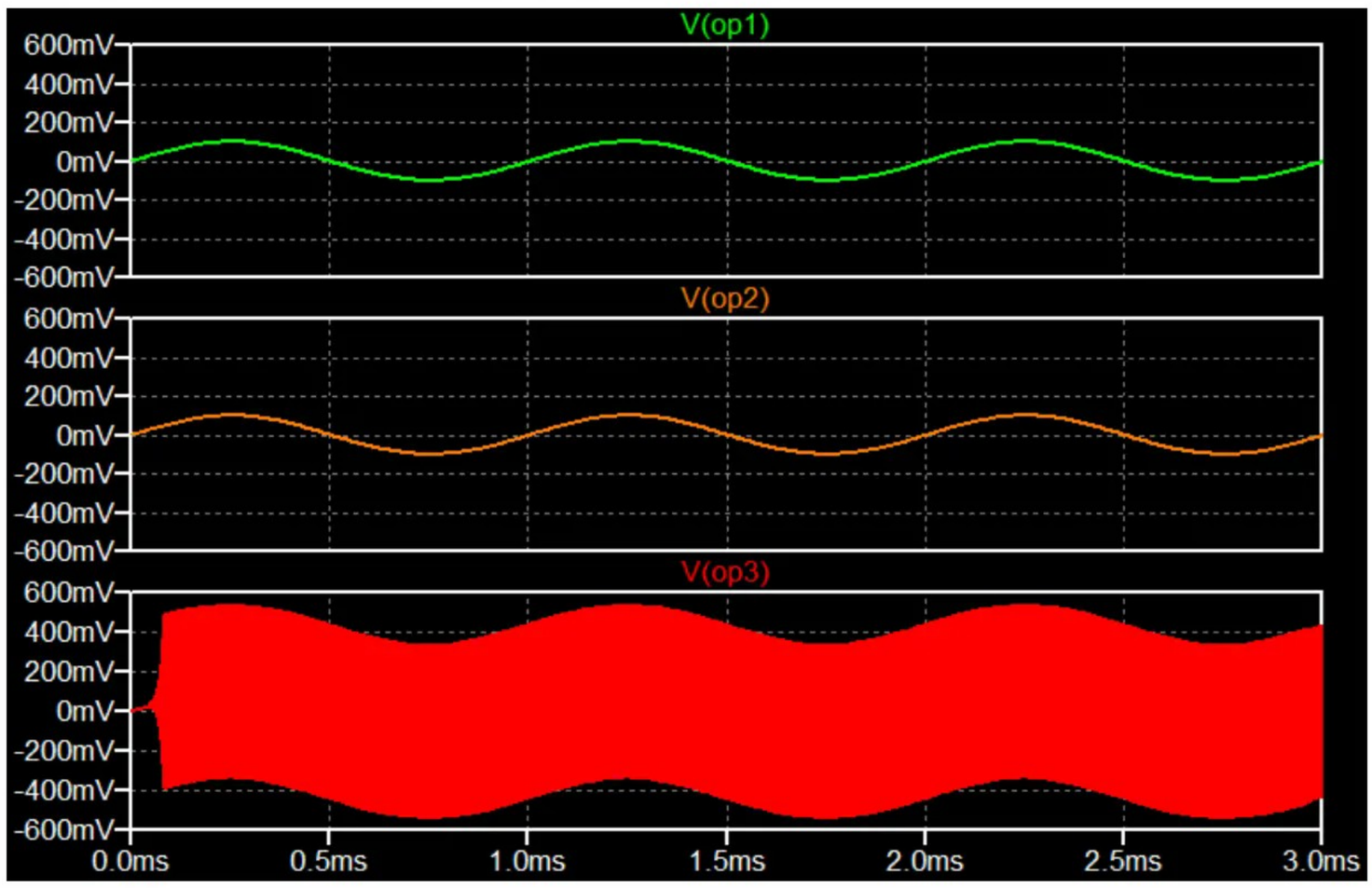


Fig.17: the response of the circuits in Fig.14 with a sinewave input.

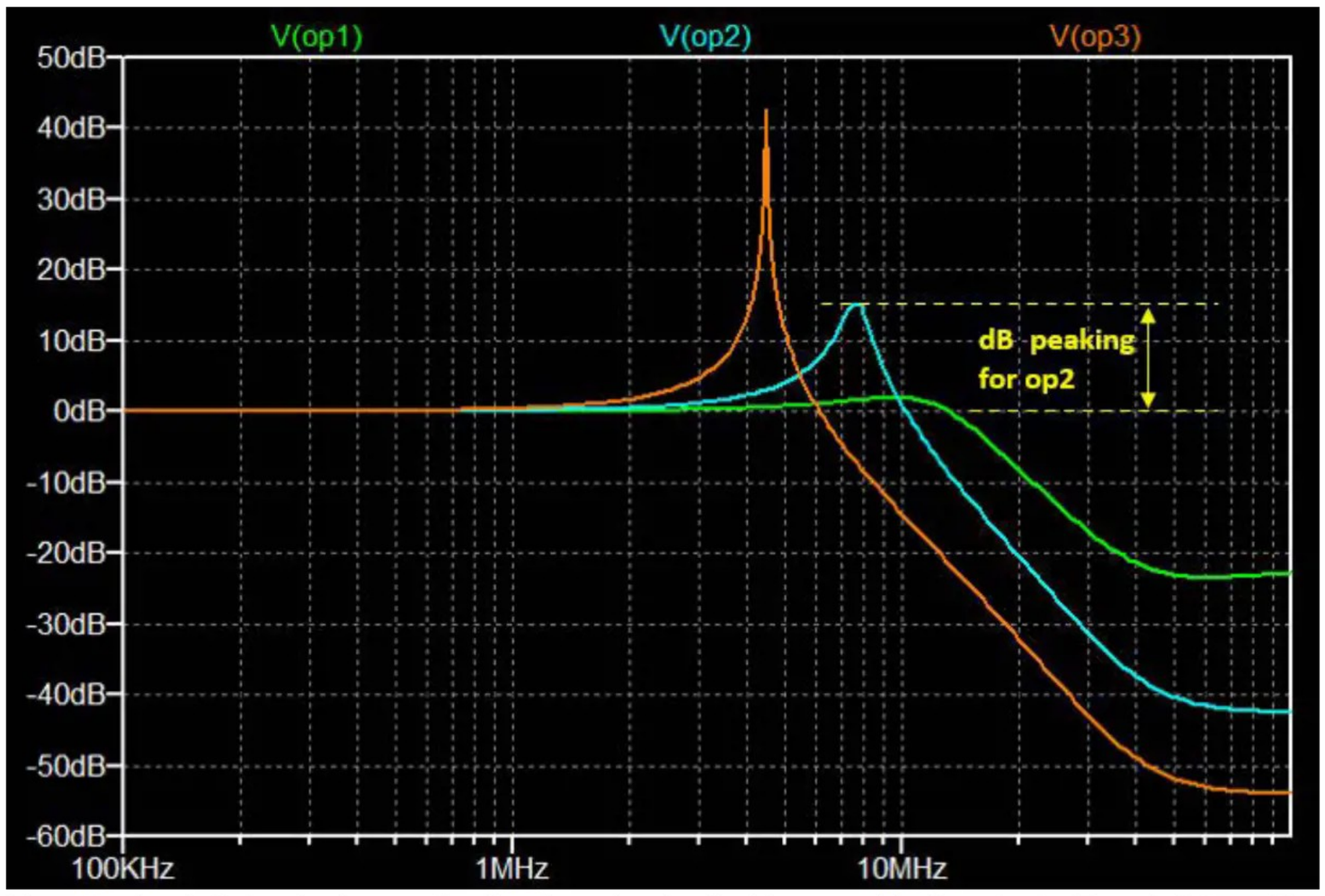


Fig.18: the AC Analysis results for the circuits in Fig.14 with C1=250pF and C2=1nF.

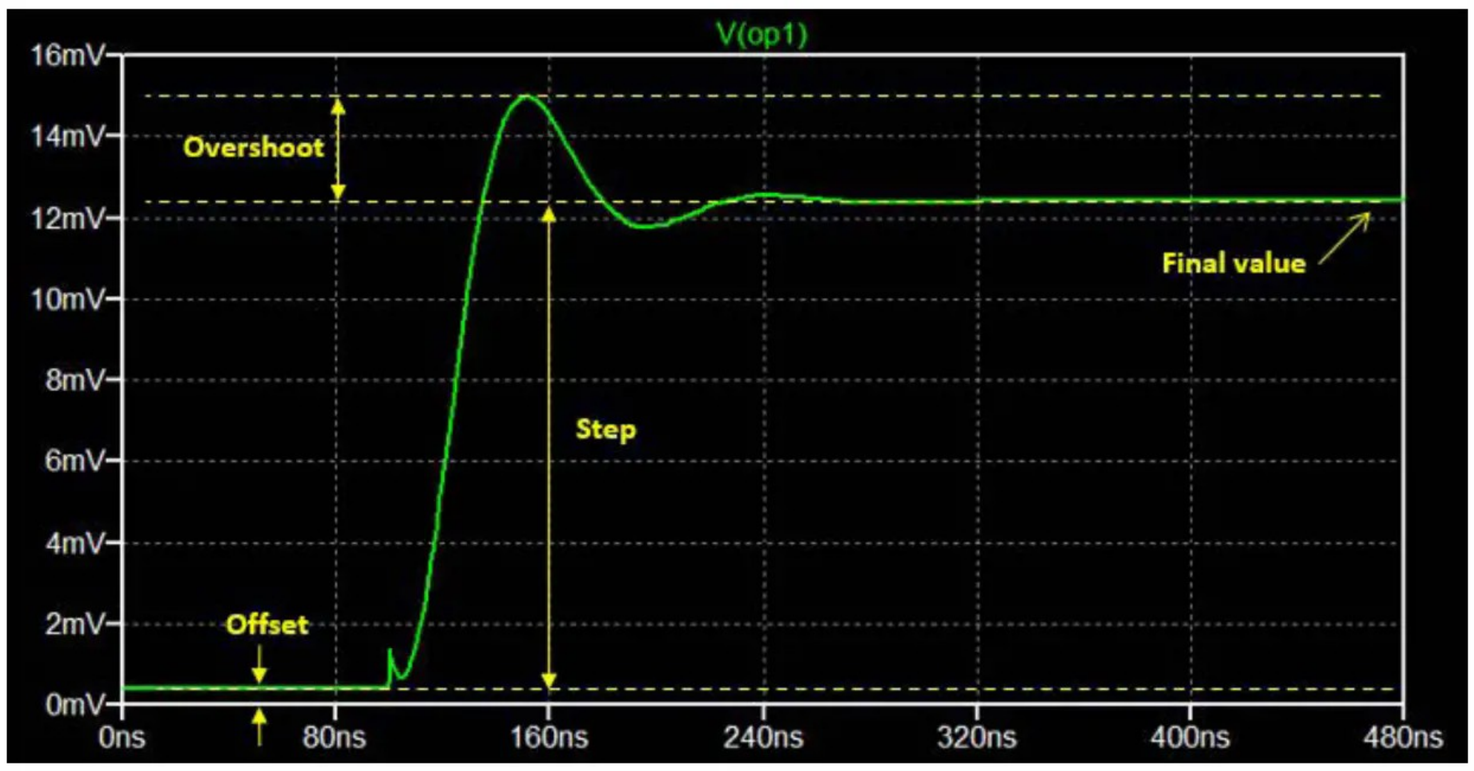


Fig.19: measuring overshoot.

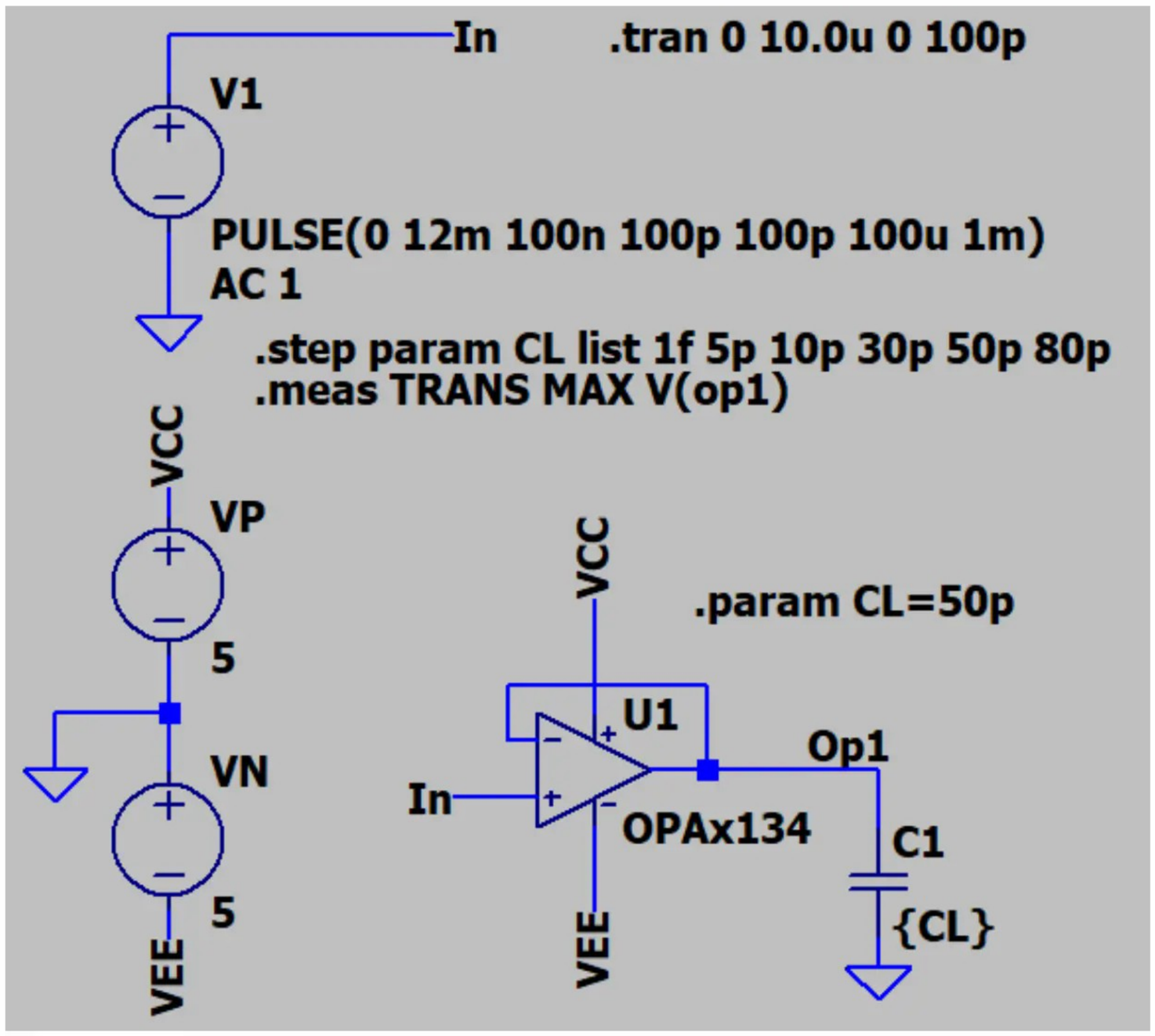


Fig.20: an LTspice schematic for measuring overshoot.

hence a poorer phase margin for this circuit) results in larger peaking in the gain.

Indirect phase margin measurement

The closed-loop results above show that overshoot and gain peaking vary with capacitive loading. It follows that there is a relationship between the amount of overshoot and peaking and the phase margin. This is potentially useful because it is difficult to measure loop gain directly on physical circuits (to obtain phase margin).

It is also not totally straightforward in simulation; you have to insert the FRA device or voltage source in the loop, as we have done in these articles.

Measuring overshoot or gain peaking could allow you to indirectly measure phase margin if you knew the relationship between them. Texas Instruments provides curves of these relationships in a presentation on measuring op amp stability – see the PDF at **https://pemag.au/link/acam**

They indicate that these curves are obtained from the mathematical relationship between the quantities, but do not give any details. It may be based on relating a generic system equation (eg, a second-order system), for which the Q (or damping factor) to overshoot or peaking relationship is known, to the shape of op amp response.

The phase margin can be related to pole frequencies, which in turn can be mapped to the generic response equation.

I was interested to see how well the Texas Instruments curves correspond to the simulation results with the OPAx134. I obtained values for overshoot and phase margin over a wide range from no load to close to fully unstable.

This was done using the approaches described above, but I used a 12mV rather than 100mV step in line with Texas Instruments' recommendation to use output steps in the 10-20mV range to prevent other aspects of op amp behaviour influencing the results.

For this approach, you have to apply the step waveform directly to an op amp input to get a valid result (intervening circuitry may distort the step), but this was already the case here anyway.

As shown in Fig.19, overshoot is the difference between the peak voltage caused by the step input and the final stable output voltage after the ringing has settled to zero (or some minimal amount). The step value is the voltage change, which takes account of any offset (Fig.19 shows a 12mV step with 0.4mV offset). Overshoot can be expressed as a percentage of the step value.

Measuring the overshoot manually is a little time consuming, but it can be automated using an LTspice `.meas` directive. Fig.20 shows an LTspice schematic for implementing this. This is similar to Fig.14, but instead of drawing multiple copies of the buffer amplifier, we use a single circuit

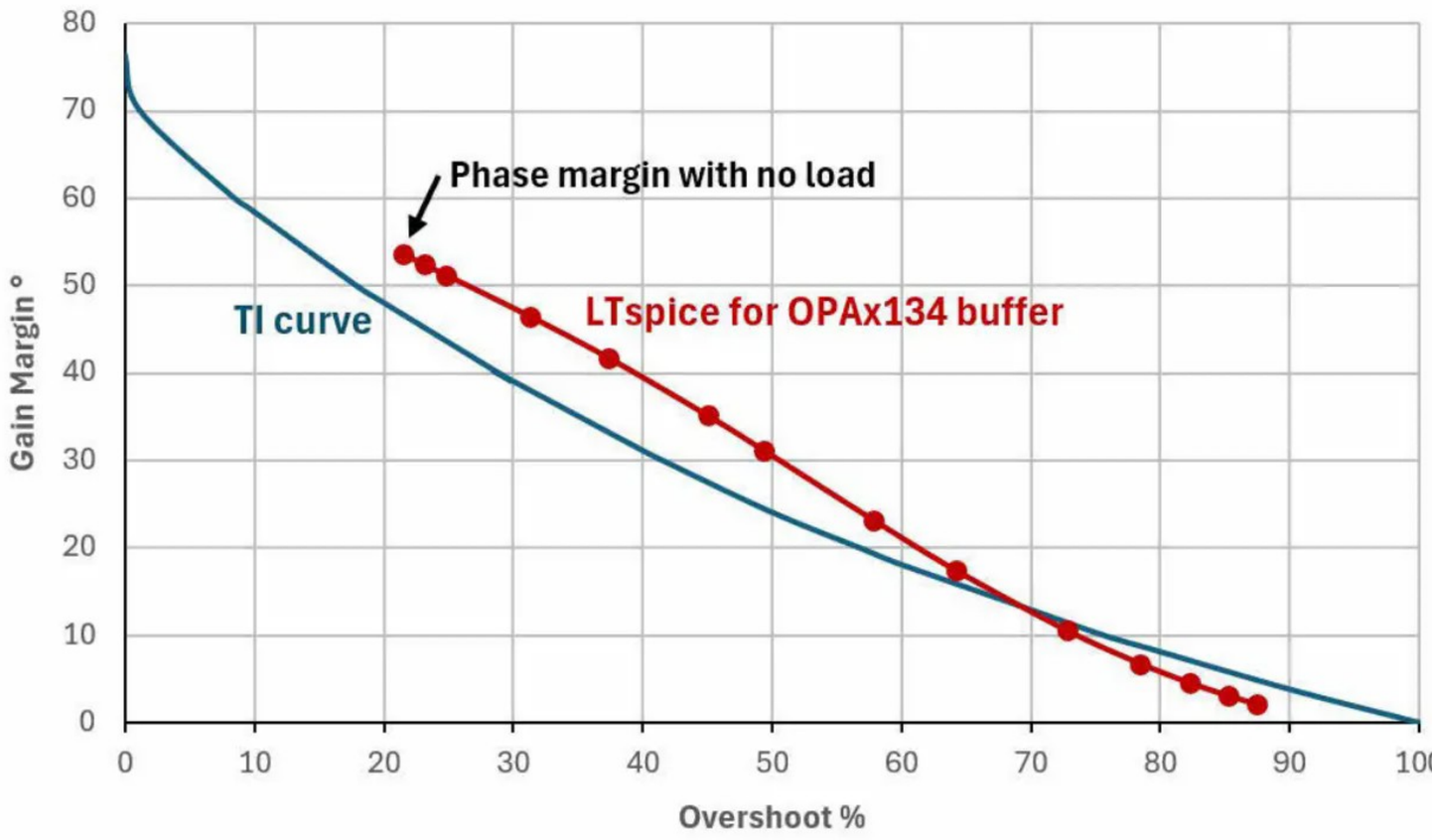


Fig.21: the approximate relationship between step response percentage overshoot and phase margin.

with a parameterised load capacitor. Only part of the capacitor value list is shown to save space.

The capacitor value for C1 is written as {CL}, where the curly brackets (braces) denote a parameter value, which can be set using the **.param** directive or stepped through multiple values using the **.step** directive. The **.meas** directive can be used to evaluate various useful values from simulation results.

In this case, we obtain the maximum value of the op amp output voltage (using **MAX V(op1)**), which is the value of the peak voltage of the overshoot.

Values obtained using **.meas** are printed in the simulation log text file, which can be viewed using View → Spice Output Log from the main menu after the simulation runs. The table of values can be copied and pasted into a spreadsheet to facilitate further calculations and plotting of data. In this case, the percentage overshoot was calculated using $100 \times (\max(V_{op1}) - V_{step} - V_{offset}) \div V_{step}$.

The same load capacitor values used in the overshoot simulation were simulated using a version of Fig.7 to obtain the corresponding phase margins.

The comparison with the Texas Instruments curve is shown in Fig.21. The match is reasonably close but not perfect. This may be because the Texas Instruments curve assumes a generic system response (as mentioned above) but the op amp has a more complex behaviour.

However, the Texas Instruments curve provides a reasonable estimate for the phase margin. I used AC analysis for the phase margin as it was quicker than FRA, but FRA may have produced results closer to the Texas Instruments curve at low phase margins.

Conclusion

You have to be a bit careful about the capacitances connected to an op amp, especially a high-bandwidth type. Sometimes an op amp will appear to be relatively stable, but a transient 'shock' could kick it into oscillation if the configuration doesn't have enough phase margin.

Worse still, you could have a prototype that works fine, but another unit built from a different batch of ICs could be unstable. Even if it doesn't enter sustained oscillation, it could still 'misbehave' under the right (or wrong) conditions unless you do your homework. PE

ETI BUNDLE (1) Teach-In 3, 4 and 5 – all on CD-ROM – only £18.95

ELECTRONICS TEACH-IN 3 – CD-ROM
Mike & Richard Tooley



The three sections of the *Teach-In 3* CD-ROM cover a huge range of subjects that will interest everyone involved in electronics – from newcomers to the hobby and students to experienced constructors and professionals.

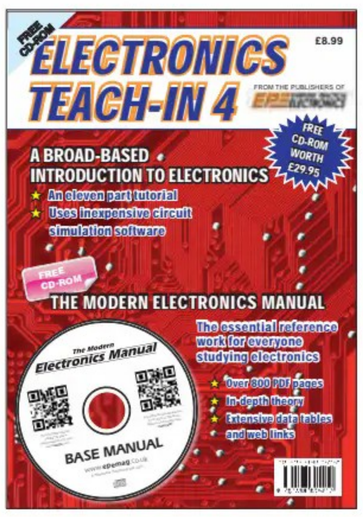
The first section (80 pages) is dedicated to *Circuit Surgery*, *EPE/PE's* regular clinic dealing with readers' queries on circuit design problems – from voltage regulation to using SPICE circuit simulation software.

The second section – *Practically Speaking* – covers hands-on aspects of electronics construction. Again, a whole range of subjects, from soldering to avoiding problems with static electricity and identifying components is covered. Finally, our collection of *Ingenuity Unlimited* circuits provides over 40 circuit designs submitted by readers.

The CD-ROM also contains the complete *Electronics Teach-In 1* book, which provides a broad-based introduction to electronics in PDF form, plus interactive quizzes to test your knowledge and TINA circuit simulation software (a limited version – plus a specially written TINA Tutorial).

The *Teach-In 1* series covers everything from electric current through to microprocessors and microcontrollers, and each part includes demonstration circuits to build on breadboards or to simulate on your PC.

ELECTRONICS TEACH-IN 4 – CD-ROM A BROAD-BASED INTRODUCTION TO ELECTRONICS
Mike & Richard Tooley



The *Teach-In 4* CD-ROM covers three of the most important electronics units that are currently studied in many schools and colleges. These include, Edexcel BTEC level 2 awards and the electronics units of the Diploma in Engineering, Level 2.

The CD-ROM also contains the full *Modern Electronics Manual*, worth £29.95. **The Manual** contains over 800 pages of electronics theory, projects, data, assembly instructions and web links.

A package of exceptional value that will appeal to anyone interested in learning about electronics – hobbyists, students or professionals.

ELECTRONICS TEACH-IN 5 – CD-ROM JUMP START
Mike & Richard Tooley



15 design and build circuit projects for newcomers or those following courses in school and colleges.

The projects are: ■ Moisture Detector ■ Quiz Machine ■ Battery Voltage Checker ■ Solar-Powered Charger ■ Versatile Theft Alarm ■ Spooky Circuits ■ Frost Alarm ■ Mini Christmas Lights ■ iPod Speaker ■ Logic Probe ■ DC Motor Controller ■ Egg Timer ■ Signal Injector Probe ■ Simple Radio Receiver ■ Temperature Alarm.

PLUS

PIC'n' Mix – starting out with the popular range of PIC microcontrollers and *Practically Speaking* – tips and techniques for project construction.

The CD-ROM also contains:
■ Complete *Teach-In 2* book, a practical introduction to PIC microprocessors
■ MikroElektronika, Microchip and L-Tek PoScope software.

Three Teach-ins for the great price of £18.95

PLUS you also get the contents of the free CD-ROM from each issue...

... so that's another TWO Teach-Ins and The Full Modern Electronics Manual!

What a Bargain!!

DOWNLOAD YOUR PE/EPE BACK ISSUES

COMPLETE-YEAR collections – all 12 issues for each year!

From just £5.95 each

VOL 28 – January to December 1999 – **£5.95**
VOL 29 – January to December 2000 – **£6.95**
VOL 30 – January to December 2001 – **£7.95**
VOL 31 – January to December 2002 – **£7.95**
VOL 32 – January to December 2003 – **£7.95**
VOL 33 – January to December 2004 – **£7.95**
VOL 34 – January to December 2005 – **£8.95**
VOL 35 – January to December 2006 – **£8.95**
VOL 36 – January to December 2007 – **£8.95**
VOL 37 – January to December 2008 – **£8.95**
VOL 38 – January to December 2009 – **£10.95**
VOL 39 – January to December 2010 – **£11.95**
VOL 40 – January to December 2011 – **£12.95**
VOL 41 – January to December 2012 – **£14.95**
VOL 42 – January to December 2013 – **£15.95**
VOL 43 – January to December 2014 – **£16.95**
VOL 44 – January to December 2015 – **£17.95**
VOL 45 – January to December 2016 – **£19.95**
VOL 46 – January to December 2017 – **£22.95**
VOL 47 – January to December 2018 – **£27.95**
VOL 48 – January to December 2019 – **£27.95**
VOL 49 – January to December 2020 – **£29.95**
VOL 50 – January to December 2021 – **£32.95**
VOL 51 – January to December 2022 – **£34.95**
VOL 52 – January to December 2023 – **£34.95 – NEW!**
VOL 53 – January to December 2024 – **£34.95 – NEW!**

FIVE-YEAR collections – 60 issues!

From just £14.95 each

PE2832: Jan 99 to Dec 03 – **£14.95**
PE2933: Jan 00 to Dec 04 – **£14.95**
PE3034: Jan 01 to Dec 05 – **£14.95**
PE3135: Jan 02 to Dec 06 – **£14.95**
PE3236: Jan 03 to Dec 07 – **£14.95**
PE3337: Jan 04 to Dec 08 – **£14.95**
PE3438: Jan 05 to Dec 09 – **£19.95**
PE3539: Jan 06 to Dec 10 – **£25.95**
PE3640: Jan 07 to Dec 11 – **£25.95**
PE3741: Jan 08 to Dec 12 – **£25.95**
PE3842: Jan 09 to Dec 13 – **£33.95**
PE3943: Jan 10 to Dec 14 – **£33.95**
PE4044: Jan 11 to Dec 15 – **£33.95**
PE4145: Jan 12 to Dec 16 – **£35.95**
PE4246: Jan 13 to Dec 17 – **£37.95**
PE4347: Jan 14 to Dec 18 – **£38.95**
PE4448: Jan 15 to Dec 19 – **£39.95**
PE4549: Jan 16 to Dec 20 – **£42.95**
PE4650: Jan 17 to Dec 21 – **£44.95**
PE4751: Jan 18 to Dec 22 – **£44.95 – NEW!**
PE4852: Jan 19 to Dec 23 – **£44.95 – NEW!**

20 YEARS OF PE!

AMAZING BUNDLE OFFER
Every issue from 1999 – 2018

240 magazines for just

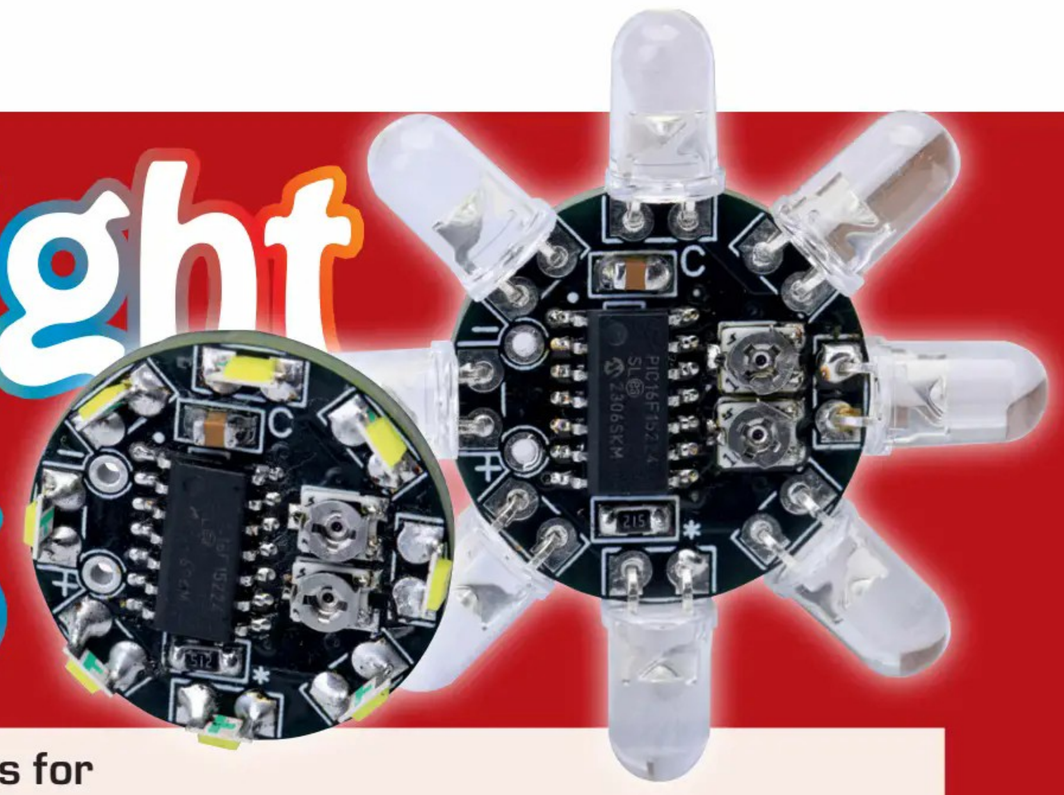
£69.95

PE2832: Jan 99 to Dec 03
PLUS!
PE3337: Jan 04 to Dec 08
PLUS!
PE3842: Jan 09 to Dec 13
PLUS!
PE3447: Jan 14 to Dec 18

Download your collections at: www.electronpublishing.com

Constructional Project

Rotating Light for Models



Here's a simple circuit that has various applications, such as for a model lighthouse, or as a siren on top of a model emergency vehicle. It sequences eight LEDs, using PWM brightness control, to form a pretty convincing imitation of a rotating light.

This project originated from a family member's desire for a white revolving light atop a miniature lighthouse. Kits to build this sort of thing are available, but we hadn't published such a circuit, and I thought it might have other uses. I also thought it could be done simply, on a tiny PCB.

We could use a single logic IC if all we wanted was essentially a circular LED chaser. However, I have seen that approach used (eg, on garbage trucks); while eye-catching, it doesn't provide a convincing illusion of rotation. Moreover, most digital logic ICs can't deliver much current, meaning the LEDs wouldn't be that bright without extra transistors.

With a microcontroller, we can fade the LEDs in and out, creating a much more impressive effect, even with just eight LEDs at 45° intervals. We can also make it adjustable; not just the rate of rotation, but also the direction and the brightness/beam angle. We can even have multiple 'beams' by lighting opposite LEDs, as shown in Fig.1.

Adjusting the beam angle effectively controls how many LEDs can be lit at once. It can range from just one (with varying brightness) up to almost all of them being lit at once, with just a dim spot rotating.

If you build it with white LEDs, it's suitable for a model lighthouse, and with a compact, black PCB that's just 20mm in diameter, it will fit in most models unless they are tiny. If you want to make a siren, you could use amber, blue, red, yellow or green LEDs, or even unusual colours like cyan or pink (yes, they're available).

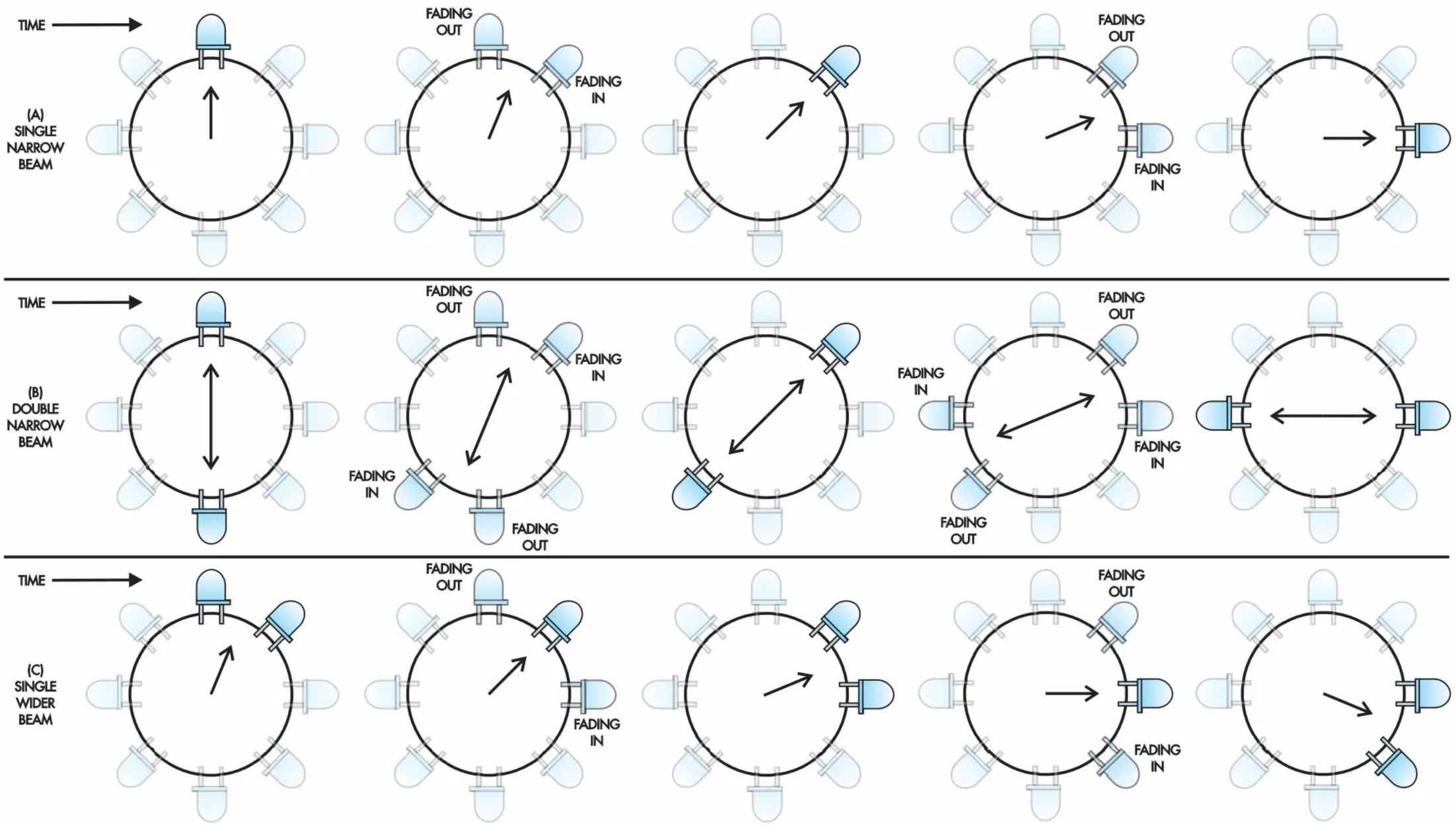


Fig.1: by fading in one LED at the edge of each beam and fading out the opposite one, we create the illusion of a smoothly-rotating light with just eight fixed LEDs. The 'beam' brightness and width varies depending on how many LEDs are lit at any given time.

Rotating Light for Models

Features & specifications

- Simulates a rotating light with one or two ‘beams’
- Adjustable rotation speed, from about 10 rotations per second to 30 seconds per rotation (0.03Hz to 10Hz)
- Reversible rotation direction
- Adjustable beam brightness
- Adjustable beam angle from 45° to nearly 360°
- Runs from 5-12V DC, typically drawing 10-20mA
- Compact circular PCB: 20mm in diameter and less than 10mm tall (with SMD LEDs)
- Can use SMD (M3216/1206/SMA) or through-hole (3mm/5mm) LEDs
- Use any colour LEDs

Project by Nicholas Vinen

Suppose you want a really special effect. In that case, you could build it with differently coloured LEDs, so the colour shifts as it rotates!

It can run from a small DC supply from 5V up to about 12V, including most small battery packs, such as standard 9V batteries or 6V batteries of four AA/AAA cells. The current draw depends on the brightness, but it’s typically around 10mA. So four AA cells would power it for quite a while; possibly as long as two weeks for really high-capacity cells. Four AAAs might last 5-7 days.

Circuit details

The complete Rotating Light circuit is shown in Fig.2. A 14-pin, 8-bit PIC16F15224 was chosen as it has just enough pins, is inexpensive and draws very little current. It can drive the LEDs directly with fairly decent brightness (its maximum per-pin current is 25mA). It’s also easy to program with the free version of Microchip’s XC8 compiler and MPLAB X IDE.

The eight LEDs are connected to eight of its digital outputs via 68Ω current-limiting resistors. Their anodes connect directly to the 5V rail, and they light when the microcontroller pulls that output pin low, to 0V. This configuration was selected as the micro’s output transistors are better at sinking current than sourcing it, as is typical. So they will deliver a higher maximum current like this.

Assuming white or blue LEDs with a forward voltage of around 3.3V and a 5V supply, there will be around 1.7V (5V – 3.3V) across the combination of the 68Ω resistors and the micro’s output transistors.

With a 3V supply, the data says that those output transistors can sink 10mA with a 0.6V saturation voltage, implying an output impedance of 60Ω (0.6V ÷ 10mA). It might be lower with the higher 5V supply voltage used in this circuit, but let’s use 60Ω as the worst-case value.

That means the 1.7V is across 128Ω (68Ω + 60Ω), so we can expect the LEDs to be driven with a peak current of about 13mA (1.7V ÷ 128Ω). LEDs with a lower forward voltage, like red or amber, would receive more current, likely around 20mA. So the peak current is limited to a safe level. The microcontroller can control the average current using pulse-width modulation (PWM).

One nice feature about this microcontroller is that its two PWM peripherals can be dynamically mapped to any of its I/O pins. So as the light ‘rotates’, we can determine the two edge LEDs and assign them to the PWM peripherals to dim them. The other LEDs are either fully off or full on, as determined by the states of the other digital outputs.

That means that all the LEDs are controlled by hardware, with the software just needing to periodically recalculate which LEDs should be lit. It can then update the PORT and PWM registers to advance the rotating light to the next position.

Two trimpots, VR1 & VR2, connect across the 5V supply with their wipers going to pins 8 & 11 of IC1. These are

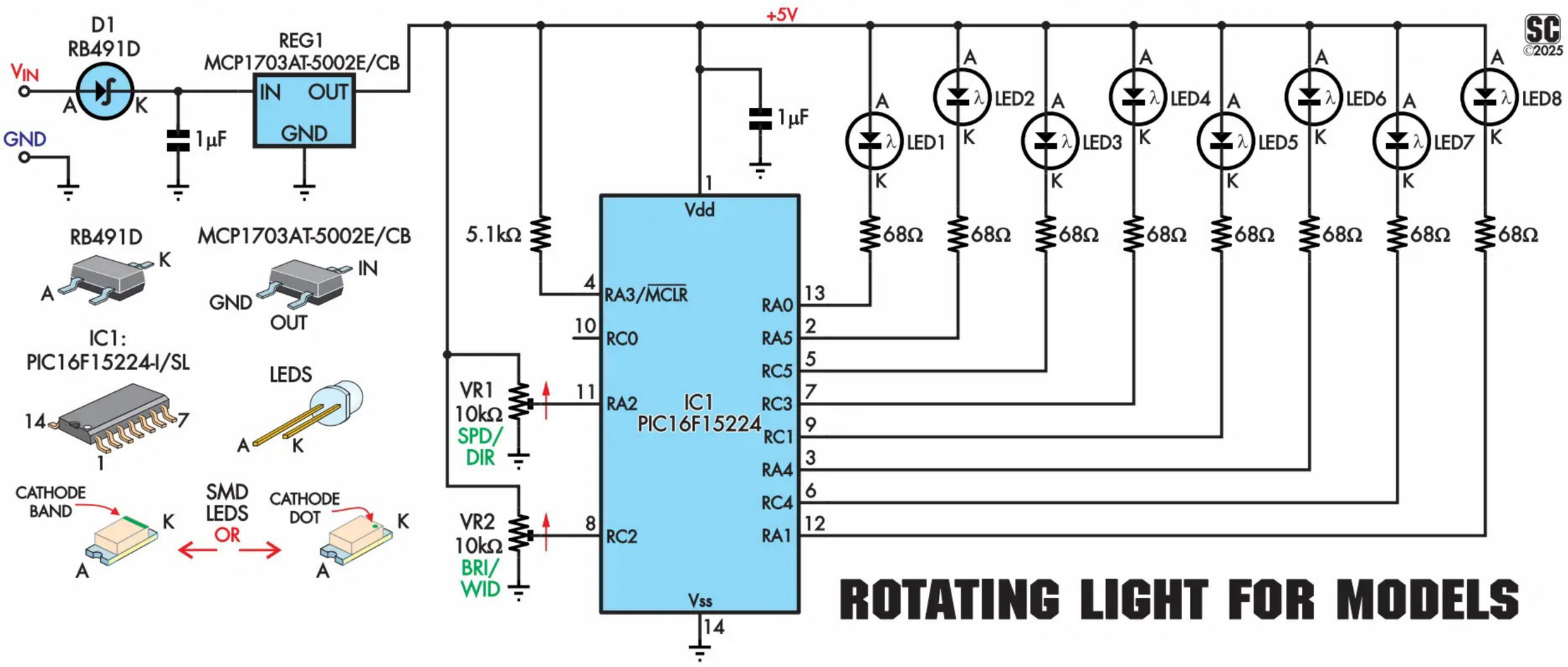


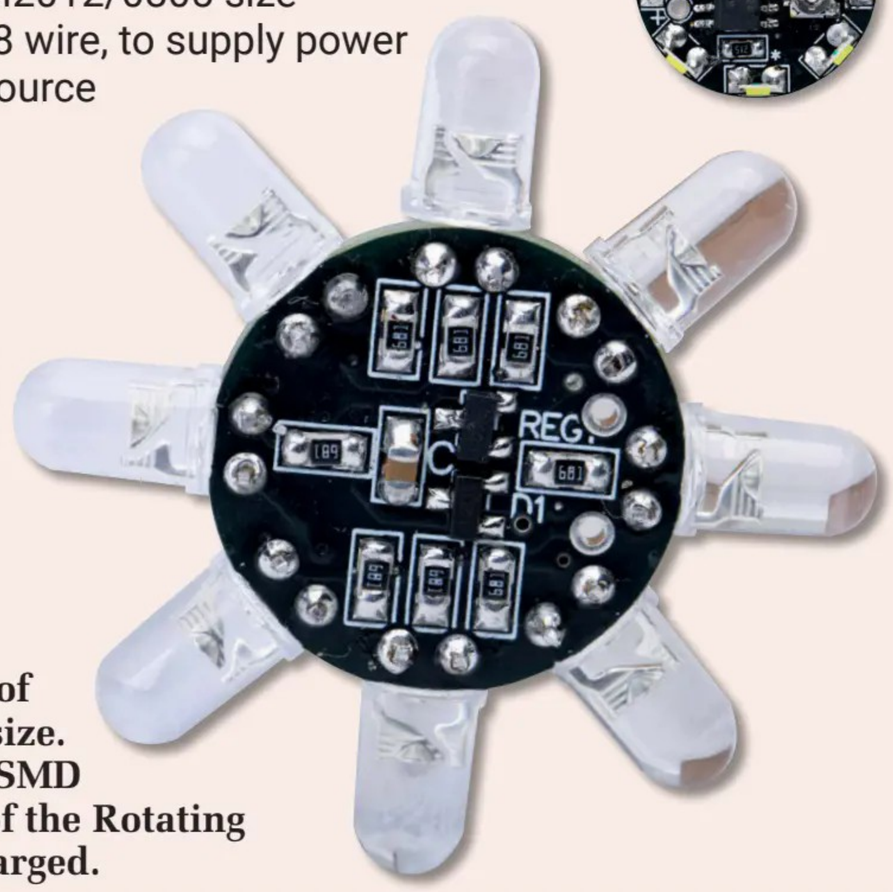
Fig.2: the circuit is little more than eight LEDs driven by the microcontroller via current-limiting resistors, two potentiometers to set the parameters and a simple linear power supply.

Parts List – Rotating Light

1 double-sided black PCB coded 09101251, 20 × 20mm
8 high-brightness LEDs, 3mm/5mm through-hole or SMD (SMA, M3216/1206 or M2012/0805 size), colour to suit application
1 PIC16F15224-I/SL 8-bit micro programmed with 0910125A.HEX, SOIC-14 (IC1)
1 MCP1703AT-5002E/CB 5V 250mA low-dropout linear regulator, SOT-23 (REG1)
1 RB491D 20V 1A schottky diode, SOT-23 (D1)
2 1µF 16V X7R ceramic chip capacitors, M3216/1206 size
2 10kΩ TC33X-2-103E SMD trimpots (VR1, VR2)
1 5.1kΩ SMD chip resistor, M2012/0805 size
8 68Ω SMD chip resistors, M2012/0805 size
1 length of light duty figure-8 wire, to supply power
1 5-12V DC 100mA power source







At upper right, the top side of the PCB is shown at actual size. The underside views of the SMD and through-hole versions of the Rotating Light project are shown enlarged.

SMD LED kit (SC7462; ~£12 + postage) | TH LED kit (SC7463; ~£12 + postage)
Both kits includes all the parts listed above, except the power supply and wire

both analog-capable pins, so we can use the micro's internal analog-to-digital converter (ADC) to measure these voltages. VR1 controls the speed & direction of 'rotation', while VR2 controls the beam width & brightness.

Usually, you would put capacitors on these pins to keep the AC impedance low, making the ADC results more precise, but there isn't a lot of room on the PCB, so we've left them off. We don't need to make super accurate measurements, and we can compensate for the lack of capacitors either by tweaking the software or by eye when making the adjustments.

In practice, we found that the ADC measurements were close enough to what you would expect based on the trimpot positions without these extra capacitors.

The 5.1kΩ pull-up resistor on the $\overline{\text{MCLR}}$ pin (pin 4 of IC1) prevents spurious resets, while the 1µF capacitor across its supply pins provides bypassing for stability. The 5.1kΩ value is not critical; it could be 4.7kΩ, 10kΩ or another similar value.

All that remains is the simple power supply. 5V low-dropout regulator REG1 ensures IC1 receives a steady and safe voltage, even if the incoming supply at CON1 varies. Schottky diode D1 prevents any damage from occurring if the supply polarity is accidentally reversed at CON1, while also having a modest (~0.3V) voltage drop.

REG1 requires an input bypass and output filter capacitor for stability, so we have provided 1µF in both cases. That is the minimum value for unconditional stability on the output, and is more than enough for input bypassing.

The circuit can be run from a 5V supply (eg, from USB), although the LED brightness will be reduced somewhat as IC1 and the LEDs will only have a supply of about 4.6V, ie, 5V minus D1's forward voltage (~0.3V) and REG1's dropout voltage (<100mV).

If you're going to power it from a regulated 5V supply like USB, you could omit REG1 and solder a bridge between its input and output pads. You could also bypass D1, or replace it with a 0Ω resistor, if you are certain that the supply polarity can't be reversed.

The maximum recommended supply voltage is 12V due to REG1's absolute maximum rating of 16V. With a 12V supply and 50mA average current draw, REG1 will dissipate 350mW ([12V – 5V] × 0.05A), giving an expected temperature rise of nearly 120°C, which would put it close to its shutdown temperature of 150°C at an ambient temperature of just 30°C.

The PCB draws enough heat away from REG1 that it's unlikely to shut down unless the current draw exceeds 50mA. Still, if you intend to run the Light with a bright, wide beam, you'd be better off with a supply voltage below 12V; 6-9V would be ideal. If you manage to overheat REG1, it won't be damaged; the light will just shut off and then restart when it cools down.

LED selection

The LEDs are arranged around the outside and can be through-hole (3mm or 5mm) or SMD types (M3216/1206 or M2012/0805). While side-emitting SMD LEDs exist, we reckon it's easier just to use regular SMD LEDs and mount them on their sides, with the emitters facing out. That's how we built our SMD prototype, shown in the photos.

Some reasons we don't think it's worth getting side-emitting SMD LEDs are:

1. They are many times pricier than the normal top-emitting type.
2. They aren't that much easier to solder than a top-emitting type facing sideways.
3. Many of them have a central pad for extra support that could short out the anode and cathode pads.
4. There are much more limited choices of size and colour compared to regular SMD LEDs.
5. Only the largest component sellers stock them.

Through-hole LEDs can be soldered on either side of the board, while SMD LEDs have to go on the top. You could perhaps get away with soldering smaller SMD LEDs across the pads on the bottom if you have a particular reason to do that.

Rotating Light for Models

PCB design

The PCB is circular with a 20mm diameter (10mm radius). By making it black, we can hide it inside models, so you only see the light when it's on.

In the middle of the top side of the PCB is the microcontroller, the two small SMD adjustment trimpots, one of the 1µF capacitors and the 5.1kΩ resistor. All the other components, like the LED current-limiting resistors and the remainder of the power supply, are in the middle of the underside.

The power connections are two solder pads to which wires can be soldered from either side of the PCB, to suit the installation.

Software

The software (**siliconchip.au/Shop/6/1837**) is just under 200 lines of C code. The PIC runs at 8MHz with its internal Timer0 used to control the rotation speed of the light and Timer2 to run the PWM peripherals used for LED dimming. At power-up, it sets the pins as analog inputs and digital outputs as required. It then initialises the two timers and the ADC.

The main loop waits for Timer0 to roll over, which happens every 4ms or so. Each time, it adds the rotation speed/direction to a 16-bit accumulator. It uses the accumulator value to calculate the brightness for the eight LEDs, then updates the output and PWM states. The 8-bit PWM runs at around 2kHz.

When Timer0 rolls over, it also measures the voltages at the two analog inputs and applies a low-pass filter to remove noise and glitches from those readings. The new readings are used for future light update calculations.

The code compiles to 1276 instruction words, taking up 2552 bytes of the 8kiB of the available flash memory (31.2%). The pro version of the XC8 compiler is not required.

The critical part that generates the 'rotating' light is actually quite simple. If you mentally unwrap the circular light pattern into a line, you end up with a bi-directional chaser that 'wraps around' from one end to the other. The mathematics to calculate that, even with the LED brightness smoothly changing, is relatively simple.

In twin-beam mode, with VR2 closer to the clockwise end than anti-clockwise, the chaser shifts so that there are two lit areas exactly four LEDs apart. Many lighthouses and sirens that use actual rotating lights will emit light from both ends, so this mode better simulates that appearance.

Construction

The Rotating Light is built on a double-sided PCB coded 09101251 that measures 20 × 20mm. The top and bottom component overlay diagrams are in Fig.3, with two versions shown to suit SMD or through-hole LEDs. Refer to those during construction to see which parts go where.

If you find the small board slides around while working on it, use a little Blu-Tack to temporarily stick it to your work surface.

None of the components are terribly hard to solder individually. We found the main challenge to assembly was to avoid accidental bridges between adjacent pads because they are quite close together due to the small size of the PCB, especially the two trimpots and the two SOT-23 devices that mount side-by-side. So it's best to feed in solder carefully and use the minimum necessary to form good fillets.

The microcontroller IC has fairly widely spaced pins, on a 1.27mm pitch and other parts have larger or more widely spaced pins. So the actual soldering of individual components is not too difficult.

As there are parts on both sides, once you have fitted the parts on one side, the PCB won't easily sit flat and will tend to rock as you work on it. To deal with this, you can either use Blu-Tack as mentioned, or you could do what we did and place the PCB on top of a roll of solder-wicking braid. This has a hole in the middle for the components to fit in, so it can rest on its edges and sit flat.

Of course, that depends on you having a similarly sized roll of braid to us, but it worked surprisingly well for us.

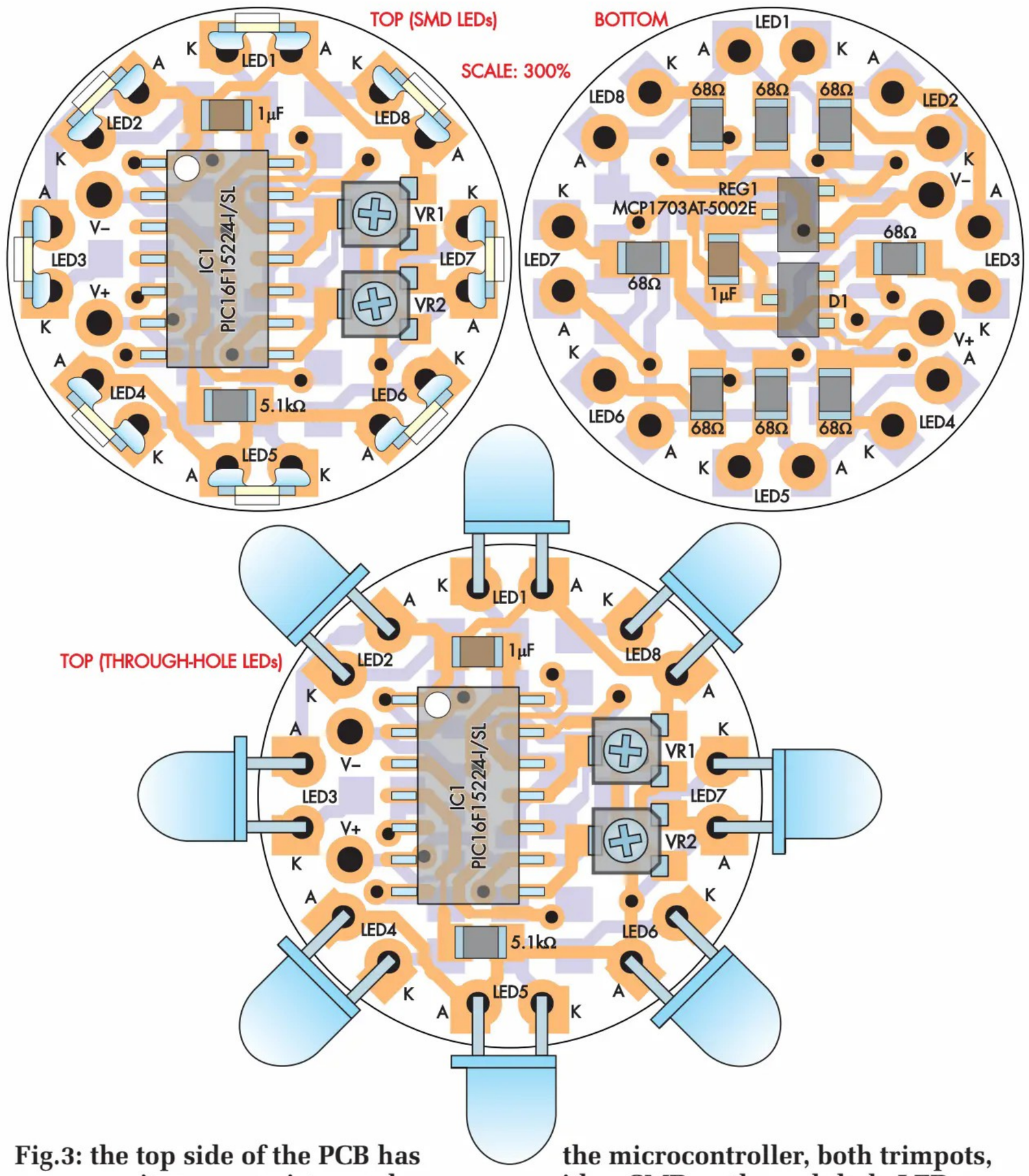


Fig.3: the top side of the PCB has the microcontroller, both trimpots, one capacitor, one resistor and either SMD or through-hole LEDs, although TH LEDs can also be inserted from the bottom side. All the current-limiting resistors are on the underside, along with most of the power supply.

There is no provision to program the microcontroller on the board, so you'll need to either purchase a pre-programmed micro (from the Silicon Chip Online Shop, either individually or in a kit), or program it yourself before soldering it. Our article on the PIC Programming Adaptor in the September 2024 issue explains how you can do it.

Once programmed, make sure you have identified pin 1 on the chip and lined it up with the marking on the PCB (very important!). Also check it against Fig.3, then tack-solder one pin. Adding a little flux paste will help the solder flow. Check the alignment of all the other pins with their pads (now is also a good time to double-check that pin 1 is in the right place!).

If the positioning is not perfect, remelt the solder joint and gently nudge the chip into position. Once it's located correctly, solder the diagonally opposite pin, then spread a little flux paste down both sides of the chip, over all the pins, and solder the remaining pins. You can drag solder them, or do them one at a time.

If solder bridges have developed between any pins, clean them up by adding a dab of flux paste and then using a clean piece of solder wick to remove the excess solder. Clean off the flux residue with a suitable solvent, then inspect the pins under magnification to ensure all the solder joints are good (solder has flowed onto both the pin and pad) and no bridges remain.

Solder the two trimpots similarly, being careful to avoid bridges to adjacent pads due to their proximity to IC pins and LED pads. We found the trimpots were the trickiest parts of all to solder because the pads didn't stick out very far from underneath them.

We've extended them in the final version of the PCB, but there was limited space available to do so. Add flux paste on both the PCB and component leads before soldering need to be careful to check that the solder has flowed down on the PCB pads before moving on to other components.

With the trimpots soldered correctly, add the sole top-side capacitor and one resistor. Finally, if you are fitting SMD LEDs, you can do that now.

Soldering the LEDs

We recommend soldering standard SMD LEDs on their side, like in our photos. First, figure out which end of the LED is the cathode. You can do this with a DMM on diode test mode. When the LED lights up, the black probe is on the cathode. It must go to one of the pads marked "K" in Fig.3.

Start soldering each SMD LED by adding solder to one of its pads. Due to the through-holes, you'll need to add more than you might expect until you get sufficient solder on the top surface. You want a visible bulge so enough solder will reach the pad on the side of the LED, rather than the bottom as usual.

The hardest part of soldering the SMD LEDs on their side was picking them up with the tweezers. We found the easiest way was to pick them up from the bench with one hand, rotate them on their side, then grab them with tweezers using the other hand. Make sure the tweezer tips don't extend past the bottom of the LED or you won't be able to get it to sit flat on the PCB.

Once we had picked them up correctly, we found that soldering them wasn't too hard. Position the LED with tweezers while keeping the solder on that initial pad molten with your soldering iron. Remove the iron for a few seconds to let it solidify, then check if the position is good. If it is, add a fillet to the other pad. The LEDs don't need to be perfectly aligned but it helps if they are close.

If you aren't happy with the LED position, you can grab it again with the tweezers, reheat the initial joint and nudge it into place.

Once both sides are soldered, you may need to add a dab of flux paste to the first pad and heat it to reflow the solder and form a nice, shiny fillet.

With all the top-side components fitted, flip the board over and add the remaining SMDs, as shown in Fig.3. Don't get D1 & REG1 mixed up. None of the other components are polarised.

If using through-hole LEDs, bend their leads consistently and solder them in place now. You can insert them from either side of the PCB but make sure when you bend the leads that the shorter (cathode) lead will always go into a pad marked "K" in Fig.3.

Now solder the power leads to their pads. They are marked with + and – symbols on one side of the PCB. You can solder them from either side.

Testing

If you have a current-limited supply, set it to 6V and 25-50mA. Otherwise, you could include a series resistor (eg, 100Ω 5W from a 12V supply) to limit the current in the event of a fault. Apply power and check the current flow. Depending on the trimpot positions, it should be around 10-20mA and should definitely not exceed 50mA. Verify that the LEDs light up and start to sequence.

If the current draw is too high, switch off and inspect the board for faults, such as solder bridges between pads or pins, or incorrectly placed or orientated components. Perform similar checks if there is no current draw or nothing happens. Also check that all solder joints have been made correctly.

If it operates but some LEDs don't light, likely they are faulty, their solder joints are bad, or they are shorted to an adjacent pad. If it appears to be working, try adjusting VR1 & VR2 to verify that you can change the rotation speed, direction and beam brightness/width as expected.

We found that many of our Phillips head screwdrivers of various sizes failed to actually rotate the trimpot. We had to search around until we found a slotted screwdriver of the perfect size before we could get sufficient purchase. After that, we could make easy and precise adjustments.

Usage

With VR1 centred, rotation is very slow; if it is perfectly centred, the light will not rotate, or just barely. It 'accelerates' in either direction as you move towards the clockwise or anti-clockwise extremes. This gives a reasonable range of speed options without making the adjustment super fiddly.

With VR2 centred, you will have a narrow (45°), dim beam. As you move it anti-clockwise, the beam will first start to brighten, then widen. At about halfway between anti-clockwise and the centre, you will have a bright 45° beam. As you move closer to anti-clockwise, the beam will get wider and wider until it occupies almost the whole circumference.

Rotating it from the centre clockwise is similar except that you will have two opposing 45° beams that get brighter, then wider.

If you want to power this board from a USB supply, we have an upcoming article on USB Power Adaptors. You would need to join the two boards with a short length of light-duty twin lead or similar. PE

Subscribe to SILICON CHIP



SILICON CHIP is one of the best DIY electronics magazines in the world. Each month is filled with a variety of projects that you can build yourself, along with features on a wide range of topics from in-depth electronics articles to general tech overviews.

	Print	Combined	Online
6 months	$105 ≈£52.50 ≈€60	$115 ≈£57.50 ≈€65	$50 ≈£25 ≈€30
1 year	$200 ≈£100 ≈€115	$220 ≈£110 ≈€125	$95 ≈£47.50 ≈€54
2 years	$390 ≈£195 ≈€220	$425 ≈£212.50 ≈€240	$185 ≈£97.50 ≈€105

- *Combined subscriptions include both the printed magazine and online access.*
- *Prices are in Australian dollars (AUD) unless otherwise specified.*
- *Prices listed are for everywhere in the world except Australia and New Zealand. View siliconchip.com.au/Shop/SubRates for a full list of current prices.*
- *Price estimates are just that and should only be used as a guide. We can only take payment in the form of Australian dollars.*

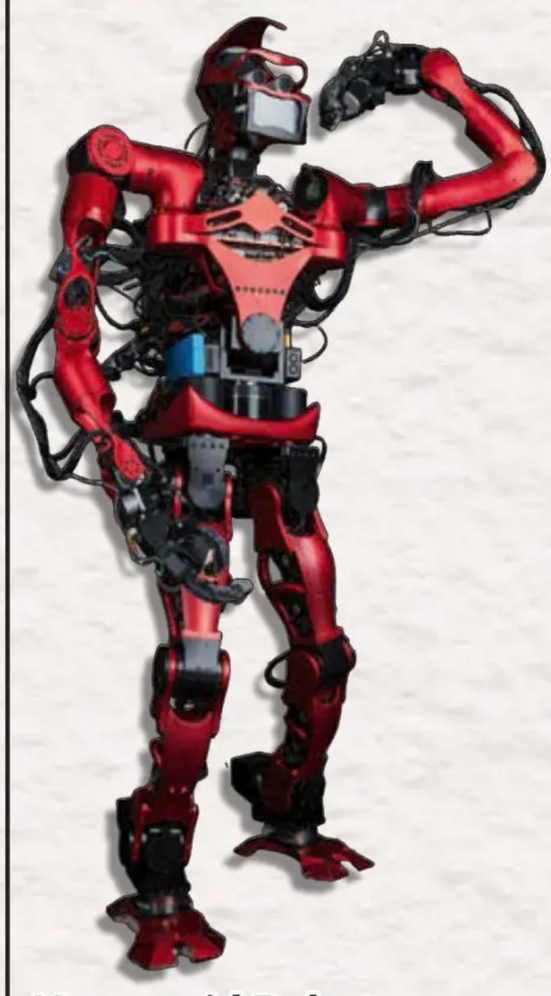
Try our Online Subscription – now with PDF downloads!

An online issue is perfect for those who don't want too much clutter around the house and is the same price worldwide. Issues can be viewed online, or downloaded as a PDF.

Published in SILICON CHIP



DCC Base Station & Remote Controller; Jan-Feb 2026



Humanoid Robots; Nov-Dec 2025

To start your subscription go to

siliconchip.com.au/Shop/Subscribe

By Max the Magnificent

Weird & Wonderful Arduino Projects

Part 15: gentlepersons, start your cartridges!

Eeek (and I mean that most sincerely)! I can't believe that it's almost March 2026. Even when I was a little kid, I always knew when it was the first day of March.

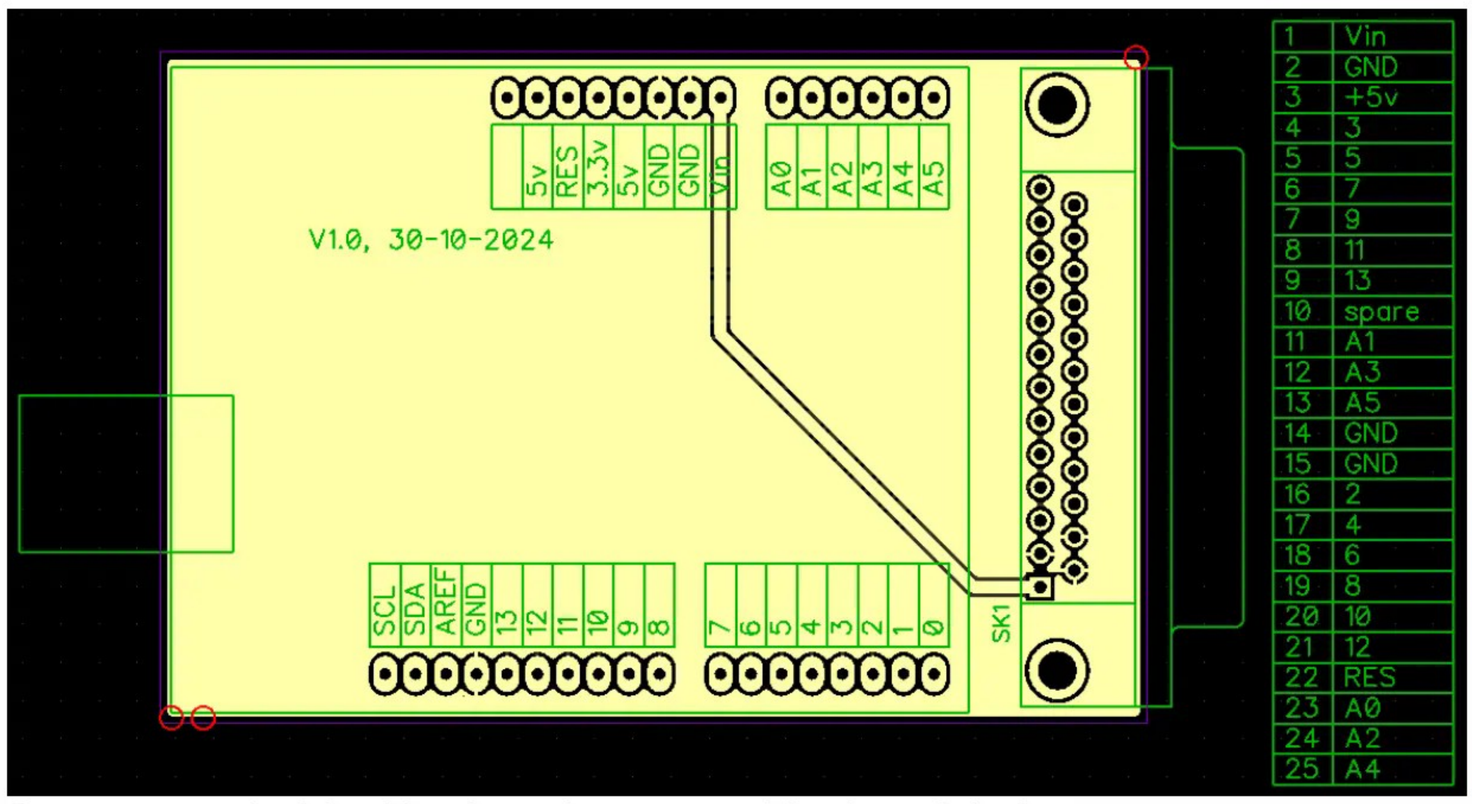
This wasn't because I was unusually precocious (well, not only that), but rather because my mum would bounce into my bedroom, shout "White Rabbits" at the top of her voice and bounce out again, leaving my ears ringing and my head spinning. One day, I'll have to ask her why she used to do this and why I'm now conditioned to carry on the tradition.

Making connections

Let's commence this column by dabbling on the hardware front. Last month, we finally finished wiring the internals of our retro games console (a thousand hurrahs!).

Since we first commenced work on this bodacious beauty, the idea has been to end up with the main console accompanied by a collection of plug-in game cartridges, much the way things were before the days of the internet, digital distribution and downloadable content.

In traditional 1980s and 1990s implementations, the 'brains' of the system (the main processor) lived in the console, while the game cartridges were essentially dumb read-only memory (ROM) modules. By comparison, we dare to be different because our console is more of a shared, unintelligent backplane that supplies power, buttons, and displays.



Screen 1: our Arduino Uno-based game cartridge board design.

In our implementation, each cartridge will contain its own 'brain' in the form of an Arduino microcontroller, with the game stored in the Arduino's flash memory.

As you may recall, our console has a 25-way female D-sub panel-mount connector at the top (see Fig.1). Our game cartridges will plug into this connector. The way we tackled things last time, to get things up and running as quickly as possible, was to use a DB25 male-to-bare-wire connector that I happened to find lurking in my treasure chest of parts (https://pemag.au/link/ac9q).

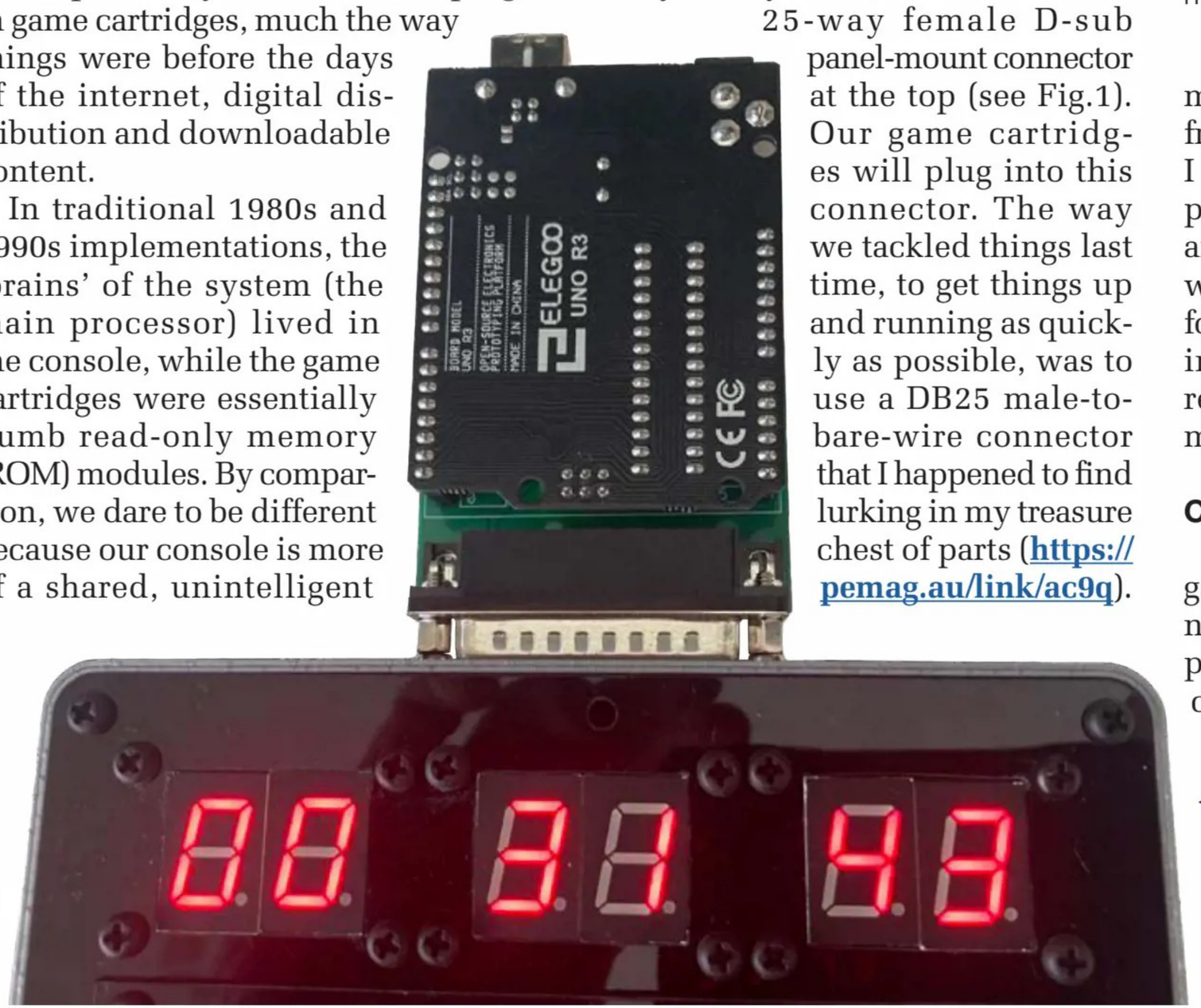


Photo 1: a 25-way D-sub right-angle male PCB-mount connector.

I plugged the connector end into my console and the appropriate wires from the other end into my Arduino. I then verified that the 7-segment display modules, the 14 × 10 tricolour LED array and the push-button switches worked as planned. They did. I performed my happy dance. I'm still wearing a foolish grin (but we shouldn't read too much into that because it's my usual look).

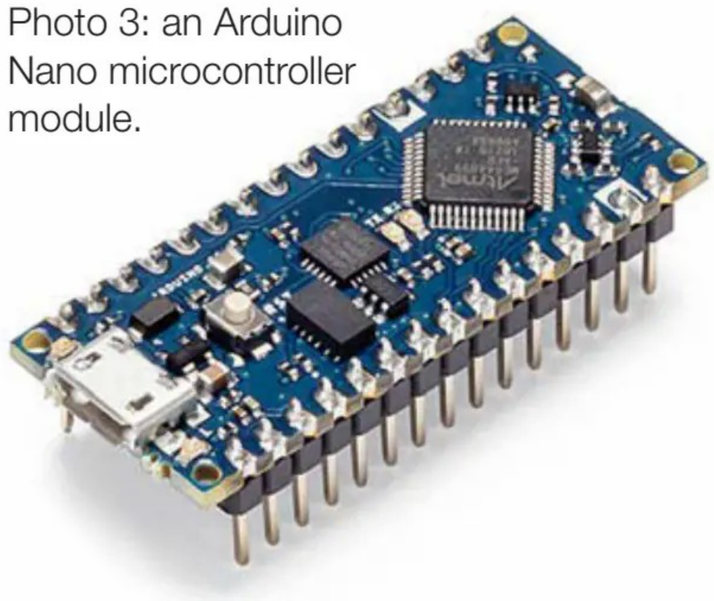
Creating a cartridge

When it comes to creating one of our game cartridges, the first thing we'll need is a 25-way D-sub right-angle male printed circuit board (PCB) mounting connector (Photo 1). Actually, we will



◀ Photo 2: the actual Arduino Uno-based game cartridge board.

Photo 3: an Arduino Nano microcontroller module.



need one of these for each of our game cartridges. I just found a pack of two for £3.15 on Amazon (**https://pemag.au/link/ac9w**).

You can probably get them cheaper if you buy them by the bushel (eg, from **DigiKey** or **Mouser** [DB25-PL-24]). You can start with a single cartridge that you can reprogram with different games and add more cartridges later if you wish.

The next thing we'll need is a cartridge-sized PCB to solder our connector to. Happily, my friend Joe Farr used the DipTrace Schematic and PCB Design Software package (**https://diptrace.com**) to create just such a board (see Screen 1). This shows a bird's-eye view of the top side of the board. Does anything look unusual to you? You're right; at first glance, all the pins appear to be the wrong way around!

This takes a little thought to wrap one's brain around. Our 25-way connector is inserted into the top side of the board and soldered on the bottom. Similarly, a bunch of male header pins are inserted on the top side and soldered on the bottom.

We use male header pins on this board because the Arduino Uno comes with female headers. This means we need to flip the Arduino upside-down to mount it on our cartridge. Photo 2 shows a completed cartridge, with the Arduino's underside facing us, plugged into my console (the Arduino is powered by 9V supplied via the 25-way connector from the console).

While I'm developing a new game, I simply leave this cartridge plugged into my console and reprogram it in place by connecting a USB cable from my host computer to the Arduino.

If you want to make your own Uno-based cartridges, Joe has kindly provided us with all the files needed to fabricate the boards. These are available in the file named **CB-mar26-uno-cartridge-v1.zip**, part of the March 2026 download package on the PE website at **https://pemag.au/link/ac9x** You can also get these boards pre-made from our shop (see page 69).

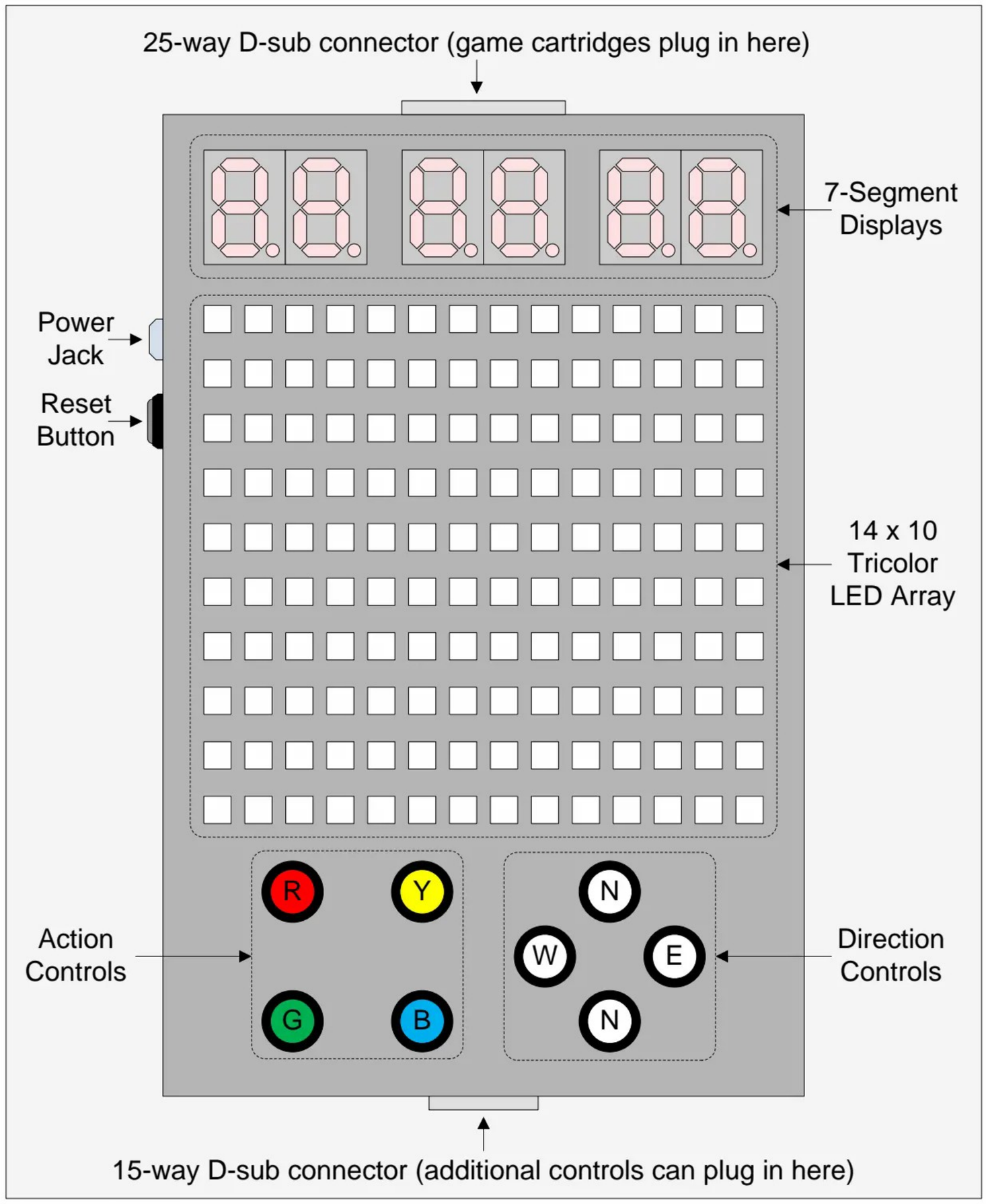


Fig.1: a view from the top of the main console.

Honey, I shrunk the board

It has to be acknowledged that our Uno-based game cartridges are a little on the 'clunky' side (I hope I'm not being too technical).

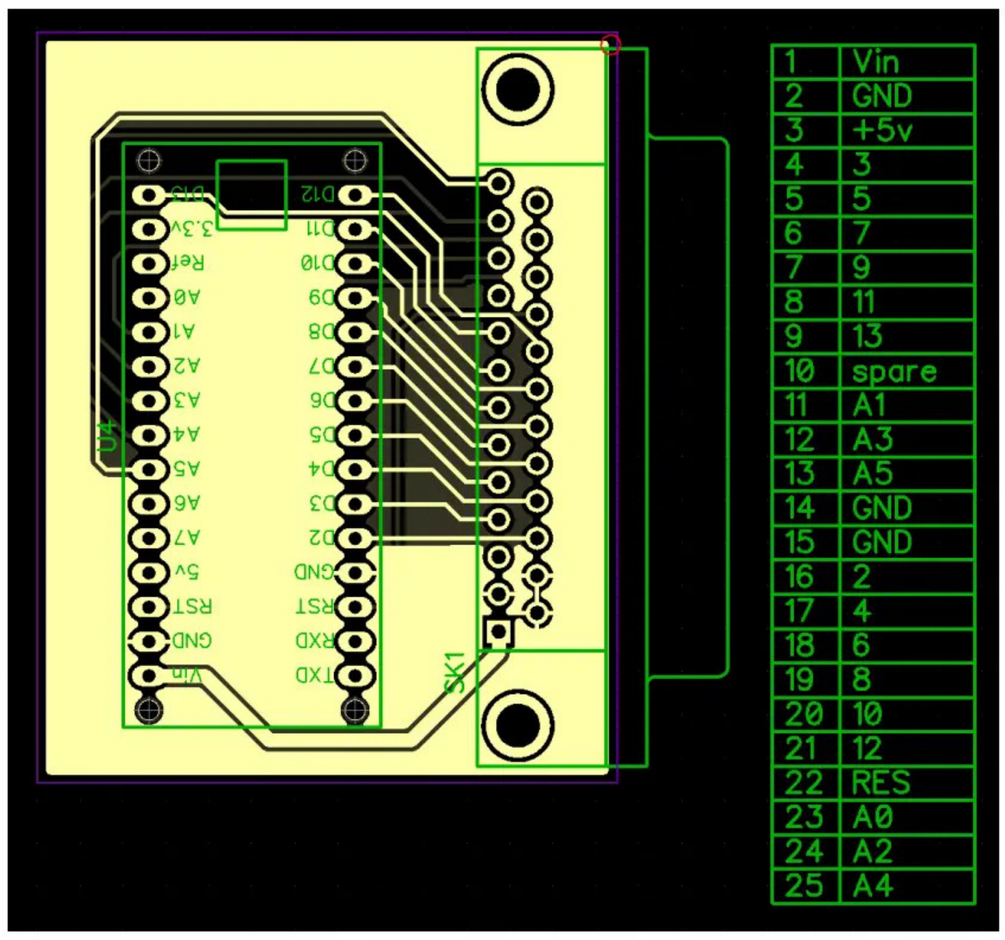
When I'm creating space-constrained projects, I often turn to the Arduino Nano (**https://pemag.au/link/ac9y**), which uses the same 8-bit processor as the Uno. In fact, it's identical in almost all respects, including its 16MHz clock, 32kiB of flash memory (for programs), 2kiB of RAM (for working memory/live data), 14 digital input/output (I/O) pins, and so on.

The main difference is its smaller form factor/footprint (see Photo 3).

Happily, Joe was on a roll when he created our Uno game cartridge board because he also threw one together for the Nano (Screen 2). Once again, this is a bird's-eye view of the top of the board.

The connector's pinout is identical to the Uno board's, or we would quickly find ourselves in a bit of a pickle. Since the Nano comes equipped with male header pins (you may have to solder these on yourself), we mount female header pins on the Nano game cartridge board.

In this case, we don't need to flip the Nano board over; we simply insert it



Screen 2: the Nano-based game cartridge board design.

into the cartridge's headers. Photo 4 shows a Nano-based cartridge plugged into my console. As we see, these new cartridges are decidedly less clunky and much easier on the eye.

Once again, Joe kindly provided the files if you want to build some of these boards yourself (file **CB-mar26-nano-cartridge-v1.zip**) and we also have these in our shop.

In our previous column (February 2026), we reused existing code to create three independent test programs: one for the 7-segment displays, one for the array, and one for the switches. The purpose of these programs was simply to make sure that things still worked as expected once we'd wired everything up in our console.

Now that we've created our first game cartridge, it behooves us to reload and rerun these programs to verify that the cartridge concept works as expected.

I've been framed

Let's turn our attention to the software side. Rather than starting from scratch every time we sit down to create a new game, it makes far more sense to begin with a 'framework'. This is a sort of "master scaffolding" that does all the heavy lifting behind the scenes.

In our case, this framework includes the general-purpose code needed to drive our six 7-segment displays, the graphics primitives (draw pixel, draw line, draw rectangle) to control the 14×10 tri-colour LED matrix, and all the switch-debouncing and event-handling routines for the eight push-button switches.

These buttons are split into two groups: the action controls [red (R), yellow (Y), green (G) and blue (B)] and the direction controls [north (N), south (S), east (E) and west (W)].

The framework frees us up to build games, not infrastructure. Once the boring-but-essential 'plumbing' has been taken care of, each new game becomes a matter of writing only the fun bits: gameplay logic, animations, scoring and whatever other scamps of whimsy our imaginations can conjure.

Even better, whenever we improve the framework in the future—add a new graphics routine, speed up the display refresh, refine the switch handler routines—every game benefits. Over time, our retro console evolves, and we get to spend more of our lives doing what we wanted to do in the first place: making gob-smacking games with gusto and abandon (and aplomb, of course).

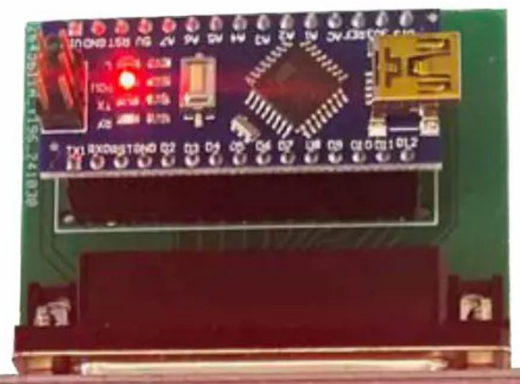


Photo 4: the Arduino Nano-based game cartridge board.

I love housework

If you look at my first framework pass (in the file named **CB-mar26-code-01.txt**), you'll see that I've gathered our three test programs together. Also, I've taken this opportunity to do a little 'housework' and tidy up some things that were niggling me.

Starting at the top of the program, I've organised things into three tidy, self-contained sections of definitions and global variables: one for the 7-segment displays, one for the 14×10 LED array and one for the push-button switches. Keeping these domains separated makes the code much easier to understand, maintain and extend.

I've also created a dedicated "pin declaration" section where all the input and output pins are defined in one place. This means that if I ever decide to reroute a signal or change a connector, I can find things in a single, well-labelled block rather than spelunking through multiple device-specific sections.

This is a simple organisational step, but it pays dividends in clarity and sanity—both of which are in short supply when you're juggling displays, LEDs, buttons and gameplay logic all at once.

I realised that we might want to use our existing UpdateScore() function to modify a variety of other metrics, so I renamed it as UpdateCounter(), which is a little more generic.

I also renamed our original LoadSR() function to DigitToDisplay() because that better aligns with our other function names.

I also deleted the original ClearSR() function we used to clear the 7-segment displays, because this was a bit of a 'one-trick' outlier. Instead, I created a new FillBuffer() function. This new routine accepts an 8-bit integer, looks up the corresponding 7-segment bit pattern in our CharSegMap[] table, and writes that same pattern into every position of our six-digit DisplayBuffer[].

Once the buffer is filled, it calls our BufferToDisplay() function to push the updated patterns to the physical display.

This means that we can now clear our display by calling FillBuffer(SPACE).

Last, but certainly not least, I deleted our original LoadSR() function because, following the aforementioned reorganisations, it ended up being called from only one place. Thus, this functionality has been subsumed by the BufferToDisplay() function.

We live in testing times

It's always reassuring when you apply power to something like our retro games console to get a quick visual 'all systems go' before the real action begins. A brief self-test gives us instant feedback that the hardware is alive and responding, saving us from chasing phantom software bugs caused by something as simple as a loose wire.

At this point, I'll freely confess that I was sorely tempted to make my power-on self-test fabulously flamboyant. For example, I could have had the 7-segment displays cycle through random numbers and then 'spin down' one by one to the number '8', slot-machine style.

Likewise, on the LED array, I could have walked vertical and horizontal bars across the display, creating something reminiscent of a retro diagnostics demo from a 1980s arcade cabinet. All very pretty… the first couple of times.

There are two reasons I decided against this 'over-the-top' approach. First, every extra animation consumes valuable memory, and we want to conserve as much flash and (especially) RAM as possible for our actual games.

Second, while flashy startup sequences are fun during development, they become teeth-grindingly annoying when you are forced to sit through them every single time you power up the console. Trust me. I've been there, done that, bought the T-shirt and got the tattoo. I still have the emotional scars.

This explains why I opted for two simple, functional power-on tests that are called at the end of the Arduino's standard setup() function.

The first routine, TestDisplayModules(), uses our new FillBuffer() function to load all six 7-segment displays with the character '8', pauses for a second, then clear them by writing spaces. This makes it trivially easy to confirm that every segment of every display is alive and kicking.

The second routine, TestArray(), calls our existing DrawRectSolid() function to set the entire 14×10 LED array to WHITE, pauses for a second, and then sets it back to BLACK. If any LED (or sub-LED) fails to fire

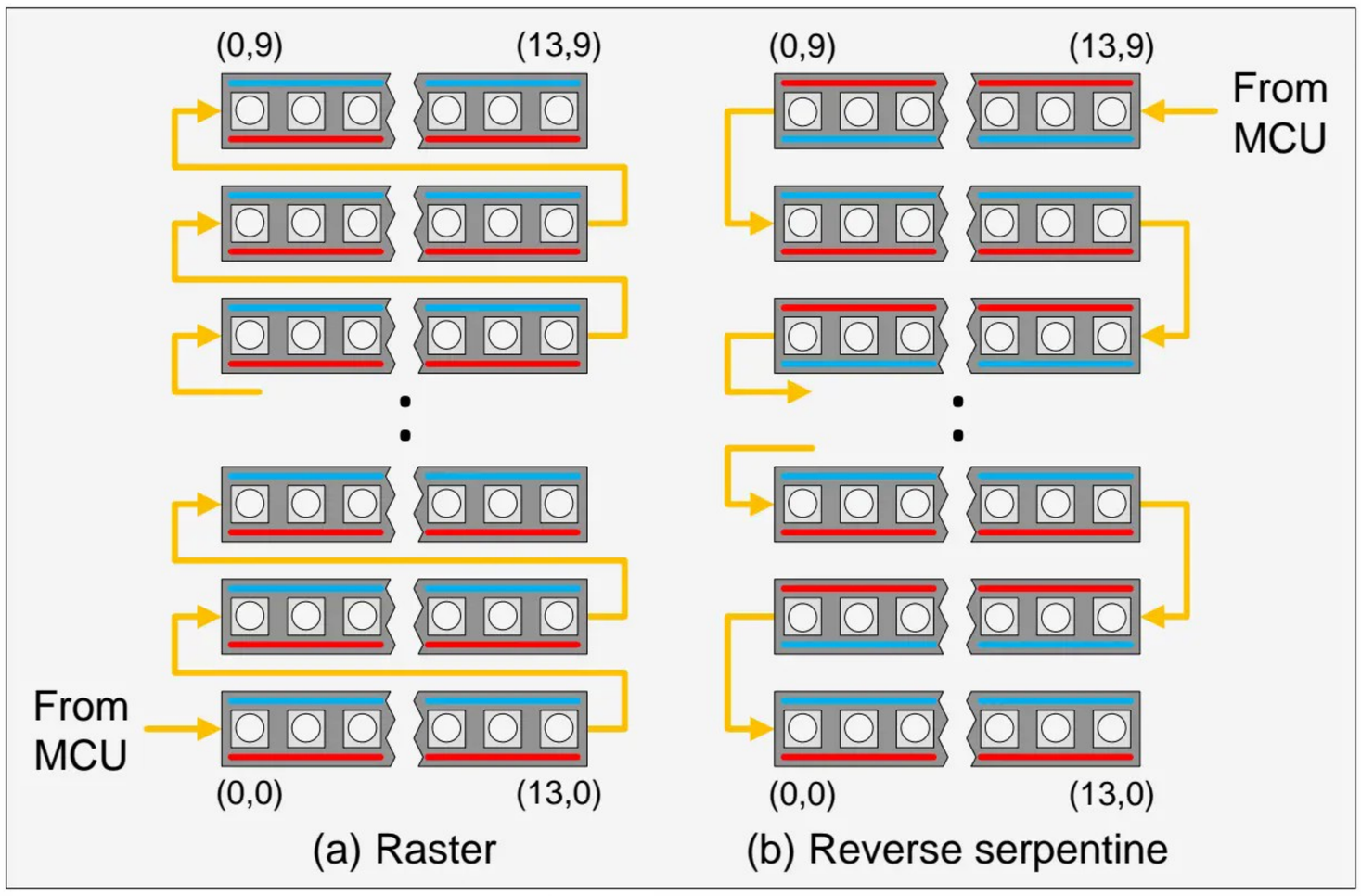


Fig.2: alternative Neopixel wiring schemes.

during this test, it will stand out like a sore thumb.

Short, sweet and sensible. No wasted memory, no unnecessary theatrics, and no irritated users (including me) drumming their fingers while waiting for the console to stop showing off and let them start to play.

Now would be a good time to load and run this first pass at our framework program and observe these two tests run.

Remember that at this stage, in addition to testing the hardware, we're also verifying that our framework works and we haven't messed anything up. As part of this, the Arduino's main **loop()** function is still cycling around, repeatedly calling our **UpdateSwitches()** and **ProcessSwitches()** functions.

In turn, our **ProcessSwitches()** function is still writing messages to the host computer's serial monitor, reporting whenever one of our push-button switches is pressed or released. This would be a good time to activate the serial monitor in the Arduino's integrated development environment (IDE) and verify that everything still works as expected.

Cut the chit-chat!

Once we are happy that our core framework is as 'sound as a pound', we need to streamline it as much as possible. You can follow along by perusing and pondering our second iteration (in the file named **CB-mar26-code-02.txt**).

My first step was to comment out the **Serial.begin()** call in the setup routine and all the **Serial.println()** calls in the **ProcessSwitches()** function.

This will speed up the gameplay. More importantly, it saves us a bunch of valuable memory bytes.

The reason for commenting these statements out rather than deleting them is that comments don't consume any space in the final machine code, and we may need one or more of them for debugging in the future.

Oh, the shame!

Now we come to the point in the story where I am once again obliged to hang my head in shame.

As you may recall, our LED array boasts 140 tricoloured 'pixels'. These are wired as a single physical string numbered from 0 to 139, but they're presented to the viewer as a rectangular matrix of 10 rows (in the Y-dimension) and 14 columns (in the X-dimension).

In my programs, I predominantly work in (x, y) coordinates, with (0, 0) at the bottom-left corner of the display, (13, 0) at the bottom-right, (0, 9) at the top-left, and (13, 9) at the top-right.

This makes it easy to think of the array as a conventional grid, even though the underlying pixels are wired as a linear strip.

It's this wiring that is the source of my chagrin. When he created his personal prototype of this console, my friend Joe opted for what I think of as a 'raster' schema, illustrated in Fig.2(a).

For reasons I no longer recall, although I'm sure they were jolly good and extremely well thought out at the time, I plumbed for what I now refer to as a 'reverse serpentine' approach, as illustrated in Fig.2(b).

As we discussed previously, we need a function we decided to call **GetNeoNum()** that accepts a pair of (**x**, **y**) values specifying the position of a pixel in the array and returns that pixel's index **n** in the physical string.

The idea was that, regardless of how we each physically organise our pixels, all we need to do is swap out our **GetNeoNum()** functions to run each other's games.

Joe did what I didn't: figure things out before he started wiring everything together.

Apart from anything else, Joe doesn't have to differentiate between odd and even rows. As a result, assuming that **NUM_COLS** has been defined as 14, the body of his **GetNeoNum()** function can be summarised as **n = (y * NUM_COLS) + x** (easy peasy lemon squeezy).

Now, let's return to my wiring scheme from Fig.2(b). My algorithm does need to differentiate between odd and even rows, and it requires some mental gymnastics to work out what's going on.

Our initial incarnation deep in the mists of time looked a little like Listing A (the lack of a listing number indicates that this code isn't derived from any of the programs in this issue).

The modulo division operation on line 127 is used to detect whether the row (**y**) we're dealing with is even or odd.

I hail from the days when C compilers were as dumb as a bag of rocks and would have implemented a statement like this as-is.

On an Arduino Uno, which doesn't have a hardware divider (meaning divisions and modulo operations must be performed using multiple simple instructions), this would consume 60-80 clock cycles.

```
122 // Return the number of the pixel in the string
123 uint8_t GetNeoNum (uint8_t x, uint8_t y)
124 {
125   uint8_t n;
126
127   if ( (y % 2) == 0 )
128   {
129     // Even Row
130     n = MAX_NEO - (y * NUM_COLS) - MAX_X + x;
131   }
132   else
133   {
134     // Odd Row
135     n = MAX_NEO - (y * NUM_COLS) - x;
136   }
137
138   return n;
139 }
```
Listing A: our original GetNeoNum() function.

Thus, our first optimisation was to replace this modulo division with a bitwise AND, as illustrated on line 509 in Listing 1(a). Remember, we're using the convention that the listing number (1 in this case) corresponds to the numerical part of the matching code file name ("01" here).

In addition to using less memory, this new approach consumes only two clock cycles if **y** is even (and the branch isn't taken) or three cycles if **y** is odd (and the branch is taken).

Sad to relate, when we compiled this new version of our routine (March 2025), we discovered that it consumed the same amount of memory (and the same number of clock cycles) as the original. What? How could this be?

Well, it turns out that modern C/C++ compilers are aware of tricks like replacing **(y % 2) == 0** with **(y & 1) == 0**, and they automatically implement these techniques without our knowing. I'm not sure whether to say "Rats!" or "Hurrah!".

And that's where we left things… until now.

Why GetNeoNum() matters

The word "furtling" is a delightfully imprecise British term meaning to fiddle, tinker, or poke about with something, typically without a grand plan, but with great enthusiasm and the faint hope that enlightenment (or at least functionality) will emerge.

Bearing this in mind, the reason for our furtling with our **GetNeoNum()** function is that it sits at the very bottom of our 'graphics stack'. Every higher-level drawing function eventually funnels through this one routine.

DrawPixel() calls it directly, while all the other helpers—**DrawLine()**, **DrawRectOutline()**, **DrawRectSolid()**, **DrawRectFilled()** and any additional routines we may add in the future—are essentially loops that call **DrawPixel()** over and over.

That means any time we touch the 14×10 NeoPixel array, we pay the cost of any inefficiencies in **GetNeoNum()**. A single full-screen fill like the **DrawRectSolid(0, 0, MAX_X, MAX_Y, WHITE)** used in our **TestArray()** function calls **GetNeoNum()** once for every one of the 140 pixels. Any future animations, scrolling, and suchlike might easily rack up hundreds or thousands of such calls.

On a little 8-bit Arduino Uno processor running at 16MHz with no cache, no pipeline and no speculative execution, even a couple of wasted instructions in a basic function like this quickly add up. That's why we want to keep **GetNeoNum()** as small, simple, and straightforward as possible.

Optimising our socks off

When we look at lines 512 and 517 in Listing 1(a), we see that we perform the same base calculation **(MAX_NEO - (y * NUM_COLS))** irrespective of whether the row is odd or even. There's no point in performing the same calculation twice, so we can rearrange things as shown in Listing B (the reason this listing doesn't have an associated number is that we aren't actually going to use it in our code).

Upon perusing the compilation reports, I discovered that doing this saved us six bytes of memory. That may not seem like a lot, but every byte counts (especially when you find you are running short).

The final version of our routine is shown in Listing 2(a). In this case, we're using the C/C++ ternary operator (**?:**). This is a compact shorthand for an **if-else** expression. It selects one of two values depending on whether a condition is true or false. The generic case is **condition ? value_if_true : value_if_false**, where 'true' is any non-zero value and 'false' equates to a value of zero.

The ternary operator does not introduce any overhead or offer any advantages in terms of memory or clock cycles. It's essentially syntactic sugar because the compiler typically generates the same machine code as an equivalent **if-else** expression. So why did we do it? To be honest, the only reason is to impress our friends and make it look like we know what we are doing!

Having said this, observe the use of the **static** and **inline** keywords on line 503. Declaring a function as static restricts its visibility to this source file only. This allows the compiler to optimise it more aggressively, since it knows the function cannot be referenced from elsewhere.

Meanwhile, the **inline** keyword provides a hint to the compiler that this function is a good candidate to be expanded in-place rather than called through a normal function call. Inlining removes the function-call overhead (pushing/popping registers, branching etc) and often results in faster, tighter assembly code.

If a function is declared **static** and is only called once or twice, even without the **inline** keyword, the compiler may decide to inline it anyway and not generate any separate function code if that's more efficient.

More meandering musings

I did ponder (and discard) a couple of other ideas. For example, consider the calculation on line 505 in Listing 2(a). One thing we could do would be to pre-calculate these values and store them in an array, along the lines of: **uint8_t BaseTable[NUM_ROWS] = {139, 125, 111, 97, 83, 69, 55, 41, 27, 13};**

This would cost us ten bytes of RAM (one byte per value) but we could change line 505 to **base = BaseTable[y]**.

The reason I didn't bother with this is that the current calculation on line 505 conceptually requires only a single 8-bit multiplication (which the AVR's hardware multiplier performs in two clock cycles) and a single 8-bit subtraction (one clock cycle), plus a few bookkeeping instructions.

I think there's a better-than-even chance that accessing the data in the array would consume at least as many,

```
504  // Return the number of the pixel in the string
505  uint8_t GetNeoNum (uint8_t x, uint8_t y)
506  {
507    uint8_t n;
508
509    if ( (y & 1) == 0 )
510    {
511      // Even Row
512      n = MAX_NEO - (y * NUM_COLS) - MAX_X + x;
513    }
514    else
515    {
516      // Odd Row
517      n = MAX_NEO - (y * NUM_COLS) - x;
518    }
519
520    return n;
521  }
```

Listing 1(a): our initial optimisation attempt.

```
504  // Return the number of the pixel in the string
505  uint8_t GetNeoNum (uint8_t x, uint8_t y)
506  {
507    uint8_t base = MAX_NEO - (y * NUM_COLS);
508    uint8_t xr;
509
510    if ( (y & 1) == 0 )
511    {
512      xr = (MAX_X - x);
513    }
514    else
515    {
516      xr = x;
517    }
518
519    return base - xr;
520  }
```

Listing B: extracting the common base calculation.

```
502 // Return the number of the pixel in the string
503 static inline uint8_t GetNeoNum (uint8_t x, uint8_t y)
504 {
505   uint8_t base = MAX_NEO - (y * NUM_COLS);
506   uint8_t xr   = (y & 1) ? x : (MAX_X - x);
507   return  base - xr;
508 }
```

Listing 2(a): the final version of GetNeoNum() (for now).

and possibly more, clock cycles (we'd have to check the assembly code generated by the compiler to make a full determination).

Having said this, this is a useful technique for us to remember for future projects involving similar concepts and more complex calculations. Remember that neither the Arduino Uno nor the Nano has a hardware divider. So, if our calculation involved a division operation, it could consume tens of bytes of flash and take on the order of 60 to 100 clock cycles.

In such a case, moving everything over into a pre-calculated table would make a lot of sense.

Furthermore, if we **were** to use a pre-calculated table, what about the ten bytes of RAM?

Well, with many microcontrollers, prefixing our BaseTable[] declaration with the const keyword (which instructs the compiler to prevent the associated value from being modified in our code) would automatically cause the table to be stored in flash, thereby freeing up RAM. Furthermore, when running below ~25MHz, there would typically be little or no timing penalty for reading from flash.

Unfortunately, with the AVR-based Uno and Nano, things are a bit more 'interesting'. Arrays and lookup tables defined with the const keyword are effectively read-only but, by default, they are still copied into RAM at start-up, so they end up consuming RAM anyway (rats!).

To keep such tables in flash and avoid burning RAM, we would need to augment their declarations with the PROGMEM keyword and use special functions to read them, which is a rabbit hole we really don't want to go down here.

As a parting thought, just to place an allegorical stake in the metaphorical sand, our current framework program (the file named **CB-mar26-code-02.txt**) consumes 3494 bytes (about 10%) of the flash program storage space available to us (the maximum is 32,256 bytes.

Meanwhile, our global variables use 95 bytes (about 4%) of the dynamic memory (RAM), leaving 1953 bytes for local variables (the maximum is 2048 bytes). To be honest, that's not bad… not bad at all.

Fabulously flashy

Before we move on to creating our first honest-to-goodness game, let's start by taking baby steps. We'll begin by using our framework code as the basis for creating a 'flashing pixel' program.

I'm envisioning our starting pixel to be at location (6, 6) for no other reason than 6 is my lucky number. This pixel will commence by flashing between black and one of our RYGB action button colours; let's say green, because that's my favourite colour. We'll consider black and green to be our 'background' and 'foreground' colours, respectively.

Let's say the on and off flash times are both 250ms (milliseconds), resulting in a total flash cycle of 0.5 seconds or a flash rate of 2Hz.

If we press any of the RYGB action buttons, the pixel will start flashing that colour. If we press any of the NSEW direction buttons, the current pixel will be forced to the background colour (black in this case), and we will start flashing the new pixel in the specified direction.

Finally, if we reach any edge of the array, we will 'wrap around' to the other side. For example, if our current location is (13, 6) on the right-hand side of the array and we press the E (east) direction button, our new flashing pixel will appear at (0, 6) on the left-hand side of the array.

Our first flash

We will take things one step at a time because it's easier to track down any problems that way (you'll find our first pass at this program in the file named **CB-mar26-code-03.txt**). We start by adding some game-specific definitions and declarations, as seen in Listing 3(a).

Next, we create the UpdateFlash() function that will make our pixel flash. We want this to have two modes: 'standard' and 'forced'. In standard mode, it turns the current pixel on and off with the appropriate flash timing. In forced mode, it will immediately turn the pixel on (to our foreground colour) and reset the flash timing. My realisation of this function is shown in Listing 3(b).

On line 408, we check whether the flash type is forced. If so, we set FlashOn to true, set the current pixel to the foreground colour, use the Neos.show() function to load the physical array, and reset the FlashThen time to the current time.

```
136 // ============ GAME-RELATED STUFF ============
137
138 // Fine cycle time
139 #define TICK             1 // Main clock tick
140 #define FLASH          250 // Flash on/off times
141
142 #define SAME             0
143 #define EAST             1
144 #define WEST            -1
145 #define NORTH            1
146 #define SOUTH           -1
147
148 #define STANDARD         0
149 #define FORCED           1
150
151 #define START_X          6
152 #define START_Y          6
153 #define START_COLOR GREEN
154
155 uint8_t  X = START_X;
156 uint8_t  Y = START_Y;
157 uint32_t ForeGndColor = START_COLOR;
158 uint32_t BackGndColor = BLACK;
159 bool     FlashOn;
160
161 uint32_t TimeThen = micros(); // Tick-related
162 uint32_t TimeNow;             // Tick-related
163 uint32_t FlashThen;           // Flash-related
```

Listing 3(a): definitions and declarations.

```
404 // ============ GAME-SPECIFIC ROUTINES ============
405
406 void UpdateFlash (bool flashType)
407 {
408   if (flashType == FORCED)
409   {
410     FlashOn = true;
411     DrawPixel(X, Y, ForeGndColor);
412     Neos.show();
413     FlashThen = millis();
414   }
415   else
416   {
417     if ( (TimeNow - FlashThen) >= FLASH)
418     {
419       FlashOn = !FlashOn;
420       if (FlashOn)
421         DrawPixel(X, Y, ForeGndColor);
422       else
423         DrawPixel(X, Y, BackGndColor);
424
425       Neos.show();
426       FlashThen = millis();
427     }
428   }
429 }
```

Listing 3(b): our UpdateFlash() function.

```
201  // Do this over and over again
202  void loop()
203  {
204    TimeNow = millis();
205
206    if ( (TimeNow - TimeThen) >= TICK)
207    {
208      UpdateFlash(STANDARD);
209      UpdateSwitches();
210      ProcessSwitches();
211
212      TimeThen = TimeNow;
213    }
214  }
```

Listing 3(c): modifying the loop() function.

```
435  void UpdateColor(uint32_t color)
436  {
437    if (ForeGndColor != color)
438    {
439      ForeGndColor = color;
440      UpdateFlash(FORCED);
441    }
442  }
```

Listing 4(a): our UpdateColor() function.

Alternatively, if this is a standard flash, on line 417 we check whether we've reached the end of the flashing pixel's on or off duration. If so, we invert the state of our FlashOn variable (if it's true, it becomes false, and vice versa), set the pixel to the appropriate on or off colour, update the physical array, and reset the FlashThen time to the current time.

We call UpdateFlash(FORCED) at the end of our setup() function to start things rolling, and we add an UpdateFlash(STANDARD) call to our main loop() function, as illustrated in Listing 3(c).

When we run our program, we see the pixel at (6, 6) happily flashing away in a cheerful green. I'm happily watching it on my desk as I pen these words, which means we're both happy.

Changing colours

Still picking the 'low-hanging fruit', let's turn our attention to changing the pixel's colour in response to pressing one of our RYGB action buttons. I bet this is going to be far easier than you might think (see the file named **CB-mar26-code-04.txt**). We start by creating an UpdateColor() function, which accepts the new colour as a parameter, as illustrated in Listing 4(a).

If the new colour differs from the existing foreground colour, we set the foreground colour to the new colour and then call our UpdateFlash() function in its forced mode.

Next, we need to modify the switch-handling code associated with each of our RYGB action switches. When any of these switches is pressed, it will enter its JUST_GONE_ACTIVE state, at which point we call our new UpdateColor() function passing in the desired colour.

This involves only a one-line addition to each of our RYGB switch routines. Consider the red switch, as illustrated in Listing 4(b), for example. The new statement appears on line 328.

```
320    // Switch Red
321    if (Switches[SW_RED].swState == JUST_GONE_ACTIVE)
322    {
323      // Mandatory
324      Switches[SW_RED].swState = WAITING_FOR_ALL_ONES;
325
326     // Optional (game-specific)
327      // Serial.println("Switch Red just went active");
328      UpdateColor(RED);
329    }
```

Listing 4(b): modifying our RYGB switch handling.

```
448  void UpdateLocation(int8_t deltaX, int8_t deltaY)
449  {
450    DrawPixel(X, Y, BackGndColor);
451    X = (X + NUM_COLS + deltaX) % NUM_COLS;
452    Y = (Y + NUM_ROWS + deltaY) % NUM_ROWS;
453    UpdateFlash(FORCED);
454  }
```

Listing 5(a): our UpdateLocation() function.

Changing location

Now let's ponder the problem of changing the pixel's location in response to pressing one of our NSEW direction buttons.

Once again, I bet this is going to be easier than you might fear (see the file named **CB-mar26-code-05.txt**). We start by creating an UpdateLocation() function, as illustrated in Listing 5(a).

This function accepts two parameters, deltaX and deltaY, which reflect whatever X-Y changes we wish to make to the flashing pixel's position. Observe that, in this case, we've declared these parameters as being of type int8_t (signed 8-bit integers) as opposed to our usual uint8_t (unsigned 8-bit integers). This is because we want to use them to represent both positive and negative values.

On line 450, we set the pixel at the current (x, y) location to the background colour. On lines 451 and 452, we update our X and Y values. Then on line 453, we call our UpdateFlash() function in its forced mode, which causes the cursor to appear in its new location.

As a reminder, we're using the % operator on lines 451 and 452 to perform 'wrap-around' (modular) arithmetic so our cursor never leaves the 14 × 10 playfield. Adding NUM_COLS or NUM_ROWS before the % ensures that the resulting value is non-negative, even if deltaX or deltaY is -1.

Next, we shall modify the switch-handling code associated with each of our NSEW direction switches. When any of these switches is pressed, it will enter its JUST_GONE_ACTIVE state, at which point we call our new UpdateLocation() function, passing in the desired direction.

Once again, this involves only a one-line addition to each of our NSEW switch routines. For example, consider the north switch, as illustrated in Listing 5(b). The new statement appears on line 256.

As you may recall from the game-related definitions shown in Listing 3(a), SAME is defined as 0, NORTH and EAST are defined as 1 (ie, +1), and SOUTH and WEST are defined as -1. These are the delta values that we end up passing into our UpdateLocation() function.

Even though this is a simple 'pseudo-game' program, it's strangely satisfying to press the RYGB and NSEW buttons and watch the flashing pixel change colour and location.

Keeping count

How many times will we press the RYGB and NSEW buttons before we get bored? I'm glad you asked. Obviously, we will need to keep count.

On my console, I have six 7-segment displays. I am going to consider the least-significant digit (0) to be on the right and the most-significant digit (5) to be on the left.

I'm going to use digits 5, 4, and 3 to keep track of the number of times I click one of the action buttons, while digits 2, 1, and 0 will be used to reflect the number of times I click one of the direction buttons. Both values will range from 000 to 999.

If you look at our latest and greatest program (in the file named **CB-mar26-code-06.txt**), you'll see that we start by adding two new definitions, MIN_COUNT and MAX_COUNT, which we set to 0 and 999, respectively. We also add two new variables, ActionCtr and DirectionCtr, and initialise them with our MIN_COUNT value.

Next, we add a FillBuffer(ZERO) call to the end of our setup() function. This loads and presents 0 values on all of our 7-segment displays.

```
248     // Switch North
249     if (Switches[SW_NORTH].swState == JUST_GONE_ACTIVE)
250     {
251       // Mandatory
252       Switches[SW_NORTH].swState = WAITING_FOR_ALL_ONES;
253
254       // Optional (game-specific)
255       // Serial.println("Switch North just went active");
256       UpdateLocation(SAME, NORTH);
257     }
```

Listing 5(b): modifying our NSEW switch handling.

All that remains is surprisingly simple. Regarding the direction buttons, we simply add three lines to our **UpdateColor()** function, which is called whenever we press one of them. This modification is shown in Listing 6(a).

On line 454, we call our **UpdateCounter()** function to increment the contents of our **ActionCtr** variable, ensuring it stays within the specified 0 to 999 range.

On line 455, we call our **NumberToBuffer()** function to copy the 3-digit value stored in the **ActionCtr** variable into our display buffer. We specify an offset of 3 digits (from digit 0), thereby ensuring that this value is copied into digits 3, 4, and 5 in the buffer. And on line 456, we call our **BufferToDisplay()** function to present the contents of the entire buffer on the display.

We perform a similar treatment in our **UpdateLocation()** function, as illustrated in Listing 6(b). In this case, however, we modify the value in the **DirectionCtr** variable. Also, when we copy this value into our display buffer, we specify an offset of 0, thereby ensuring it ends up in the buffer's 2, 1 and 0 digits.

Did we do well?

So, how well did we do with this first test case of our framework program? Well, before we started, our core framework required 531 lines of source code while consuming 3494 bytes of flash and 95 bytes of SRAM.

Our new program, with all its 'bells and whistles', is now 618 lines of source code, which means we've added only 87 lines, several of which were blank. Also, this latest incarnation consumes 4430 bytes (~13%) of flash and 126 bytes (~6%) of RAM.

I don't know about you, but I, for one, am feeling pretty good about all this. Apart from anything else, all the foundational routines we've created seem to be working well for us (touch wood).

I also think it's important to say that nothing we've done here is time wasted. The new routines we've created here—**UpdateFlash()**, **UpdateColor()** and **UpdateLocation()**—will undoubtedly come in handy (perhaps in modified form) for more ambitious programs in the future.



Next time

I was planning to start a new project next month, but we still have a lot more functionality to explore with our retro games console, not least implementing our first real 'win-or-lose' game.

I have a great one in mind, but I'd love to hear your suggestions.

Even when we do start a new project, we can still return to this Retro Games Console and add a new game cartridge now and then!

As always, if you have any thoughts you'd care to share on anything you've read here, please feel free to email me at max@clivemaxfield.com PE

```
446 void UpdateColor(uint32_t color)
447 {
448   if (ForeGndColor != color)
449   {
450     ForeGndColor = color;
451     UpdateFlash(FORCED);
452   }
453
454   ActionCtr = UpdateCounter (ActionCtr, 1, MIN_COUNT, MAX_COUNT);
455   NumberToBuffer(ActionCtr, 3, 3);
456   BufferToDisplay();
457 }
```

Listing 6(a): adding action button count functionality.

```
459 void UpdateLocation(int8_t deltaX, int8_t deltaY)
460 {
461   DrawPixel(X, Y, BackGndColor);
462   X = (X + NUM_COLS + deltaX) % NUM_COLS;
463   Y = (Y + NUM_ROWS + deltaY) % NUM_ROWS;
464   UpdateFlash(FORCED);
465
466   DirectionCtr = UpdateCounter (DirectionCtr, 1, MIN_COUNT, MAX_COUNT);
467   NumberToBuffer(DirectionCtr, 3, 0);
468   BufferToDisplay();
469 }
```

Listing 6(b): adding direction button count functionality.

Devices and components used in this issue

Arduino Uno R3 microcontroller module	https://pemag.au/link/ac2g
Arduino Nano R3 microcontroller module	https://pemag.au/link/ac9y
PCB spacer kit (black, M3)	https://pemag.au/link/ac4u

5-year collections

2019-2023
£49.95
2018-2022
£49.95
2017-2021
£49.95
2016-2020
£44.95

Purchase and download at:
www.electronpublishing.com

The dark arts of technology

Techno Talk

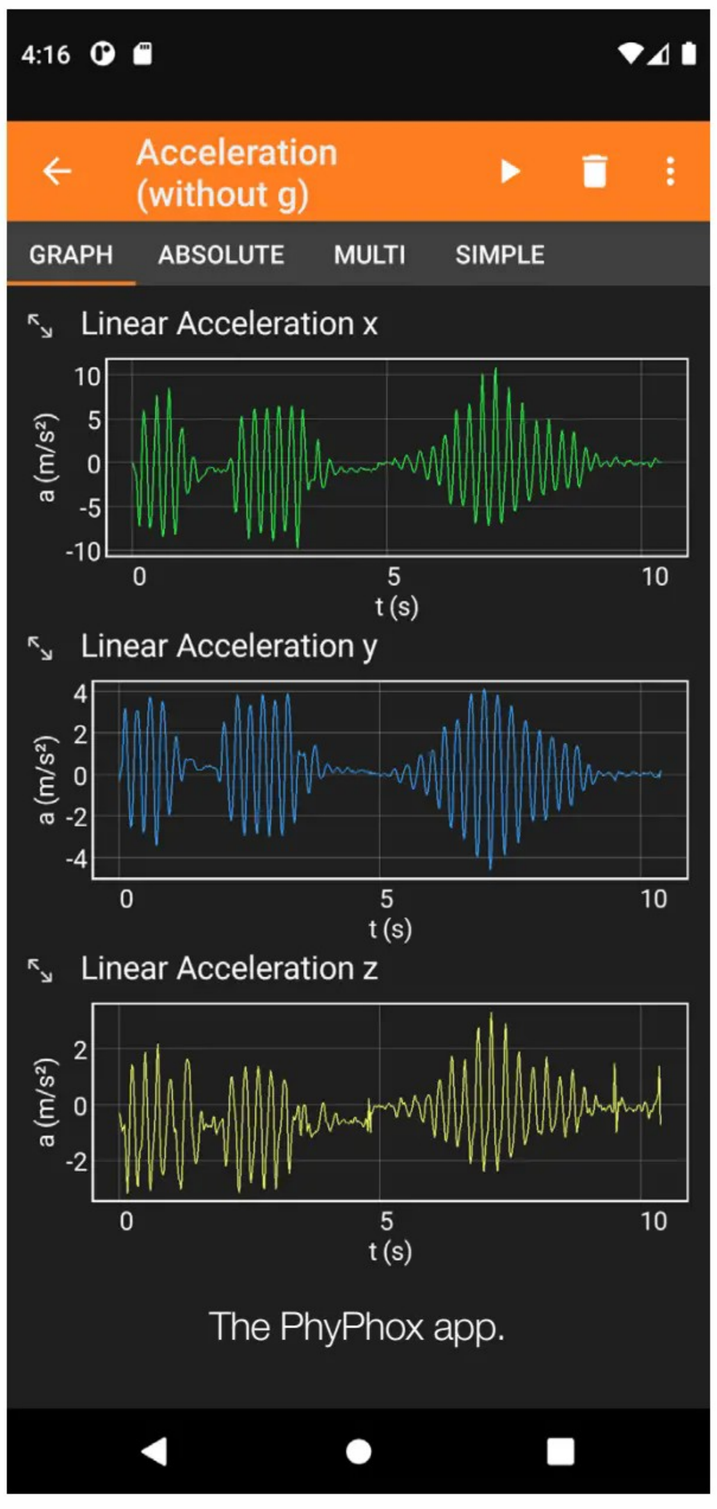
From physics experiments to smartphone sensors, reverse engineering, obfuscation and generative AI—an exploration of how abstraction explains, empowers and occasionally conceals modern technology.

Max the Magnificent

When I was in high school, in those days of yore, the sort of experiments we performed were, in hindsight, laughable in their simplicity. We're talking about trolleys rolling down inclines, falling weights and pulleys, vibrating strings and tuning forks, ripple tanks and water waves, and... need I say more?

The rudimentary sensors at our disposal were along the lines of mercury thermometers and stopwatches. Our data recorders were plain paper and graph paper. No accelerometers, no velocity plots, no real-time graphs. Just paper, pencils and patience.

The overarching reality of 1970s high school physics was that sensors were expensive and rare (a single analog meter or a monstrous CRT-based oscilloscope often served an entire class). Data was sparse, slow and hand-recorded. Experiments focused on validating theories, not exploration; students adapted to the equipment, not the other way around.



The PhyPhox app.

The PhyPhox app

Returning to the present day, smartphones boast many advanced sensors: accelerometers, gyroscopes, magnetometers, cameras, microphones, a pressure sensor, light sensors, tempeature sensors, satellite navigation... the list goes on.

Matching the sensors found in a modern smartphone would have required a fully equipped university lab in my school days, and that's assuming such sensors were available at all!

This leads us to a free app called PhyPhox (**phyphox.org**), which turns one's smartphone into a portable physics lab, using its built-in sensors to run science experiments and collect real-world data. I'm trying to imagine the expression on my physics teacher's face if I were to travel back in time and demonstrate this to him.

How low can you go?

At the lowest level of abstraction, digital computers run machine code, the most basic form of instruction they can understand. This code consists of binary numbers (strings of 0s and 1s) that tell the processor exactly what to do, such as moving data, adding two values, or jumping to another location in memory. Each instruction directly controls the hardware, with no translation needed.

In the very early days of computing, circa the late 1940s and early 1950s, programs were written in machine code that precisely matched the processor's instruction set. On machines like ENIAC, this could even mean setting physical switches or using plugboards to define how the machine was to behave.

The next level up the abstraction ladder is assembly language. This language uses short mnemonic words (like ADD, LOAD and JMP) and symbolic names for memory locations. In a sense, assembly language is machine code written with words instead of numbers.

The first assembly-language programs were written on paper and translated by hand into machine code. Later, programs called assemblers automated this process, taking assembly language as their input and producing machine code. Early assemblers were themselves written in machine code or hand-translated from assembly, after which no sane programmer wanted to go back.

It wasn't long before we climbed still higher up the abstraction ladder, in the form of languages like FORTRAN in the late 1950s and C in the early 1970s.

Moving to higher levels of abstraction shifts our focus from mechanism to intent. It reduces the cognitive load while programming, improves our ability to visualise and explore the solution space, accelerates iteration and enables portability (ie, code is no longer tied to a single machine and its instruction set). Still, this can come at the cost of some loss of low-level visibility and control.

Programs called compilers take these high-level representations as input and generate assembly or machine code as output. What most people don't realise is that compilers almost invariably generate an intermediate representation.

Initially, this was assembly language, but most modern compilers translate source code into one or more intermediate representations (IRs), perform analysis and optimisation on those IRs, then produce the final machine code.

One advantage of this is that, by separating the compiler into a front end (specific language to IR) and a back end (IR to assembly/machine code), the back end can be used for multiple different language compilers, and the front end can target multiple different machine architectures.

Foiling reverse engineering

'Reverse engineering' is the analysis of a finished system to infer its structure, function and design without access to the original specifications, drawings or code.

Whether applied to hardware or software, reverse engineering by bad actors sits at the heart of supply-chain attacks, malware, intellectual property theft, product cloning and contemporary cyber warfare.

The word "obfuscation" comes from the Latin "ob" (against, toward, in the way of, over) and "fuscāre" (to darken). So "obfuscāre" literally meant "to darken over" or "to obscure". This word entered the English language in the early 1500s.

In the context of computer programming, 'obfuscation' typically refers to deliberately making source code hard to read or understand while keeping it functionally correct.

Consider a function called IsEven() that accepts an integer (whole number) value of x and returns true if the number is even, or false if it isn't. This could be implemented as "return (x % 2) == 0", for example.

Renaming this function to something like aj53t() and changing the code to "return !((x & 1) ^ 0)" will cause people's eyes to glaze over while they try to wrap their brains around what's going on.

This leads us to the competition called The International Obfuscated C Code Contest (**ioccc.org**). Since the 1960s, people have used obfuscation to protect their source code. Techniques include deliberately awful and/or misleading variable and function names, excessive use of macros and indirection, confusing control flows and even self-modifying code (yes, really).

All this makes reverse engineering painful and debugging one's code… let's say, "character building". The irony was that, until recently, most obfuscation was relatively shallow; skilled programmers could still unravel it, but the original authors often couldn't understand their own work six months later.

As a result, source-code-level obfuscation was frequently more effective against maintenance than against piracy.

Yet another consideration is that modern compilers are ruthlessly clever. If any obfuscation that's been applied doesn't change program semantics, can be proven redundant, or produces predictable results, the optimiser will happily remove it in the name of speed, size, or clarity (well, the optimiser's idea of clarity).

From the compiler's point of view, obfuscation often looks just like bad code. In this context, most of my own code is self-obfuscating, which means I can boast a hitherto unrecognised talent (my mum will be so proud).

Things get worse because there are disassemblers that can convert machine code into human-readable assembly code, and decompilers that can accept assembly code and generate easy-to-understand pseudo source code.

As one example, consider the open-source software reverse-engineering (SRE) suite Ghidra (**ghidra.net**). This little scamp can analyse compiled executables (machine code) and reconstruct a human-readable representation of how they work.

Darkness as a defence

From ancient Greek mythology, Nyx is the goddess and personification of night. She isn't merely darkness; she represents obscurity, concealment, and what lies beyond ordinary perception. In many mythological tales, night wasn't just the absence of light; it offered protection, uncertainty, and strategic advantage. That which couldn't be clearly seen couldn't be easily challenged.

This metaphor carries surprisingly well into modern software security.

Recently, I was chatting with Dr. Nils Albartus, Embedded Security Specialist and Technical Solutions Director at Emproof (**emproof.com**). The company's flagship product, Nyx, is designed to make software behave a little like its mythological namesake: make it harder to see, harder to understand, and therefore harder to attack.

Emproof Nyx offers both passive and active protection techniques and technologies. In the case of passive protection, Nyx can read a program in machine code and obfuscate it into complex, misleading forms that confuse or break disassemblers and decompilers like Ghidra.

Nyx can also augment the code with active protections, including runtime guards that can detect breakpoints, tracing hooks and virtualised execution. If the code detects tampering, it can halt execution, revert to a safe path, or (very sneakily) continue running, but with misleading data.

Marching to a different beat

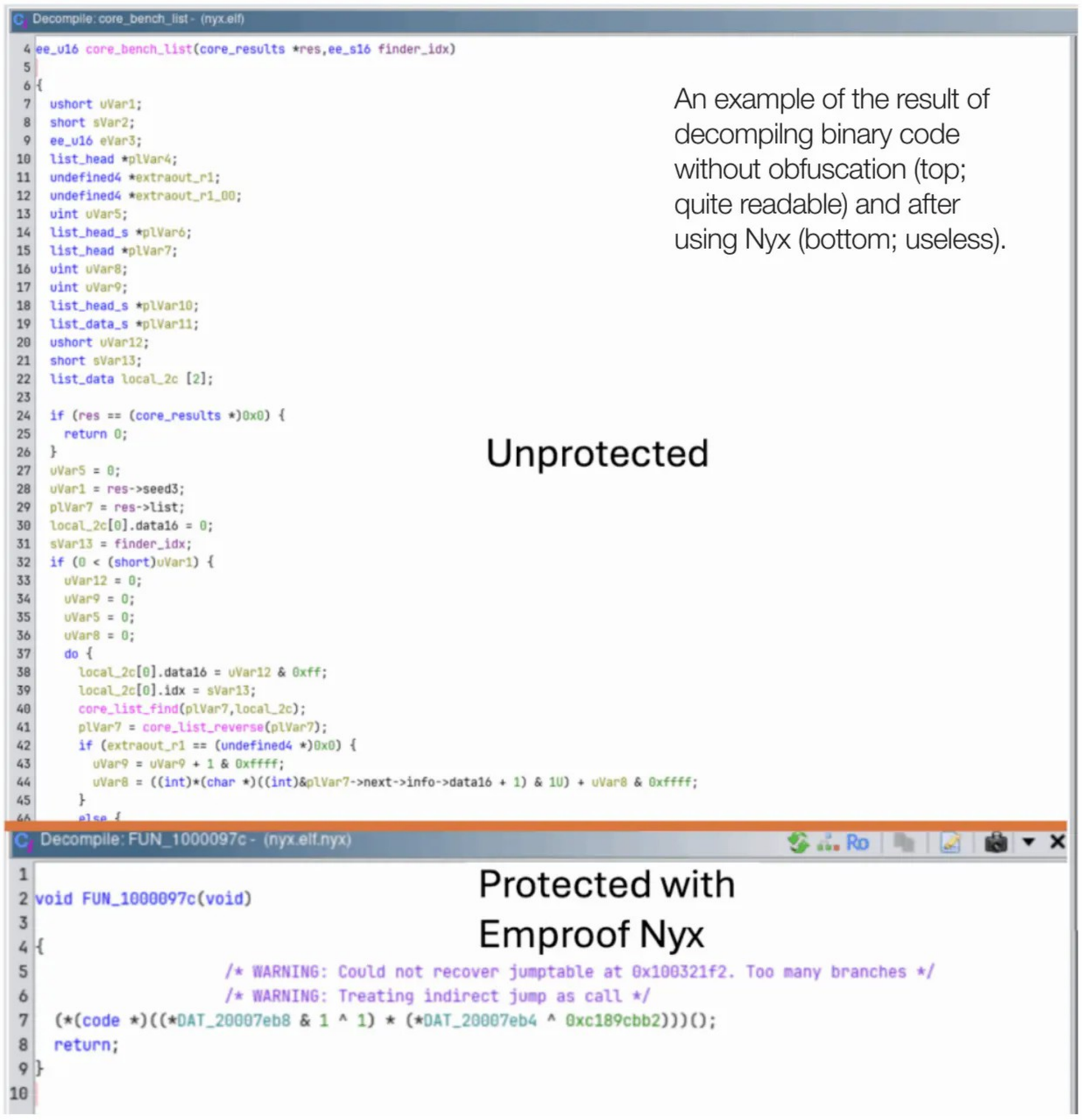
I've long been a fan of XMOS's multi-core XCORE devices (**xmos.com**). Each physical core runs multiple hardware-scheduled threads, so a single core behaves like several 'virtual' cores running in parallel. These XCORE chips offer clock-cycle-accurate determinism and reaction times measured in nanoseconds, making them ideal for real-time, timing-critical applications where predictability is king.

However, XCORE devices demand that you use a parallel programming mindset. Developers must think in terms of concurrent tasks, message passing and precise timing. These devices are technically well-suited to a far broader range of applications than they're currently used for, but the learning curve has acted as a throttle.

I was recently chatting with Mark Lippett, president and CEO of XMOS. Mark told me that the need to overcome this learning curve prompted the folks at XMOS to introduce GenSoC. This tool provides a generative-AI interface that allows developers to specify an XCORE-based design using natural language rather than low-level parallel code.

For example, a user can describe requirements like, "I want to create an audio pipeline with USB input, I²S output, sample-rate conversion and deterministic low latency", and GenSoC will generate a complete, working XCORE-based SoC design in seconds.

And so we find ourselves armed with pocket laboratories, AI design assistants, and tools that can turn darkness into clarity (or vice versa). I don't know about you, but I'm almost scared to think what we'll discover next. PE



An example of the result of decompilng binary code without obfuscation (top; quite readable) and after using Nyx (bottom; useless).

This month we write about an important piece of mostly invisible internet infrastructure: data centres. They are rarely seen and little is known of them by the general public.



By Dr David Maddison

DATA CENTRES, SERVERS & CLOUD COMPUTING

Every time you use a search engine, watch an online video, use an email service, use social media, read or write blogs, buy products online, use an AI system, or even read *Practical Electronics* or most other magazines or newspapers online, you are almost certainly using a data centre.

Data centres contain large numbers of computer servers where information is received, stored, managed, processed and disseminated. A server is a computer on which software runs remotely, to 'serve' other computers called 'clients' over a network. Such software applications include web servers, email servers, databases, custom servers and more.

Small companies might start with their own central computer system with an in-house server to store and process their data. As they grow, it might become more economical to move these services to offsite data centres, especially for companies with multiple locations. Companies can:

- pay a data centre to host their own hardware
- rent hardware from a third party but manage the software themselves
- have their off-site hardware and software managed entirely by a third-party or multiple parties

More and more these days, individuals also pay companies to manage off-site services and data for them, often referring to those services as being in 'the cloud'.

For example, you might be a Google customer and use Google Docs, Gmail, Google Drive etc; or an Apple customer using iCloud, Apple Mail etc; or a Microsoft customer using OneDrive, Office 365 etc.

Those services may use local apps (or run in a web browser) but most of the 'heavy lifting' is done in servers located in data centres. In most cases, those servers are distributed around the world, so there will always be a local server for fast access (and also so that the entire service doesn't go down due to one network outage).

In some cases (or in certain areas), it is also necessary to store data locally to comply with local laws.

Cloud services providers can be huge; they might contain tens of thousands of servers, or even millions, as they service numerous companies (and individuals) from all over the world.

The origins of data centres

Early computers were room-sized, used large amounts of power and needed a specialised environment with air conditioning, raised floors for cables, provision of a large power system and a building capable of taking the weight of the computer. Such computers were known as "mainframes" (see Fig.1). They were typically accessed via a 'dumb terminal', as shown in Fig.2.

That was the case from the late 1940s through to the 1970s. Only large businesses, government organisations and scientific establishments could afford such computers. Due to the cost, computing was often done through 'time-sharing' arrangements, where many users accessed a portion of the power of one large computer through a terminal at their desk or some common location.

In the 1970s, the microcomputer was invented, and it was popularised in the 1980s. Software could then be run by individuals from their personal

Data Centres



Fig.1: the NASA Mission Control computer room in 1962 which used two IBM 7094-11 computers. *Source:* https://archive.org/details/S66-15331



Fig.2: a typical way to interact with a computer in the early 1960s was via a printing Teletype, such as this ASR-33 model, introduced in 1963. *Source:* https://w.wiki/B5fn

computer (PC), which is also where data was stored. Software was developed that did not need specialised training to use (it was more 'user-friendly').

Unfortunately, having a computer on every desk led to other problems, such as organisations losing control of their IT resources. This created an incentive to again centralise computing resources.

Some larger government and corporate entities still maintained special rooms with traditional mainframe computers where critical data was stored, even with the rollout and acceptance of microcomputers. Still, by and large, desktop PCs were widely used throughout the 1980s and 1990s until the internet started to expand rapidly.

The expansion of the internet and the resulting vast requirement for data storage and e-commerce created a need for centralised computing and data storage. This coincided with the so-called 'dot.com bubble', from about 1995 to 2000, with large investments in IT-related companies.

Central data storage was expensive, and eCommerce companies needed a fast internet connection, which at the time was costly. There was also the need for backup power for the computers and dedicated staff to maintain the systems.

It thus became preferable for organisations to subcontract their data storage and computing requirements to an external organisation, such as a data centre, where economies of scale helped to minimise costs.

In a way, the modern data centre represents a return to the earliest days of computing via centralised systems with dedicated staff.

What is a data centre?

A data centre is a dedicated facility that houses computers, storage media, networking, power & telecommunications infrastructure, cooling systems, fire suppression systems, security systems, staff facilities and anything else required to run networked computers.

What is "the cloud"?

This expression is often used in reference to computers running in data centres. 'The cloud' represents the availability of computing resources to an end user anywhere that an internet connection exists. That generally implies that the resources are located in one or more data centres.

While cloud resources could be hosted in one central location, more likely, they will be distributed over a range of locations for redundancy, to reduce bandwidth requirements over long-distance connections and to reduce latency (access time).

Most commonly, a 'cloud' service is a type of Software as a Service (SaaS), as per the following section on delivery models. That means that both the cloud hardware and software (including the operating system, applications etc) are managed by a third party. All the customer has to do is access it. It is a somewhat nebulous concept (like a cloud!).

Clouds may be public, such as many of the services operated by Microsoft, Google and Apple, or 'private', where only specific customers with contracts can access them. Hybrid clouds contain a mix of public and private data and/or services.

Service delivery models

A data centre or cloud can be managed in various ways, as shown in Fig.3. It can either be completely in-house, or with infrastructure as a service (IaaS), platform as a service (PaaS) or software (applications) as a service (SaaS) representing reducing levels of customer management and increasing levels of data centre or cloud provider management.

As an example, the Silicon Chip website (and some of our other software) use the IaaS model. They chose this model to retain maximum control over our systems, without having to worry about provisioning high-speed internet, backup power, cooling etc. It also saves money because they only need a fraction of the power of a computer, so they can share hardware with others to split the costs.

Tenancy refers to the sharing of resources. Multi-tenancy is popular on public cloud services, such as Microsoft Azure. In this case, an

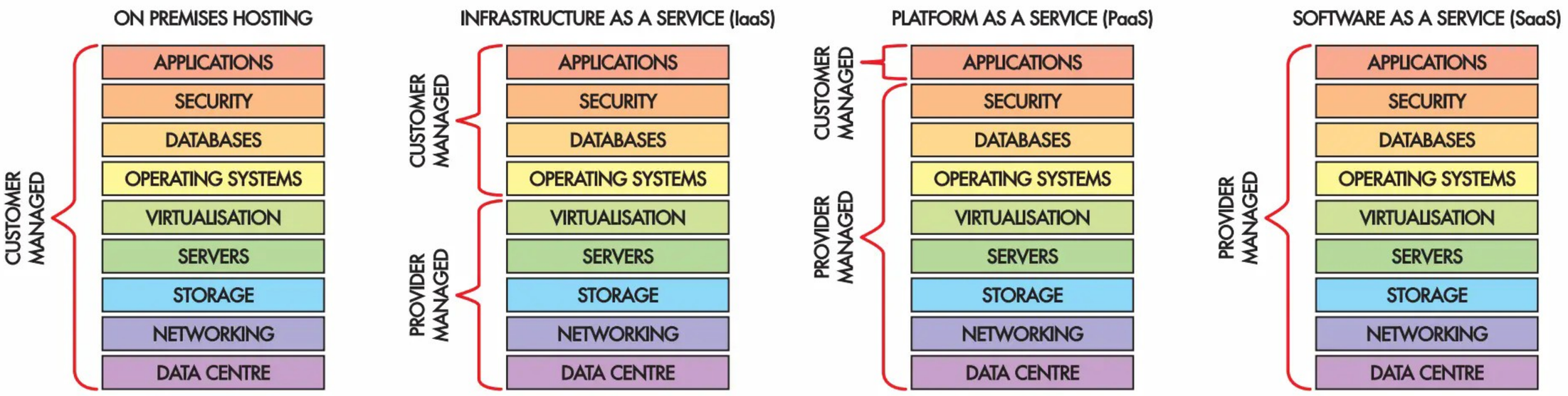


Fig.3: four different data centre service delivery models (related to the concept of tenancy).
***Original source:* https://w.wiki/B5fq**

individual customer's data remains invisible and inaccessible to others, but they share hardware, networking, other infrastructure, databases and memory. In that case, there are limited possibilities for the customisation of application software.

Examples of multi-tenancy software providers include Google Apps, Salesforce, Dropbox, Mailchimp, HubSpot, DocuSign and Zendesk.

With single-tenancy, there is no sharing of resources, which means maximum control over the software – see Fig.4.

Virtual machines and servers

A virtual machine or virtual server is an emulated version of a physical computer running within a physical computer.

To put it another way, from the customer's perspective, they have access to an entire computer, with which they can do whatever they like. But it doesn't exist as a physical computer; instead, it is software running on a physical computer, alongside many other customers' virtual machines.

Businesses can create their own virtual server, which can run software and operating systems, store data, perform networking functions and do other computing functions as though it was a real physical computer.

This virtual server runs under a software layer known as a 'hypervisor', which manages the memory, CPU, storage, networking and other physical resources of the physical computer and allocates them to the virtual machines as required.

Why use a data centre?

We touched on this earlier when we explained why we use IaaS, but there are other reasons, including:

- lower costs (due to economies of scale)
- lower latency and faster transfer speeds
- hardware maintenance performed by third parties with access to experts and parts
- multi-tenancy allows costs and resources to be shared among a large pool of users
- data centres typically have a lot of redundancy, making them resistant to power outages and natural or human-induced disasters

Why use the cloud?

These reasons include those for using a data centre, plus:

- device independence; applications can be typically via a web browser, so will work from any operating system, including mobile devices
- software maintenance, including updates, performed by expert third parties
- performance monitoring and security by expert third parties
- scalability and elasticity so resources can be increased as required

How many data centres exist?

There are currently around 523 data centres in the United Kingdom and approximately 139 in Ireland. Europe has approximately 3346 data centres spread among 44 countries.

Worldwide, there are approximately 11,000 data centres, with the United States of America having the most at 5387.

Data centre infrastructure

Data centres have major network infrastructure to connect the data centre to the outside world with plenty of bandwidth.

The internal network is also handy for transferring data between multiple computers operated by the same customer (and sometimes even different customers, eg, web crawlers for search engines).

There is also significant storage infrastructure for storing data and software; it may be integrated with the computing nodes, or separate and accessed through internal high-speed networking.

Of course, there are plenty of computing resources for data processing with onboard memory, with connections to data and applications storage, plus internet infrastructure. These are supported by cooling systems, power supplies and fire suppression systems. The work of a data centre is done in various forms of processing units:

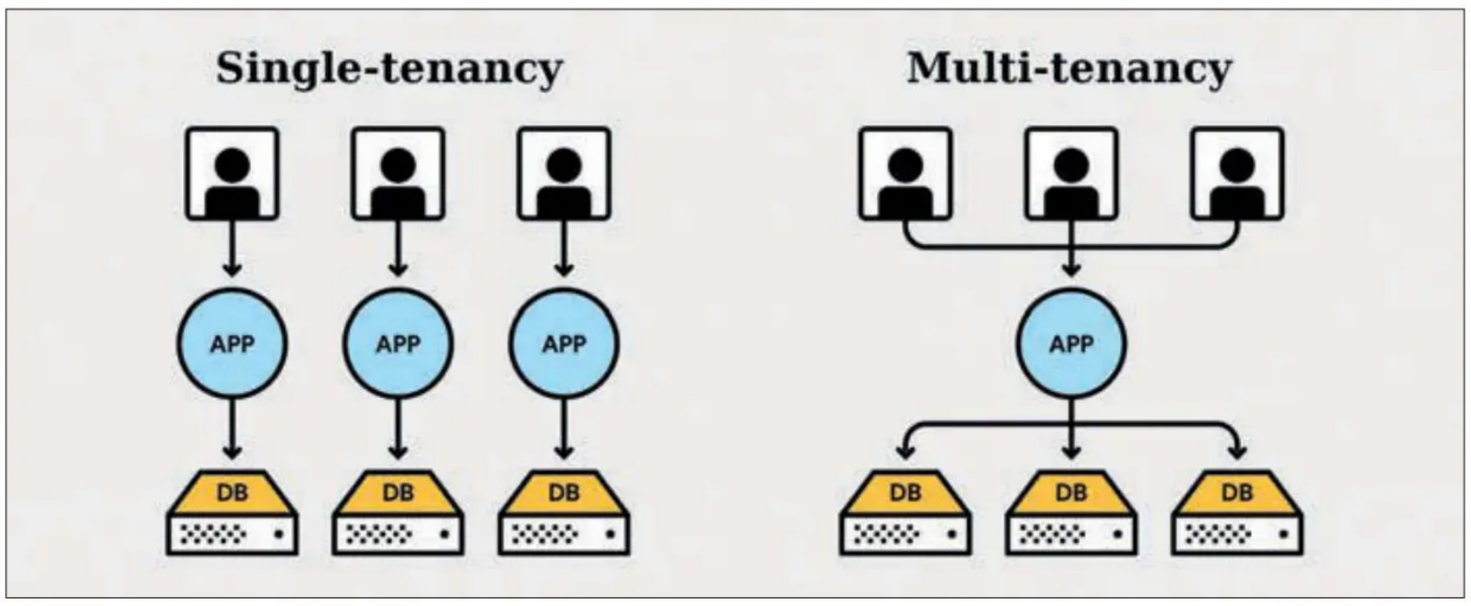


Fig.4: the single tenancy vs multi-tenancy models for data centres. DB is short for database. *Original source:* https://resources.igloosoftware.com/blog/multi-tenancy-database

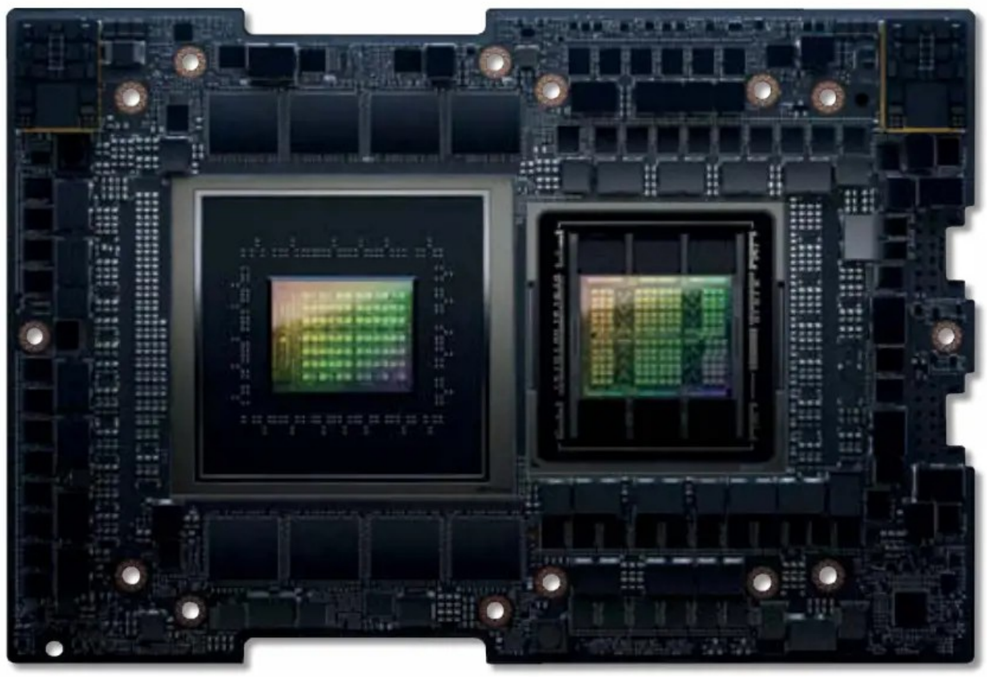


Fig.5: the NVIDIA GH200 Grace Hopper platform, based on the Grace Hopper Superchip. This board is capable of four petaflops (4×10^{15} floating point operations per second) and includes 72 ARM CPUs, 96GB of HBM3 memory for the CPUs plus 576GB for the GPUs. *Source: TechSpot* – *https://pemag.au/link/ac19*

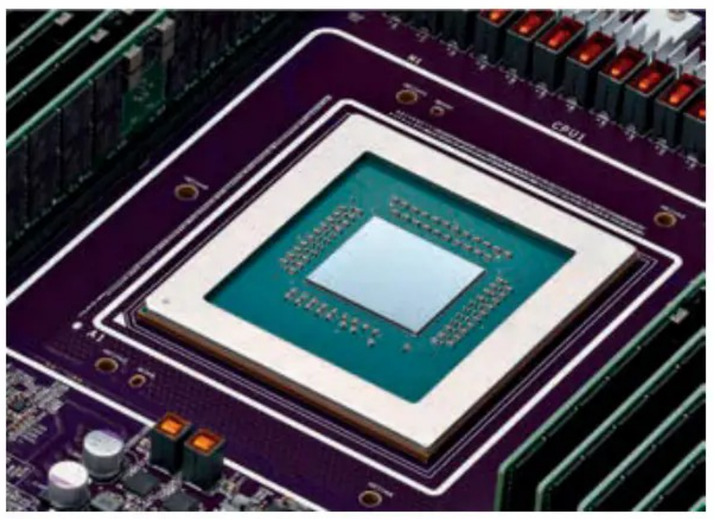


Fig.6: Google's v5p TPU chip. *Source: https://thetechportal.com/2024/04/09/google-ai-chip*

CPUs (central processing units)

CPUs are at the heart of traditional computers and generally continue to be, including in data centres. They may be supplemented by GPUs, TPUs and DPUs (each described below) to improve performance or provide new capabilities.

An example of a CPU designed for data centres is the fourth-generation AMD EPYC based on the x86 architecture, as used in most PCs and servers (Fig.7). It is designed to be energy efficient, secure and give high performance. Each of these processors may include up to 128 Zen 4 or Zen 4c cores, allowing each server to potentially handle thousands of requests at any time.

GPUs (graphics processing units)

GPUs are special processors to accelerate the rendering of images, including 3D scenes. They are also capable of image processing.

While they were originally designed for graphics applications, they are highly suitable for non-graphics applications such as parallel processing, accelerated computing and neural networks as needed in machine learning and artificial intelligence (AI). As such, they are commonly found in AI systems.

The term 'accelerated computing' refers to using specialised hardware such as GPUs to more efficiently performing complex computing tasks than traditional CPUs can.

An example of a GPU used in accelerated computing and AI data centres is the NVIDIA Grace Hopper Superchip processor, which forms part of the GH200 Grace Hopper platform (Fig.5). It is specifically designed for accelerated computing and generative AI, primarily in data centres. It utilises the latest HBM3e high bandwidth memory technology that provides 10TB/sec of memory bandwidth.

TPUs (tensor processing units)

TPUs are proprietary ASICs (application specific integrated circuits) by Google, optimised for neural network machine learning and artificial intelligence. Various versions have been produced since 2015. They are designed for high computational throughput at low precision, handling numbers with as few as eight bits.

The chips (see Fig.6) are designed specifically for Google's TensorFlow framework for machine learning and artificial intelligence, and are incorporated into 'packages', as shown in Fig.8.

A notable application was Google's use of TPUs to find and process all the text in the pictures of Google's Street View database in under five days. Google has developed what they call the Cloud TPU v5p AI Hypercomputer (Fig.9).

DPUs (data processing units)

DPUs, also called infrastructure processing units (IPUs) or SmartNICs (NIC stands for network interface controller) are used to optimise data centre workloads and to manage networking, security and storage. They relieve system CPUs of these workloads.

An example is the SolidNET DPU, an ARM-based software-defined DPU with a PCIe half-height-half-length (HHHL) format. It is based on an off-the-shelf 16-core NXP LX2161A System on Card (SOC) and uses open standards (see Fig.10). For more information, see **https://pemag.au/link/ac0b**

Power supply

A typical data centre power system includes:

- transformer(s) to reduce the utility voltage, if necessary
- automatic switching gear to switch to backup power sources such as a generator in the event of a utility failure

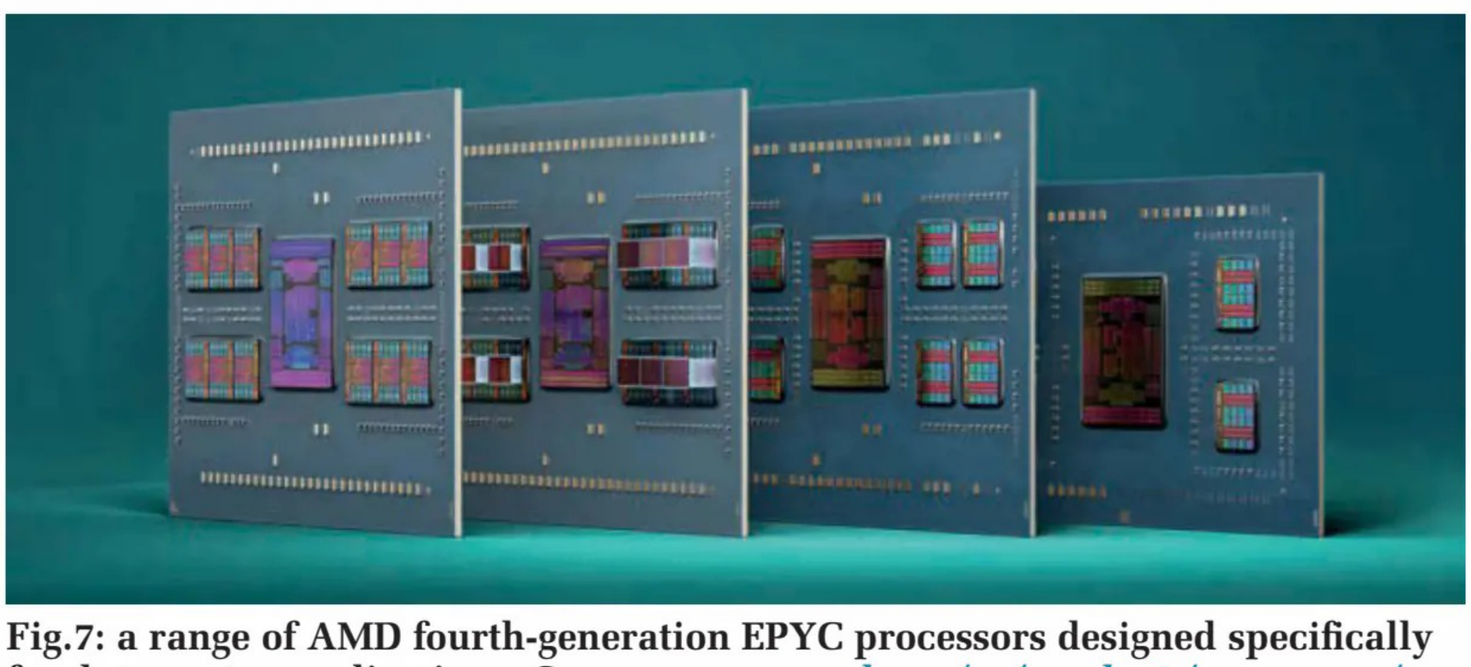


Fig.7: a range of AMD fourth-generation EPYC processors designed specifically for data centre applications. *Source: www.amd.com/en/products/processors/server/epyc/4th-generation-9004-and-8004-series.html*

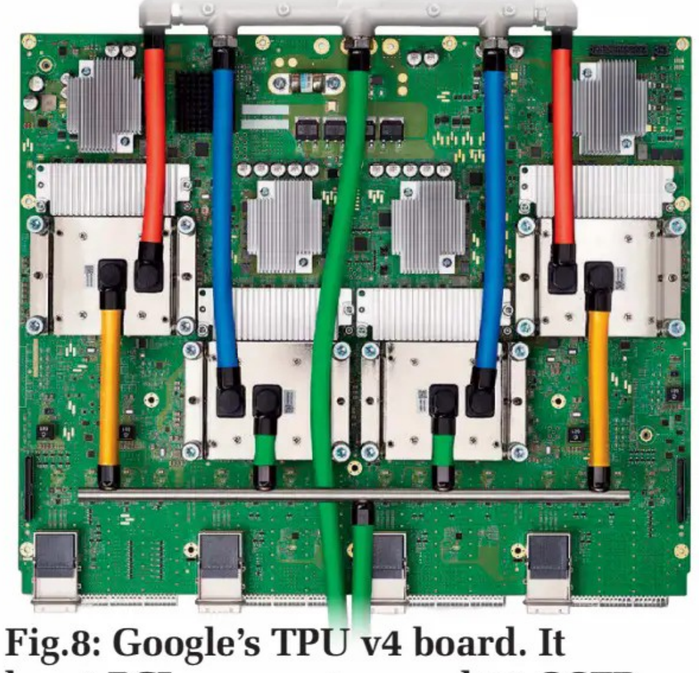


Fig.8: Google's TPU v4 board. It has 4 PCIe connectors and 16 OSFP connectors. *Source: https://w.wiki/B5fr*

- a UPS (uninterruptible power supply) supplied by a battery bank to provide backup power in the event of a utility failure, until the generator starts, as well as to condition power and remove voltage spikes in normal operation
- power distribution units (PDU), an electrical board to distribute power from the UPS to equipment locations
- a remote power panel (RPP), an electrical sub-board to distribute power from the PDU to individual rack-mounted power distribution units (rPDU)

rPDUs are much like power boards. Individual servers or other equipment are plugged into them. Some of these components may be absent, depending on the size and sophistication of the data centre. All of the above has cables, wiring, circuit breaker boards etc.

Some data centres use flywheel energy storage rather than a battery-based UPS (see https://pemag.au/link/ac1b). They can be slightly more costly to install, but they don't degrade over time as much as batteries do.

Power consumption

Data centres, especially AI data centres, use an enormous amount of electrical power. That's both to power the computers themselves, particularly their CPUs, GPUs and TPUs, as well as their cooling systems. So it is important that these be designed to be as efficient as possible to minimise power consumption.

Data centres need access to inexpensive, reliable 24/7 power supplies.

They consume a significant amount of the world's electrical power; one estimate is 1%-1.5% (https://pemag.au/link/ac0i). According to another estimate (https://pemag.au/link/ac0j), AI currently uses 8% of the world's electrical energy.

The IEA predicts that data centres will consume 6% of electrical power in the United States by 2026, and 32% in Ireland by 2026, up from 17% in 2022 (https://pemag.au/link/ac0k).

A typical 'hyperscale' data centre consumes up to 100MW according to Oper8 Global (the largest is up to 960MW). But that is just internal consumption. Given a power usage effectiveness (PUE) of 1.3, 130MW will need to be provided from the grid.

At a time when dispatchable (on demand) power capacity is diminishing in many countries and being replaced with intermittent solar and wind production, plus the energy demand for charging electric vehicles, it is not clear where all this power will come from.

The shortage of power has been recognised. According to the CBRE Group (https://pemag.au/link/ac0l):

A worldwide shortage of available power is inhibiting growth of the global data center market. Sourcing enough power is a top priority of data center operators across North America, Europe, Latin America and Asia-Pacific. Certain secondary markets with robust power supplies stand to attract more data center operators.

Data centres are being set up in New Zealand with access to 200MW of relatively inexpensive hydroelectric, gas and geothermal energy, from which 79% of New Zealand's total production is derived (https://pemag.au/link/ac0m).

In the United States, Equinix, a data centre provider, signed a 20-year non-binding agreement with Oklo to purchase up to 500MW of nuclear power (https://pemag.au/link/ac0n).

Microsoft is proposing to use nuclear power for its data centres (see https://pemag.au/link/ac0o), as is Google (https://pemag.au/link/ac0p). Amazon purchased a nuclear-powered data centre in Salem Township, Pennsylvania, USA (https://pemag.au/link/ac0q). It consumes an almost unbelievable 960MW of electrical power.

According to Funds Europe, the rapid growth of data centres is putting an unsustainable strain on the European electrical grid (https://pemag.au/link/ac0r). They already use 2.7% of their power, expected to increase to 3.2% by 2030. It has been suggested they use small modular reactors (SMR) and micro modular reactors (MMR) to power data centres. There is a growing interest in using nuclear power for AI data centres: https://pemag.au/link/ac0j

Cooling

One of the most critical aspects of a data centre, apart from the computing resources, is the provision of cooling. This is because the vast majority of the enormous amount of power used by data centres ultimately gets converted into heat.

Data centres are cooled by air conditioning the rooms the computers are in, and also possibly some type of liquid cooling of the servers themselves.

A data centre can be designed with hot and cold aisles between server racks to help maximise the efficiency of the cooling system. Cold air may be delivered from beneath perforated floor tiles and into the server racks

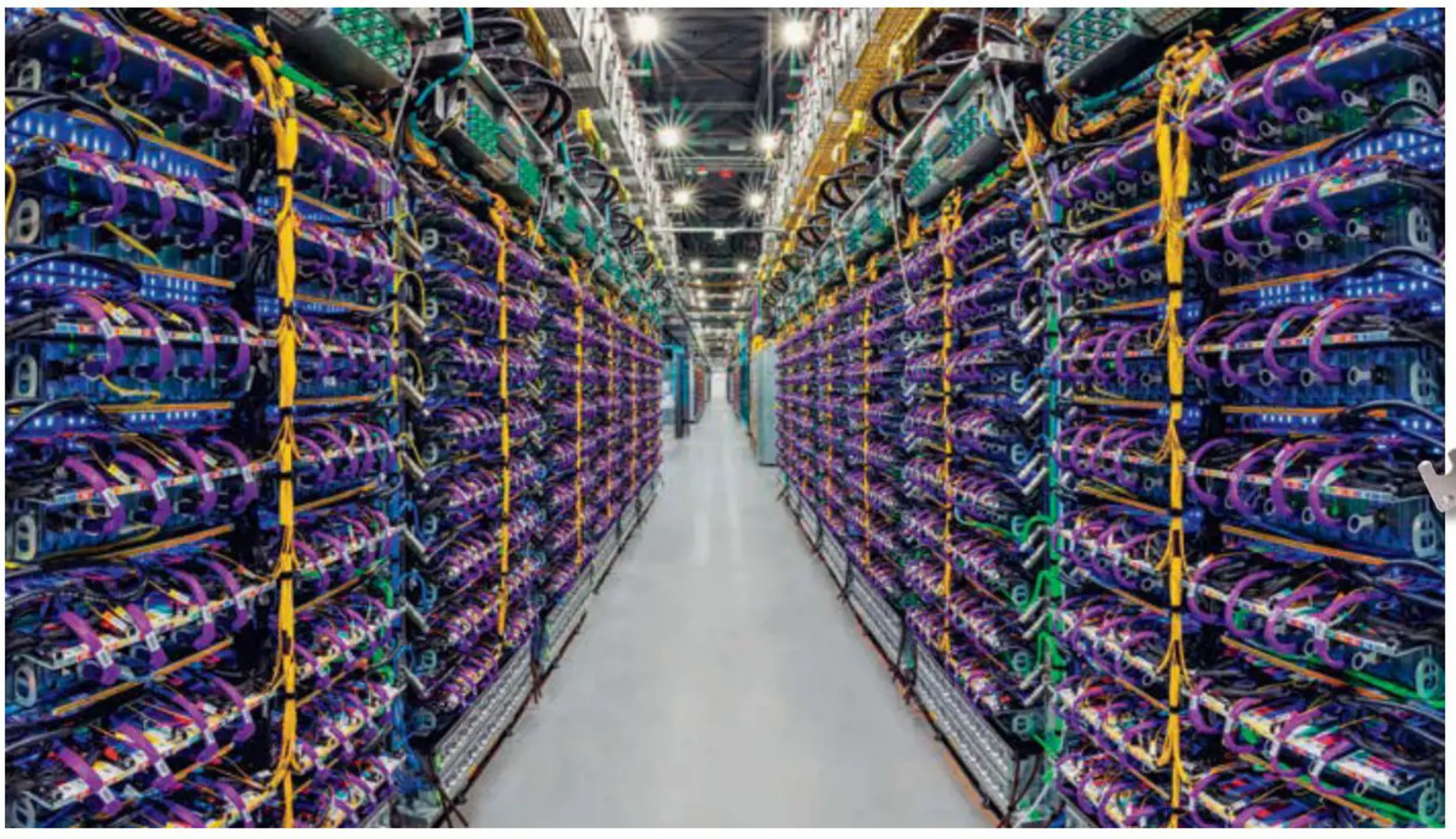


Fig.9: inside part of Google's 'hypercomputer' based on v5p TPUs arranged into 'pods'. Each pod contains 8960 v5p TPUs. *Source: Axios* – *https://pemag.au/link/ac1a*



Fig.10: a SolidRun SolidNET Software-Defined DPU (data processing unit). *Source: www.storagereview.com/news/solidrun-solidnet-software-defined-dpu-for-the-edge-unveiled*

before being discharged into the hot aisles (see Fig.12). Alternatively, hot air may be collected at the top of the server racks rather than being blown into an aisle.

Some data centres are using emerging technologies such as immersing the computer equipment in a fluid to efficiently remove heat (Fig.13). In two-phase cooling, a volatile cooling liquid boils and condenses on a coil which is connected to a heat exchanger to remover heat, after which it drips down into the coolant pool.

Silicon Chip magazine published an article in the November 2018 issue on the DownUnder GeoSolutions supercomputer in Perth that was immersed in an oil bath for cooling at **https://siliconchip.au/Article/11300**

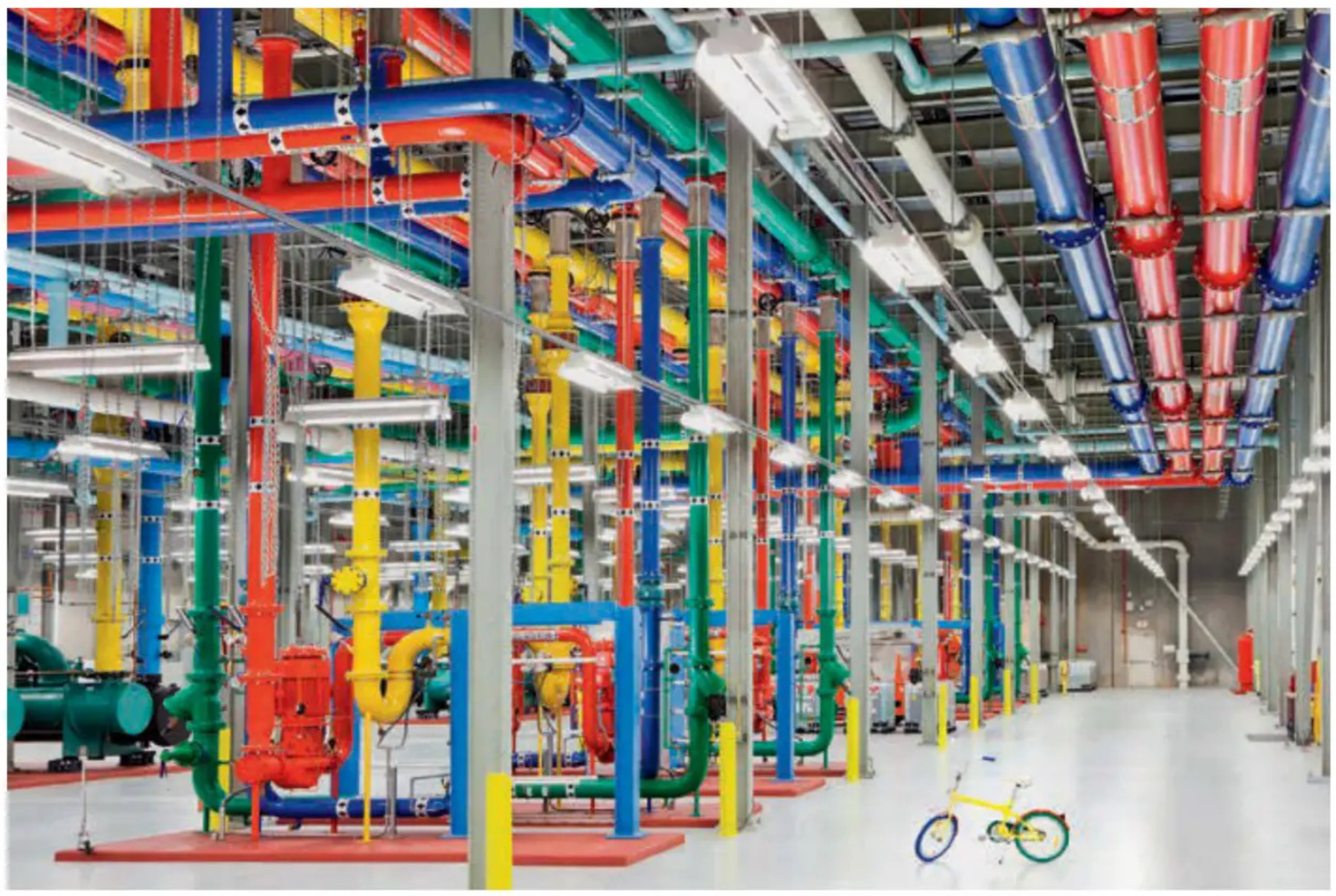


Fig.11: part of the elaborate plumbing for the cooling system for the Google data centre in Douglas County, Georgia. *Source: www.google.com/about/datacenters/gallery*

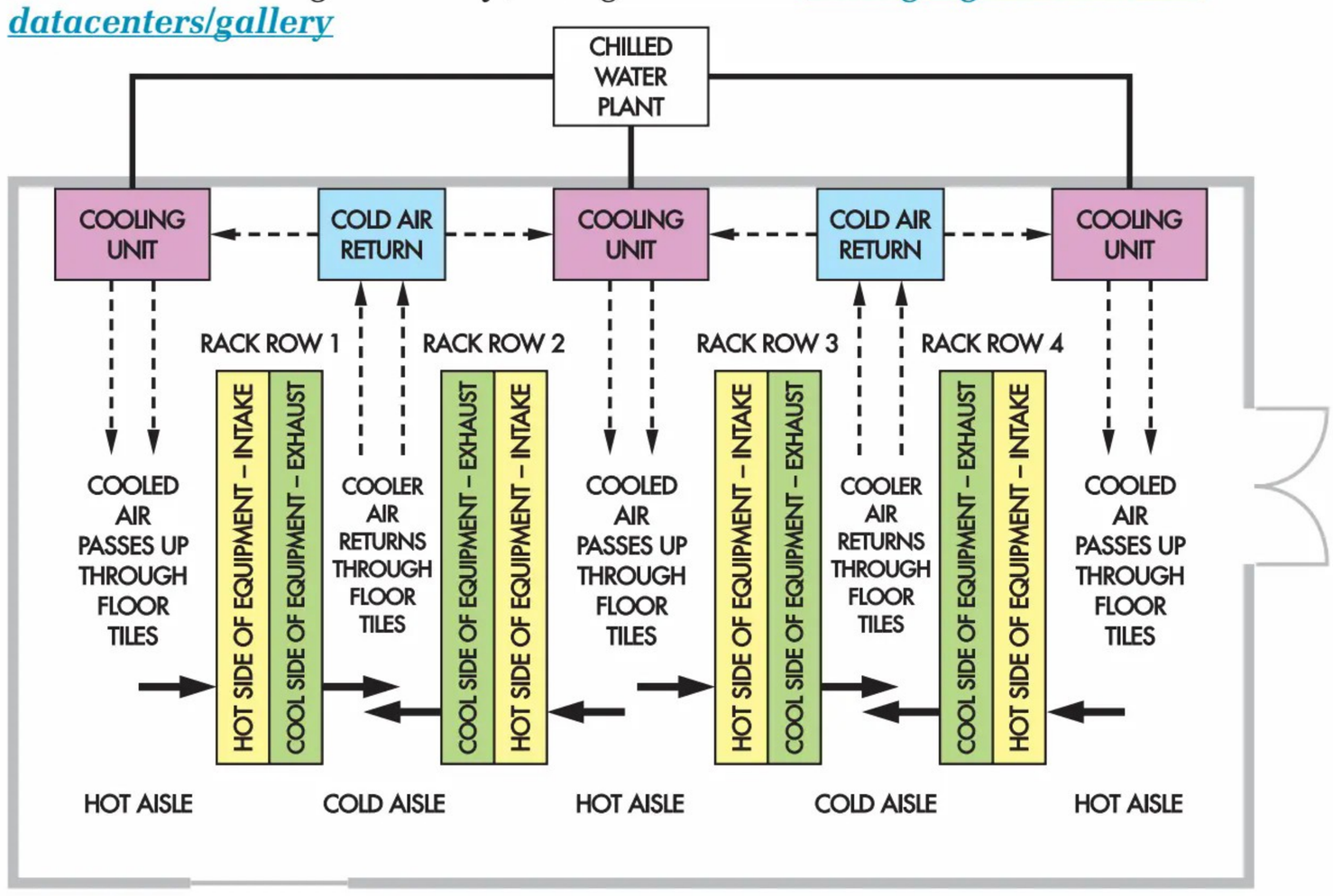


Fig.12: one possible configuration of a data centre using the concept of hot and cold aisles between rows of servers. *Original source: www.techtarget.com/searchdatacenter/How-to-design-and-build-a-data-center*

Water usage

Some data centres, especially those used for AI, consume water for cooling and hydroelectric generation as well. One would think that cooling a data centre would mostly involve a closed loop system, like a typical car. But apparently that is not always the case, as many data centres use large amounts of water. Nature magazine states:

...in July 2022, the month before OpenAI finished training the model, the cluster used about 6% of the district's water. As Google and Microsoft prepared their Bard and Bing large language models, both had major spikes in water use — increases of 20% and 34%, respectively, in one year, according to the companies' environmental reports... demand for water for AI could be half that of the United Kingdom by 2027 – https://doi.org/10.1038/d41586-024-00478-x

Details of Microsoft's water consumption for AI is at **https://pemag.au/link/ac0u**

About ⅔ of the water used by Amazon data centres evaporates; the rest is used for irrigation (**https://pemag.au/link/ac0v**). That source also states that the amount of water to be consumed by a proposed Google data centre is regarded as a trade secret!

Fire detection and suppression

Due to the very high electrical power density inside a data centre, if a fire breaks out, it could get serious very quickly. Fire detection systems need to give early warning to prevent major damage, and fire extinguishing systems

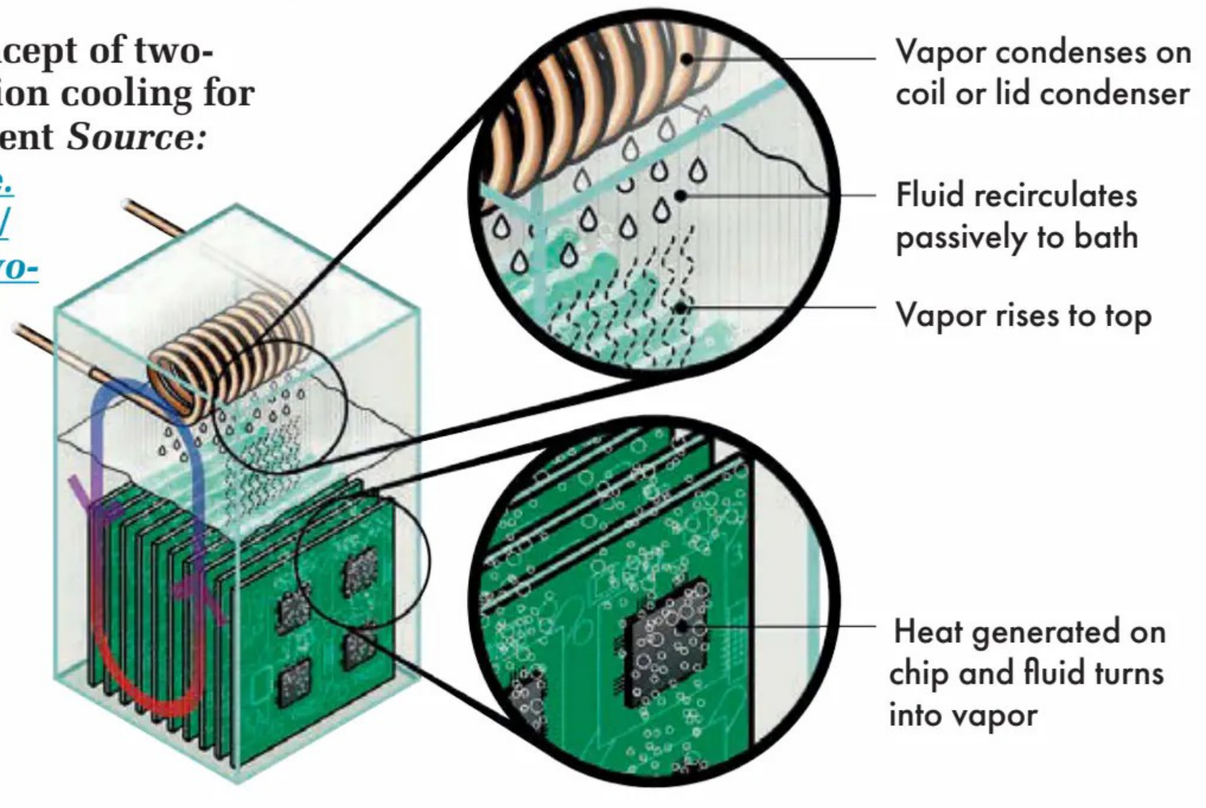


Fig.13: the concept of two-phase immersion cooling for server equipment *Source: www.gigabyte.com/Solutions/liquidstack-two-phase*



Fig.14: a comparison of the VESDA early warning smoke detection to conventional fire detection systems. *Source:* *https://xtralis.com/product_subcategory/2/VESDA-Aspirating-Smoke-Detection*



Fig.15: an artist's impression of the Victaulic Vortex fire suppression system in operation, discharging a water and nitrogen fog. *Source:* *https://youtu.be/qmhO7E4c0tM*



Fig.16: the entry lobby of a Google data centre uses a Circlelock door and retinal scan, emphasising the high security requirements of data centres. *Source:* *www.google.com/about/datacenters/gallery*

need to cause minimal damage to electrical equipment.

VESDA (Very Early Smoke Detection Apparatus) is a highly sensitive smoke detector (Fig.14), at least 1000 times more sensitive than a typical smoke alarm. It sucks air through perforated pipes that are routed around a protected area, then analyses the sample for the presence of smoke with sensitive detectors. It is an Australian invention in use in many data centres for the early detection of fires.

Victaulic Vortex is a fire suppression system used in many data centres (Fig.15). It is a combined water and nitrogen fire extinguishing system. Tiny droplets of water and nitrogen gas, like a fog, are discharged from nozzles to absorb heat, reduce oxygen and extinguish the fire.

It causes minimal or no wetting and therefore no equipment damage, avoiding a costly clean-up. After rectifying the fire damage, the data centre can be quickly returned to operation.

Security

Physical security, data security, environmental security (avoiding flooding, earthquakes etc) and power supply security are all important considerations for data centres. Human entry usually requires some type of biometric system (like a retinal scan) via a secure doorway – see Fig.16. That shows a Circlelock door, which is described at **https://pemag.au/link/ac0c**

Server racks

Server racks are standardised frames (typically made from metal) that hold computer servers, network switches or other equipment. They help to organise wiring, airflow or plumbing for cooling, provide access for service & maintenance, and sometimes physical security – see Fig.17.

Server racks are mounted together in single or multiple rows in whatever number is required, as shown in Fig.18. An important feature of server racks is that they allow a very high density, with up to 42 individual systems in one standard rack, or over 100 with a 'blade' configuration.

A server rack is designed to accommodate equipment that is 19 inches (482.6mm) wide; that standard was established in 1922 by AT&T. The height of equipment is standardised in heights representing multiples of 1.75 inches (44.45mm). A single-height unit

Data Centres

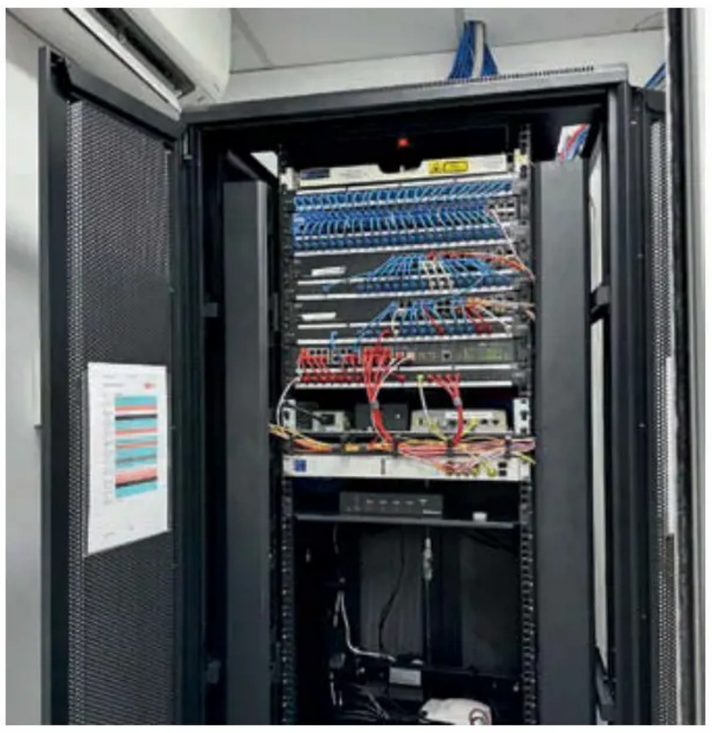


Fig.17: this server rack is mostly populated with network switches and patch panels. *Source: Fourbs Group – https://pemag.au/link/ac1c*



Fig.18: a group of server racks in a data centre. *Source: https://kpmg.com/jp/en/home/insights/2022/03/datacenter-business.html*



Fig.19: removing a 1U rack-mounted server mounted with sliding rails. *Source: https://youtu.be/fWaW9lA_pA0*

is designated 1U (see Fig.19), double height 2U etc.

Equipment might be mounted on rails so it can easily be slid out for service. Alternatively, and more simply, it may be bolted to the edges of the rack using 'rack ears'.

Almost all aspects of server racks are covered by CEA, DIN, EIA, IEC and other standards. The so-called 19-inch rack is used for many other types of equipment as well.

There are some other rack standards. One example is Open Rack, an initiative of the Open Compute Project. This rack was specifically designed for large-scale cloud deployments and has features such as a pair of 48V DC busbars at the rear to power the equipment.

It is designed for equipment that is 21-inches (538mm) wide instead of 19in (482.6mm), with a vertical spacing of 1.89in (48mm) instead of 1.75in (44.45mm) to improve cooling.

The racks are strong to accommodate the extra weight of equipment, all cables connect at the front rather than the back, and IT equipment is hot pluggable. See Fig.20 for a typical Open Rack configuration.

Data storage

While there is a general move to solid-state drives (SSDs) for data storage, hard disk drives (HDDs) retain some advantages over SSDs such as lower price, especially for higher capacities; they last longer, with little degradation with constant read/write cycles; and data recovery is easier for certain failure modes.

According to Seagate (https://pemag.au/link/ac0d), over 90% of online data stored in data centres is on hard disk, with the remainder on SSDs.

Western Digital sells a drive intended for use in data centres, the Ultrastar DC HC680, with a capacity of 28TB. Seagate's Exos X series of hard drives have capacities up to 32TB.

Tape drives are also used in data centres for archiving data and backups. They have great durability and longevity, and can provide an 'air gap' (no physical connection to the rest of the system) to protect stored data against hacking attempts and ransomware. They are also low in cost for their high capacity.

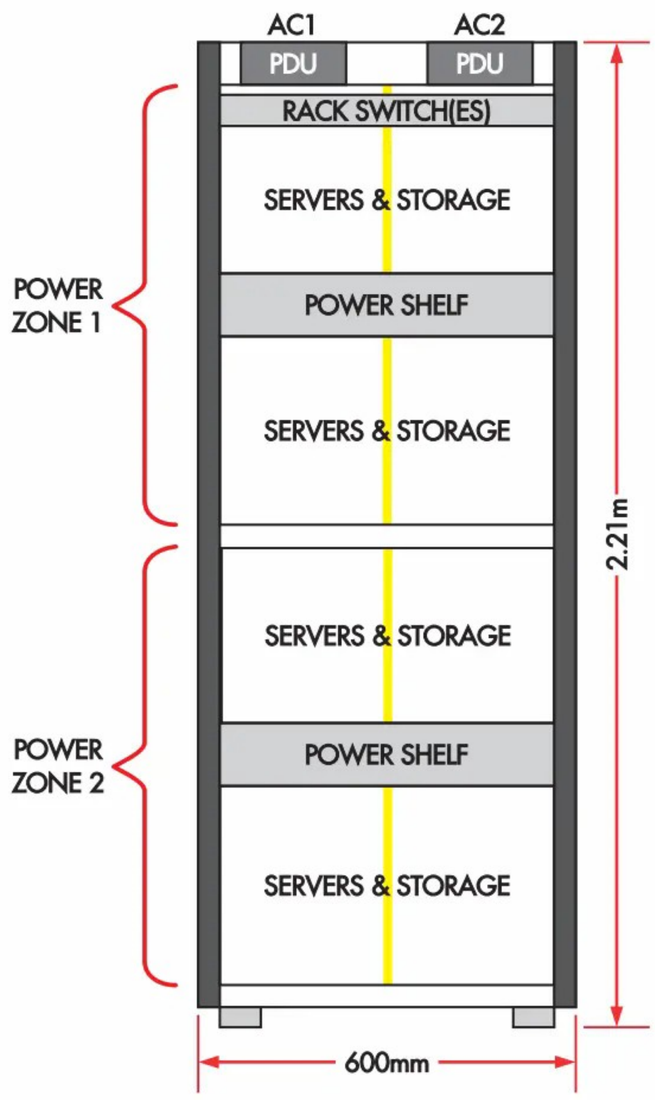


Fig.20: a typical configuration for an Open Compute Project V2 rack. *Original source: Mission Critical Magazine – pemag.au/link/ac1e*

Enterprise and Datacenter Standard Form Factor (EDSFF) is a specification designed to address the limitations of the 2.5-inch and M.2 sizes for solid-state drives. EDSFF drives provide better signal integrity, can draw more power and have higher maximum read/write speeds.

Standards for data centres

Various international standards exist for the design of data centres and their security and operational efficiency. Examples include:

- ISO/IEC 22237-series
- ANSI/TIA-942
- ANSI/BICSI 002-2024
- Telcordia GR-3160

Data centre ratings

Data centres can be rated according to the TIA-942 standard:

Rated-1: Basic Site Infrastructure

The data centre has single-capacity components, a non-redundant distribution path for all equipment and limited protection against physical events.

Rated-2: Redundant Component Site Infrastructure

The data centre has redundant capacity components, but a non-redundant distribution path that serves the computer equipment.

Rated-3: Concurrently Maintainable Site Infrastructure

The data centre has redundant capacity components and redundant distribution paths that serve the computer

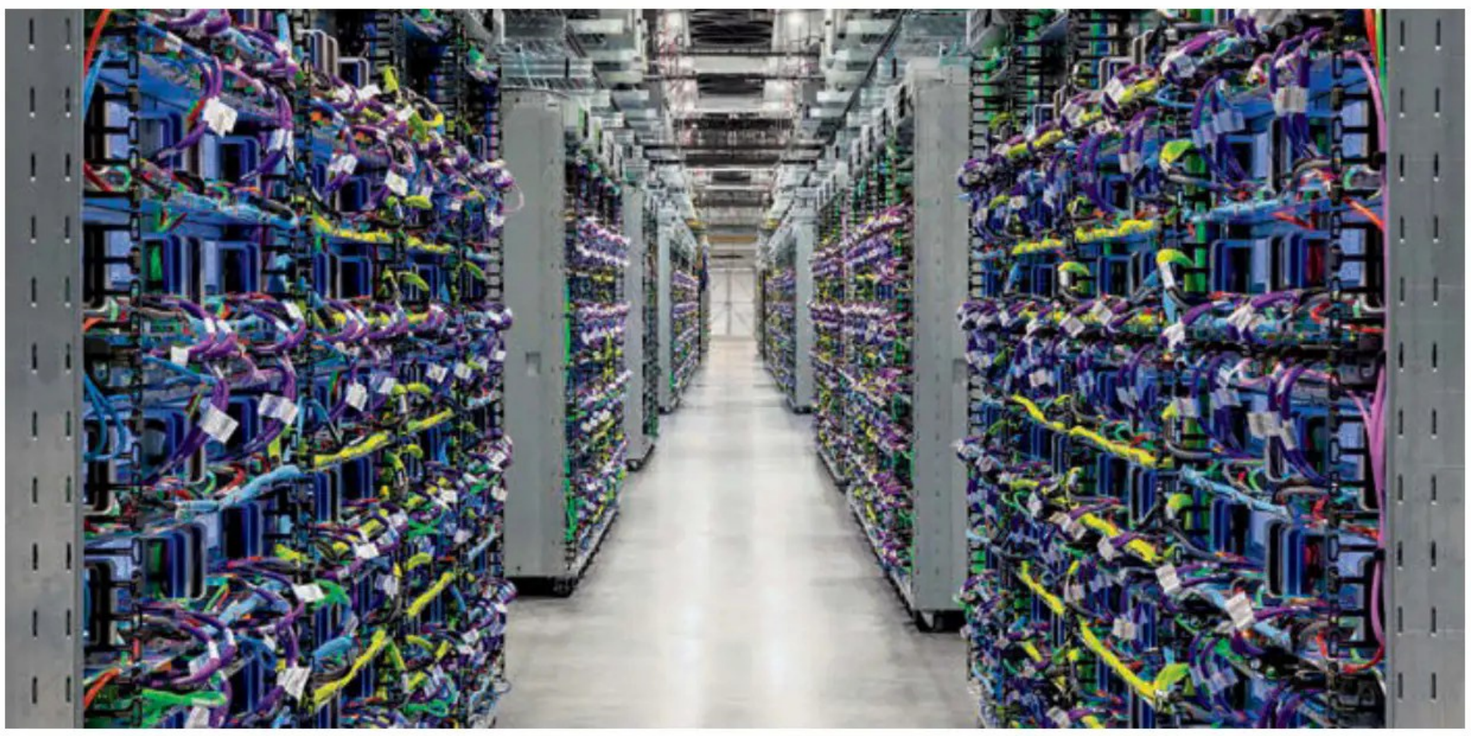


Fig.21: the Google Cloud TPU v5e AI infrastructure in a data centre. *Source:* *https://cloud.google.com/blog/products/compute/announcing-cloud-tpu-v5e-and-a3-gpus-in-ga*

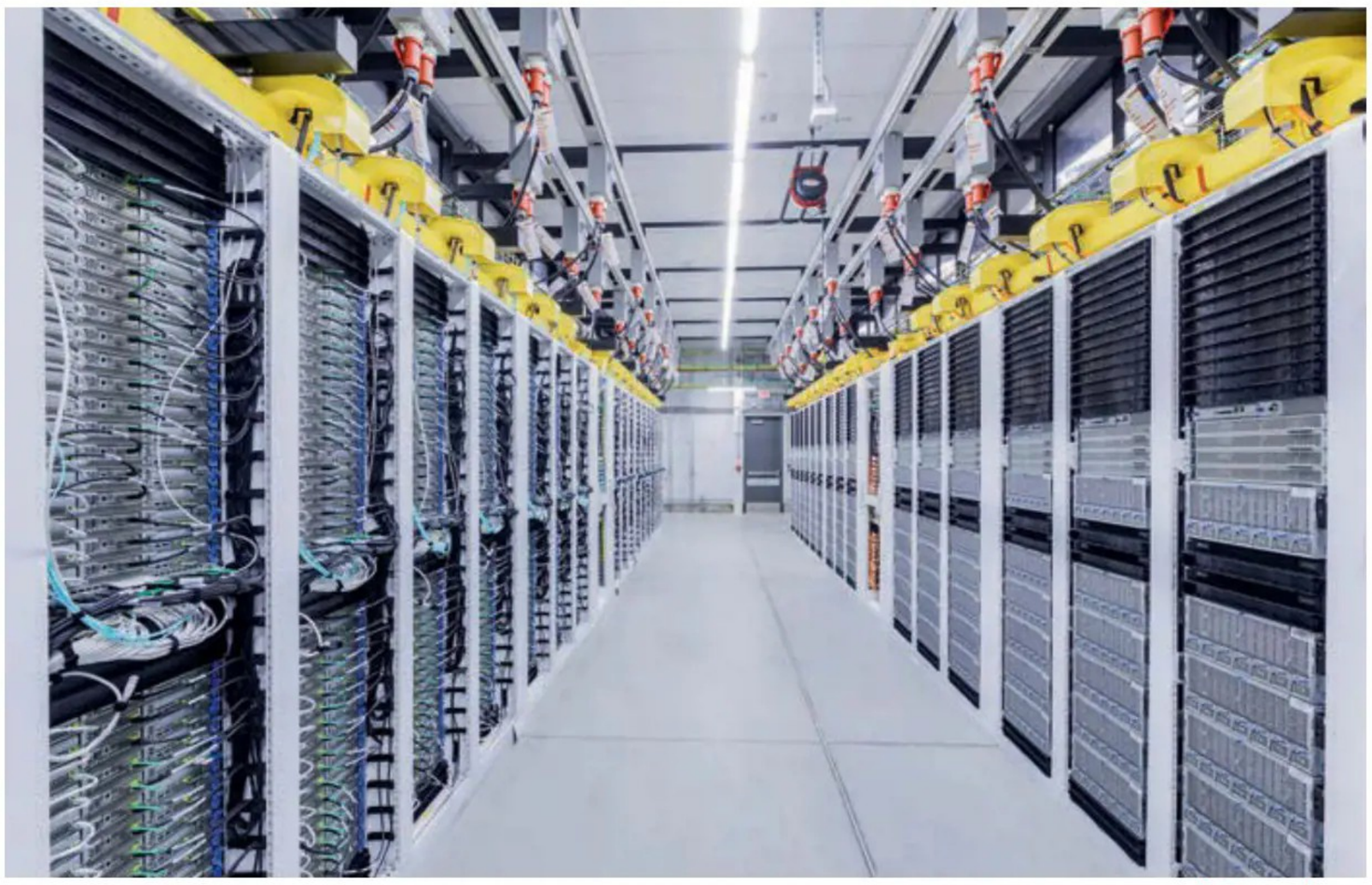


Fig.22: the Microsoft Azure infrastructure that runs ChatGPT. *Source:* *https://news.microsoft.com/source/features/ai/how-microsofts-bet-on-azure-unlocked-an-ai-revolution*

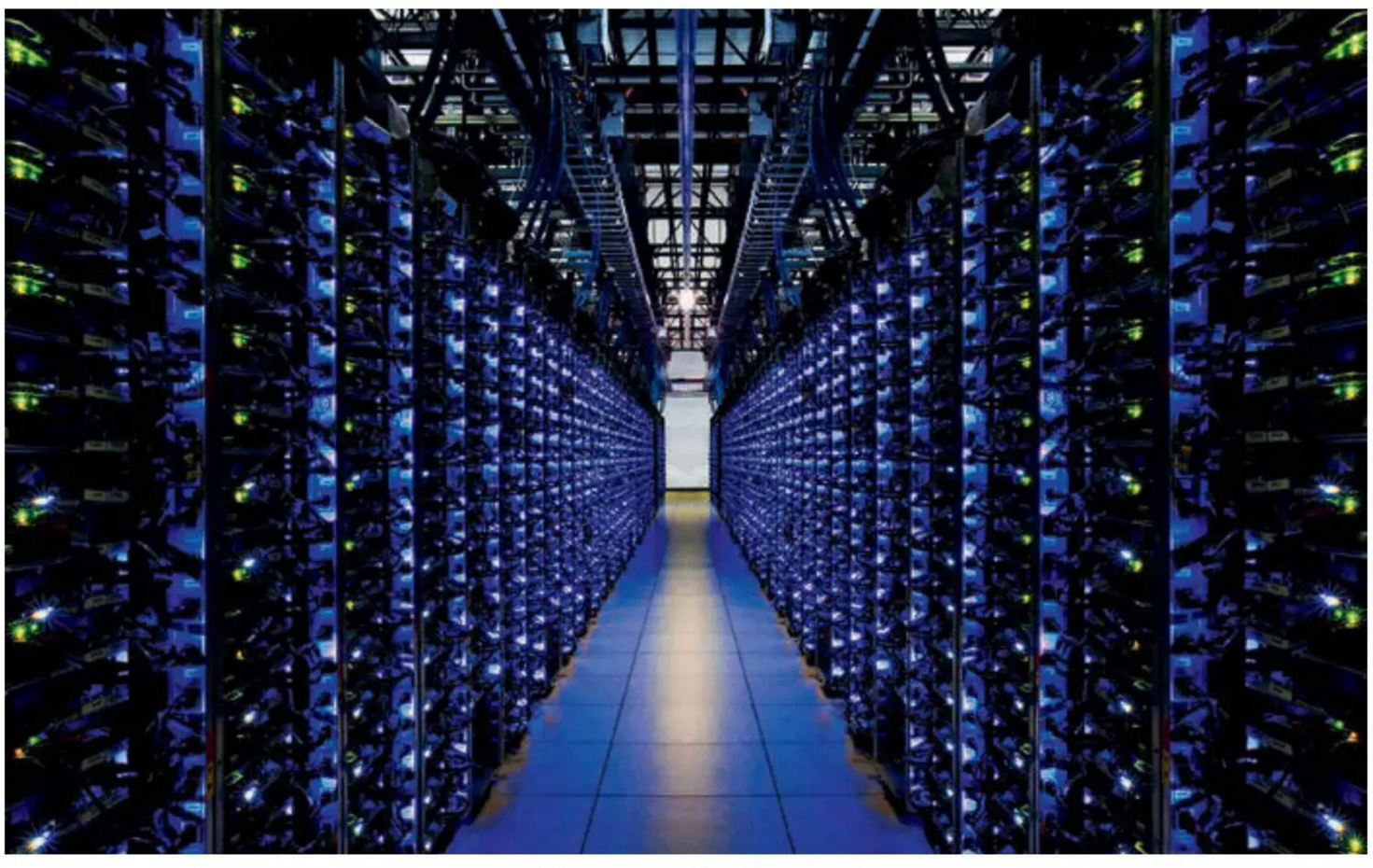


Fig.23: inside a small section of the Google data centre in Douglas County, Georgia, USA. *Source:* *www.google.com/about/datacenters/gallery*

equipment, allowing for concurrent maintainability of any piece of equipment. It also has improved physical security.

Rated-4: Fault Tolerant Site Infrastructure

The data centre has redundant capacity components, active redundant distribution paths to serve the equipment and protection against single failure scenarios. It also includes the highest level of security.

A 'hyperscale' data centre is one designed to accommodate extreme workloads. Amazon, Facebook, Google, IBM and Microsoft are examples of companies that use them.

Artificial intelligence (AI)

Some data centres are specialised for AI workloads.

AI data centres are much like regular data centres in that they require large computing resources and specialised buildings. However, the resource requirements for AI are substantially more than a conventional data centre.

According to Australia's Macquarie Data Centres, conventional data centres require around 12kW per rack, but an AI data centre might require 60kW per rack. Oper8 Global (**https://pemag.au/link/ac0w**) states that an 'extreme density' rack can have a power consumption of up to 150kW!

An AI data centre requires far more computing resources. Instead of mainly using CPUs, it will also contain a significant number of GPUs and TPUs.

Deep learning & machine learning

AI data centres can use either machine learning or deep learning. Machine learning uses algorithms to interpret and learn from data, while deep learning uses similar algorithms but structures them into layers, within an artificial neural network simulating how a brain learns.

A neural network is hardware and/or software with architecture inspired by that of the human (or other) brains. It is used for deep learning, a form of artificial intelligence. Large versions of these are run in data centres. Machine learning does not necessarily use neural networks (but it can).

Machine learning is best for structured tasks with small datasets, with thousands of data points, but may

Data Centres

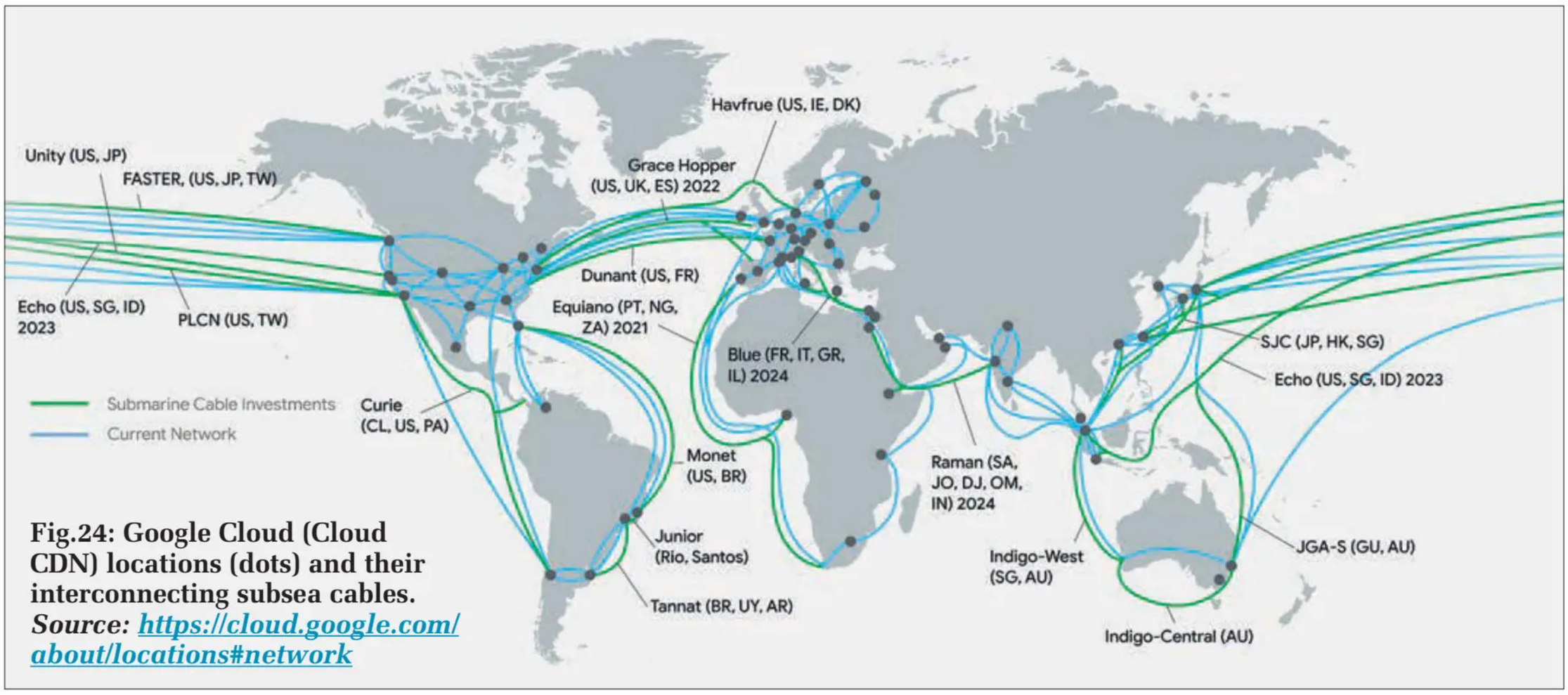


Fig.24: Google Cloud (Cloud CDN) locations (dots) and their interconnecting subsea cables. *Source:* *https://cloud.google.com/about/locations#network*

require human intervention if a learned prediction is incorrect. Deep learning is best for making sense of unstructured data with large datasets and millions of data points. Deep learning can determine for itself whether a prediction is wrong or not.

Machine learning is relatively quick to train but less powerful; deep learning can take weeks or months to train, like a person.

CPUs have advantages for implementing recurrent neural networks (RNNs). Typical applications for RNNs are for translating language, speech recognition, natural language processing and image captioning.

GPUs have advantages for some fully connected neural networks. They are probably the most common type of processor used for neural networks, hence the huge stock value of companies that make GPUs like NVIDIA, which at the time of writing is one of the most valuable publicly listed companies in the world at US$2.6 trillion.

Fully connected neural networks are suitable for deep learning and have applications in speech recognition, image recognition, visual art characterisation, generating art, natural language processing, drug discovery and toxicology, marketing, medical image analysis, image restoration, materials science, robot training, solving complex mathematical equations and weather prediction, among others.

TPUs have advantages for convolutional neural networks (CNNs). Applications for CNNs include pattern recognition, image recognition and object detection. Fig.21 shows part of the Google Cloud TPU data centre artificial intelligence infrastructure. Also see the video titled “Inside a Google Cloud TPU Data Center” at **https://youtu.be/FsxthdQ_sL4**

ChatGPT

This popular AI ‘chatbot’, developed by OpenAI, is hosted on a Microsoft Azure cloud computing data centre infrastructure (see Fig.22). It runs on tens of thousands of NVIDIA’s H100 Tensor Core GPUs with NVIDIA Quantum-2 InfiniBand networking.

Google data centres

Google is among the largest owners of data centres, storing vast amounts of the world’s data. Fig.23 shows the inside of a part of a Google data centre, while Fig.24 shows the location of Google Cloud data centres and their interconnection via undersea cables. The locations of data centres for delivering media such as videos (such as for YouTube) can be seen at **https://pemag.au/link/ac1d**

Open Compute Project (OCP)

The OCP (**www.opencompute.org**) was founded in 2011 with the objective of sharing designs for data centre products and practices. Companies involved include Alibaba Group, Arm, Cisco, Dell, Fidelity, Goldman Sachs, Google, Hewlett Packard Enterprise, IBM, Intel, Lenovo, Meta, Microsoft, Nokia, NVIDIA, Rackspace, Seagate Technology and Wiwynn.

Their projects include server designs, an accelerator module for increasing the speed of neural networks in AI applications, data storage modules (Open Vault), Open Rack (mentioned previously), energy-efficient power supplies and network switches based on SONiC (Software for Open Networking in the Cloud).

Underwater data centres

Because of the significant cooling requirements of data centres and the need for physical security, experiments have been made in placing data centres underwater.

They would be constructed within a pressure-resistant waterproof container, with only electrical and data cables coming to the surface. They would not have any staff. With no people, there is no need for a breathable atmosphere, so it can be pure nitrogen to reduce corrosion of connectors and other parts.

There is also no possibility of accidental damage such as people dislodging wires etc. Also, there would be no dust to clog cooling fans or get into connectors.

The underwater environment has a stable temperature, resulting in fewer failures than when the temperature can vary a lot.

It is much easier and more efficient to exchange heat with a fluid such as water than with air, reducing the overall power consumption.

An underwater environment also provides protection from some forms of nuclear radiation, which can cause errors in ICs, as water is a good absorber of certain types of radiation. Water can also absorb electromagnetic pulses (EMP) from nuclear explosions. The fact that the electronics are also effectively housed in a Faraday cage will also help with disaster resistance.



Fig.25: cleaning Microsoft's underwater data centre after being on the seabed for two years, off the Orkney Islands in Scotland. *Source:* *https://news.microsoft.com/source/features/sustainability/project-natick-underwater-datacenter*



Fig.26: an IBM modular data centre built into a standard 40ft (12.2m) long shipping container. *Source:* *https://w.wiki/B5ft*



Fig.27: looking like somewhere where Superman might live, this 65-storey data centre is proposed to be built in Iceland. *Source:* *www.yankodesign.com/2016/04/01/the-internets-fortress-of-solitude*

Physical security is improved as being underwater, even if a diver could get to it, there would be no practical way to get inside without flooding the whole container.

An underwater data centre can also contribute to reduced latency (response time) because half the world's population lives within 200km of the sea, so they can be optimally placed near population centres and possibly undersea cables.

Underwater data centre projects include:

- Microsoft Project Natick (**https://natick.research.microsoft.com**), an experiment first deployed in 2015 with a data centre built within a pressure vessel 12.2m long, 3.18m in diameter, and is about the same size as a standard 40ft (12.2m) shipping container – see Fig.25.

Its power consumption was 240kW. It had 12 racks containing 864 standard Microsoft data centre servers with FPGA acceleration and 27.6 petabytes of storage. The atmosphere was 100% nitrogen at one bar. Its planned operational period without maintenance was five years.

- Subsea Cloud (**www.subseacloud.com**) is proposing to put data centres 3km below sea level for physical security.
- Chinese company Highlander plans to build a commercial undersea data centre at the Hainan Island free trade zone, with a facility for 100 airtight pressure vessels on the seabed.

Modular data centres

A modular data centre is designed to be portable and is built into a structure like a shipping container – see Fig.26. They might be used to supplement the capacity of an existing data centre, for disaster recovery, humanitarian purposes or for any other reasons where a data centre has to be moved into a place it is needed.

Iceland data centre

A 65-storey data centre has been proposed to be built in the Arctic (see Fig.27). It was designed by Valeria Mercuri and Marco Merletti.

If built in Iceland, it could take advantage of inexpensive geothermal energy and be close to international cable networks. The low temperatures would minimise cooling costs, and the vertical design would minimise land usage. PE

Practical Electronics PCB SERVICE

PROJECT	CODE	PRICE
MARCH 2026		
USB-Programmable Frequency Divider PCB	04108241	£3.99
Rotating Light PCB (black)	09101251	£2.99
433MHz Transmitter PCB	15103251	£2.99
Arduino Uno Game Cartridge PCB (Cool Beans)	CBmar26uno	£3.99
Arduino Nano Game Cartridge PCB (Cool Beans)	CBmar26nano	£2.99
Silent Stomp Switch PCB pair (Audio Out)	AO-MAR26-*	£3.99
FEBRUARY 2026		
Power LCR Meter PCB	04103251	£7.99
Versatile Waveform Generator PCB	04104251	£5.99
Penta-Fuzz 'bender' PCB	AO-FEB26-1	£2.99
JANUARY 2026		
High-Bandwidth Differential Probe	9015-D	£5.99
NFC-Programmable IR Remote Keyfob	15109231	£2.99
Pre-driver Overdrive Pedal main PCB	AO-JAN26-1	£4.99
Updated jack PCB for the Pre-driver	AO-JAN26-2	£3.99
DECEMBER 2025		
Variable Speed Drive (black)	11111241	£11.99
Digital Capacitance Meter	04111241	£5.99
Battery-Powered Model Train transmitter	09110241	£3.99
Battery-Powered Model Train through-hole receiver	09110242	£3.99
Battery-Powered Model Train SMD receiver	09110243	£3.99
Battery-Powered Model Train battery charger	09110244	£3.99
Transmanium Fuzz Box jack PCB	AO-DEC-25	£3.99
NOVEMBER 2025		
Capacitor Discharger	9047-01	£4.99
5MHz, 40A Current Probe	9049-01	£5.99
Transmanium Fuzz Box main PCB	AO-NOV-25	£4.99
OCTOBER 2025		
Seaside Sound Simulator PCB (blue)	01111241	£7.99
Fender Jazz Bass pickguard	23109241	£5.99
J&D Luthiers T-Style pickguard	23109242	£5.99
Music Man Stingray pickguard	23109243	£5.99
Fender Telecaster pickguard	23109244	£4.99
3D Printer Filament Dryer control PCB	28110241	£4.99
SEPTEMBER 2025		
Automatic LQ Meter PCB	CSE240203A	£5.99
Automatic LQ Meter front panel (black)	CSE240204A	£5.99
Compact Hi-Fi Headphone Amplifier PCB (blue)	01103241	£4.99
Discrete Ideal Bridge Rectifier TH PCB	18108241	£2.99
Discrete Ideal Bridge Rectifier SMD PCB	18108242	£2.99
Dual Rail Supply Protector PCB	18109241	£3.99
Audio Out Cascode Amplifier PCB	CASCODEA	£4.99
Audio Out Cascode Amplifier Power Supply PCB	CASCODEP	£4.99
AUGUST 2025		
Micromite Explore-40 PCB	07106241	£2.99
Micromite Explore-40 Audio Breakout PCB	07101222	£2.99
8-Channel Learning IR Remote Receiver PCB	15108241	£5.99
FlexiDice PCB set	08107241-2	£4.99
JULY 2025		
Compact OLED Clock & Timer PCB	19101231	£4.99
180-230V DC Motor Speed Controller PCB	11104241	£9.99
Mains Power-Up Sequencer PCB (same as Feb '25)	10108231	£11.99
JUNE 2025		
Styloclone (case version)	23106241	£6.99
Styloclone (standalone version)	23106242	£7.99
Dual Mini LED Dice (TH LEDs, white solder mask)	08103241W	£2.99
Dual Mini LED Dice (TH LEDs, black solder mask)	08103241B	£2.99
Dual Mini LED Dice (SMD LEDs, white solder mask)	08103242W	£2.99
Dual Mini LED Dice (SMD LEDs, black solder mask)	08103242B	£2.99
Dual Hybrid Supply regulator board	18107211	£5.99
Dual Hybrid Supply front panel control board	18107212	£2.99
Dual Hybrid Supply CPU board	01106193	£3.99
Dual Hybrid Supply LCD adaptor	01106196	£2.99
MAY 2025		
Skill Tester 9000	08101241	£9.99
USB-C Serial Adaptor (black solder mask)	24106241	£2.99
DC Supply Protector (SMD adjustable version)	08106241	£3.99
DC Supply Protector (TH adjustable version)	08106242	£3.99
DC Supply Protector (TH fixed version)	08106243	£3.99
APRIL 2025		
ESR Test Tweezers PCB set	04105241+	£5.99
Fan Speed Controller Mk2	10104241	£4.99
Calibrated Microphone (SMD version)	01108231	£3.99
Calibrated Microphone (TH version)	01108232	£3.99
MARCH 2025		
Pico Gamer	08104241	£9.99
Compact Frequency Divider	04112231	£4.99
FEBRUARY 2025		
Laser Communicator transmitter	16102241	£3.99
Laser Communicator receiver	16102242	£3.99
Model Railway Points Controller transmitter	09101241	£3.99
Model Railway Points Controller receiver	09101242	£2.99
Mains Power-Up Sequencer	10108231	£11.99
JANUARY 2025		
USB to PS/2 Converter for VGA PicoMite	07111231	£3.99
ps2x2pico USB to PS/2 Converter	07111232	£3.99
Compact Microphone Preamplifier (case version)	01110231	£5.99
Compact Microphone Preamplifier (integrated version)	01110232	£5.99
DECEMBER 2024		
Raspberry Pi Clock Radio PCB pair	19101241/2	£12.99
VGA PicoMite (blue)	07107221	£5.99
Secure Remote Switch receiver	10109231	£4.99
Secure Remote Switch transmitter (module version)	10109232	£2.99
Secure Remote Switch transmitter (discrete version)	10109233	£2.99
NOVEMBER 2024		
Ideal Diode Bridge Rectifiers (28mm square)	18101241	£3.99
Ideal Diode Bridge Rectifiers (22mm square)	18101242	£3.99
Ideal Diode Bridge Rectifiers (4-pin SIL)	18101243	£3.99
Ideal Diode Bridge Rectifiers (tiny SOT-23)	18101244	£3.99
Ideal Diode Bridge Rectifiers (standalone SMD)	18101245	£4.99
Ideal Diode Bridge Rectifiers (standalone TH; 2oz copper)	18101246	£6.99
Multi-channel Volume Control (volume PCB)	01111221	£4.99
Multi-channel Volume Control (control PCB)	01111222	£4.99
Multi-channel Volume Control (OLED PCB, black)	01111223	£4.99
Coin Cell Emulator (black)	18101231	£3.99
OCTOBER 2024		
TQFP-32 Programming Adaptor	24108231	£4.99
TQFP-44 Programming Adaptor	24108232	£4.99
TQFP-48 Programming Adaptor	24108233	£4.99
TQFP-64 Programming Adaptor	24108234	£4.99
2m VHF FM Test Signal Generator	06107231	£4.99
SEPTEMBER 2024		
PIC Programming Adaptor	24105231	£4.99
Salad Bowl Speaker crossover	01109231	£7.99

(continued on page 78...)

PCBs for most recent *PE/EPE* constructional projects are available. From the July 2013 issue onwards, PCBs with eight-digit codes have silk screen overlays and, where applicable, are double-sided, have plated-through holes, and solder mask. They are similar to photos in the project articles. Earlier PCBs are likely to be more basic and may not include silk screen overlay, be single-sided, lack plated-through holes and solder mask.

Always check price and availability in the latest issue or online. A large number of older boards are listed for ordering on our website.

In most cases we do not supply kits or components for our projects. For older projects it is important to check the availability of all components before purchasing PCBs.

Magazine back issues are available – see **https://www.electronpublishing.com/product-category/pe-back-issues-print-and-digital/**

Part 2 by Phil Prosser

POWER LCR METER



We introduced this new device last month. It isn't just another LC meter; it can deliver a range of currents up to 30A to determine how an inductor behaves as its core starts to saturate. This tester can also measure very high capacitances and very low resistances. This article covers its assembly, testing, calibration and use.

The Power LCR Meter has two basic modes: it either applies a fixed current or a fixed voltage to the device under test (DUT) and samples the voltage across it and current through it many times over a short period. It then examines those samples to determine either its resistance, capacitance or inductance.

Because it can control the current used for the test, for power inductors, it can step through a range of currents and calculate the inductances, allowing you to see how it changes. For a typical inductor with a ferrite, iron or mu-metal core, the inductance will remain relatively steady until a certain current level is reached, then it will fall off as the core saturates.

Having this information is invaluable as it allows you to determine whether the inductor will be suitable for applications that demand a certain inductance up to a certain current level, like a loudspeaker crossover or switch-mode power supply.

Construction

The Power LCR Meter is built on a double-sided 156 × 118mm PCB coded 04103251. It mostly uses through-hole parts, but there are a few SMDs, which should be fitted first. During assembly, refer to the component overlay diagrams, Figs.10 & 11, to see which parts go where.

You can see in the photos that we didn't have a 5W 0.39Ω resistor, so we used two smaller resistors in series. We only installed one 47,000µF capacitor on this prototype, which was enough for the test inductors used. Fit both if you want to test large, low-resistance inductors.

You will also see that we have used 1µF & 10µF SMD tantalum capacitors, while the final parts list suggests ceramic capacitors instead. You can use either, but the specific ceramic capacitors should be cheaper, more reliable and perform better. If you use tantalums, make sure you orientate them with the positive stripes as shown on the PCB and in the photos.

We always like to fit all the power supply parts before the remaining active semiconductors to make testing easier. So start by mounting all the parts in the power supply section, which is everything to the left of the white vertical line on the silkscreen (the black line in Fig.10, including the parts in the lower-left corner).

It's easiest to start with low-profile components like resistors and then work your way up to the taller ones, ending with the bulky and heavy inductors.

Watch the orientations of the diodes, electrolytic capacitors, regulators and transistor. For the regulators and transistor, pay attention to which side the metal tab goes (REG3 & REG5) or flat face (the others) so that they match Fig.10.

There is space for a heatsink for the LM2576 (REG5), but it is not required. The average dissipation is low enough that it will be fine without it.

With all the power supply components installed, you can connect a 12-20V DC power supply to CON4 (with the positive lead nearest the fuse) and check the following:

1. Check the 10V filtered rail is 9-11V; our four prototypes all measured about 9.8V. You can measure this on the DUT+ terminals. There is a GND test point just next to the power switch; we found it convenient to solder a

Power LCR Tester, part two

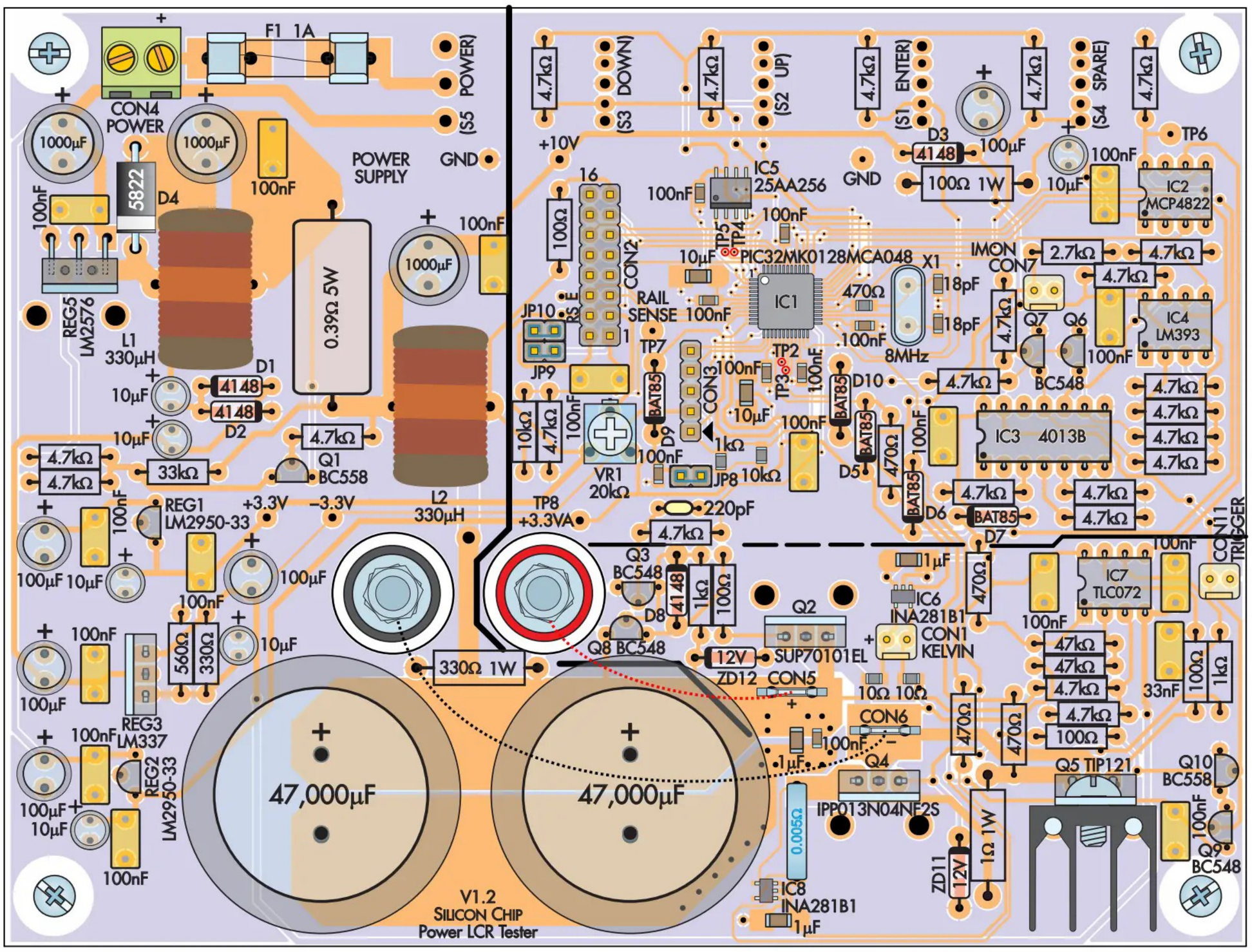


Fig.10: we recommend you fit the power supply components first (the whole left-most section) so you can verify that is all working before adding the rest of the parts. Be very careful to orientate IC1 correctly, with its pin 1 dot at upper left, before soldering it. Also watch the orientations of the other ICs, diodes, electros, and transistors (including the Mosfets).

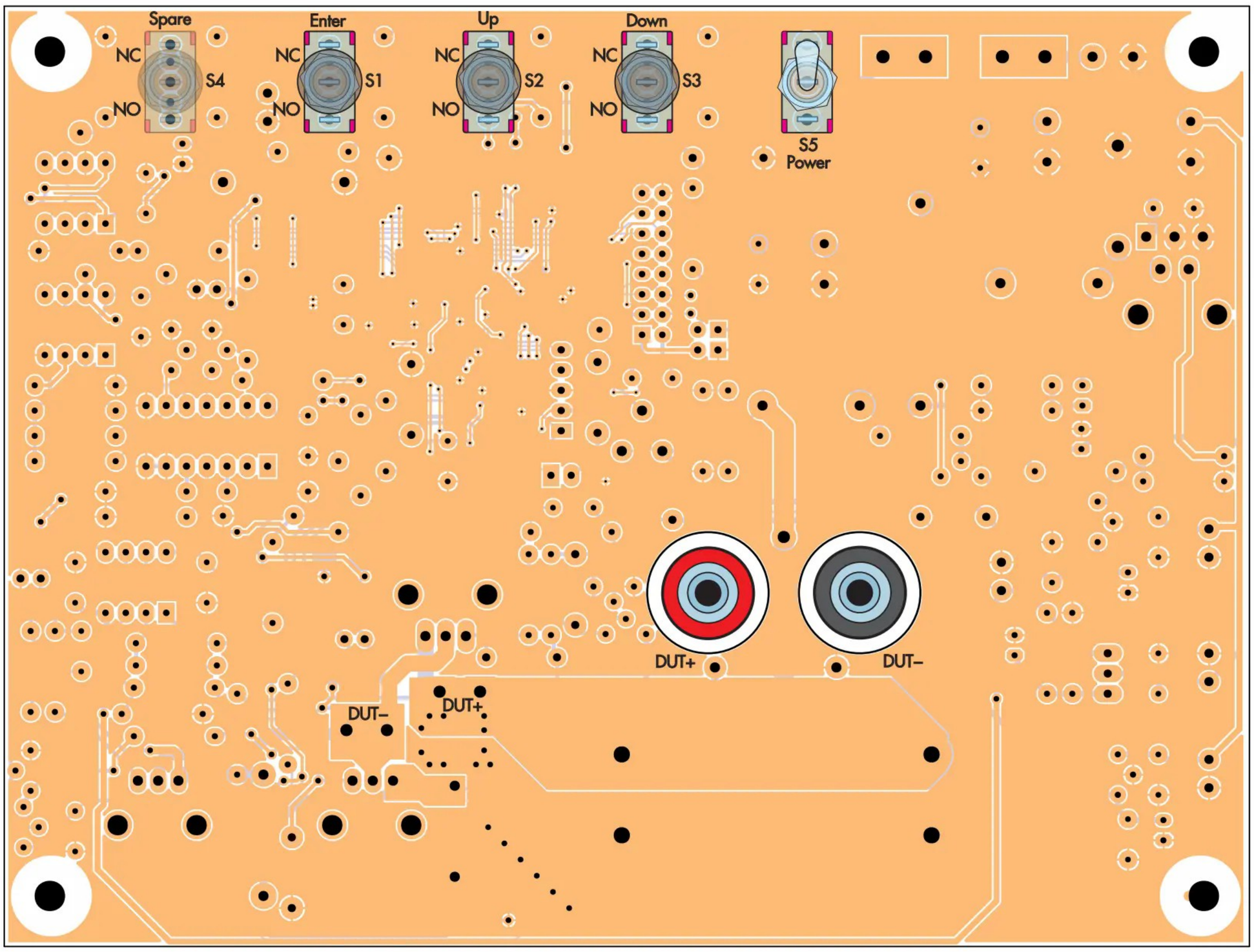


Fig.11: there aren't many parts on the back of the PCB; just the four or five switches. The main measurement terminals pass through the two large holes near the middle.

piece of tinned copper wire into this to clip onto.

2. Check the +3.3VD, +3.VA and -3.3V voltages. Test points for these are just above the circular cutouts for the DUT connectors. We expect the two positive rails to be within 100mV; note that in normal operation running from 12V, these regulators get quite warm.

If any of these are off significantly, or something gets hot, check the orientation of all capacitors and diodes. We tried to keep all capacitors orientated the same way, but because switch-mode power supplies have exacting layout requirements, the diode placement in that area is not so consistent.

The 330Ω resistor just above the 47,000µF capacitors is there to put a sufficient load on the switch-mode power supply that it runs continuously. We need this to generate the -3.3V supply. If your -3.3V supply does not come up properly, but everything else looks OK, check it.

The following surface mount parts can go on next. With the power supplies behaving, it is time to get the fiddly bits on while there is still room. That includes:

- The PIC32MK0128MCA048 (IC1).
- The two 10µF surface-mount capacitors.
- The eight 100nF surface-mount bypass capacitors, which are mostly around IC1.
- The two 18pF SMD capacitors near the crystal oscillator.
- The three 1µF SMD capacitors, which are next to the INA281s and across the DUT terminals.
- The 25AA256-I/SN serial EEPROM.
- 470Ω series resistor for the crystal oscillator.
- The 10kΩ and 1kΩ resistors next to the reset header.
- The two 10Ω resistors for the Kelvin connection option.
- The two INA281B1 devices (IC6 & IC8).

The INA281 devices are in SOT-23-5 packages, which are a little on the small side. However, if you approach them with some care, they are not too difficult to solder.

The PIC microcontroller is in a 48-pin thin quad flat pack (TQFP), which has a 0.5mm lead spacing. This was the most easily soldered IC in the series we could find, alternative devices being in leadless packages, which are daunting to solder.

We have provided soldering guides for TQFP and SOT-23 packages in the past. Our key tip is to use plenty of flux paste and to use a magnifying loupe to check for bridges between pins when you've finished. Use solder wick to remove any bridges you find. If the joint on a pin looks a little dry, resolder it before it causes you trouble later.

When you've finished construction and apply power, if the LCD does not fire up immediately, come back and double check those pins for shorts. We have had to fix plenty of solder bridges ourselves in the past; the PIC microcontrollers are very tolerant of shorted pins and we have not managed to blow one up yet from a solder bridge (but it's still better to clear them before applying power).

Pro tip: after soldering all the SMDs, you will probably have flux residue that gets in the way of a proper inspection. Clean it off using a flux solvent (or isopropyl alcohol or methylated spirits if that's all you have) and it will be much easier to spot any problems. Your board will also look a lot nicer and be less sticky!

Mounting the LCD

We want to connect the 16×2 LCD to the main PCB with a 16-way ribbon cable. To fit neatly in the case, we directly soldered the ribbon cable to the 14 through-holes on the LCD. This was a nuisance, but there was not room in the case for the IDC header we wanted to use. We say 14 and not 16 because the backlight connections are at the other end of the LCD.

We show how we connected this in the photo below. Ensure that the red wire on your ribbon cable goes to pin 1 at both ends. Also make sure that once crimped, the IDC cable comes out in the right direction. The total length of ribbon cable we used was 300mm, with about 200mm between the IDC header and LCD board, leaving that extra length to connect to the backlight on the LCD board.

Pins 1-14 of the ribbon cable are connected to the same pin number on the LCD. Note that the pins alternate between the two columns on the LCD.

For the two remaining wires on pin 15 and 16 from the main board, strip the end of these and solder them to the anode and cathode backlight pads.

Importantly, for the Altronics screen, you must place jumpers horizontally on JP9 and JP10 on the main board as shown in Fig.10. This applies 3.3V to V_{DD} (pin 2) on the LCD and grounds pin 1.

If you are using a different display, check its data sheet, as these pins are sometimes swapped between manufacturers. If this is the case, you can install JP9 and JP10 vertically, which will swap the rails.

Getting the microcontroller working

At this point, we can install the remaining parts in the microcontroller section. That is the section at upper-right bordered by a solid vertical line on the left and a broken horizontal line below.

The four pushbutton switches mount to the rear of the PCB (S4 is not needed). For these, it is important that you rotate them so the normally open (NO) pins are at the bottom. Double-check this using a continuity meter; if on startup the system always goes into calibration mode, you almost certainly have the switches in the wrong way around.

Also watch the orientations of the BAT85 protection diodes as they are not consistent. We also note that you can save quite a bit purchasing these from the larger online suppliers.

We have used a lot of 4.7kΩ resistors to make it easier to purchase and manage the parts for this project. However, there are some 470Ω resistors as



This shows how to solder the ribbon cable to the Altronics 16×2 LCD. We tried to use an Altronics P5162A 14-way IDC-to-PCB adaptor, but it wouldn't fit in the space available. If you are installing it in a larger case, you may be able to use it.

well, which will have similar colour codes, so take care not to mix them up. Mount all the 4.7kΩ resistors at once and you can be confident you won't confuse them.

Plug the LCD onto the main PCB, making sure that you get the pin 1 ends right at both the PCB and display end. We can now test this part of the board.

Apply power and check the power rail voltages again. The voltages should be about the same; if any are very low, look for things getting hot or capacitors in the wrong way around.

You should see the LCD backlight come on. If not, check the connections on the LCD from the header to the backlight LED and check that the headers are plugged in the right way around.

You now need to adjust trimpot VR1, which controls the LCD contrast. Start at one end and turn it until you get good contrast on the display. There should be legible text if everything is fine, but if the LCD has not fully booted, you will still see lines of boxes or characters. If you get no display at all, double-check your LCD data sheet to make sure JP9 and JP10 are in the right locations.

If the LCD is not displaying anything at all, check the soldering on the microcontroller and your cabling. If this all looks good, you probably want to check for activity on the LCD RS and E lines with an oscilloscope (if you have one). We put test points on the PCB for these – although we didn't have to use them, as the 16×2 LCDs seem to mostly just work.

If you still think nothing is happening and the display is blank, check the crystal oscillator drive on its associated 470Ω resistor. There must be an 8MHz sinewave here; if it is missing, double-check the microcontroller solder joints. You should now have a screen with text on it.



You can see how we wired the sockets to the PCB, all via polarised plugs or screw terminals.

Installing the measurement section

You can solder all the remaining parts in place now. The only heatsink that you need to attach is on Q5, as shown in Fig.10. The other devices don't dissipate enough power to warrant heatsinks, even though we have space for them on the board.

With all parts mounted, you should be able to fire the meter up and get a screen saying "Resistance < 300R, Enter to Meas" and similar for Capacitance, Inductance and Inductance Saturation.

If you press the Enter/OK button, the meter will display "Measuring Resistance", "Measuring Capacitance", "Measuring Inductance" or "Measuring Inductance Sat'n" respectively and go off and measure the value. Note on our case we labelled "Enter" as "OK".

The standard measurements take about a second, while the inductor saturation tests need to perform quite a lot of measurements and take longer. Because we are dealing with inductors carrying a lot of current, we also need to provide a decent charge and decay time. So the inductance saturation test can take a few seconds, depending on the value of the inductor under test.

The results are displayed on the screen and, once presented, you can press Enter/OK to repeat the measurement. If you want to change between resistance, capacitance and inductance measurements, press the up/down keys to cycle through the options, then press Enter/OK to measure.

After a saturation current measurement is complete, you can cycle through the 10 inductance values across the range the meter can provide. The maximum current the meter will test to is 30A, plus readings from 5% to 90% of the maximum current.

We have selected this range to ensure that noise at the start of the measurement does not grossly affect results (although it may still affect it if the inductor rings badly). By pressing up

and down, you can review:

- The current at which the measurement is made.
- The percentage of the inductance value of the second inductance measurement, which is considered 100%. We chose the second measurement, as this was always ‘clean’ in our tests.
- The value of inductance at the displayed current.

Calibration

If you don’t calibrate the meter, it will load defaults, which will work but definitely compromise accuracy. To calibrate the meter, apply power and hold both the up and down buttons continuously. The meter will present the question “Calibrate meter?”, “Y/N, Up/Dn”. Press the up button, and a series of help screens will walk you through the process.

As you will see in operation, inductance values are ‘all over the shop’ with current, so we have kept calibration focused on the few key parameters. We can calibrate critical parameters, but we do not seek to create a ‘lab standard’; this is more of a working measurement system for power devices where a few percent precision is sufficient. **There are five steps to calibration:**

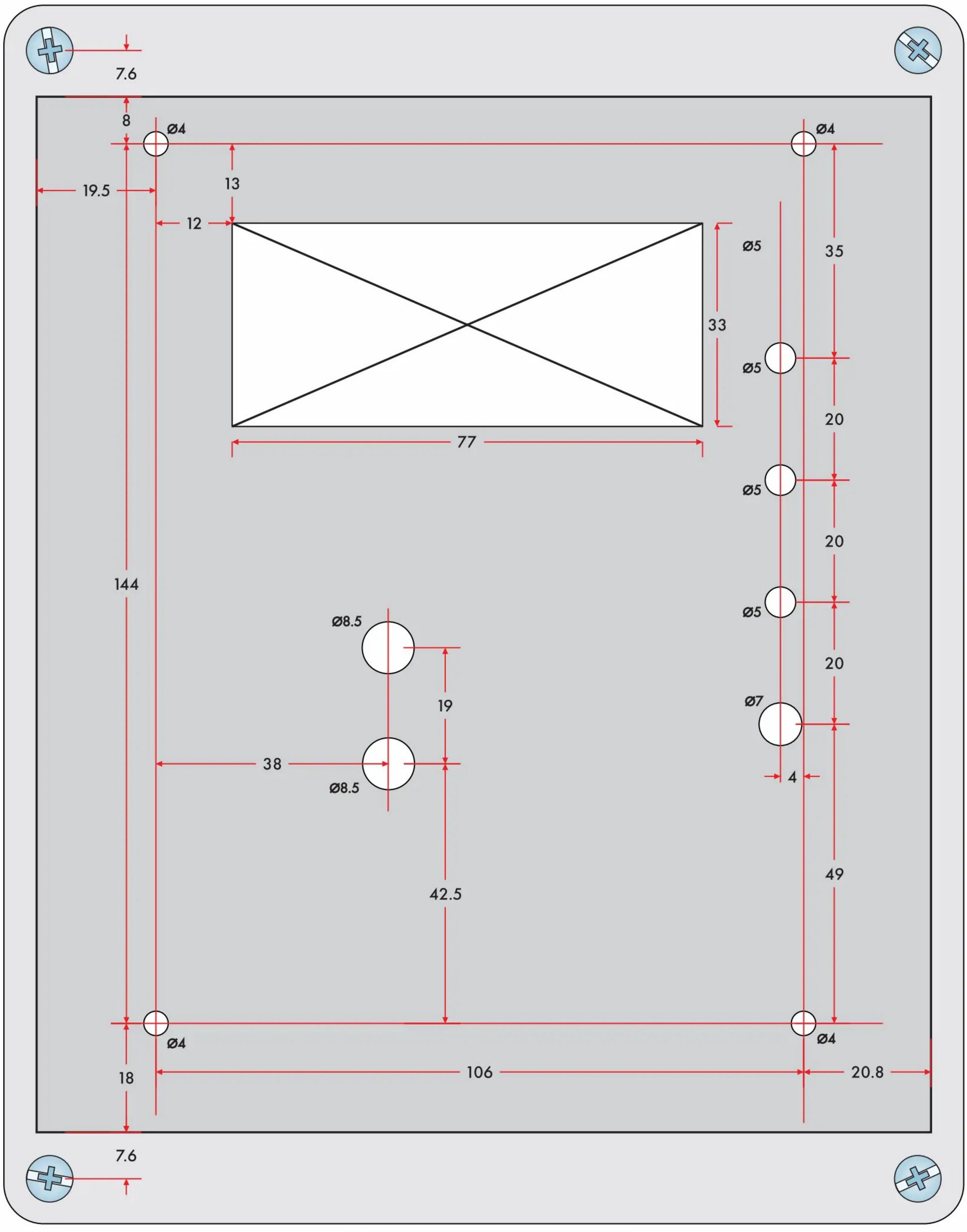


Fig.12: drill the holes in the lid as shown here. It’s best to start with pilot holes and then enlarge them to size. For the rectangular cutout, you could use something like a jigsaw, but you can also drill lots of small holes within the outline, knock the centre out, then file it to shape. It doesn’t have to be perfect as the bezel will cover minor imperfections.

ALL DIMENSIONS IN MILLIMETRES SCALE: 100%

Power LCR Tester, part two

#1: 10mA constant current test

The current measurements in steps 1-3 are important for resistance and capacitance tests.

Connect a milliammeter across the DUT terminals. The Meter will drive a 10mA current. Measure this and use the up/down buttons to enter your measured value. Get this to within 0.1mA of your meter reading.

#2: 100mA constant current test

This is the same as step 1 but at 100mA.

#3: 1A constant current test

The meter pulses the current on for two seconds, then off for about eight. This reduces heating in the constant current sink. Make sure your meter is not on a low-current range when you connect it. Adjust the value displayed until it is within 1mA of your meter's reading.

#4: Measure 3.3VA

This voltage defines the full-scale value for the ADC and affects all measurements. Measure the voltage between ground and the 3.3VA rail at TP8. Enter this into the meter using the up and down buttons.

#5: Null capacitance

Leave the DUT terminals open circuit for this stage. This measures the internal minimum capacitance and uses it to correct low readings. In our prototypes, the minimum measurable capacitance was around 20nF, and we achieved reasonable performance for values of 100nF and above. This is a power device tester, and does not seek to measure low-value capacitors.

Once this is all done, it stores the new calibration factors in EEPROM, and you are ready to start testing!

Putting it in the case

We housed our tester in an Altronics H0310 ABS box. The board mounts on the lid, with onboard buttons and switches passing through holes in that lid. The specified switches all have the same height, so provided you make holes in the lid that all align with the switches, this provides a very neat mounting arrangement.

We have always struggled with mounting 16×2 LCDs as they don't generally come with a bezel. Therefore, we designed a bezel that can be 3D-printed to match the Altronics Z7018 LCD. You can download the STL file from **siliconchip.au/Shop/6/605**

If you use a different LCD screen, you might want to design a similar bezel to match yours, as it makes assembly easier and neater.

Fig.12 shows the front panel/lid cutouts and drilling details, while Fig.13 (overleaf) shows the drilling required for the side of the case. The Kelvin probe connectors mount on the side; we used banana sockets, allowing us to use Kelvin probes when we want to measure really low resistances. You don't need to use them for normal inductor and capacitor tests.

We also installed BNC connectors so that we could use an oscilloscope to monitor the current waveform – see the photo below. These are optional.

You do need to mount a power socket. This meter needs a minimum of 12V. We selected a socket that matched our power supply; there are many options. We chose a convenient spot on the side of the enclosure for this, as shown.

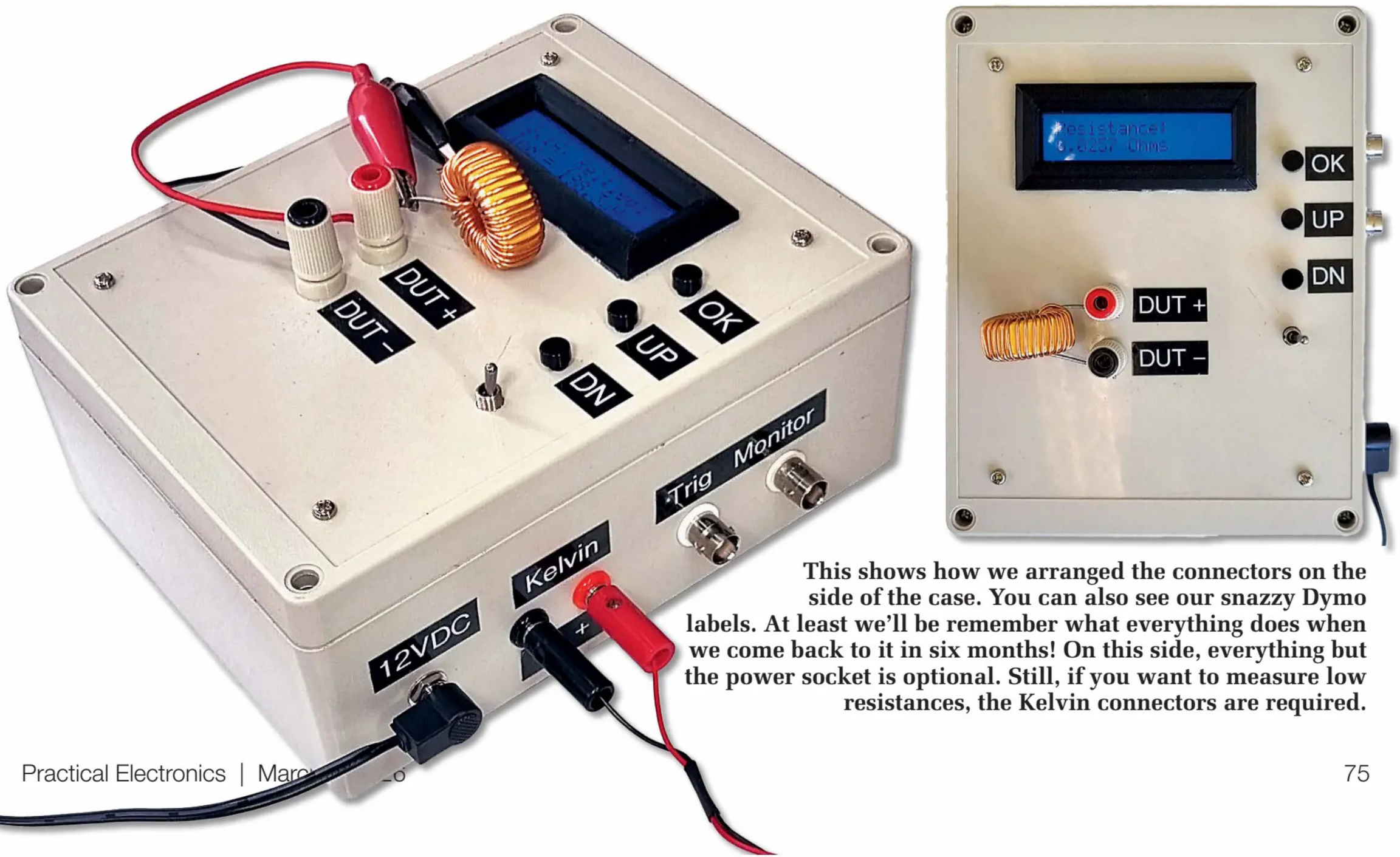
The arrangement of holes and connectors on the side is what we recommend, but you can customise this to your needs. Ensure that all holes are centred in the lower half of the case so the connectors will not interfere with the PCB.

Fit the LCD bezel to the LCD now. Test-fit it before gluing anything in place, as we have seen 16×2 LCDs in so many configurations. Make sure that yours will fit before committing to glue.

If you use the Altronics screen and our 3D-printed bezel, it should be fine. The bezel is a tight fit, so expect to jiggle the display to get it on.

If necessary, you can use a knife to scratch/trim the printed bezel. Those who have used a 3D printer will be used to this fettling process.

Glue the bezel in place with a few drops of superglue on the inside of



This shows how we arranged the connectors on the side of the case. You can also see our snazzy Dymo labels. At least we'll be remember what everything does when we come back to it in six months! On this side, everything but the power socket is optional. Still, if you want to measure low resistances, the Kelvin connectors are required.

the enclosure. Then install the LCD in the bezel and glue that in place after double-checking that you have the LCD the right way up.

The DUT screw terminals affix to the front panel and project through two matching holes in the PCB. Mount them and do them up tight; we will wire them up later.

Mount the four 10mm standoffs to the PCB using machine screw and shakeproof washers, then jiggle the PCB to get the pushbuttons through the holes in the front panel. Make sure the back of the LCD is clear of your PCB. The LCD ribbon cable comes out to the side of the PCB and will reach the header. The PCB mounts to the front panel as shown in the adjacent photo.

Now you can install 80mm of 7.5A or 10A rated wire between the DUT+ and DUT– terminals on the PCB and the red and black screw terminals. We soldered ours directly to the PCB to minimise resistance, but the board accepts 6.3mm spade terminals and you could crimp 6.3mm spade lugs to these wires. If doing that, make sure the connections are nice and tight.

We need to make provision for Kelvin connections required for measuring low resistances accurately. These connect to the PCB via CON1. We simply ran two 150mm wires to banana sockets on the side of the case.

For monitoring the operation via an oscilloscope, we recommend mounting two BNC sockets. One connects to CON11 and provides a trigger signal, while the other goes to CON7 for current monitoring. We used 150mm ribbon cable offcuts to wires these up. We put these on the side of our case next to the Kelvin connectors as we don't use them much and that was where there is room.

These oscilloscope connections are optional but present some interesting data. You need a digital 'scope set to single-shot mode to capture the data. Set the trigger level to 1V.

The vertical scale of the current curve is 100mV per amp. Most pulses are pretty short; for low-value inductors, they are in the 10s of microseconds. Large inductors can be tested over a few milliseconds.

If you look at the waveforms presented last month, you will see that inductor current curves are almost never straight. Where there is a reasonably high DC resistance but no saturation, they curve downwards, while if the inductor saturates, they curve upwards.

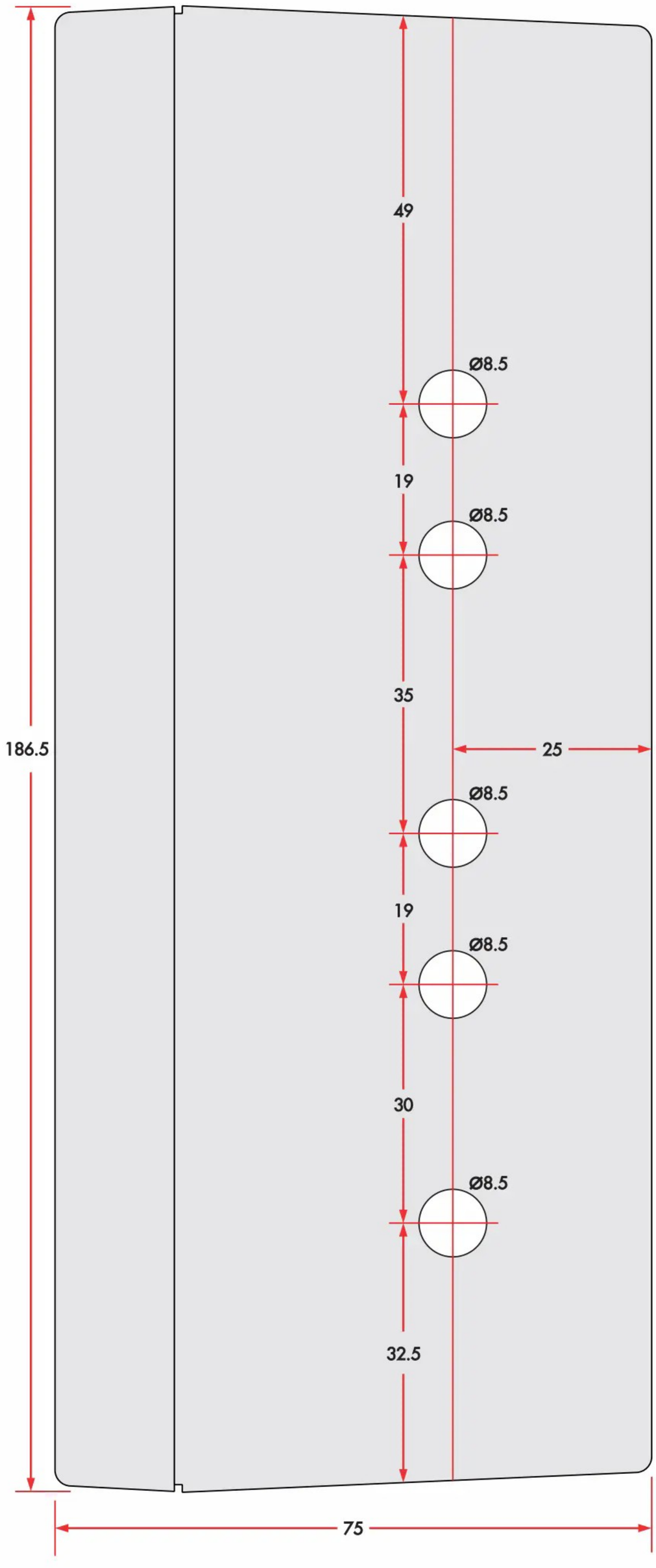


Fig.13: this is how we arranged the connectors on the side of the case. You might decide to leave some of these out so verify which connectors you actually need before drilling the holes.

The front and side panels are shown opposite. The front panel is shown at 40% actual size, while the side panel is at full size. You can download both of them from siliconchip.com.au/Shop/11/1832

Usage hints

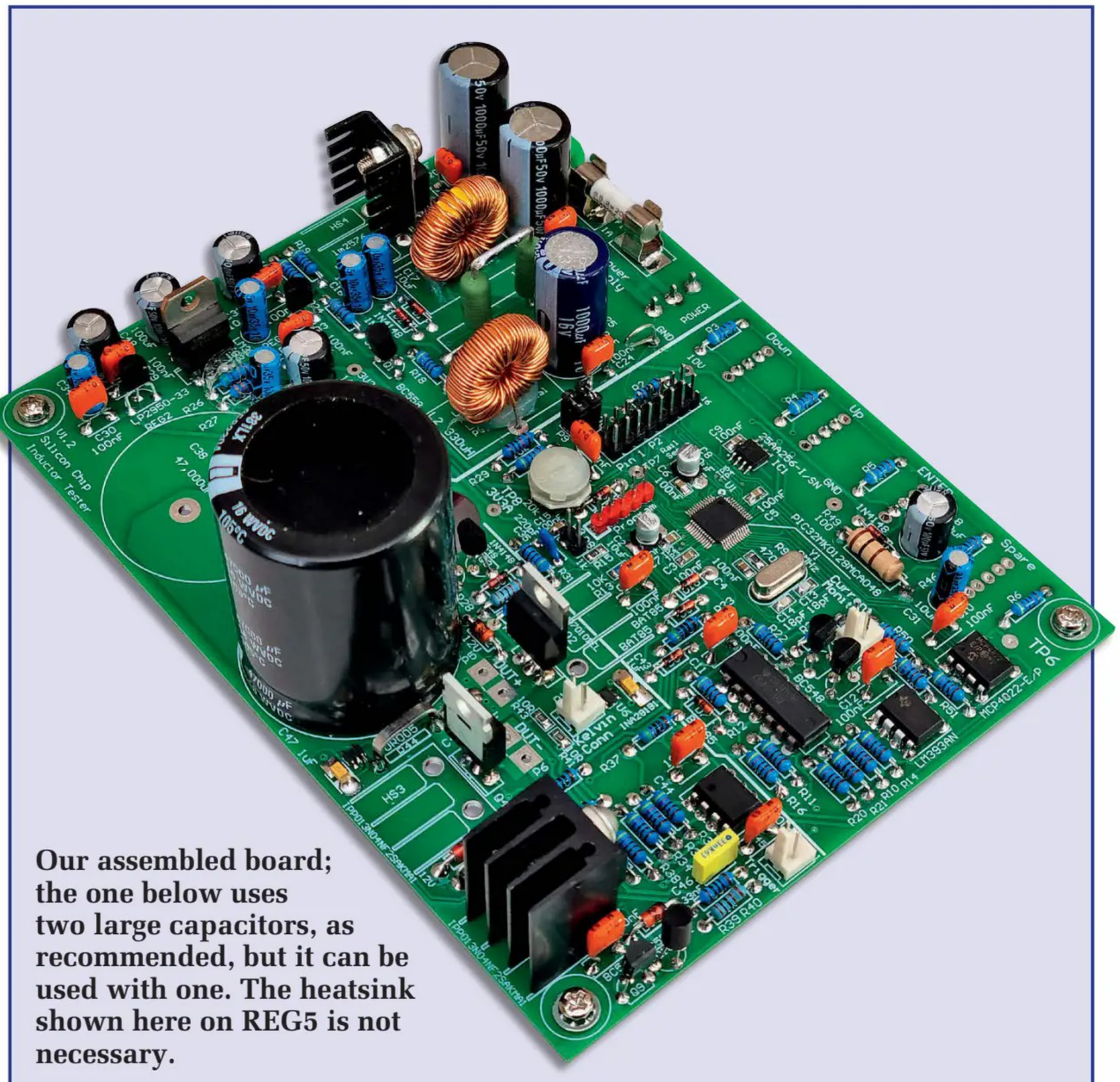
Never use this meter to test components in circuit. The currents it drives may destroy something.

Never connect this meter to powered circuitry. We have protection for inductor back-EMF, but if the input is driven above the 10V rail, you will damage the Meter.

Always discharge capacitors before connecting them – if they hold a charge above 10V, you might damage the Meter.

You don't need to use the Kelvin connections for anything but low resistances. If you want to measure resistances below a few ohms, you really should use them. With these, you can measure right down into the milliohm region. PE

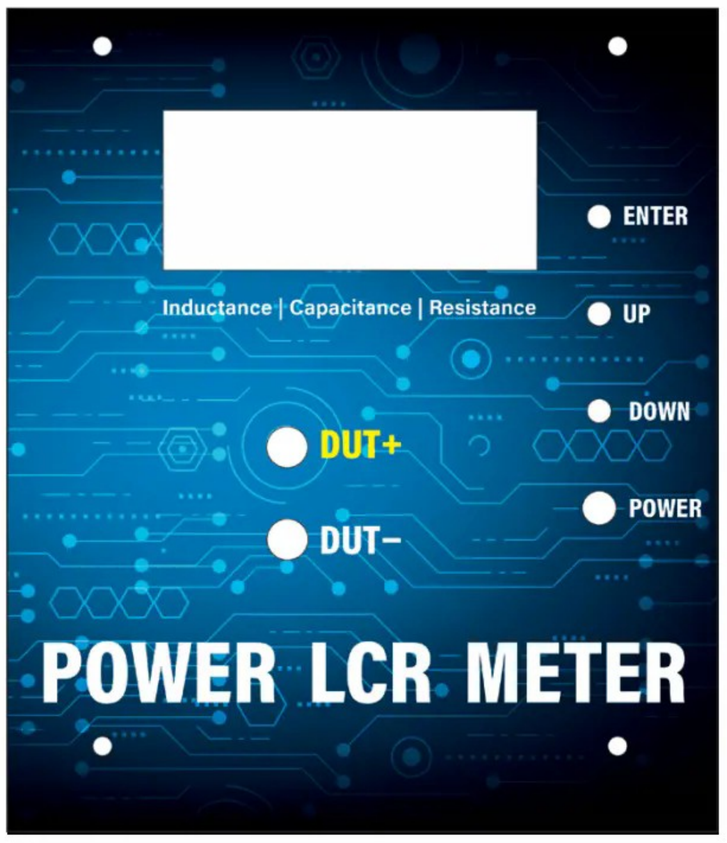
Power LCR Tester, part two

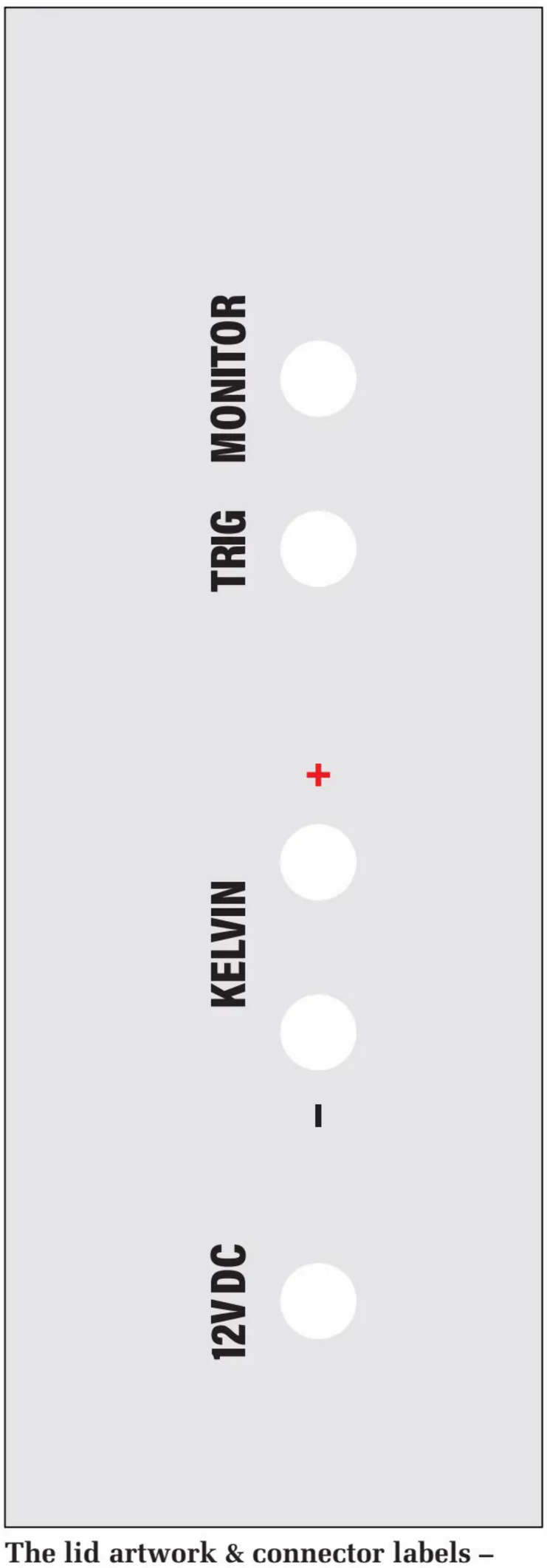


Our assembled board; the one below uses two large capacitors, as recommended, but it can be used with one. The heatsink shown here on REG5 is not necessary.



This shows the PCB mounted to the inside of the case lid, with the LCD ribbon cable in place.





The lid artwork & connector labels – see the Fig.13 caption for details.

Double-sided | plated-through holes | solder mask

PROJECT	CODE	PRICE
(continued from page 69...)		
AUGUST 2024		
Arduino ESR Meter combined L/C-ESR	04106182	£9.99
Arduino ESR Meter separate ESR	04106183	£7.99
Watering System Controller	15110231	£11.99
JULY 2024		
Dynamic NFC Tag (22 × 31mm)	06101231	£5.95
Dynamic NFC Tag (86 × 54mm)	06101232	£7.95
Reciprocal Frequency Counter	CSE230101C	£8.95
Reciprocal Frequency Counter front panel (black mask)	CSE230102	£7.95
Pi Pico-based Thermal Camera	04105231	£6.95
JUNE 2024		
Loudspeaker Test Jig	04106231	£12.95
MAY 2024		
GPS-Disciplined Oscillator	04103231	£12.95
Dual RF Amplifier	CSE220602A	£5.95
Songbird	08103231	£7.95
APRIL 2024		
Basic RF Signal Generator	CSE221001	£8.99
Basic RF Signal Generator front panel (black solder mask)	CSE220902B	£9.99
Silicon Chirp Cricket	08101231	£7.99
Test Bench ‘Swiss Army Knife’	04110221	£9.99
MARCH 2024		
Digital Volume Control Potentiometer SMD version	01101231	£8.95
Digital Volume Control Potentiometer through-hole version	01101232	£9.95
Advanced SMD Test Tweezers body	04106221	£6.95
Advanced SMD Test Tweezers legs (pair)	04106212	£7.95
FEBRUARY 2024		
Active Mains Soft Starter	10110221	£10.95
JANUARY 2024		
Q Meter	CSE220701	£8.95
Q Meter front panel (black solder mask)	CSE220704	£7.95
Raspberry Pi Pico W BackPack	07101221	£8.95
DECEMBER 2023		
Digital Boost Regulator	24110224	£8.95
Dual-Channel Power Supply for Breadboards	04112221	£11.95
Display Adaptor for the Breadboard PSU	04112222	£9.95
NOVEMBER 2023		
LC Meter Mk3	CSE220503C	£9.95
DC Supply Filter for vehicles	08108221	£8.95
Discrete Audio Op Amp PCB	AO1-JUL23	£9.95
Discrete Audio Op Amp PCB + essential components	AO2-JUL23	£17.95
OCTOBER 2023		
Buck/Boost Charger Adaptor	14108221	£9.95
PIC Breakout Board for SOIC parts	24110225	£5.95
PIC Breakout Board for DIP parts	24110222	£5.95
AVR64DD32 Breakout board	24110223	£5.95
Automatic Train Controller	09109221	£6.95
Chuff Sound module	09109222	£6.95
SEPTEMBER 2023		
Mini LED Driver	16106221	£7.95
New GPS-Synchronised Clock	19109221	£9.95

PROJECT	CODE	PRICE
AUGUST 2023		
110dB RF Attenuator	CSE211003	£9.95
Universal Battery Charge Controller (2023 update)	14107192	£9.95
Wide-Range OhmMeter	04109221	£12.95
JULY 2023		
Multimeter Checker/Calibrator	04107221	£10.95
JUNE 2023		
Arduino Programmable Load	04105221	£8.95
Buck-Boost LED Driver	16103221	£8.95
MAY 2023		
Precision AM-FM DDS Signal Generator	CSE211002	£14.95
Improved SMD Test Tweezers programmed PIC	0410621PIC	£12.95
AO 2x-dual-to-quad through-hole (pack of five)	AO1-MAY23	£6.95
AO 2x-dual-to-quad SMD (pack of five)	AO2-MAY23	£6.95
AO 2x-single-to-dual through-hole (pack of five)	AO3-MAY23	£6.95
AO 2x-single-to-dual SMD (pack of five)	AO4-MAY23	£6.95
APRIL 2023		
500W Amplifier Module	see p.22, April 2023	
Clipping Indicator (per channel)	01112211	£7.95
CD Welder Power Supply (one needed)	29103221	£9.95
CD Welder Controller (one needed)	29103222	£9.95
CD Welder Energy Storage module (several needed)	29103223	£7.95

For the many earlier PCBs that we stock please see the PE website: www.electronpublishing.com

PE/EPE PCB SERVICE

Order Code Project Quantity Price

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Name . . . . . .

Address . . . . . .

. . . . . .

Tel . . . . . .

Email . . . . . .

Card No . . . . . .

Valid From . . . . . Expiry Date . . . . . .

Card Security No . . . . . MasterCard VISA

You can also order PCBs by phone, email or via the shop on our website: www.electronpublishing.com

No need to cut your issue – a copy of this form is just as good!

Silicon Chip kits & parts for projects in this issue (posted from Australia):

USB-Programmable Frequency Divider kit	https://siliconchip.au/Shop/20/6959	AUD $72 (about £36.00) + P&P
Rotating Light kit (TH LED version)	https://siliconchip.au/Shop/20/7463	AUD $24 (about £12.00) + P&P
Rotating Light kit (SMD LED version)	https://siliconchip.au/Shop/20/7462	AUD $24 (about £12.00) + P&P

20% VAT is included in the above price (it will be removed for customers outside the UK). Postage & packing cost is typically around £12-15.
Please allow 1-2 weeks for delivery (longer in some cases).

CLASSIFIED ADVERTISING

Practical Electronics

If you want your advertisements to be seen by the largest readership at the most economical price then our classified page offers excellent value.

The rate for semi-display space is £10 per centimetre high, with a minimum height of 2·5cm. All semi-display advertisements have a width of 5.5cm. The prepaid rate for classified adverts is 40p per word (minimum 12 words).

For rates, further information on display and classified advertising or to book an ad, please email us at **pe@pemag.au**

Practical Electronics reaches more UK readers than any other UK monthly hobby electronics magazine. Our sales figures prove it.

We have been the leading monthly magazine in this market for more than thirty years.

W5SWL Electronics

Technical Products and Small Component Parts, RF and Audio Connectors, Hobby Electronics, Ham Radio to Microwave Products, General Hobby Supplies, Technical Books, Tools, and More. **Located in the United States.**

Visit **www.W5SWL.com**

BOWOOD ELECTRONICS LTD

Suppliers of Electronic Components

www.bowood-electronics.co.uk

Unit 10, Boythorpe Business Park, Dock Walk, Chesterfield, Derbyshire S40 2QR. Sales: 01246 200 222

Send large letter stamp for Catalogue

COAST ELECTRONICS

BREAKOUTS-COMPONENTS-CONTRACT DESIGN-3D PRINTER PARTS-MUSICAL-MICROCONTROLLERS

WWW.COASTELECTRONICS.CO.UK

MISCELLANEOUS

VALVES AND ALLIED COMPONENTS? For free stock list and/or advice, please contact me: geoffdavies337@gmail.com Telephone: 01788 574774

PIC DEVELOPMENT KITS, DTMF kits and modules, CTCSS Encoder and Decoder/Display kits. Visit www.cstech.co.uk

Electrical Industries Charity (EIC)
We help people working in the electrical, electronics and energy community as well as their family members and retirees.

We use workplace programmes that give the industry access to financial grants and a comprehensive range of free and confidential services.

www.electricalcharity.org

electrical industries charity

ADVERTISING INDEX

AOSHOP 10
ENIGMA COMPONENT SHOP. 9
HAMMOND MANUFACTURING. 31
MICROCHIP TECHNOLOGY . . . . . Outer back cover
PCBWAY. . . . . Inner front cover
PDF BACK ISSUES 37, 41, 55
SILICON CHIP MAGAZINE. 47
TAG CONNECT 9

Advertising mailing address

Electron Publishing Australia
PO Box 194
Matraville NSW 2036
Australia

Email pe@pemag.au

Web www.electronpublishing.com

For editorial contact details, see page 2.

Next Month – in the April issue



Power Rail Probe

A low-cost device that help you examine the ripple and noise on power supply rails with an oscilloscope.



Versatile Battery Checker

Check a battery's internal resistance and estimate its current delivery capability.

Pico 2 Audio Analyser

Generates test signals and measures distortion using a Rasyberry Pi Pico 2 module.



On sale 5th of March, 2026

Content may be subject to change

Practical Electronics

Subscriptions hotline

To subscribe or renew online, paying by credit/debit card or PayPal, please visit our shop at:
www.electronpublishing.com

For those wishing to pay by direct debit, cheque or over the telephone, please call 'Select Publisher Services' on 01202 087631, or send a cheque (payable to: '**Select Publisher Services Ltd**') with your details to:

Practical Electronics Subscriptions, PO Box 6337, Bournemouth BH1 9EH, United Kingdom

Remember, we print the date of the last issue of your current subscription in a box on the address sheet that comes with your copy.

Digital subscribers, please call 07918 614662 or visit: www.electronpublishing.com

Published on approximately the first Thursday of each month by Silicon Chip Publications, PO Box 194 Matraville NSW 2036 Australia. Printed in England by The Manson Group, Reynolds House, 8 Porters Wood, Valley Road Industrial Estate, St Albans AL3 6PZ. Distributed by Seymour, 86 Newman St, London W1T 3EX. Subscriptions UK: £33.99 (6mo); £59.99 (12mo.); £114.99 (2yrs). EUROPE: airmail service, £38.99 (6mo); £69.99 (12mo); £129.99 (2yrs). REST OF THE WORLD: airmail service, £46.99 (6mo); £84.99 (12mo); £164.99 (2yrs). Make payments to 'Practical Electronics', Practical Electronics Subscriptions, PO Box 6337, Bournemouth BH1 9EH United Kingdom. Email: pesubs@selectps.com. PRACTICAL ELECTRONICS is sold subject to the conditions that it shall not, without the written consent of the Publishers first having been given, be lent, resold, hired out or otherwise disposed of by way of Trade at more than the recommended selling price on the cover, and that it shall not be lent, resold, hired out or otherwise disposed of in a mutilated condition or in any unauthorised cover by way of Trade or affixed to or as part of any publication or advertising, literary or pictorial matter.

Did you know our online shop now sells the current issue of *PE* for £5.99 inc. p&p?



You read that right! We now sell the current issue of your favourite electronics magazine for exactly the same price as in the High Street, but we deliver it straight to your door – and for UK addresses we pay the postage. No need to journey into town to queue outside the newsagent. Just go to our website, set up an account in 30 seconds, order your magazine and we'll do the rest.

www.electronpublishing.com



Versatile Arm® Cortex®-M0+ MCUs

Delivering Efficiency, Reliability and Touch Innovation

Unlock new possibilities in IoT, consumer, industrial and automotive applications with the PIC32CM GV00 and PIC32CM JH00 microcontroller families. Built on the efficient Arm® Cortex®-M0+ core with up to 48 MHz performance, these MCUs deliver an ideal balance of performance, ultra-low power operation and advanced integration, tailored to meet the demands of today's connected world.

The PIC32CM16/32 GV00 MCU is engineered for sensor-rich, energy-conscious devices, combining ultra-low power operation (operating from just 1.62V to 3.63V) with market-leading features like a 256-channel capacitive touch controller, precision analog and flexible connectivity. This makes it an excellent choice for IoT and consumer products where responsiveness and efficiency are paramount.

For environments where reliability and stability are critical, the PIC32CM32/64 JH00 MCU stands out with robust 5V operation, strong analog integration and Driven Shield+ enhanced touch technology. This ensures dependable capacitive touch performance, even with wet surfaces, or high-interference conditions, making it the trusted choice for industrial controls, automotive systems and advanced home appliances.

Whether you're developing compact, cost-sensitive devices or demanding applications in harsh environments, the PIC32CM GV00 and PIC32CM JH00 MCU families deliver the features, reliability and performance you need to compete and succeed.



MICROCHIP

microchip.com/pic32cm-gv-jh

The Microchip name and logo and the Microchip logo are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries. All other trademarks are the property of their registered owners.
© 2025 Microchip Technology Inc. All rights reserved.
MEC2634A-UK-12-25