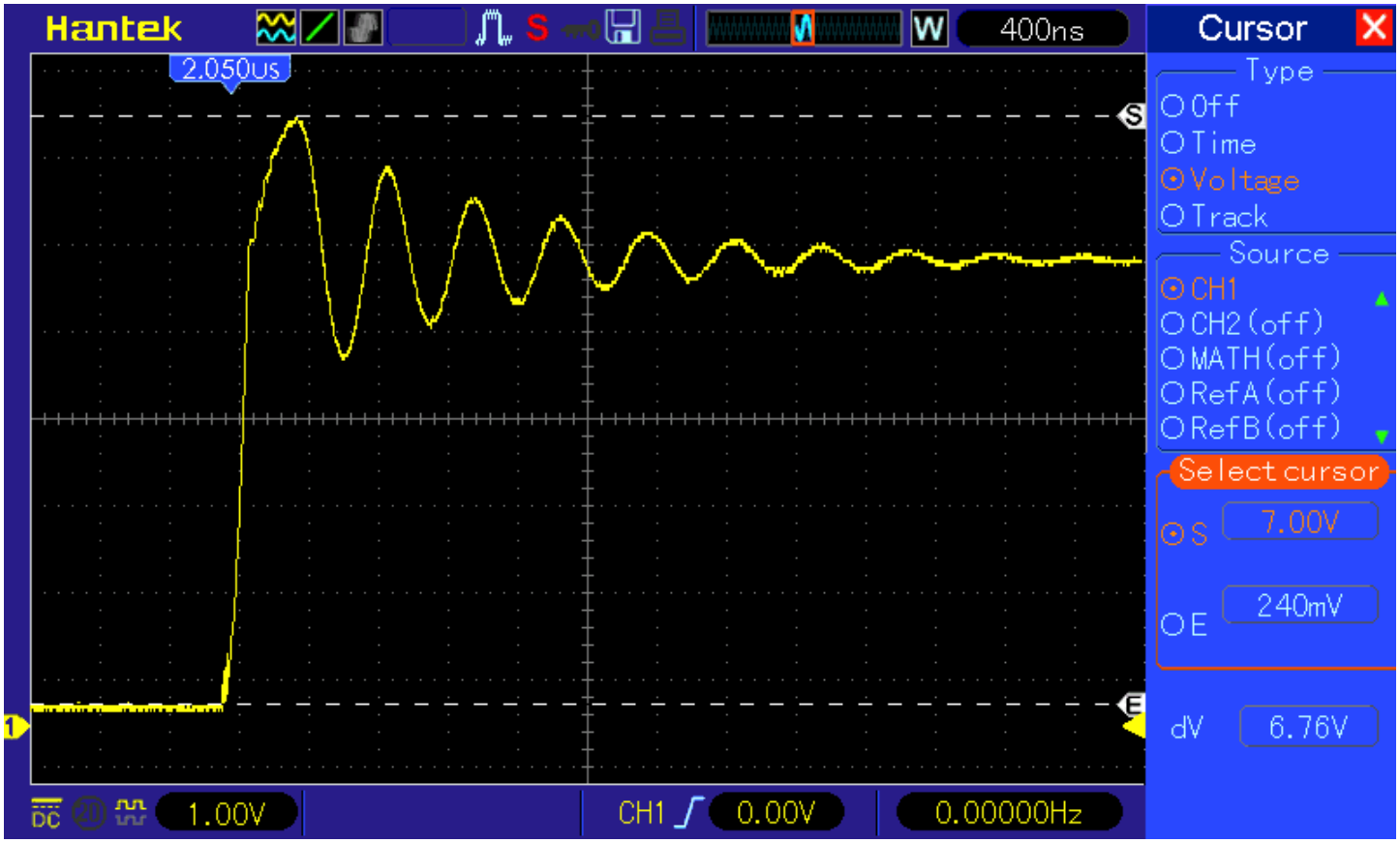
Post Publication Notes

So-called dodgy plug-packs

In the articles, I mentioned that I found that one of my plug-packs was dodgy because of poor regulation. It would trip the over-voltage protection circuit almost every time I connected it and turned it on. I also mentioned that I worked around this by paralleling a spare 10,000uF electrolytic capacitor across the Phoenix (green) power plug that mates with the clock's main board.

Most of my 5V plug-packs are OK but I've since discovered another 5V plug-pack that also causes the over-voltage protection to trip. When put on the oscilloscope, I see transients exceeding 6.5V (as shown in the CRO screen grab) when this plug-pack is connected to the clock, even if it has already been powered on and outputting what appears to be a good 5V level before connection! Its these instantaneous transients that are tripping the protection circuit so I am glad I designed it in.



Only two of my plug-packs tripped the clock over-voltage protection circuit, but given that I did find those two, I now strongly expect some of you will also encounter the same problem. I diagnosed the problem with an oscilloscope, but even if you've just got a multimeter to measure plug-pack output voltage, you can confidently diagnose it. You'll know you're experiencing the issue if you can see what appears to be a good 5V level before **and after** you have connected the supply to your clock, but you **also see** that the clock's over-voltage protection circuit trips when you connect it.

If you encounter this, grab yourself an electrolytic capacitor, say 470uF or higher and screw the + and – legs of the capacitor into the back of the clock's Phoenix plug along with the same polarities of your power supply wires. In the case of my plug-pack, 470uF turned out to be sufficient. If 470uF is not enough capacitance in your case, increase the value until it reliably eliminates these transients and your clock powers up every single time.

Given that I've now found a couple of these plug-packs, I was wondering if there may be a different explanation than poor regulation as I'd assumed in the original articles.

Over the years, I have used many plug-packs and have noticed that some use a twisted pair power cable and others coaxial. I've seen many different conductor weights in these cables and I've never thought much about it until now. The

different conductor arrangements in the power cables lead to varying levels of self-inductance (in the case of twisted pairs) or mutual inductance (in the case of coax). I won't go into the maths other than to say that the inductance of the power lead could potentially amount to (of the order of) hundreds of nH where as ideally, you'd want a power lead to have low inductance. I guess ultimately this is another example of getting what you pay for.

I'm now thinking that when the clock and its Pi power up, there will be an inrush of current to charge their distributed power-bypass capacitors. As the capacitors reach their nominal maximum voltage, the current will diminish and then settle down to the normal operating current. The initial current inrush will cause energy to be stored in the inductance of the plug-pack's power lead. The greater the inductance of the plug-pack power lead, the greater the stored energy will be. As the in-rush current settles down, this energy stored in the cable's inductance will present itself as overshoot and ringing at the clock's power input terminals.

The bottom line is that you should probably factor on including an electrolytic capacitor at the clock's power input.

Starting the media player when a Bluetooth device is connected

You can connect a bluetooth device to the clock at any time and stream audio from your device. The bluetooth device will remain connected to the clock even when the audio stream stops so you can play another stream without having to reconnect your device again.

However if at the same time you are streaming bluetooth audio you start a media player stream on the clock, including when an alarm trips and plays a media source other than the radio, the clock will disconnect from any currently connected bluetooth devices so that it can play the media stream correctly. Afterwards, you'll need to reconnect your bluetooth device to the clock in order to stream using bluetooth once more.

Alternate analogue audio devices

By default, the clock is configured to use the Raspberry Pi's analogue audio output jack. If you try running this software on either a Pi5 or a PiZero2W, there is no analogue audio port. I have verified on a PiZero2W that the default audio device is set to the HDMI port and I have verified that using the Lite OS image, the clock will stream Internet radio and play local media to the audio of a connected TV. I have not however verified the use of an HDMI to analogue audio adaptor or USB to analogue audio dongle.

If your chosen audio arrangement is not auto-configured by default, you will need to configure the audio settings yourself. Firstly determine the correct configuration commands (google may help) and try them out in an SSH shell using the CLI. Then to make these configurations permanent, edit the commands into the clock's audio configuration file at /usr/local/bin/set-clock-audio.sh.

When wi-fi goes down

If your clock is in a marginal wi-fi reception area of your home, or if the wi-fi LAN in your home goes down (or the fixed Ethernet LAN goes down if you're using that), then the time display will blink intermittently. The clock will still operate and will be able to play alarms that are stored on the clock's own SD-MMC file system, but if the LAN is down, the clock will be unable to stream Internet radio or play files from a network file-server.

The clock should maintain good and accurate time when the LAN is down assuming it has been up and running and connected to the Internet for a reasonable amount of time prior.

If the LAN is down for more than one week, the clock will auto-reboot. At that point in time, the clock will lose around 13 seconds (or whatever time it takes for your hardware to reboot), and this will not be recovered until an Internet connection is restored. Each further week the clock remains disconnected from the LAN, it will lose an additional 13 seconds each time it auto-reboots.

Note that you can always configure a second or subsequent backup SSID under the wi-fi settings on the web interface. If the currently connected wi-fi link goes down, the Pi will hunt for any other configured SSIDs to try and connect with one of those.

About the software

The web serving component of the project is implemented upon a well-known application called Apache and the clock is implemented in a programme called alarm-clock. All of the GPIO activity is implemented in a third programme called pigpiod (the Pi GPIO daemon). There are other support functions such as the network time keeping implemented in something called chrony, and the Linux operating system itself has many other tasks of its own such as managing the wi-fi, samba share, usb ports, other network functions and looking for time zone daylight savings updates, all of this done through what ultimately is dozens of different programmes and modules that seemingly run simultaneously.

When you ssh to the Raspberry Pi and type the following command, it displays information about each of the CPU cores within the Pi's ARM microprocessor.

```
cat /proc/cpuinfo
```

For most Linux Pis, you'll see four different cores meaning the hardware supports four simultaneous programmes running 'at full CPU speed' across its four cores. If a programme supports multi-threading, different parts of the same programme can be running at precisely the same time and if you can arrange its different tasks to take advantage of this, the programme will run faster overall than it could using just a single CPU core.

Apache is an example of a multi-threaded programme and can connect with many different browsers at exactly the same time. If you try, you'll see the clock supports this too and the fair test is to launch multiple browsers to the clock's web page simultaneously.

The alarm-clock programme is written in multi-threaded C. It integrates the media player functions by incorporating the open sourced 'mpv' media player libraries. For this reason alone, multi-threading was worthwhile and because there are many other things the clock needs to do, multi-threading seemed to be a good fit. Each of the threads are somewhat like different programmes, but you can also think of them as different subroutines running on (potentially) different CPU cores at (potentially) the same time. The sequencing of clock software events can therefore change depending upon what other things the Linux operating system is running on the different CPU cores alongside the clock. Those that are interested can explore the source code, but you can be a spectator without looking at any code at all and just watch it play out as it runs.

Hardware versions of the Pi that have been tested

Because I don't own every variant of the Pi, I've not been able to verify that the software installs and works correctly on every variant. However I've installed and tested the hardware with a couple of Pi3s, Pi4s, and a PiZero2W. I also installed and unsuccessfully tried to run the software on one of the early Pi Model Bs.

As the article was being prepared for publication, a new model Pi (the Pi5) was announced along with a new operating system release called Bookworm. Using the Pi5 is not recommended for the following reasons:

- Although the Pi5 is faster than any of the earlier hardware platforms, the additional speed is not necessary for this clock.
- The Pi5 does not support an analogue audio output.
- The lead author of the pigpio library has indicated that it may be some time before the pigpio library supports the Pi5. That's because the development team are not only waiting for the new hardware to be released, but the documentation for the new chips that will drive the GPIO pins. Its not clear when a pigpio library that supports the Pi5 will be available.

Other than on the very old Model B, the installer and the alarm-clock programme ran fine on the Pi3, Pi4 and Pi Zero 2W devices I have available to me, including working with the hardware design as published. The Model B processor is slow and could not keep up, and it crashed.

In terms of performance, the speed difference is very noticeable between the different models of Pi, most so when accessing the in-built web pages. The Pi4 is obviously the fastest, followed by the Pi3 and then PiZero2W. I suspect that the Pi2 and PiZeroW will be noticeably slower again.

If you would like to stream media, using either a Pi3 or Pi4 is recommended. If you're instead planning to only use the external audio input from an external audio device, the Pi2 or PiZero models should support your needs, or the Pi5 once the pigpiod library is released and is stable for this platform.

Timezone and daylight savings support

The clock uses a unix database called 'tzdata' to keep track of dates and times of upcoming daylight savings changes. Updates are issued to the database from time to time as various jurisdictions make changes to their upcoming daylight savings schedule. While the operating system remains in support, the unattended updates function enabled by default in the clock will fetch and implement any daylight savings changes without your intervention. After the version of operating system in your clock passes the end of its support life, those daylight savings updates will no longer be automatically applied.

The following example command will list all of the daylight savings time transitions stored in the database for the Australia/Canberra timezone. You can modify the command to reflect your local timezone if you are interested.

```
zdump -v /usr/share/zoneinfo/Australia/Canberra
```

If you run that command, you'll see there are time transitions that have been defined up until the end of the year 2499. This implies that when your clock's operating system passes the end of its support life, it will continue to make automatic daylight savings adjustments at the database's scheduled times.

If the future scheduled times change for any reason (such as law changes etc), then those future changes will not be reflected by the clock. If this concerns you, the unix timezone database is and always has been public so you should be able to google and manually update it when you need.