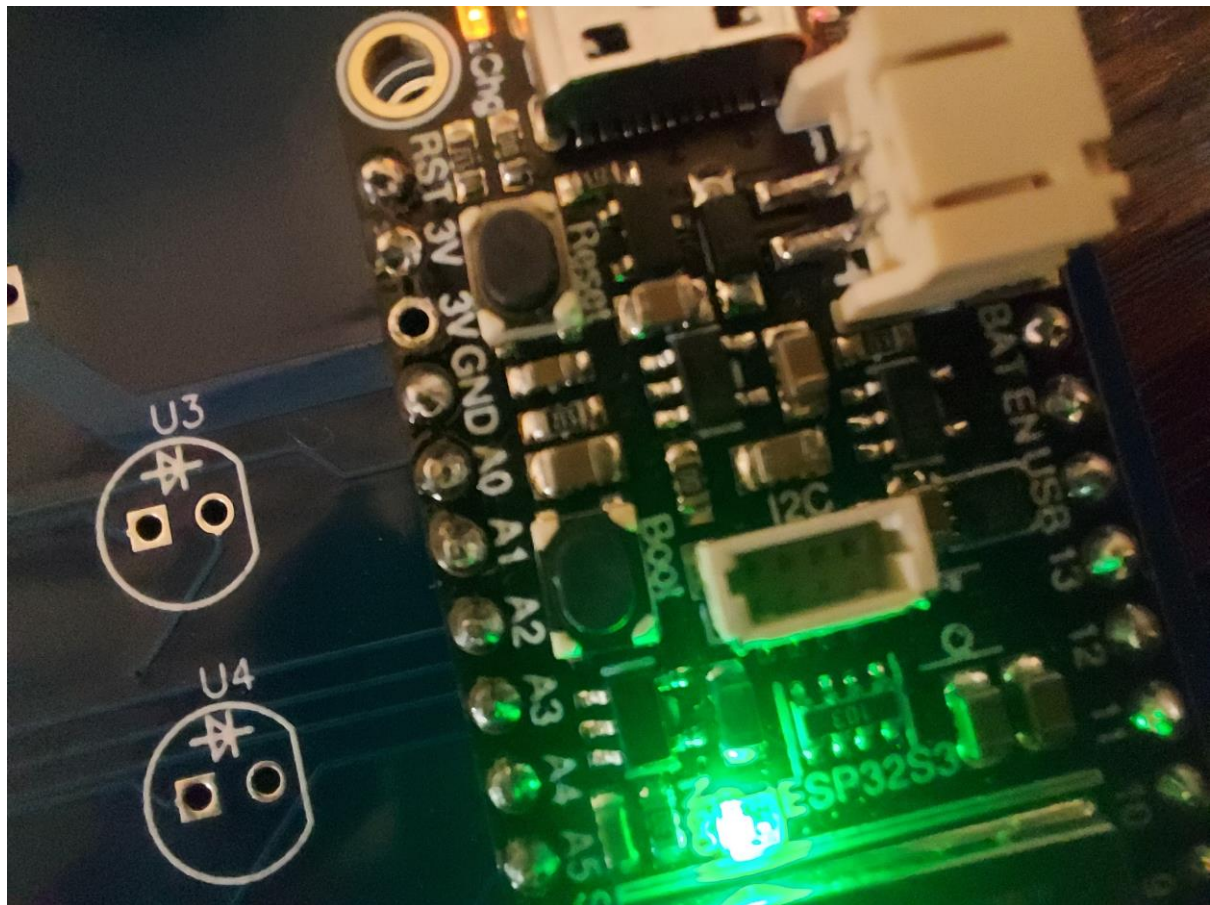


Ed's binary clock – 6th April 2024

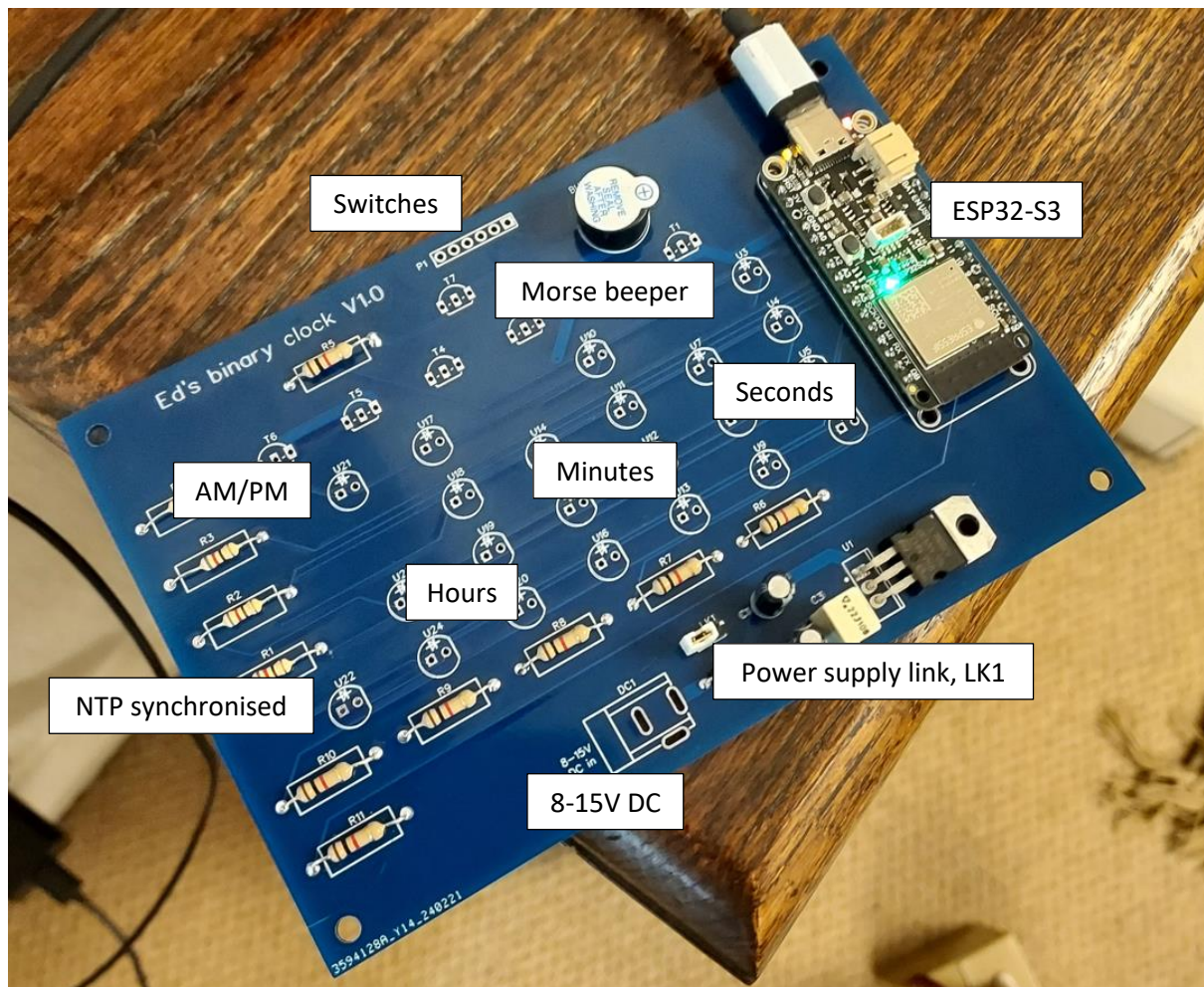
Deliberate mistakes that would be fixed if there was ever a V1.1 of the PCB

The “builtin” LED pin on the Arduino is used as one of the display column select lines so the red LED on will always be dimly lit. Should really use a different GPIO...

THIS ONE IS REALLY, REALLY IMPORTANT!!!! Pin 3 on some Feather boards is the reference for the analogue to digital converters, on this one it's joined to the output of the 3V regulator. On the PCB, this pin is connected to ground which would short out the regulator on the Arduino. Remove the pin from the header before soldering the headers to the board and the Arduino to the headers. This is the pin labelled “3V”, next to the one labelled “GND”.



The board



Parts list (see circuit diagram)

- 7x, BC337/338 transistor (more or less any low voltage NPN transistor with the same pinout will do)
- 4x, 82 ohm resistor or 120 ohm to make the display dimmer if required
- 7x, 1K resistor
- 1x, 7805 5V regulator
- 1x, 1N4001 diode
- 1x, 10uF electrolytic capacitor (anything that fits in the board, 10-100uF)
- 1x, 4.7uF electrolytic capacitor (2.2-6.8uF)
- 1x, 0.1uF (0.1-0.47uF)
- 1x, 5V piezo buzzer (make sure it's the sort that beeps when you put 5V on it, some are just piezo speakers and need a square wave to make a noise)
- 1x, Adafruit ESP32-S3

Adafruit ESP32-S3

This is the processor and WiFi module. It's based on a 32-bit, "ESP32-S3-MINI". The module has some FLASH memory, some RAM, a dual-core processor and 2.4GHz WiFi with an on-board antenna. It has more processing power than is strictly necessary but it's cheap.. The FLASH memory is used to hold my software and a small FLASH disk to store the configuration.

The yellow LED to the right of the USB connector, labelled "CHG", is part of the battery charger which isn't used. It can't be controlled by software or disabled and, when there's no battery, it does its own thing. It might be off, it might light dimly or flash, which is normal and nothing to worry about.

The red LED on the other side of the USB connector is connected to one of the column select lines used for the clock display, so it will be dimly lit all the time. This is normal and nothing to worry about.

The two buttons are used together when updating the software using the USB port. Software updates can also be done using wifi.

The one labelled "RESET" (nearest the USB connector) restarts the board. Pressing the one labelled "BOOT" (nearest the microcontroller) lights all the LEDs on the clock board.

The USB connector may be used to power the board – see below. It can also be used to display software debugging information on a connected PC, otherwise it's only used when updating the software.

Power

You can use either the USB connector on the Arduino board or an external supply. The neatest way is to use an external supply and the on-board regulator, that way all the wiring can be done from behind the board and you don't have to fit a USB connector inside the box.

If you want to use an external supply, it should be 8-15V and the link LK1 must be fitted to the PCB. There is a diode to protect against connecting it up the wrong way round. Bend the pins and mount the 7805 flat on the board. The regulator shouldn't need a heatsink.

If you want to use the USB connector, either to power the board or to re-programme the Arduino, then make sure LK1 is removed first.

If you never want to use an external supply, then you don't need either the 7805 regulator or any of the components around it. The only disadvantage of that might be the routing of the power cable – using an external supply, you can omit the power connector and solder the cables directly to the back of the board, leaving the front clear – who doesn't want to use a transparent lid on their box??

I used an XLR connector on the box for the power connector because they're big and have a latch to stop them falling out (and we have a big box of second hand ones, saved from the bin, at work!).

Switches

Connect the switches to P1 as below, pin 1 is the square pad. The other sides of all the switches go to 0V, pin 6:-

- Pin 1, toggle switch, 12 or 24 hour mode
- Pin 2, toggle switch, enable or disable hourly “time in morse code” mode
- Pin 3, not used
- Pin 4, push button, show the date instead of the time when pressed
- Pin 5, push button, send the time in morse code
- Pin 6, 0V return for all switches

If both push buttons are pressed at the same time, provided the board is connected to your wifi network, it will send the IP address of the board in morse code

If you want to use a clear lid on your box, you might want to wire the switches from the back of the board to hide the wires a bit....

If you have loads of RF swilling around, then the board will probably benefit from extra pull-up resistors on the switch wires. The ones inside the microcontroller are quite a high value, try 4.7K ones to pull the switch wires to 3V if you get spurious operation – make sure it’s to 3V and not 5V otherwise it’ll break....

Display

If you sleeve the shorter of the LED leads, then it’s much easier to get them all mounted at the same height above the board to give the best display.

If you want to change the brightness of the LEDs, change the values of resistors R1 – R4.

The box

The board is designed to be mounted, using standoffs, on the lid of a standard diecast metal box – 192x112x61mm – that’s the one with six screws in the top, like the old Microwave Modules transverters (<https://cpc.farnell.com/hammond/1590r1/box-diecast-192x111x57/dp/EN82046>). I mounted the switches on the back of the box (opposite side to the lid), with an XLR connector for the power. The box sits on its side. I drilled a couple of 3mm holes in the back to let the noise of the morse beeper out – they can be quite loud, so the hole was in the back and not over the top of the beeper on purpose... I also replaced the lid with a piece of Perspex so everyone could admire how brilliant the board inside was. It works either with the LEDs all behind the Perspex or with holes drilled and the LEDs slightly poking out. The LEDs are spaced 15mm horizontally and 10mm vertically. The drilling template/diagram shows where to make the holes – don’t take any notice of the outline of the lid on the template because there are several variations of box which are all “about” the same size. The plastic lid is also “quite important” since a diecast box with the metal lid fitted would block the wifi signal to the board quite well.... If you wanted to hide everything away, then you’ll have to use a plastic box instead. If you use a diecast box with a transparent lid, then you can change the screws for pan head ones instead of countersunk to look a bit nicer. The size for an Eddystone or Hammond box is UNC 6-32x1/2 inch although there are some Hammond ones which use 3mm – I used stainless steel ones.

Operation

When the board is powered, the multicolour LED on the Arduino will light in the sequence blue-green-red for one second each.

The LED will then go off for one second and then light up red. That turns blue until the board connects to your wifi network. Once connected, the LED turns green.

To configure the board to suit your network. Hold down either push button whilst powering on to start the configuration mode. The multicolour LED lights a pinky-purple colour whilst in configuration mode. This needs to be done only once, the configuration is saved in a file on the board. Once it exists, the configuration file is loaded each time the board is powered.

In configuration mode, the board works as an open wireless access point with the SSID "NTPClock". Connect to that with your laptop. Once connected, point your web browser at <http://192.168.1.1> to see the configuration page. Fill in the values to match your network. If you have an NTP server on your network (some routers do it...), then you can change the address of the NTP server on the page to match yours or just leave it as **ubuntu.pool.ntp.org**. You can use either an IP address or a hostname.

You probably don't need to change the hostname but if you already have something called "ntpclock.local" on your network, then you should.

The number probably don't need changing – if you set the update times to be too short (less than 20 seconds) then some Internet time sources will block you from using them.

Alternatively, remove LK1 and connect a USB cable to the Arduino and use a terminal emulator for configuration.

If want to do that and don't already have an emulator, download and install Teraterm from <https://github.com/TeraTermProject/teraterm/releases/download/v5.1/teraterm-5.1.exe>. When installing, to install just the basic terminal emulator, deselect all the options apart from Teraterm (at the top of the list). Configure Teraterm, set "New-line" in Setup->Terminal to be "Receive: AUTO". You can make the text bigger under Setup->Font. Speed is 9600 baud, which is the default for Teraterm.

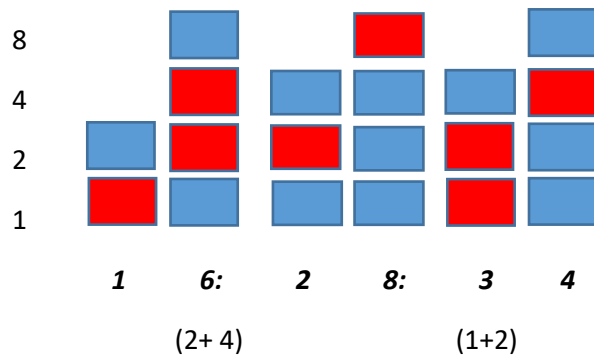
Start the command line by pressing any key and use "?" to list the available commands but don't forget to remove LK1 before connecting the USB cable...

- "load" to read currently saved configuration
- "ssid SSIDNAME" to set wifi SSID to "SSIDNAME"
- "ssid" to show configured ssid
- "password" to show/set wifi password
- "hostname" to show/set the name the board can be accessed by from your PC
- "ntpserver" to show/set NTP server to be used
- "save" to save configuration – you have to use "save" to write the configuration file!!
- "exit" to end the command interpreter and restart with new configuration

Don't format the disk or delete the files "index.html" or "favicon.ico". Deleting "config.dat" allows the configuration to be reset to the default values. If you want to do that, use "delete /config.dat", "load", change the settings and then "save".

Once setup, restart the board and it should connect to your wifi network and start telling the time. The yellow NTP sync LED (U22) will come on once the time synchronisation is stable. The rest of the LEDs show the time in binary coded decimal. In 12 hour mode, U21 lights to show PM and is off for AM, in 24 hour mode, it's always off. The rest of the left hand LEDs show the hours part of the time, the middle is the minutes and the right hand end is the seconds. The top most LED of each column is the most significant bit.

Example:-



Morse code

The characters are sent at approximately 20wpm with a much bigger than normal spacing – the board only ever sends numbers so you'll soon get up to speed and the spaces give you thinking time!

If you need to change the speed, then the software needs re-compiling. The value is defined as MORSE_DELAY in the file config.h. The default is 60ms. Much better to just learn morse code numbers fast enough 😊

The timing is as below:-

- One dot is "MORSE_DELAY"
- One dash is "3 * MORSE_DELAY"
- Space between dots/dashes in same character is "MORSE_DELAY"
- Space between characters is "5 * MORSE_DELAY"
- Space between each complete number is "15 * MORSE_DELAY"

For the hourly chime, the hours part of the time is sent in morse at the same speed but the space between the two digits is only 3 * MORSE_DELAY, so it'll sound faster than the full time or the IP address....

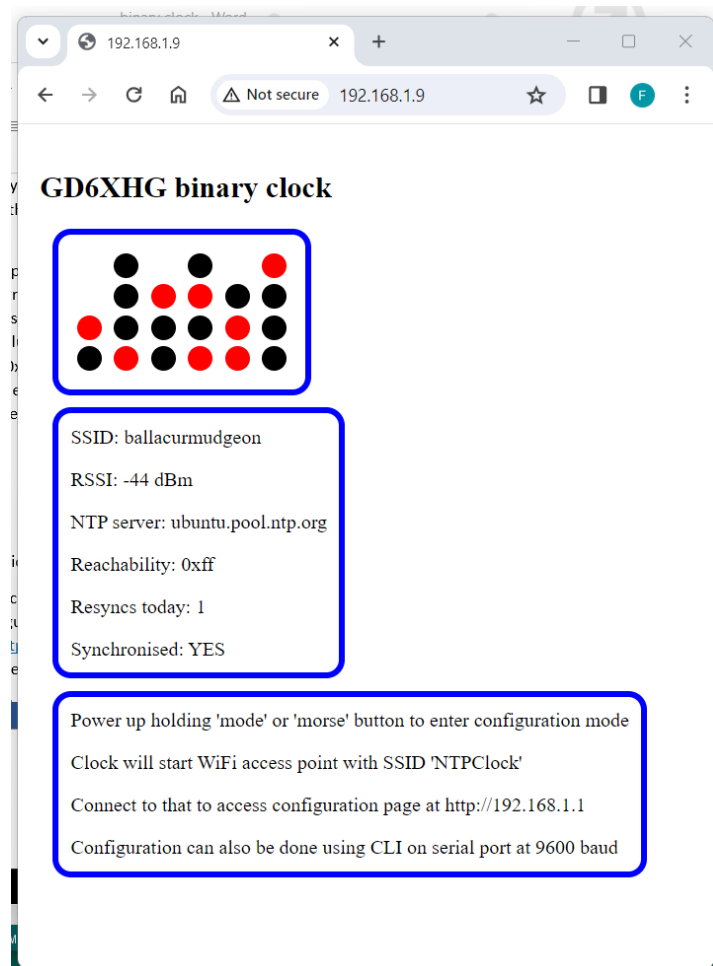
Remote access(!)

You can access the board from a computer on the same network using either “Telnet” or a web browser.

You can connect using either the hostname defined on the configuration page (for Windows/MacOS) or the IP address (if the hostname doesn’t work – eg some Linuxes)

You can find the IP address of the clock by logging in to your router or you can press both push buttons to get the clock to send its IP address in morse code. The easiest way to make that work is to first press the “date” button and then the “time in morse” button whilst holding the “date” button down. Otherwise, if you get the “time” button first and not both at exactly the same time, the board will start sending the time in morse code before the IP address which is terribly confusing....

If you use a web browser to point to the IP address of the board, you will see this continually updating “status page”.

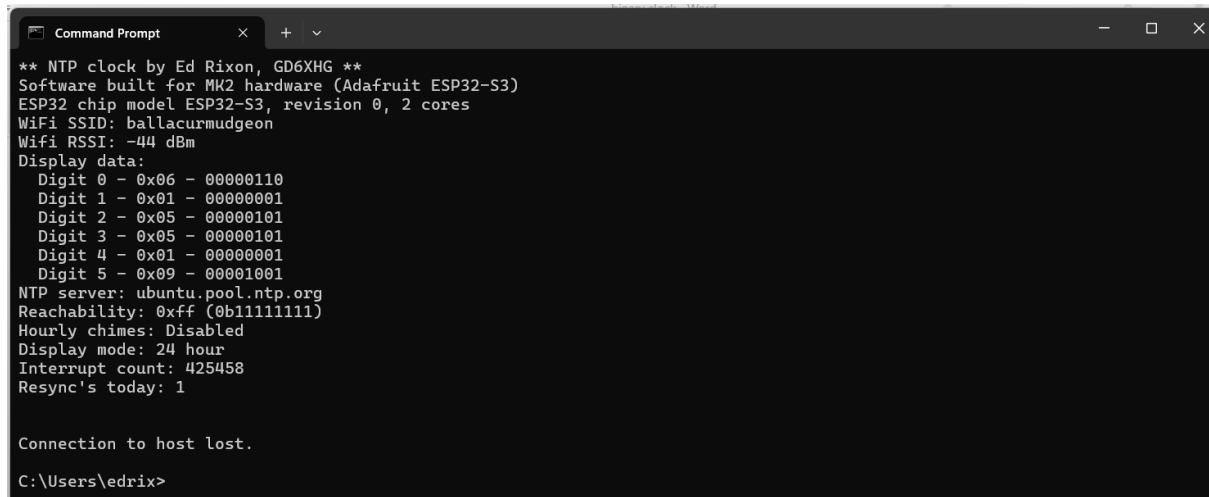


That shows which LEDs are lit, the signal strength of the wifi signal and a “reachability” value. The reachability shows how reliable the NTP server you are using is – each time the board requests the time, the least significant bit of this value is updated with a “1” if the server replies or a “0” if it doesn’t. Normally, this value will be 0xff (hexadecimal, indicating every one of the last eight time requests were responded to), any other value shows some missed replies.

For example, 0xfd means the last but one NTP time request didn't get a reply. (convert to binary and look at 1's and 0's. Hexadecimal FD = 11111101 in binary, "1" means server replied, "0" means it didn't).

One or two missed replies are OK but more than that means you probably need to use a different server or the wifi signal is a bit weak.

You can see similar information by using "telnet" from the Windows command prompt.



```
Command Prompt
** NTP clock by Ed Rixon, GD6XHG **
Software built for MK2 hardware (Adafruit ESP32-S3)
ESP32 chip model ESP32-S3, revision 0, 2 cores
WiFi SSID: ballacurmudgeon
WiFi RSSI: -44 dBm
Display data:
  Digit 0 - 0x06 - 00000110
  Digit 1 - 0x01 - 00000001
  Digit 2 - 0x05 - 00000101
  Digit 3 - 0x05 - 00000101
  Digit 4 - 0x01 - 00000001
  Digit 5 - 0x09 - 00001001
NTP server: ubuntu.pool.ntp.org
Reachability: 0xff (0b11111111)
Hourly chimes: Disabled
Display mode: 24 hour
Interrupt count: 425458
Resync's today: 1

Connection to host lost.
C:\Users\edrix>
```

The telnet page doesn't continually update and the board disconnects the session once the information has been sent.