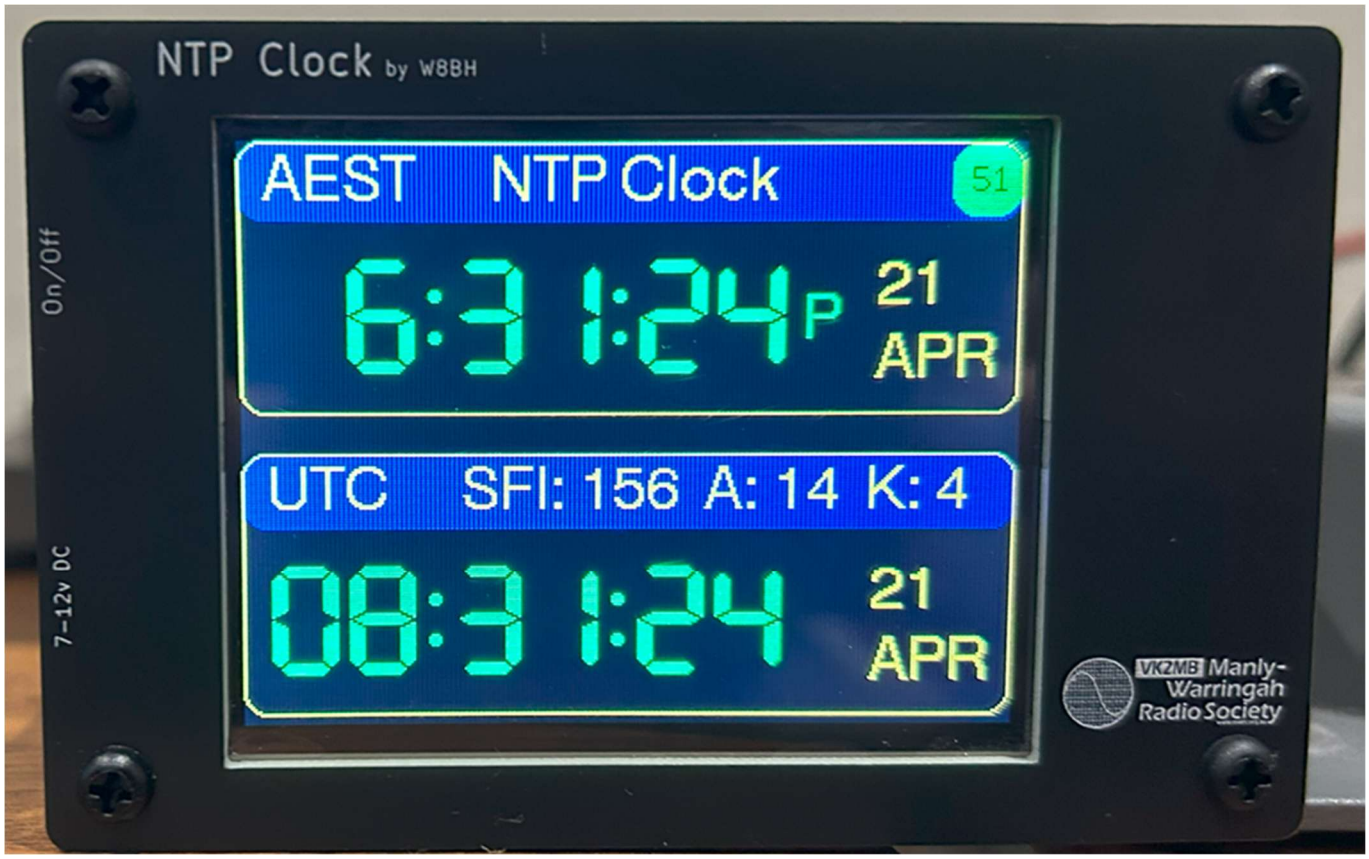
NTP Clock Kit Build Notes

VK2ARH Version 2.2



By Richard VK2ARH

The NTP Clock Build Notes, Documentation and firmware for this kit can be downloaded from:

https://github.com/VK2ARH/NTP-Clock/tree/main

Table of Contents

Introduction

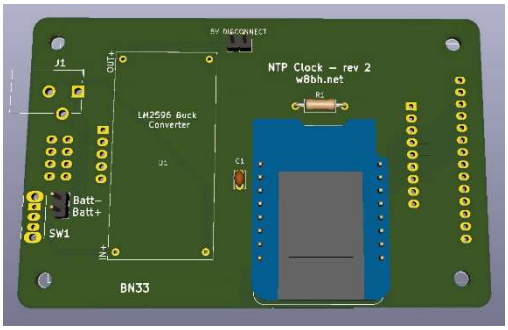
An NTP clock uses the Network Time Protocol (NTP) to display accurate date and time, which is periodically updated to ensure that the clock maintains an accurate time display. When restarted the clock following loss of power, or when a time changes resulting from the move to and from summer time, the NTP clock automatically sets and adjusts the time for your time zone, whist constantly displaying UTC.

The **Network Time Protocol (NTP)** is a networking protocol designed for clock synchronization between computer systems over packet-switched, variable-latency data networks. Developed by David L. Mills at the University of Delaware in 1981, NTP is one of the oldest Internet protocols still in use.

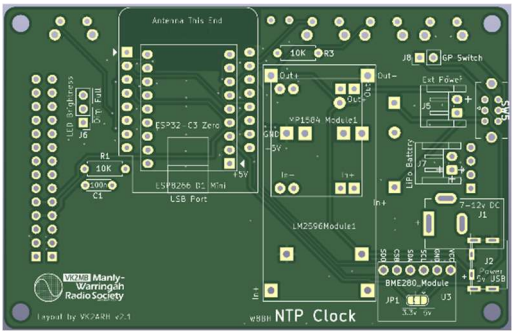
The NTP clock connects to your WiFi network, and synchronizes its time using NTP, further software enhancements have added solar weather data display, and the software continues to be developed to improve and deliver more functionality.

This project is drawn from the NTP clock developed by Bruce Hall W6BH and a wealth of information relating to this project can be found on his website : http://w8bh.net/

Bruce's design supported three TFT displays (2.2", 2.8" and 3.2"), we chose the 2.8" TFT display for the MWRS project as it was felt this provide the best compromise between price and viewing performance. The boards in this kit were developed by Richard VK2ARH and included several enhancements over the original W8BH board to support additional functionality and experimentation.



W8BH's Original Board



VK2ARH v2.1 Board

The VK2ARH board supports more components than the original W8BH board, but it provides the same basic functionality with additional optional features. This board supports the 2.8"TFT display (which comes in two variants), operation from a 5v USB-C supply (whilst enabling a choice of two buck converter modules to operate from 13.8v), 4 push button switches to support additional functionality and software developments, light dependent resistor (LDR) circuitry to support dimming the main screen, and an environmental monitor module to support temperature/humidity and pressure measurements. The option is also available to use an ESP32-C3-Zero processor. The VK2ARH PCB also has companion top and bottom panels to enable mounting the clock in a PCB Sandwich enclosure for those who don't wish to use a 3D printer or custom enclosure.

These notes were not intended to be a detailed construction manual (but have seemed to grow since originally conceived) but are supplied to direct the NTP Clock builders to the relevant documentation and provide additional information to enable a successful build using the NTP Clock V2.1 PCB.

Version 2.1 is a further developed main board enhancing the original V1 board designed for the Manly Warringah Radio Society (MWRS) build program in 2025. https://www.mwrs.org.au/ . The project comprises a 'basic kit' or a basic set of components to build a fully working NTP clock operating from a 5v USB-C power source with dimming and night mode capability, as well as providing a number of push button switches that software developers can use to deliver further functionality. The board supports additional components which can be added to support operation from 13.8v, LiPo battery backup and environmental monitoring if desired.

The design and software for this project are all open source, so that individuals and radio/electronics clubs can easily assemble their own 'kits' to build this project either individually or as a group project. The kits are not sold commercially. If you develop additional functionality, please let me know and publish it so that others can similarly leverage from the activity. My email contact details can be found in QRZ.com.

Bruce Halls original project documentation can be downloaded from his website http://w8bh.net/ or from the VK2ARH Github repository https://github.com/VK2ARH/NTP-Clock/tree/main which also contains the gerber files for the PCB's used in this project. Look for the following documents:

NTP_DualClock.pdf – Provides the background to the operation and programming of the firmware for the W8BH NTP Clock.

NTP_clock_build Instructions.pdf – A detailed construction manual of how to build the NTP clock using Bruce's original PCB provided for the project. NOTE: These are different to the MWRS PCB's covered in this document.

CalQRP_Club_-_NTP_Clock_ReadMeFirst 2025-04-04.pdf This document provides a series of guidelines provided for the California QRP Club NTP clock project in April 25 which used Bruce Hall's original project PCB's. These guidelines relate to configuring the time zone for the USA. The full list of parameters for use in other time zones can be found here: https://github.com/nayarsystems/posix_tz_db/blob/master/zones.csv

How to install the ESP8266 Board into Arduino IDE.pdf The title is self-explanatory. You will be uploading the firmware for this project to a ESP8266 WEMOS D1 mini microcontroller, and this document shows you how to set up the Arduino IDE environment to work with this board.

Software

The Original NTP Clock software has been enhanced as follows:

- Robert Kincaid AI6P added solar indices to the UTC display header after downloading solar data from the 'hamsql.com' website run by Paul Herrman N0NBH. These are updated approximately every 30 minutes with data pulled down from Paul's solar weather server.
- Simon VK2YU added WiFi Manager to better manage Wi-Fi connections and allow screen-based input from your cell phone rather than hard coding the wifi credentials into the NTP Clock firmware.
- Mark KD5RXT developed code to support the dimming functionality along with the 'Night Mode' display.
- AI6P and WA2FZM have further refined the code used by the CalQRP group which provides an extended range of sola data display, restructured the code and provided a permanent server certificate to access the solar data (this has also been incorporated into the MWRS code), but does not yet include the dimming, night mode or WiFi manager functions. This code can be found here:

  https://github.com/WA2FZW/W8BH-NTP-Clock-Revisited-by-WA2FZW-and-AI6P

As software continues to develop it is likely that features of both versions of the software may be incorporated into a single version, but this is yet to be undertaken. Discussion in the rest of this document relates to the MWRS version of the software.

The software will need to be configured with your callsign (or any other header you wish to display) and the local time zone if you wish to operate the clock in any location other that Australian Eastern Standard Time (Sydney).

Details of how to change the code to support other time zones can be found in the CalQRP_Club-NTP Clock_ReadMeFirst 2025-04-04.pdf document and a full list of global time zone 'code string' can be found here:

https://github.com/nayarsystems/posix_tz_db/blob/master/zones.csv

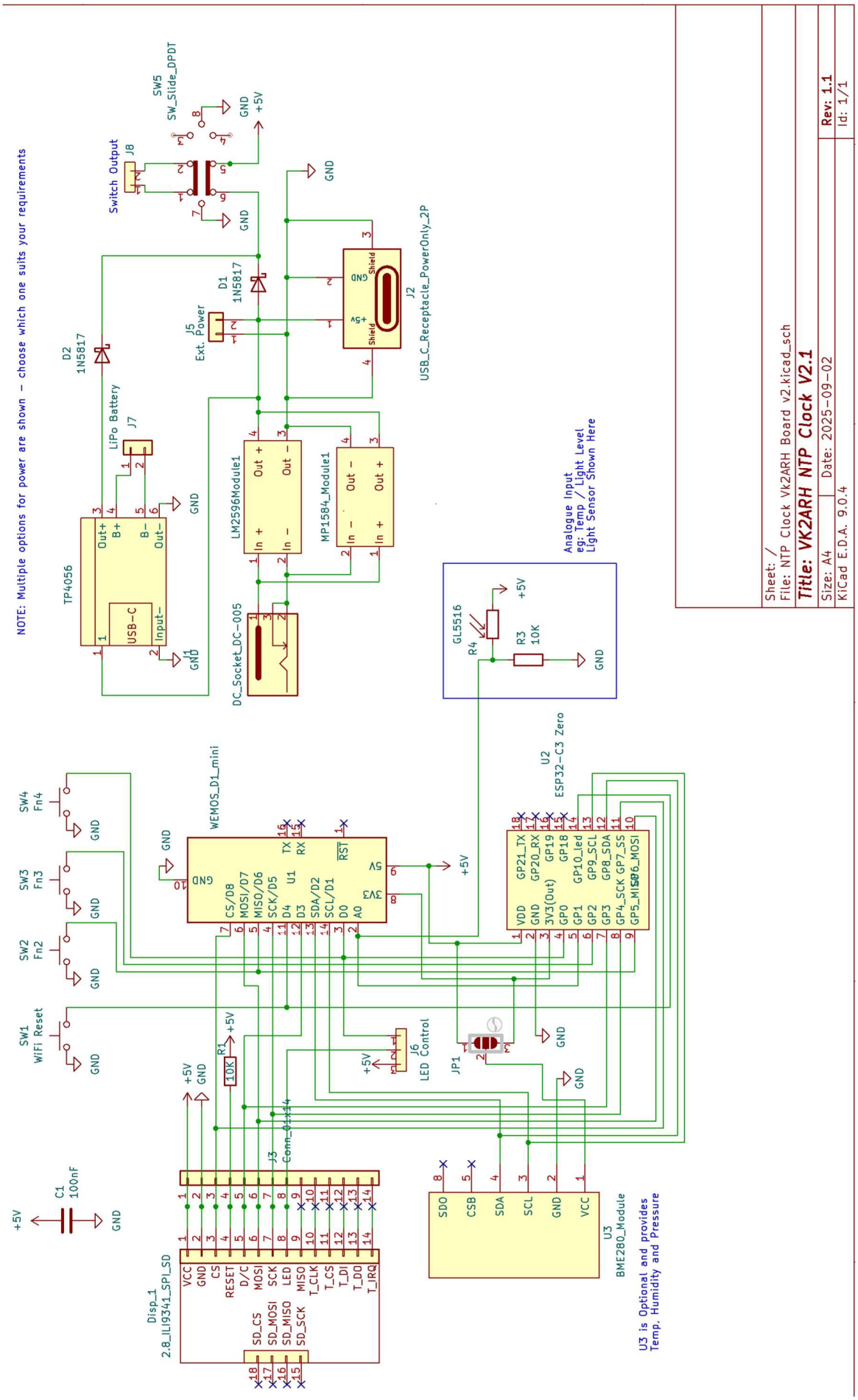
Further instructions to assist uploading the software to the ESP8266 microcontroller can be found later in this document.

Bill of Materials – NTP Clock

Part	Qty
Main Board	1
Top and Bottom Cover	2
2.8" LCD Display	1
ESP8266 Wemos D1 Mini	1
DPDT Slide Switch SK22D02	1
100nF MLCC Capacitor	1
10k Resistor	2
40 Pin Female Header	1
4 Pin Male Header	1
USB Socket - Power Only	1
Spacer 16mm M-F	4
Spacer 15mm F-F	4
Plastic 2mm Nut (for spacing)	4
LDR GL5516	1
6mm x 6mm Momentary Push Button Switch	4
2mm JST Header Socket	1
Panel Mtd USB Socket	1
Plastic Screw 12mm	4
Plastic Screws 6mm	4
1N5817 Diode	1
Optional - Not included in Kit	
LM2956 Power Module or	1
MP1584 Power Module	1
TP4056 Battery Management Module	1
BME280 (or BMP280) Module	1
Male Pin Headers	Qty
5.5mm x 2.1 mm Socket	1
5.5mm x 2.1 mm Plug	1
2mm JST Header Socket	1
1N5817 Diode	1

Note: various combinations of the spacers may be supplied to support the PCB sandwich based on what was in stock at the time of kitting. The number of pin headers required for optional items will be determined by the options chosen. A standard 2.5mm 40 pin header strip will provide enough headers to solder the optional modules to the board.

MWRS NTP v2.1 Clock Schematic

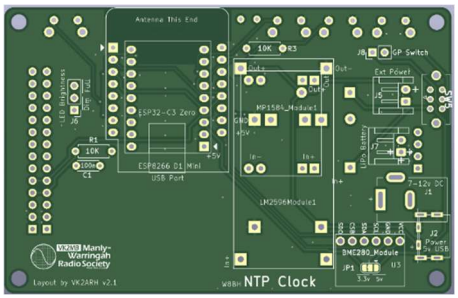


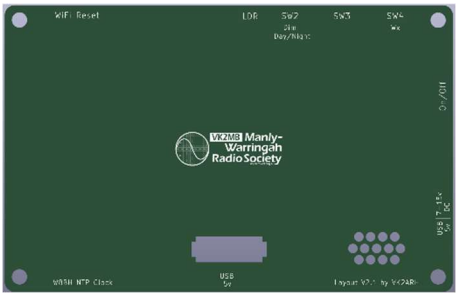
MWRS PCB's

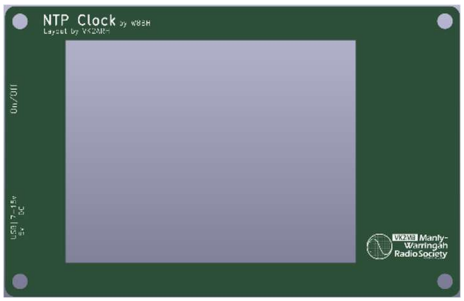
The Basic MWRS project kit provided all the necessary hardware to build a fully functional version of the NTP clock powered by a 5v USB-c power supply using the ESP8266 D1 Mini microprocessor. The kit also included a Light Dependent Resistor (LDR) and resistor to support display dimming functionality and four momentary contact push button switches to support further enhancement of the clock function via software development. The latest gerber files for the PCB's can be found on my github site: https://github.com/VK2ARH/NTP-Clock/tree/main/Gerbers

The V2.1 board supports a range of options should the builder wish to add these at a later stage:

- Supports for an external DC power source ranging from 7-15v supplied via a 5.5mm x 2.1mm DC plug. This option uses an onboard MP1584 (small) or LM2596 (larger) buck converter module to supply the 5v required for operation and enables operation from a traditional 13.8v shack power supply, portable use with a LiFePO4 or 12v lead acid car battery power source.
- Battery backup is supported by using a single LiPo cell managed by a TP4056 battery management module. This module will manage the charge and discharge of a single LiPo cell, and boosts the cell voltage to 5v for operation of the clock when the external power source is lost
- The PCB has been modified to support the ESP32-C3 Zero microprocessor if the builder wishes to port the firmware to that processor and undertake firmware development that exceeds the capabilities of the ESP8266 based D1 Mini.
- The PCB supports the installation of a BME280 (or BMP280) temperature/humidity/pressure module should the builder wish to develop code to support that functionality. The BMP280 doesn't support humidity measurement – only temperature and pressure.







The kit includes three PCB's, the main PCB and a top and bottom panel which enables the NTP clock to be mounted in a free-standing housing, without the need for further mounting in an enclosure.

If you wish to mount the clock in an enclosure, Simon VK2YU has developed a 3D printed enclosure his design can be found here:

https://www.thingiverse.com/thing:7013343

Building the MWRS NTP Clock

The basic NTP clock takes approximately one hour to assemble, and could be considered a easy project for beginners with some experience soldering, or a newcomer soldering under supervision. These build notes assume that you are familiar with soldering components onto a PCB.

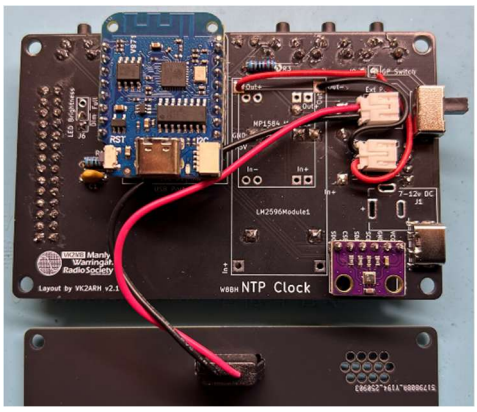
The basic ‘kit’ provides all the components necessary to build the NTP clock using a 5v USB-C external power supply, push button switches to support further user interface functionality development and the LDR and resistor to support the dimming and night mode functions.

You may choose to omit the push button switches, LDR and resistor and the clock can be built and deliver the same functionality as the original W8BH design but powered from a 5v supply.

If you decide to add 13.8v DC power supply support, LiPo battery backup or the environmental monitoring module, these can be added later with ease.

> NOTE: All the additional components can be soldered to the PCB at a later stage if you later decide to start experimenting. The ESP32-C3-Zero microprocessor, MP280, TP4056 LM2596 or MP1584 modules, are NOT supplied with the basic kit, but can easily be obtained on line.

NTP Clock fitted with LiPo Battery Backup



Top view with D1 Mini, BME280 and rear mounted USB port

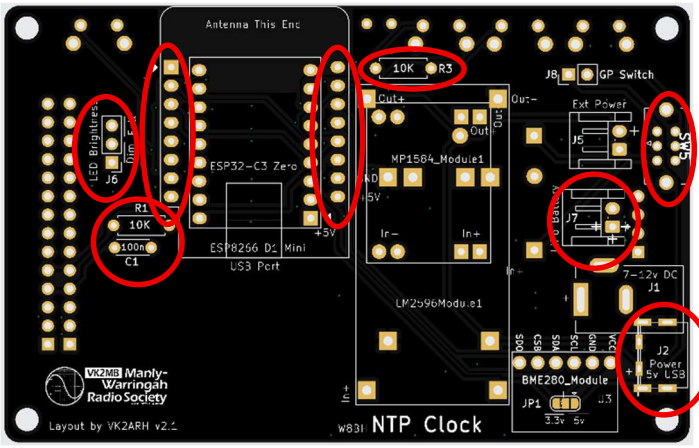
NOTE: You will need to file off approx. 1mm off two corners of the Display PCB closest to the ‘4 Pin Header’ end to allow it to fit onto the board without interfering with the plastic standoff’s holding the top panel onto the clock. You can also cut off the corners using wire cutters. This can be done with confidence, and it will not impact the operation of the 2.8” display.

If you don’t want to file your display board, you will need to find a couple of 3mm machine screws of appropriate length and replace the M3 plastic standoff supplied with the kit as shown in the photograph above. I recommend taking a file or wire cutters and just clipping the corner of the Display PCB (leaving the mounting holes intact) so that it can avoid interference with the plastic standoff when fitting into the ‘PCB Sandwich’ housing.

Building the NTP Clock using a 5v USB-C Power Supply (Basic Kit)

Build the NTP clock using the following components on the main board – it is important that you install the components on the correct side of the PCB. The location of each component is shown on the PCB's screen print.

PCB Front (Side with the MWRS Logo)

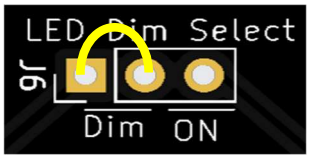


Install the following components:

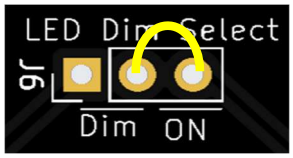
- SW5: DPDT Power Switch
- J2: USB-C Power Socket
- J5: 2mm JST Socket
- R1, R3: 10K Resistor
- C1: 100nF MLCC Capacitor
- 2 x 8 Pin Female Headers (supplied with the WEMOS D1)
- Solder a jumper bridge to J6 to either enable or disable the dimming function. Use a cutoff component lead soldered in a loop so that it can be easily 'cut' and removed later if required.

TFT Display Dimming Jumpers

If you install the LDR and SW2 on the rear of the board to support dimming and the night mode function, solder (jumper) J6 to the DIM selection as shown below left. If you are not using the dimming function install the J6 jumper to ON to enable the display at full brightness as shown below middle. The right hand picture shows the recommended 'loop' jumper.



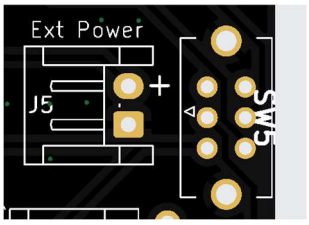
Dimming Function Activated



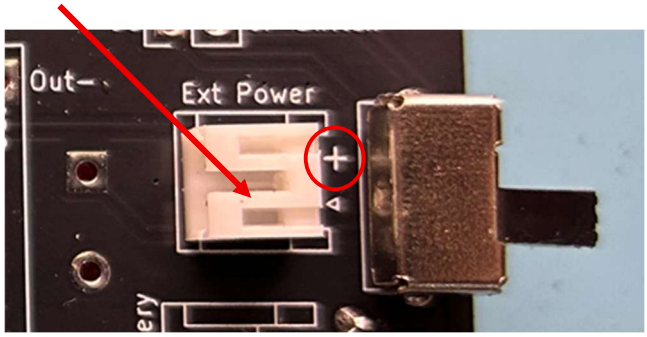
Dimming function Bypassed – Display Max Brightness

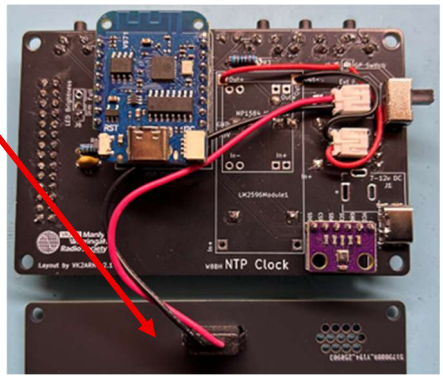
Example of 'loop' jumper installed

J5 is optional and supports a rear USB-C connector. Power can be provided by the J2 USB-C socket mounted on the side of the board, but if you wish to supply power from the rear of the clock, the panel mounted USB-C connector can be installed on the rear panel and connects via a 2mm JST plug to the J5 socket.



Ensure that the correct polarity of J5 is observed so that it aligns with polarity of the power cable from the rear mounted USB-C socket. When installed the connector should look like that shown below and the connection to the rear mounted USB-C socket is shown on the lower right.

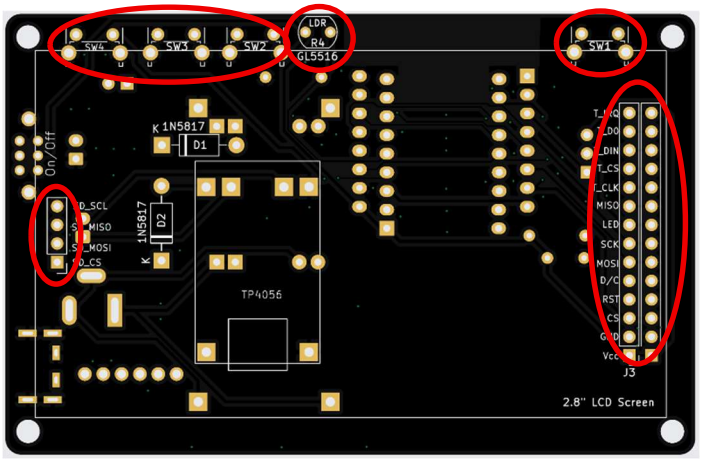




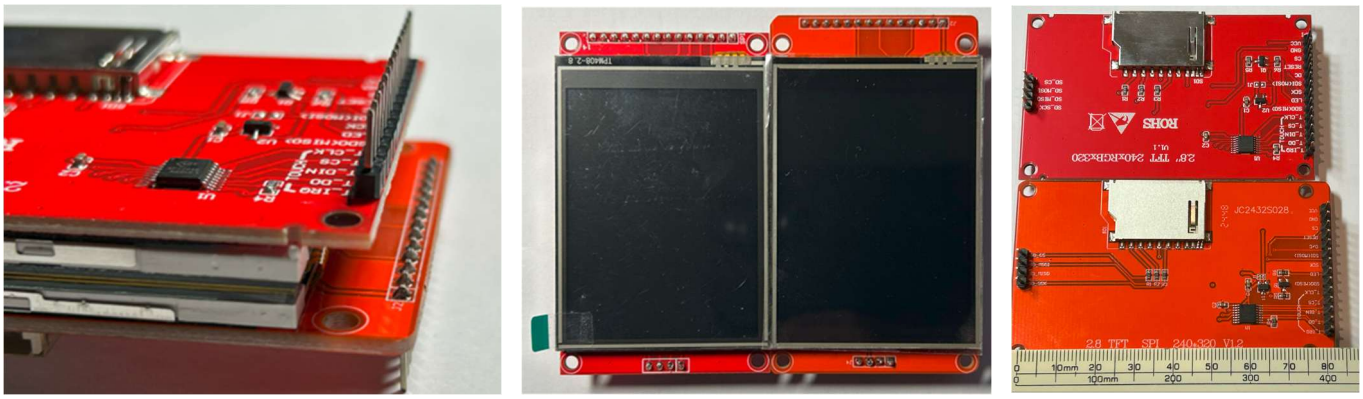
All the legs (power and shield) on the **J2 USB-C connectors** are very short and only just protrude through the PCB. Take time using a fine tipped soldering iron to heat and then flow solder **into** the solder pad so that it bonds with the pad and the USB-C socket legs. Once completed you will have a mechanical and electrical connector on your PCB.

PCB Rear

- 1 x 14 Pin Female Header
- 1 x 4 Pin Female Header
- 1 x GL5516 LDR
- 4 x 6mm x 6mm Right Angled push button switches
- 1 x Jumper across D1 or install a 1N5817 diode



As different manufacturers provide 2.8" TFT screens with different 14 pin connector positions, the version 2.1 board provides two sets of 14 pin connectors to enable use of either of the common pin configurations found on the 2.8" TFT displays.

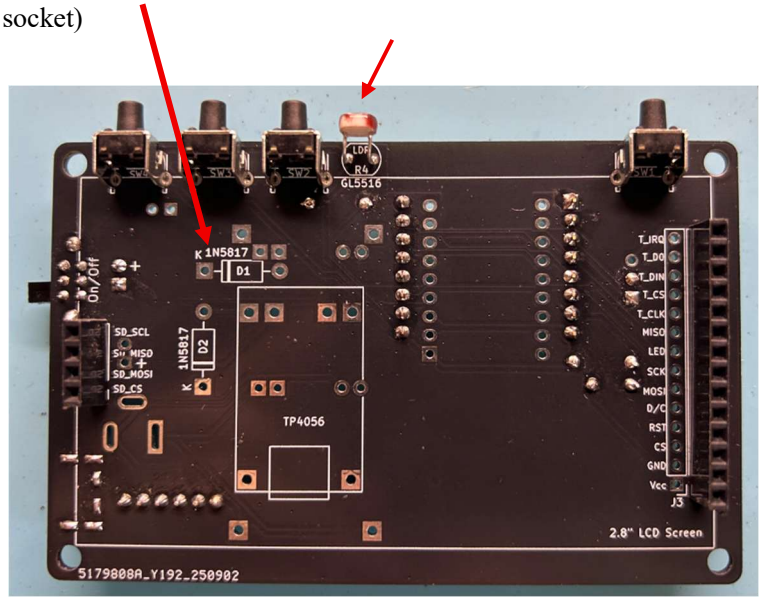


Different TFT Displays have the 14 pin header in different locations. The longer board is the original factory specification.

You will need to cut the 14 pin and 4 pin female headers from the 40 pin female header supplied with the kit. The easiest way to do this is to sacrifice the 15th pin (and 5th pin for the 4 pin header) when cutting off the header and then file down the end of the header to neaten it up.

- Install the 14 pin female header in the connector which aligns with your TFT display.
- Install the 4 pin female header on the opposite side of the board. These connections do not provide any electrical connections to the board but simply hold the TFT display in position.
- Install the 6mm push button switches (SW1, SW2, SW3, SW4).
- Install R4 after bending the legs 90 degrees approx. 5mm below the LDR so that it is facing upwards.
- Install a jumper to bridge across D1 (you may choose to install a 1N5817 diode instead to provide reverse polarity protection if you are supplying power via the J5 socket)

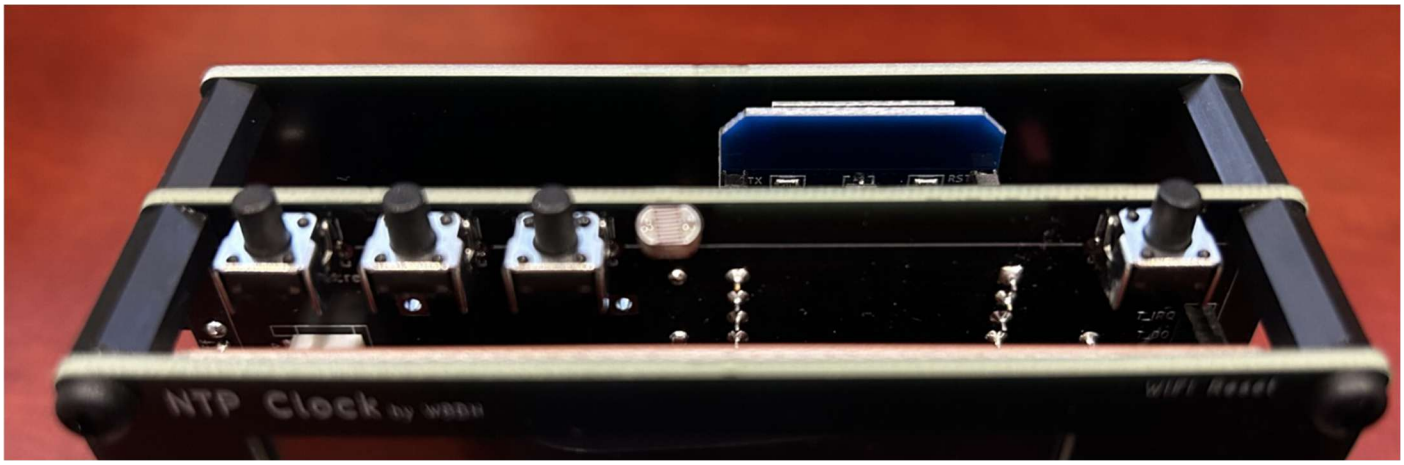
The completed rear board (basic kit) will look like that shown on the right (although no jumper or diode is shown at D1). The installation position of the 14 pin header will be determined by the location of the pins on your TFT display. Measure / check twice before installing your headers. There are sufficient female headers supplied with the 40 pin header strip to install two 14 pin headers (one in each position) to allow for either 2.8" TFT display variants to be accommodated should you wish top provision for both.



Ensure that the pin headers are perpendicular to the PCB – the easiest way to align these is to solder a single pin to the board – check alignment and if adjustment is needed, reheat and melt the soldered pin, adjust the

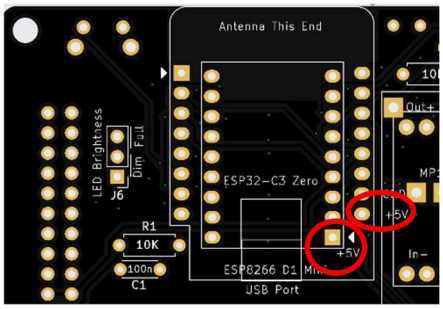
position of the header and once in position proceed to solder the remaining pins to the board.

The LDR is installed after bending the leads at right angles before soldering to the board. The following photograph shows the position of the LDR and push button switches and illustrates how the switches and LDR are installed.



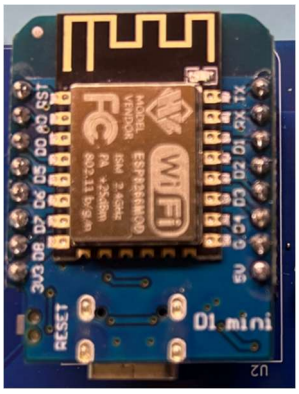
Installing the WEMOS D1 Mini (or ESP32 Zero).

- Solder the 8 pin male header pins to the D1 Mini **or**
- Solder the 9 pin male header pins to the ESP32 Zero

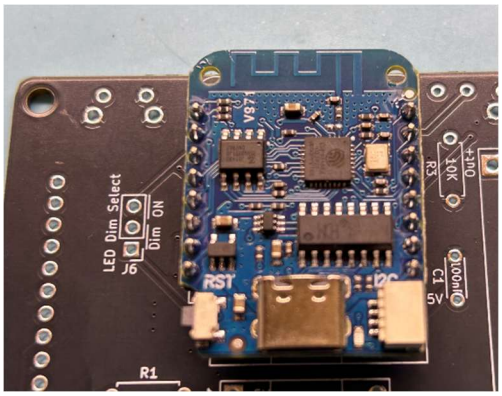


You may find it easier to do this by inserting the male header pins into the female headers you installed on the front of the PCB and once aligned solder the D1 Mini to the header pins.

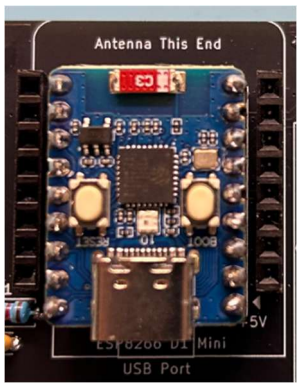
NOTE: it is important that you install the headers and the D1 mini in the correct orientation. The easiest way to check orientation is to check that the WiFi antenna is located at the top of the board, and the +5v power pin aligns with the bottom right pin identified on the board. Depending upon which processor and version you are using the USB connector may be on the top or underneath the board. Examples of the correct orientation are shown below:



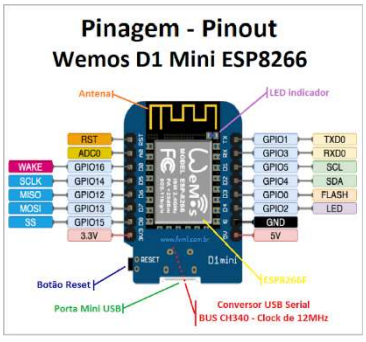
WEMOS D1 Mini Version 1

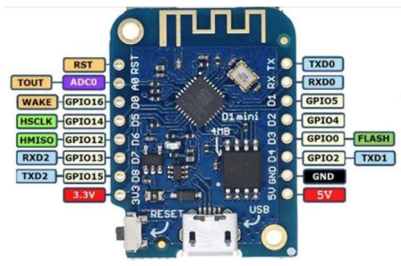


WEMOS D1 Mini Version 3 and 4

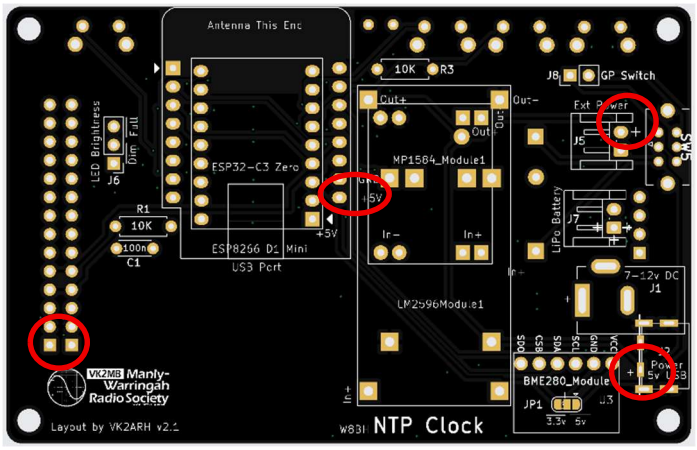


ESP32 Zero





This completes the construction of the basic NTP Clock. Before connecting power, remove the TFT display and WEMOS D1 mini. Check that you have no shorts between +5v and Ground using a multimeter between any +5v point and ground. If you have a short start to investigate and resolve the short before proceeding. Connect a USB-C 5v power source to J2 (USB-C port on the board) and verify that 5v is being correctly applied to the points on the PCB circled in red. If you installed J5 and the panel mounted USB-C socket, check the correct polarization of the supply by checking the +5v points on the PCB with the panel mounted socket supplying power via J5. If you don't have any power to the ESP8266 port, check that you installed a jumper bridging D1 or D1 if you are using J5 to supply external power. If you are using J5 I suggest that you also test polatity after testing with power connected to J5.



Install the firmware into the ESP8266 WEMOS D1 Mini (see below) and then install the D1 Mini and the TFT Display onto the PCB. Depending upon which settings you configured in the firmware will determine your clocks start sequence. See details later in this manual.

Installing and Running the Software

The software to operate the NTP clock needs to be uploaded into the WEMOS D1 Mini after 'personalizing' your installation. By default, the MWRS firmware will provide a display as shown below left – with the banner set to "NTP Clock" and the time zone set to Australian Eastern Standard Time (AEST). You can edit the firmware and insert your own callsign as shown in the photo below right as well as configuring your local time zone:

These instructions assume that you have a working knowledge of using the Arduino IDE to compile and upload the firmware to the WEMOS D1 Mini. There are many YouTube videos and tutorials on the internet if you don't yet have this knowledge.

Details of how to modify and install the firmware can be found in the documents listed on P4 of these notes. Read these documents to gain an understanding of the process involved. In short:

- You need to configure the Arduino IDE to program the WEMOS D1 Mini.

  See: How to install the ESP8266 Board into Arduino IDE.pdf

- Download the latest firmware from my GitHub site: https://github.com/VK2ARH/NTP-Clock and load it into the Arduino IDE, **make sure you have installed the libraries identified in lines 35-50 of the code**. If you are not familiar with loading libraries, the Arduino documentation covering library installation can be found here: https://docs.arduino.cc/software/ide-v1/tutorials/installing-libraries/ There are lots of YouTube videos available on the topic as well. I have also included some notes on this on p21 of these notes.

  NTP_Dual_Clock_MWRS_V2.ino

- Modify the entry for the banner eg: insert your callsign at line 84 in the code (eg: replace “NTP Clock” with your callsign).

```
#define TITLE                 "NTP Clock"
```

- Change the code at line 91 which configures your time zone if you are in any time zone other than Eastern Australia Standard Time (AEST). The character string you need to use for your time zone can be found https://github.com/nayarsystems/posix_tz_db/blob/master/zones.csv

  Eg: for Easter Standard Time USA you need the string “EST5EDT,M3.2.0,M11.1.0”

```
#define TZ_RULE               "AEST-10AEDT,M10.1.0,M4.1.0/3"
```

- If you wish to hard code your WiFi network credentials into the firmware, uncomment lines 87 and 88, and insert your SSID and Password between the quotes. Eg:

```
#define WIFI_SSID             "myNetwork"
#define WIFI_PWD              "password1234”
```

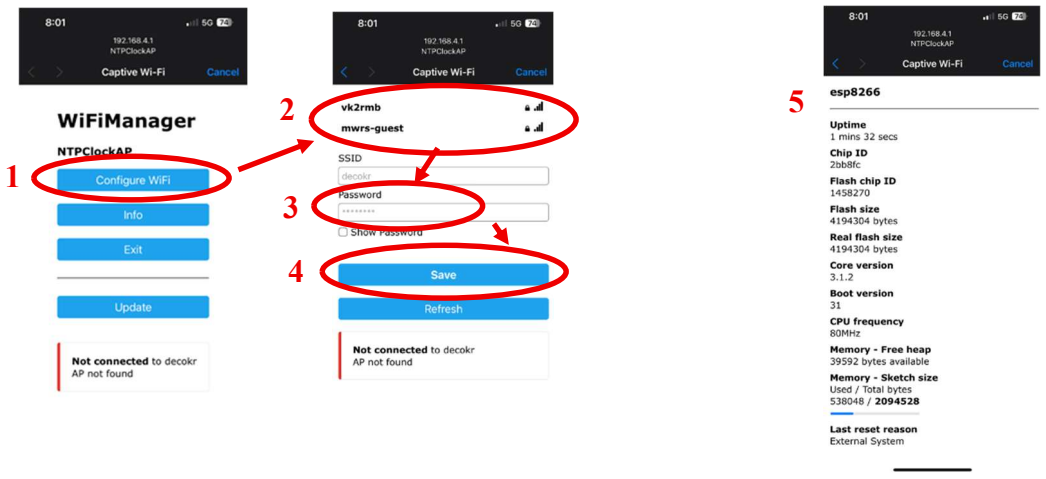
  I recommend that you use WiFi Manager to enter your WiFi credentials, and leave lines 87 and 89 as they were in the code.

- Compile and upload the firmware to the D1 Mini.

Setting up your WiFi Credentials on First Time Startup

Upon first power up, (when no WiFi credentials are stored in the D1 mini, or if the stored network is not available) you need to enter new Wifi Credentials using the WiFi Managers capability. After about 15-20 seconds after power up, you will still be looking at the WiFi Connecting Screen on the NTP Clock or you may be viewing a blank screen.

- Open up your mobile phone and look for a WiFi access point “NTPClockAP”
- Connect to the NTPClockAP
- If you are using an iPhone – you will automatically be directed to a WebPage which will allow you to configure the Wifi Credentials. If you are using an android phone, you may need to put the IP address http://192.168.4.1 into your Web Browser to bring up the configuration screen.
- The screen will look similar to those shown below.
  - Select Configure WiFi (1)
  - Identify the WiFi network SSID that you wish to connect to by tapping on the listed network (2), this list will show all the WiFi networks that your NTP clock can view.
  - Enter the password for your chosen network (3).
  - Hit the SAVE button (4) and your NTP Clock will now after an approx. 10 second delay connect to your chosen WiFi network and commence operation.
  - The information screen (5) can be displayed by selecting the Info button on the first WiFiManager screen



Using the Dimming Function: (Updated 31 July 2025.)

Mark Culross KD5RXT has provided an update to the firmware to support the use of the LDR and SW2 to enable either manual or automatic dimming of the TFT display. This provides a significant improvement in the day-to-day use of the NTP Clock. This feature has now been incorporated into the firmware available in the software folder on my GitHub site:

https://github.com/VK2ARH/NTP-Clock/tree/main

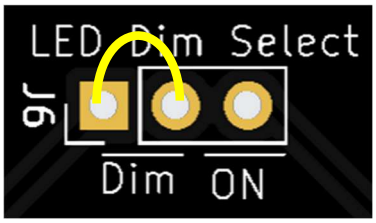
In Mark's own words:

> *I decided to tackle the desire to control the screen brightness, & I succeeded. The sketch allows adjusting the brightness from 1-7 (7 is the brightest & is also the power-on default), as well as an AUTO setting (active between "7" & "1"). The AUTO setting uses the detected light level from the LDR sensor to set the brightness. Note that the on-board LED Dim Select jumper must be moved from ON to DIM for this to work.*
>
> *Mark KD5RXT*

A video demonstration of using the dimming function can be found here:

https://youtu.be/ylc6PuMD5ms?si=vuxyQWi9VpEJZjZy

NOTE: If you are using the dimming function the LED Dim Select jumper (J6) must be set to Dim as per the diagram on the right, whilst the brightness setting is shown on the photo below:



The brightness level is set by pressing S2, each time it is pressed the brightness level changes from 1 through to 7 and then Auto. When set to auto, the level is determined by the amount of light hitting the LDR.

HAMSQL Certificate

The HAMSQL Certificate currently used in the software has been updated and should provide you with continuous access to the solar data which is displays next to the UTC banner.

If the certificate has changed or expired or you are unable to reach the solar data server, the ***no data*** message will display on the UTC banner as shown on the right:

Night Mode: Updated 7th August 20025 (By Mark KD5RXT)

The latest version of the firmware version 2.2 makes provision for 'Night Mode' using a red display so as not to destroy your night vision. To switch between 'Day' and 'Night' mode – hold down SW2 for more than 3 seconds and on release the mode will toggle between Night and Day modes. A video outlining this functionality can be found here:

https://youtu.be/N1oHZ7b6xN0?si=A_FLzp0qJcr7Nnyy

Installing additional Optional Features

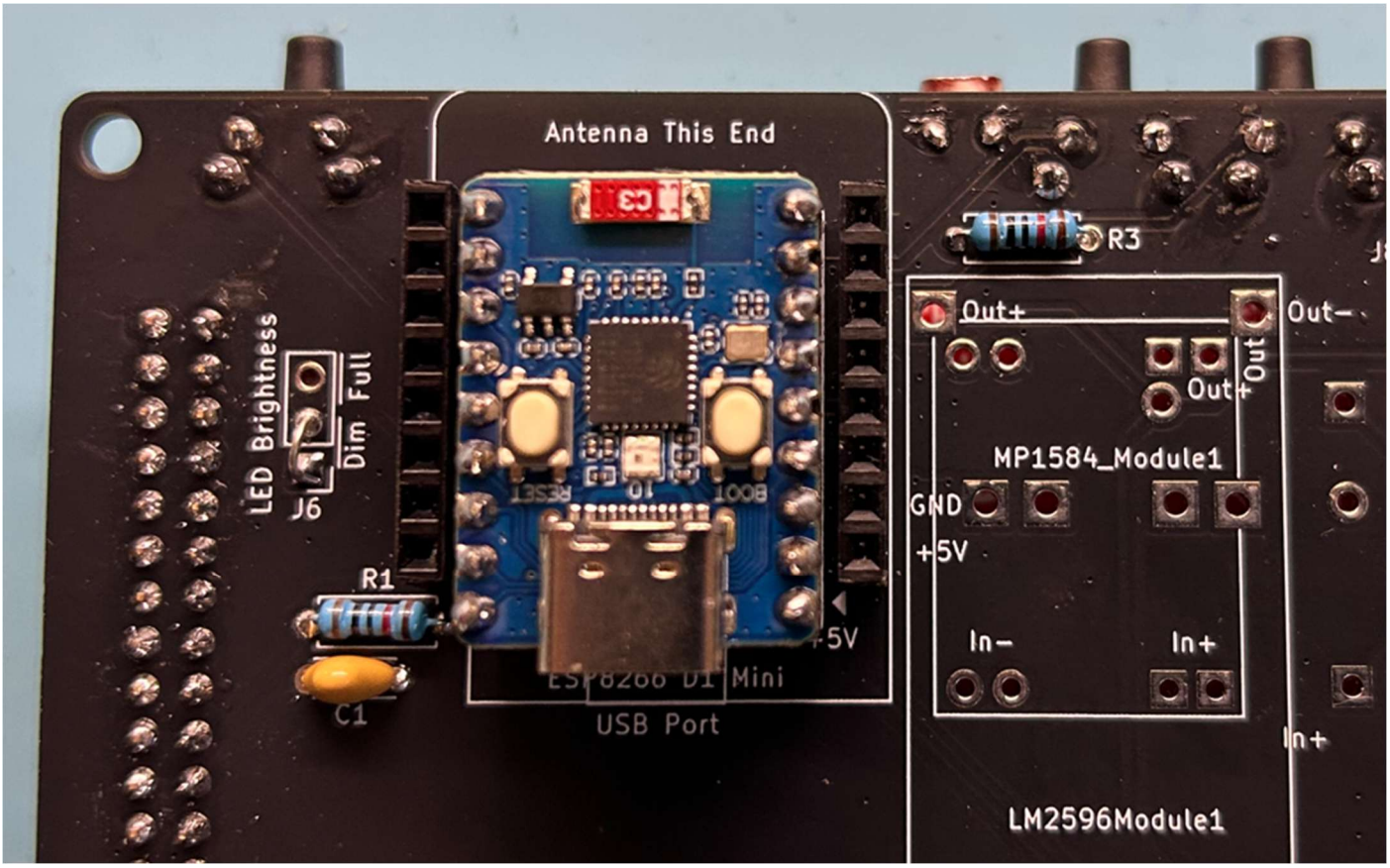
The basic NTP clock kit built by the MWRS included the LDR (Light Dependent Resistor) and a series of push button switches to enable builders to experiment with the code and develop new functionality if they chose to do so. The dimming, night mode and WiFi Manager capabilities are a result of their inclusion, and this capability is now represented in the latest code. Further developments of the original W8BH code by AI6P and WA2FZW included additional solar data display capabilities. The software continues to develop, and I will list the latest software on my GitHub site should you wish to download it at a later stage and update your NTP clock to take advantage of the new capabilities.

The current v2.1 board supports several optional features, these include:

- ESP32 Zero – should you wish to take advantage of the increased processing power and memory capabilities of and ESP32 processor, the ESP4266 (WEMOS D1 Mini) can be replaced with an ESP32 Zero. The code will need to be modified to run on the ESP32.
- Operation from a 7-15v DC power supply, this enables powering the NTP clock from your regular shack 13.8v supply or when portable using LiFePO4 batteries or vehicle power supply systems. This is achieved by installing either the LM2956 or MP1504 module to convert the power supplied by a 5.5mm x 2.1mm DC plug.
- Battery backup using a single cell LiPo battery with TP4506 battery management module.
- Installation of a BME280 or BMP280 modules to provide Temperature, Pressure and Humidity (BME280 Only) readings to display on your clock. NOTE: The software to support this function is yet to be developed, but several individuals are 'playing' with that capability.

Using an ESP32 Zero

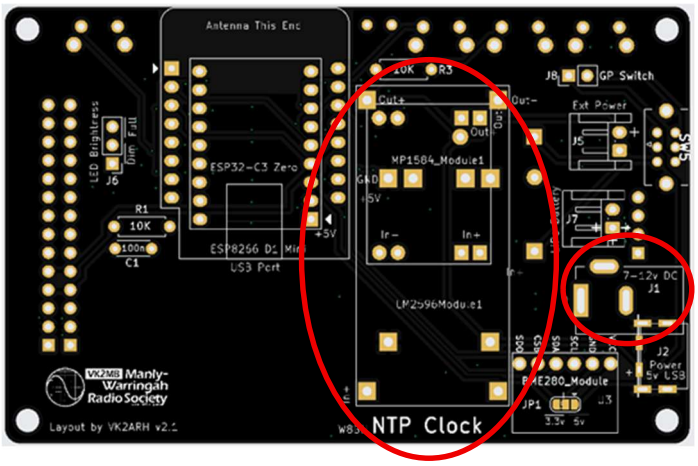
The only modification required to use an ESP32 Zero module is to install the 9 pin headers onto the main board to support insertion of the ESP32 Zero Module instead of the ESP4266 (WEMOS D1). When installed in the correct orientation the ESP32 Zero Module looks like this:



You will need to recompile your firmware to operate on the ESP32. The ESP8266 firmware will not operate on the ESP32 without slight modification and recompiling.

Building the NTP Clock using a 7v-15v DC Power Supply

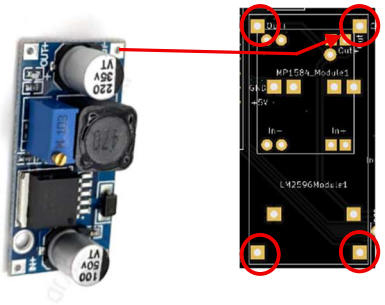
Follow the same building instructions as per the USB-C power supply powered basic 'kit' for the front and rear of the PCB, BUT do not install the USB-C power only socket. If you have already installed the USB-C power socket (J2) when constructing the basic build, you will need to remove it to install J1 – the DC power socket.



Remove the microprocessor and the TFT display before installing the power modules, as the voltage needs to be adjusted to 5v before installing voltage sensitive components.

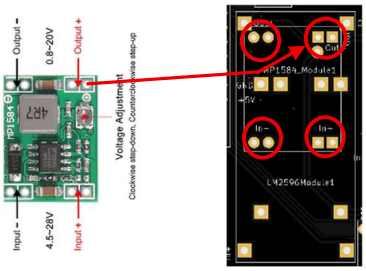
The PCB supports either an LM2596 Power Module – the module used by W8BH in his original design, or the smaller MP1584 Module. Both can power the NTP clock with ease.

- The LM2596 Power Module uses four single header pins, ensure that you align the module in the correct orientation so that the input and output pins match the details provided on the screen print on the PCB.



In both examples the location of the modules top right hand pins location on the PCB is identified. The module is mounted on the front of the PCB with the component side of the module facing upwards.

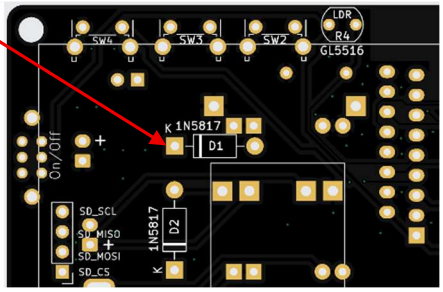
- The MP1584 Power Module has a total of 8 input and output pins with two pins used for each function. Using a minimum of 4 single pin headers to solder the module to the front of the PCB.



NOTE: Do not install the TFT display or the WEMOS D1 onto your PCB until you have calibrated your power module output to 5v.

- Install J1 – a 5.5mm x 2.1mm DC plug onto the front side of the PCB as shown in the location diagram on the top right of this page.

- Install D1 – a 1N5817 polarity protection diode onto the rear of the PCB observing the correct polarity shown on the screen print. If you installed a jumper across D1 during an initial built (without a power supply module) remove the jumper and replace it with a 1N5817 diode.

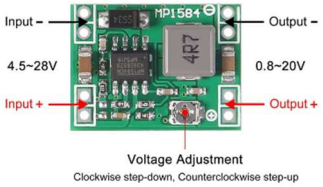


Calibrating the Power Module Output to 5V

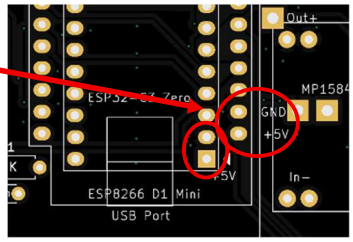
With the TFT Display and the WEMOS D1 **REMOVED FROM (OR NOT INSTALLED)** on the main PCB, connect a 7v-15v DC supply to the DC Power Socket (Center pin is +ve, shield is -ve).

Monitor the output voltage with your multimeter and adjust the LM2596's multiturn variable resistor until you achieve an output of 5v DC. You may require multiple rotations of the screw before the module's voltage output begins to change.

If you are using the MP1584 power module, the small trimmer resistor is used to adjust the output voltage.

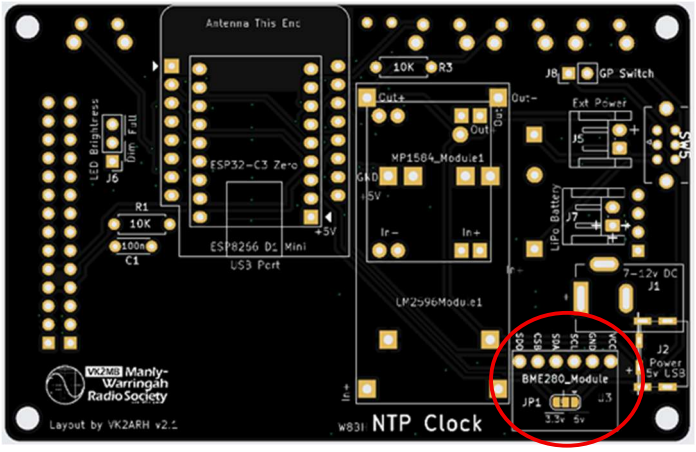


Once you have set 5v, double check the voltage and polarity being supplied to the WEMOS D1Mini using the power supply pins to the D1. Both the +5v and GND pins are noted on the front panel screen print and are duplicated on the footprint for the ESP32 -C3 Zero.



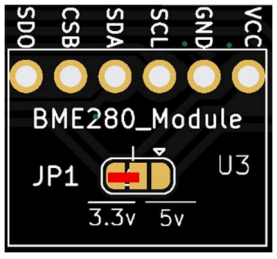
Installing the BME280/BMP280 Sensor

BMP280 is a barometric pressure sensor that measures air pressure and temperature, whilst the more expensive BME280 is a precision environmental sensor that measures humidity, temperatures, and barometric pressures.

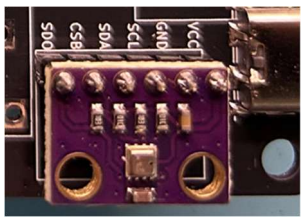


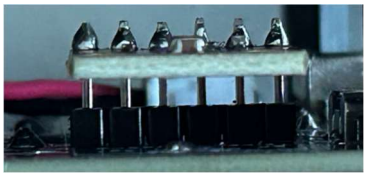
Both devices can be supplied to operate either from 5v or 3.3v and communicate with the microprocessor using the I2C bus. The 5v versions of the sensor have an onboard voltage regulator to drop the voltage down to 3.3v.

The NTP Clock main board supports both sensors and enables the appropriate voltage to be delivered to the module after soldering a jumper on JP1 to select operation with 3.3v or 5v. The easiest way to solder a jumper on JP1 is to use excess solder when soldering onto the middle pad and either the 5v or 3.3v pad, then drag the solder across the middle and selected voltage pads to form a solder 'bridge'. To remove the jumper, simply heat the solder and remove with solder wick or similar.

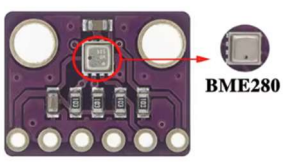


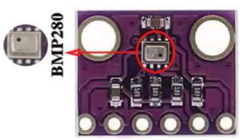
The example shown configures the board to deliver 3.3v to the Vcc input pin on the BME280 module. If you have a 5v module, simply short out the center pin to the 5v pin on the right of JP1. Once installed the BME280 module looks like that shown below: You may wish to raise the position of the BME280 module a few mm above the main PCB to provide better air circulation and reduced thermal impact from any components heating the main PCB. The easiest way to differentiate the two modules is to note the square shape of the BME280 or the rectangular shape of the BMP280.





Raise the BME280 module above the PCB





NOTE: at the time of writing software to display the BME280 output on the clock is not in the current firmware but can be expected in a future release. Some modules have only 4 pins, the SDO and CSB are not required to work with the NTP Clock.

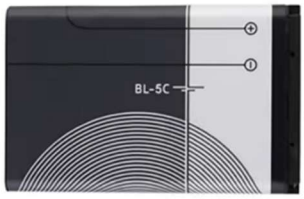
NTP Clock Backup with LiPo Battery

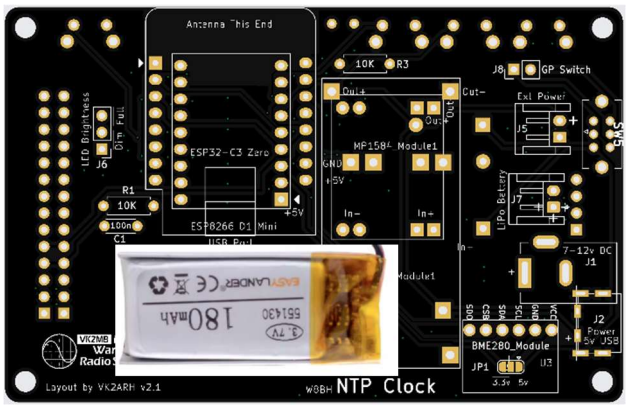
The NTP clock can be backed up using a single LiPo 3.7v cell and a TP4506 battery management module. The choice of battery is a personal preference; I chose to use what I had available which was an old camera battery which fitted easily in the clock without the need to adjust the size of the spacers between the PCB's. You should check the TP4506 data sheet to determine if the charge rate is suitable for your chosen battery.

The TP4506 module and the battery fit on the bottom side of the main PCB between the TFT Display and the PCB as shown on the right. I secured the battery to the PCB using a piece of double-sided tape and fed the battery leads which had been soldered to the battery terminals through two spare holes on the PCB to the front of the board.

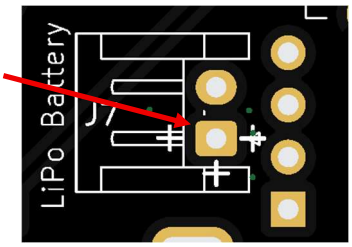
Space is available on the front side of the PCB under the processor and the MP1584 (if you are using it) to mount a small slim LiPo battery as shown below right. The use of a common BL-5C can also be squeezed in under the TFT display– but at the end of the day it up to the individual builder's choice.

The NTP clock will draw around 170mA when operating from 5v, so a 760mA battery will provide between 4-5 hours of operation, enough to keep the clock operating during a brown out.



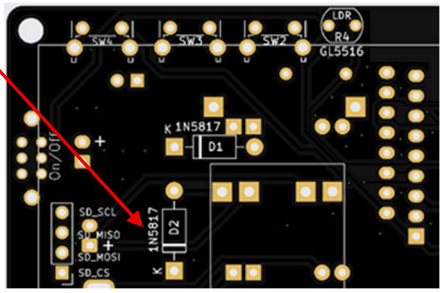


When connecting the battery to the clock – if you use the J7 JST socket position you will need to carefully consider which side of the board you locate J7. It may be necessary to bend the pins 180 degrees to ensure the correct polarity connection to the board. If you are plugging the battery into J7, J7 should be mounted on the same side of the board you mount your battery, and **you should manage the connection polarity accordingly.**



NOTE: the pin for the +ve battery connection is marked on the PCB Screen Print, J7 uses the square pad for +ve (this differs from the round pad being used as +ve on J5 for the External Power connection) – both J5 and J7 connectors have the +ve pad marked on the PCB screen print.

- Install D2 – a 1N5817 protection diode observing the polarity shown on the screen print



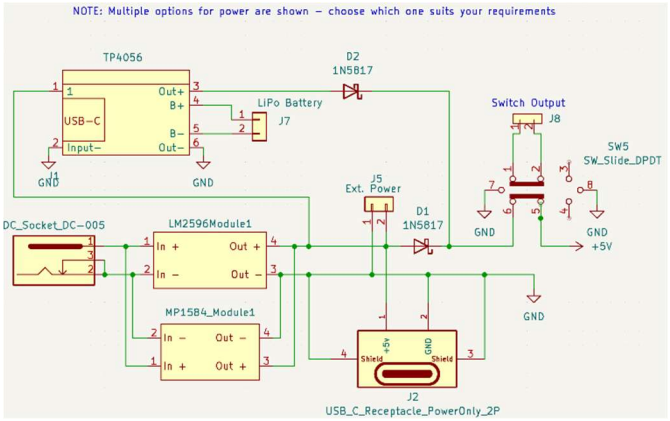
Isolating the LiPo Battery When SW5 is Switched Off

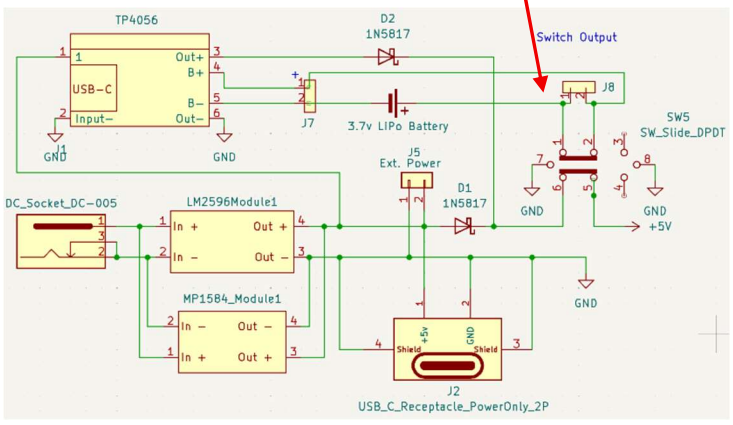
When the LiPo battery is connected to J5, it's wired directly to the TP4056 battery management module. When switching SW5 OFF, external power is still supplied to the TP4056 enabling the battery to be recharged and LED status lights on the TP4056 remain illuminated, red for charging and blue of fully charged.

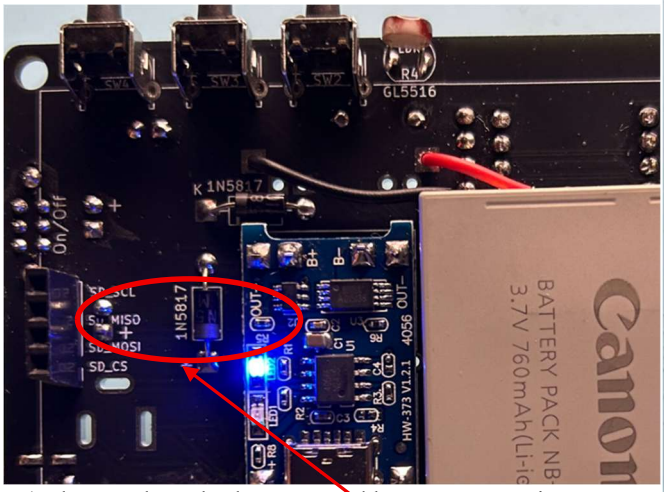
If external power is lost and SW5 is in the ON position, the TP4056 acts as a boost converter and delivers 5v to the NTP clock until it shuts down due to low battery voltage. When power returns the TP4056 starts recharging the LiPo battery.

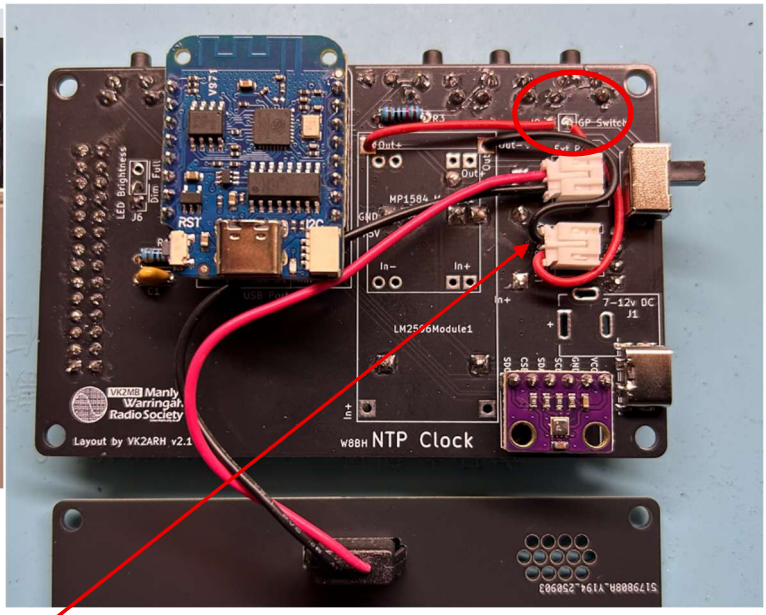
If you want to physically disconnect the LiPo battery from the TP4056 when the SW5 is switched off, you need to use J8 to isolate **ONE of the leads** of the LiPo battery. When wired this way the battery is isolated when the clock is switched off, and recharges when SW5 switches the clock back on.

To isolate the LiPo battery when SW5 is switched off, break the +ve LiPo battery connections and wire J8 between the battery and the positive input terminal of J7 as shown in the schematics below:









(I chose to have both power and battery connection on the front side of the PCB as the position of D2 and the TP4056 module were likely to interfere with the J7 connector should it be mounted on the rear of the PCB.)

In this example the LiPo battery leads are connected to J7 (NOTE the polarity and the orientation of the J7 JST connector). Both leads feed through to the bottom side of the PCB using the unused Out+ and Out – holes for the LM2596 module. The -Ve LiPo battery lead is wired directly to J7's negative terminal, whilst the +ve LiPo battery lead is wired via J8 (circled on the right hand photo) to J7's +ve terminal.

Installations will vary depending upon the builder's preference and LiPo batteries used. Just be careful to ensure that you wire in a way that ensures the correct polarity of the battery connection and there is no possibility of shorting the LiPo battery.

Good luck with your build, and I hope you get as much pleasure from using the NTP Clock as we did.

73's

Richard VK2ARH

An earlier version of the MWRS NTP Clock running on 13.8v in my shack 😊

Adding Libraries to the Arduino IDE for the NTP Clock

To successfully compile and upload the NTP clock software, you need to add several nonstandard libraries to your Arduino IDE environment. The libraries required are listed in the early part of the firmware as shown below: look for a code segment like this – depending upon which version of the code you are looking for the line number may be different. This is the section of code from version 2.3.

```
35  ********************************************************************/
36
37  #include <FS.h>        // to be used later -VK2YU
38  #include <TFT_eSPI.h>  // https://github.com/Bodmer/TFT_eSPI
39  #include <ezTime.h>    // https://github.com/ropg/ezTime
40  #if defined(ESP32)
41  #include <WiFi.h>  // use this WiFi lib for ESP32, or
42  #elif defined(ESP8266)
43  #include <ESP8266WiFi.h>        // use this WiFi lib for ESP8266
44  #include <DNSServer.h>          // requiresd for WifiManager
45  #include <ESP8266WebServer.h>   // requiresd for WifiManager
46  #include <WiFiManager.h>        // requiresd for WifiManager
47  #include <ArduinoJson.h>        // to be used later -VK2YU
48  #include <WiFiClientSecure.h>   // used for secure TLS connections VK2YU
49  #include <ESP8266HTTPClient.h>  // used for collection of solar data https://github.com/rkincaid/NTP_DualClock_Solar/tree/main
50  #endif
51
```

As you are using the ESP8266 the following list of libraries are required:

```
<FS.h>        // to be used later -VK2YU
<TFT_eSPI.h>  // https://github.com/Bodmer/TFT_eSPI
<ezTime.h>    // https://github.com/ropg/ezTime
<ESP8266WiFi.h>        // use this WiFi lib for ESP8266
<DNSServer.h>          // requiresd for WifiManager
<ESP8266WebServer.h>   // requiresd for WifiManager
<WiFiManager.h>        // requiresd for WifiManager
<ArduinoJson.h>        // to be used later -VK2YU
<WiFiClientSecure.h>   // used for secure TLS connections VK2YU
<ESP8266HTTPClient.h>  // used for collection of solar data https://github.com/rkincaid/NTP_DualClock_Solar/tree/main
```

Plus you will also need the following library to use the push button switch functionality:

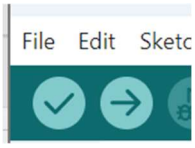
```
<Button.h>
```

Full details of how to install additional libraries in the Arduino IDE can be found here:

https://docs.arduino.cc/software/ide-v1/tutorials/installing-libraries/

However, I have detailed below how to do this specifically for the NTP Clock libraries.

Some of the libraries are already part of the IDE (particularly after you have added support for the ESP8266), the remaining are installed either from the list of libraries waiting to be installed into the IDE, or manually by downloading a Zip file from the website referenced next to the library name, or all the libraries (which may not be the latest) from my GitHub site. The easiest way to determine the status of the required library is to compile the firmware first by clicking the tick button. Error messages will identify any missing libraries. EG:

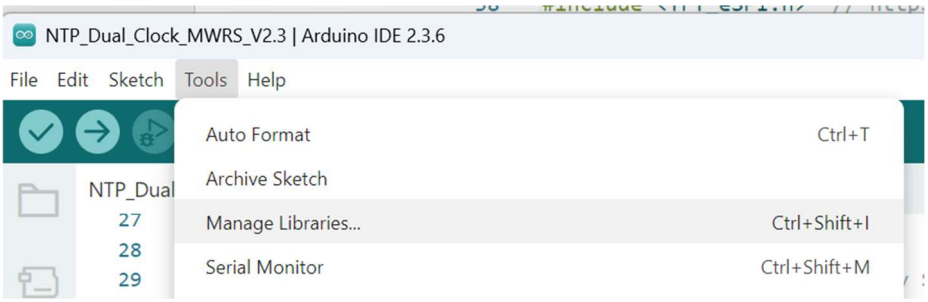


Missing libraries can be installed automatically by selecting them from library manager list of available libraries or by downloading the required Zip file and manually installing that file.

Installing a Library from the Available Library List:

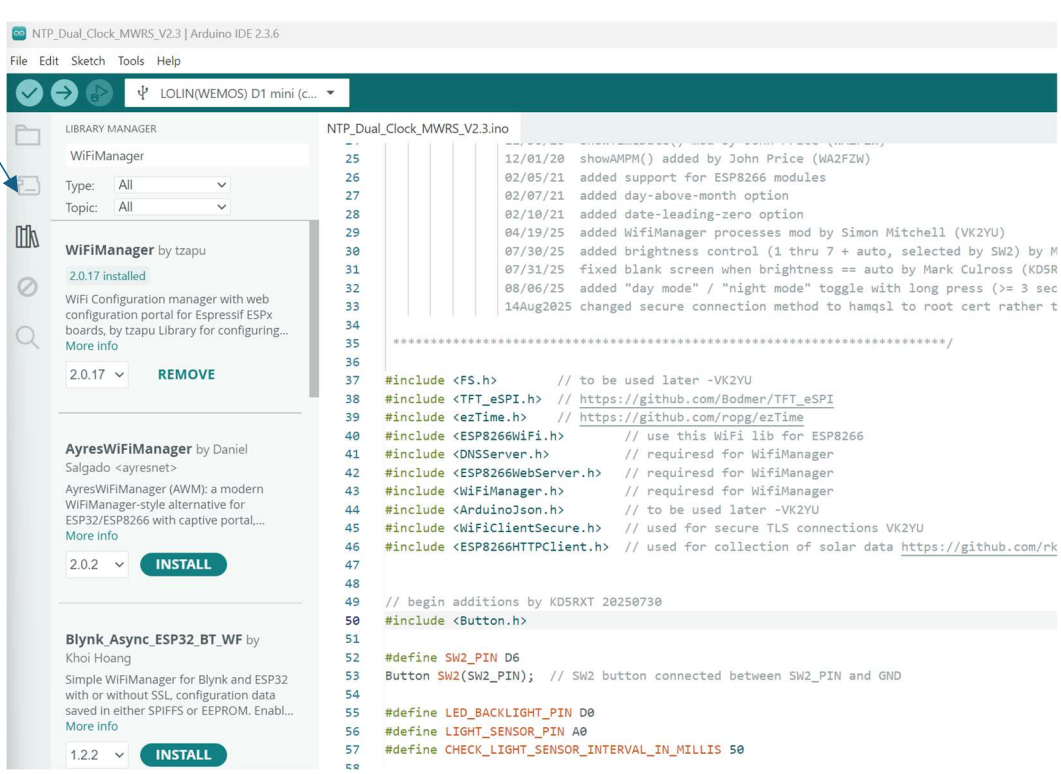
You install available libraries using the following process. I'll use the WiFiManager.h library as an example.

From the menu select: Tools > Manage Libraries…



The library manager panel will appear on the left hand side of the IDE and list all the libraries currently available for uploading in alphabetic order.

Type the library you are looking for (or want to install) in the search panel, in this case WiFiManager and it will list any library that matches that description you have already installed first and then a list of libraries that match that description.



If not already installed scroll down the list of libraries available and click the install button to install the required library.

It may take some time to install the library depending on the speed of your internet connection and processor etc., but you will get a message in the bottom of your IDE as well as an installed tag under the library name in your library list: eg:

Repeat this process for each of the libraries you want to ins 2.0.17 installed of available libraries.

Manually Installing a Library:

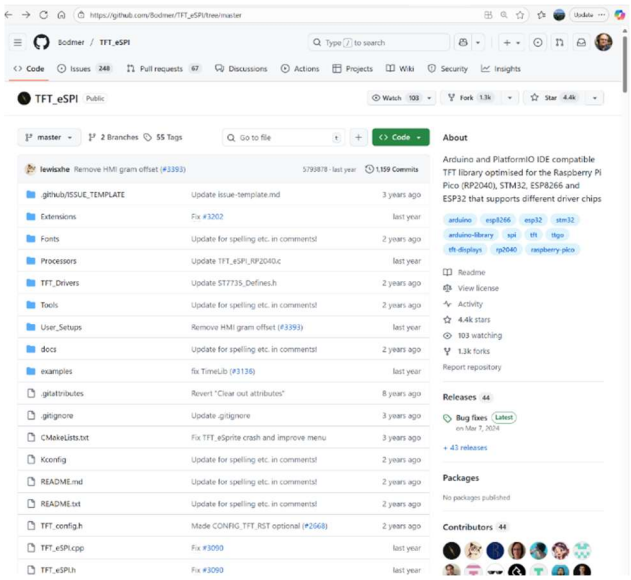
If the required library is not in the list of libraries, you will need to manually install it using the URL shown in the library list eg:

```
<TFT_eSPI.h>  // https://github.com/Bodmer/TFT_eSPI
<ezTime.h>    // https://github.com/ropg/ezTime
```

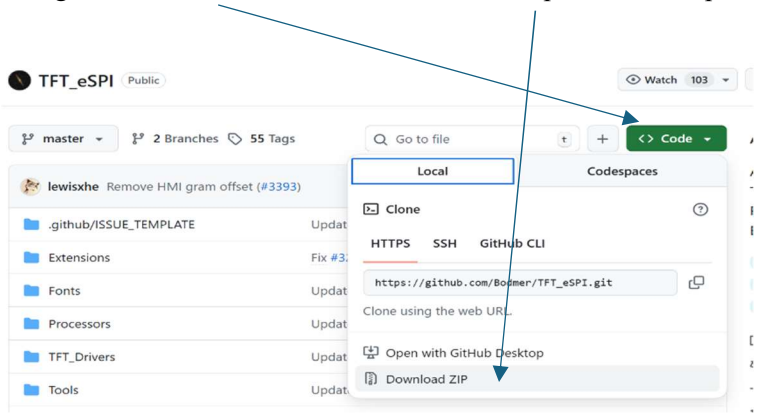
To manually install a library (a library not in the list of available libraries):

First go to the URL shown in the firmware code using your web browser *(we'll use the TFT_eSPI as an example)*;

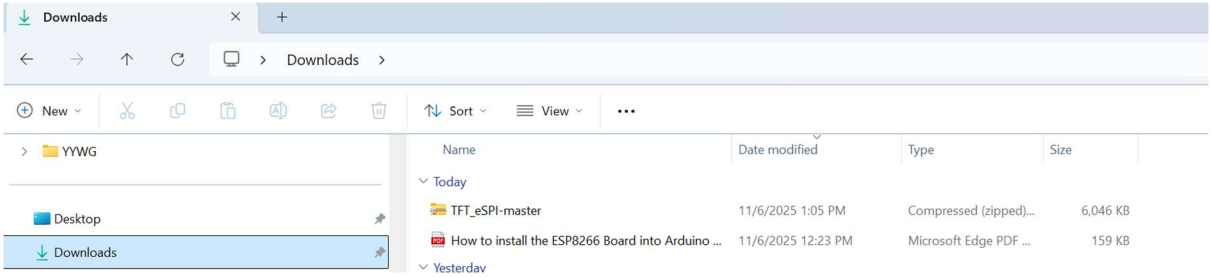
You'll see a screen similar to this:



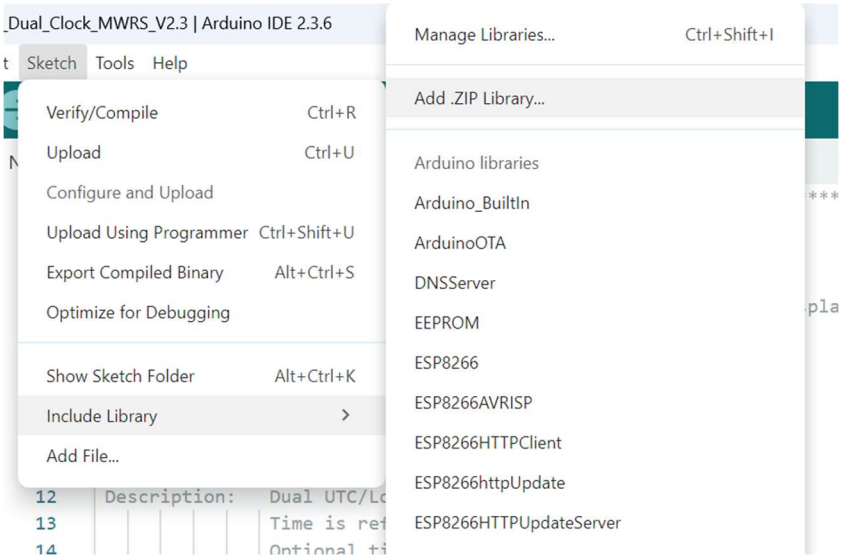
Click on the green code button and select : Download Zip from the dropdown box.



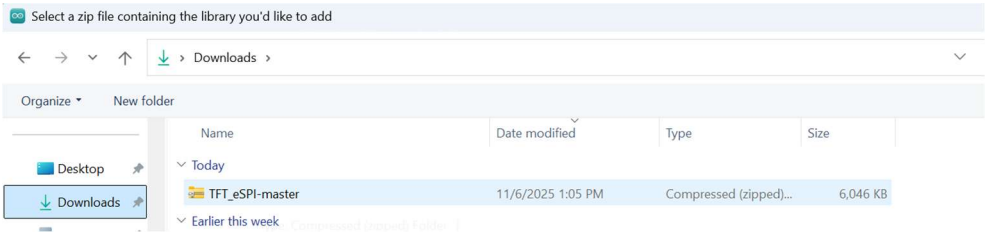
A zip file will be downloaded to your 'Download' folder.



Now go to the Arduino IDE and select: Sketch > Include Library > Add .ZIP Library

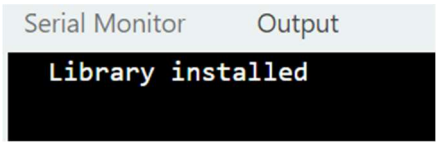


When the dialogue box opens, navigate to your download folder and select the zip file you have just downloaded:

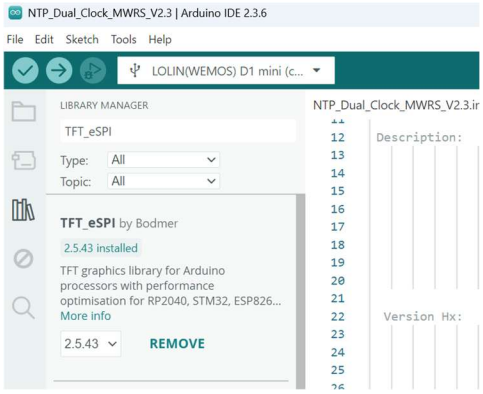


The IDE will then start importing the new library. If you have already installed this library a dialogue box will ask you if you wish to overwrite the previously installed library.

If you overwrite any previously installed library or not, when the library is installed you will receive a simple confirmation message at the bottom of your IDE screen as shown below:

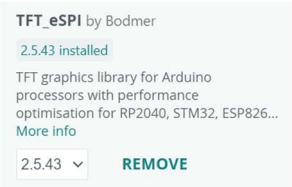


You can check that the library has been installed correctly by checking in the Library Manager as before:

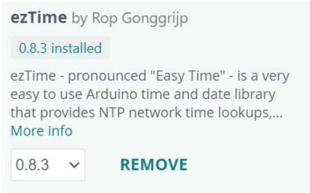


Sometimes it is difficult to choose the correct library from the available libraries in the standard Arduino list so here is a list of the libraries that I am currently using in my IDE to support the correct compilation of the NTP Clock firmware for the ESP8266:

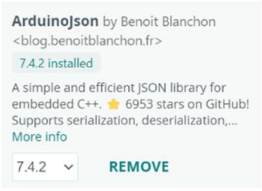
TST_eSPI.h



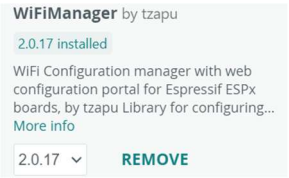
eZTime



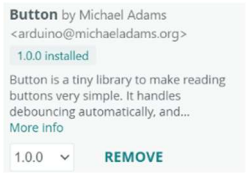
ArduinoJson.h



WiFiManager



Button.h



The following libraries are not listed in my library list, so I presumed they were automatically part of the standard Arduino IDE environment or installed with the ESP8266 environment as standard available libraries.

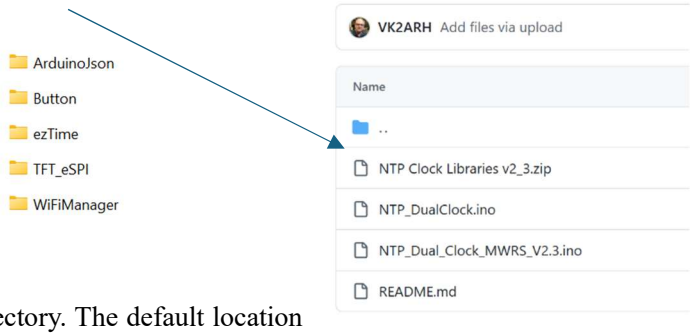
FS.h ESP8266WiFi.h DNSServer.h ESP8266WebServer.h WiFiClientSecure.h ESP8266HTTPClient.h

Directly Copying Libraries to Your IDE

If you are struggling with the previous two methods which will ensure that you are able to install the latest library – the final alternative is to copy the required library files directly to your IDE's library directory and they will be picked up when you compile the firmware.

To do this first download the Library Zip File from my Github site, then unzip the file in your downloads directory.

The Zip file contains all the additional libraries used in the code. →



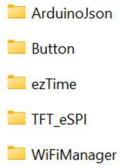
Identify the location of your IDE's library directory. The default location can be found in your documents folder:

Documents > Arduino > Libraries

When you open this directory, you will see a number of library folders listed.

Copy the library directories extracted from the Zip file and past them into your Arduino Library directory. You may be asked if you wish to overwrite the existing file if you have already installed that library. (Use your own discretion here, as the library you have already installed may be a later library than the one I have in the Zip File).

Copy to your Arduino Library Directory



➔ Default location: Documents > Arduino > Libraries