NTP Clock with Solar Data
Version 3.1
©John Price – WA2FZW - November, 2025

Table of Contents

Introduction

Several years ago, Bruce Hall (W8BH) designed an ESP32 based digital clock that kept time by querying an NTP (Network Time Protocol) server. His original design as I recall was either simply a breadboard or hand wired implementation. I designed a circuit board for it; Bruce has also designed one since then. Bruce's notes and the Gerber files for my original PCB and the original software can be found at W8BH.net.

Recently, the California QRP Club did a version of Bruce's clock that runs on an ESP8266. Robert Kincaid (AI6P) modified Bruce's ESP8266 version of the software to add the capability to download solar data from the 'hamqsl.com' website run by Paul Herrman (N0NBH).

Unfortunately, the approach Robert used to access the data was incompatible with the method required for the ESP32 processor due to differences in the software libraries available for each of the processors.

I spent a couple of weeks working with Robert to figure out how to make the software run on either processor.

Once that problem was solved, I continued to do some cleanup on the software and added the capability for the user to define a number of items from the solar data that could be displayed in a rotating sequence on the clock at a rate chosen by the user; the user can add other items if desired.

We also added the capability for the user to define multiple WiFi networks, so that if a connection cannot be established on one, we will try others.

As there were a couple of things I didn't like about the ESP32 PCB that I had originally designed and Bruce's ESP8266 design, I have also designed new hardware for both the ESP32 and the ESP8266.

There are actually three different hardware versions. This document will describe both the new hardware and software.

In addition to this document, the Gerber files for all three versions of the clock and the software is available on Github. The TFT_eSPI setup files are also available there.

Here’s a picture of my clock running on the ‘Cheap Yellow Display’:

The STL files for the case and stand (not visible from this angle) are from this site.

Modification History

Version 3.1 (November, 2025) adds the capability to display a rotating sequence of local times in different timezones.

As distributed, the clock will alternate between US Eastern time and Australian Eastern time every 5 seconds.

The detailed instructions for how to change the sequence are in the comments in the ‘UserSettings.h’ file

The Hardware

There are actually three variations of the Version 3.0 hardware implementation presented here.

The first is for the ESP DEVKIT-32D, which is basically the same as my original version except I eliminated the voltage regulator and ability to power it from 12V, electing instead to simply power it via the processor's USB connector which is how I've always powered my original version.

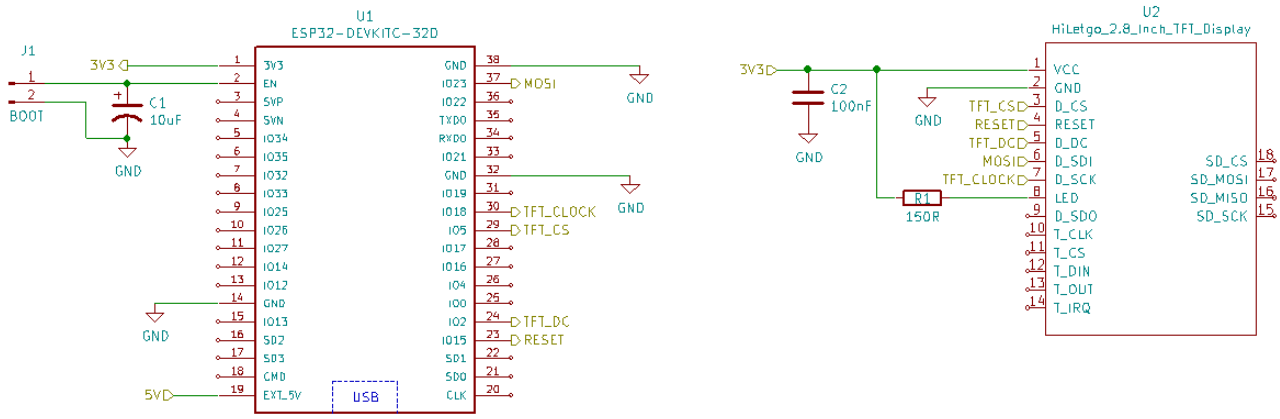
The second and third versions are both based on the ESP8266; one uses the Hiletgo ESP8266 NodeMCU board and the other the ESP8266 D1 Mini board.

Other than using different processors, the circuits for all three versions are essentially the same except the ESP32 and ESP8266 versions use different GPIO pins for the display.

There is no particular reason to choose to build one over the others except possibly what processor you might already have in the parts bin!

ESP32 DEVKIT-32D Version

Here's the schematic for the Version 3.0 ESP32 DEVKIT-32D PCB:

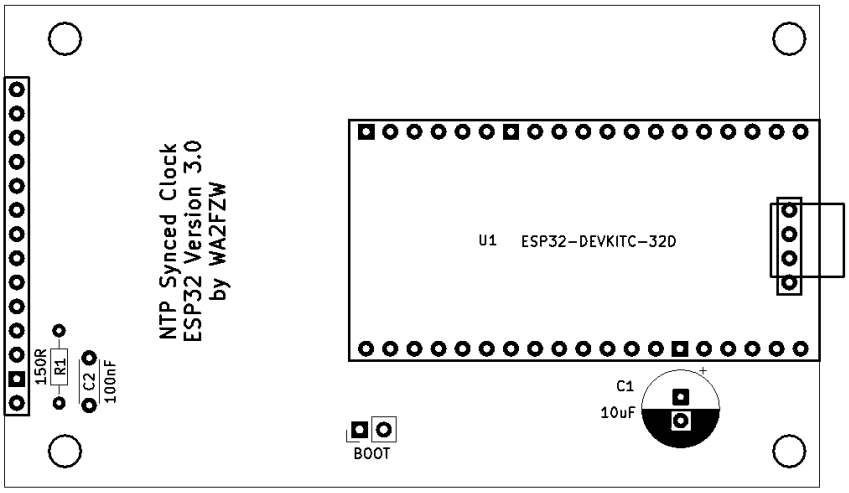


It's pretty simple; just the processor, the ILI9341 TFT display and a couple of other parts.

Perhaps the only thing that needs some explanation is the 10uF capacitor. Unlike the Arduino processors, some versions of the ESP32 require you to operate the ‘BOOT’ or ‘EN’ button (I forget which) in order to load new software. Placing the capacitor between the ‘EN’ pin and ground seems to eliminate the need to do that; makes it much easier to update the software when the unit is in a box.

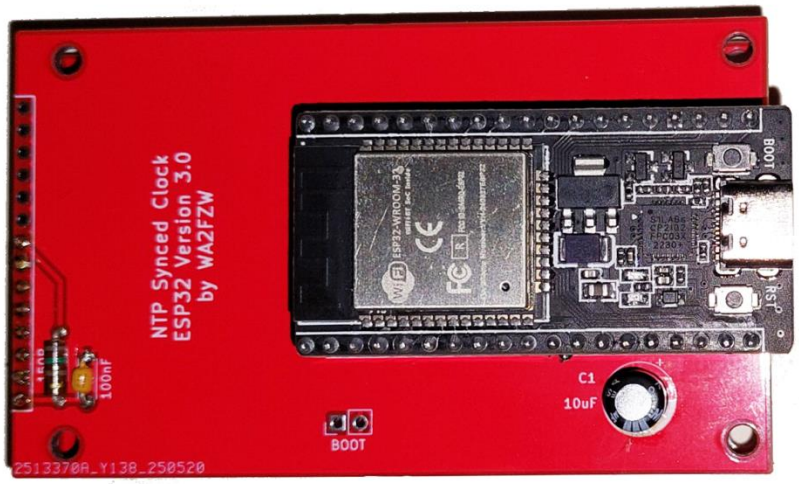
The original circuit board included a 5V regulator so the clock could be powered from a 12V source. I only know of one person who built the original version that used a power source other than the USB connection, so I eliminated the regulator on this version.

Here’s what the Version 3.0 ESP32 DEVKIT printed circuit board layout looks like:



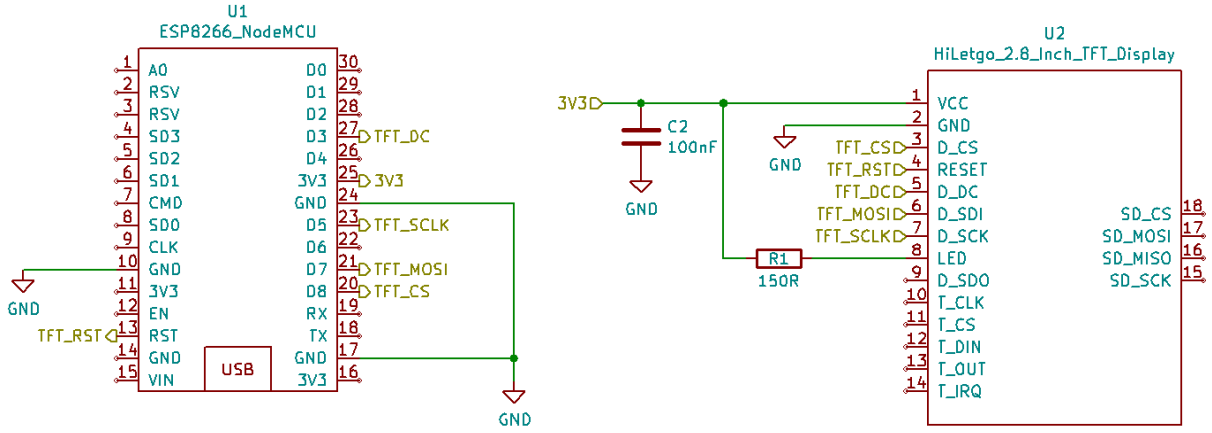
You can connect a pushbutton to the ‘BOOT’ pads to give you the capability to manually re-boot the processor. I only left that on the PCB for compatibility with the earlier ESP32 PCB. I did not include it on either of the ESP8266 versions.

And here’s the assembled circuit board:

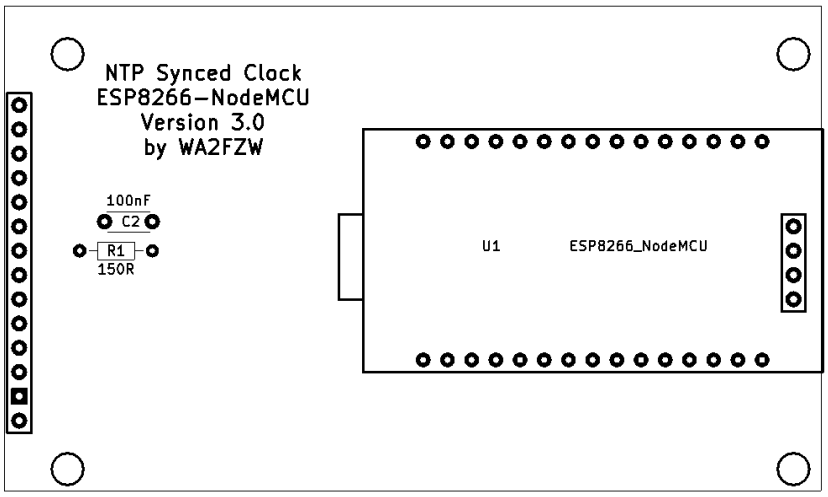


ESP8266 NodeMCU Version

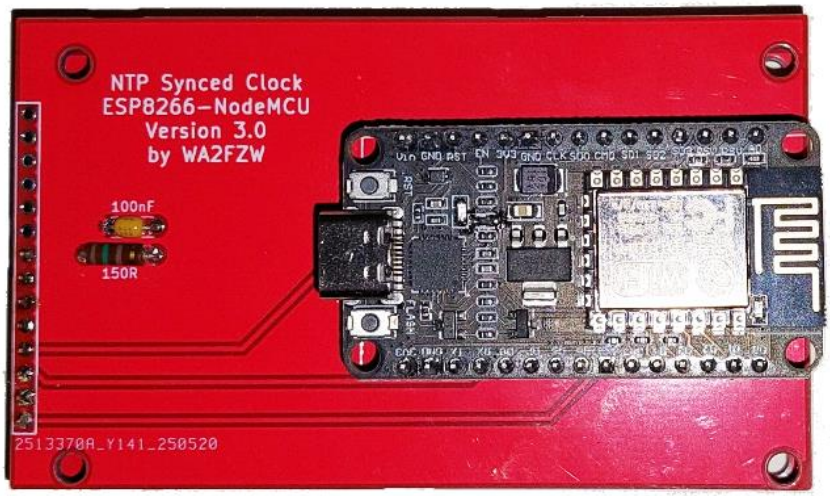
Here’s the schematic for the Version 3.0 ESP8266 NodeMCU board:



Here’s the PCB layout:

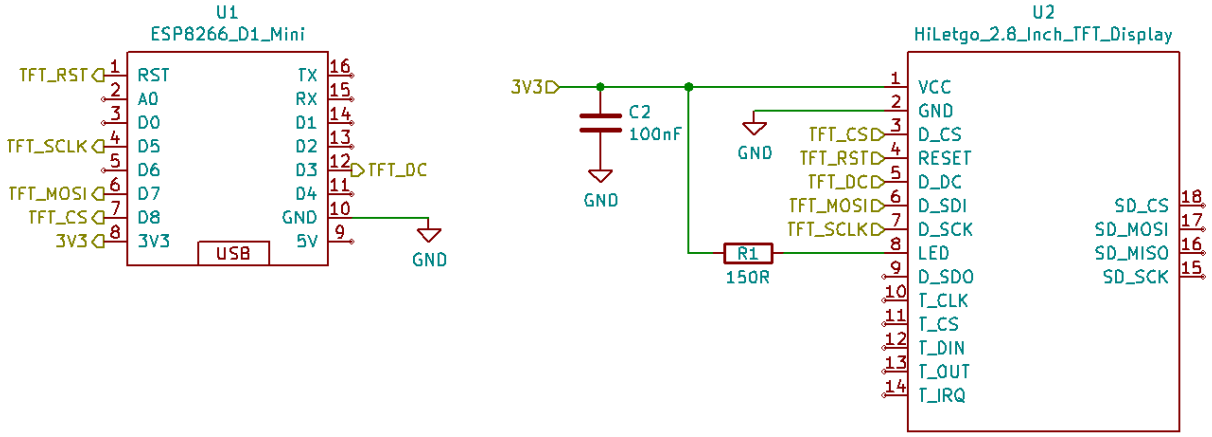


And here’s the assembled circuit board:

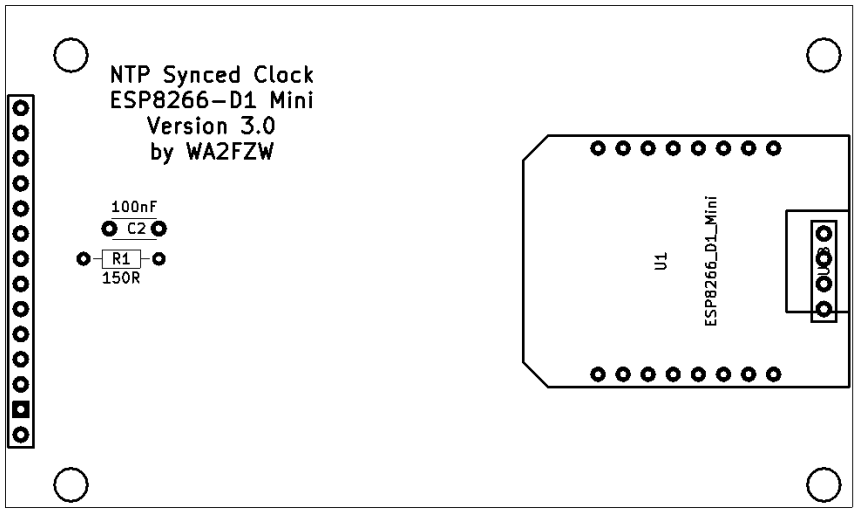


ESP8266 D1 Mini Version

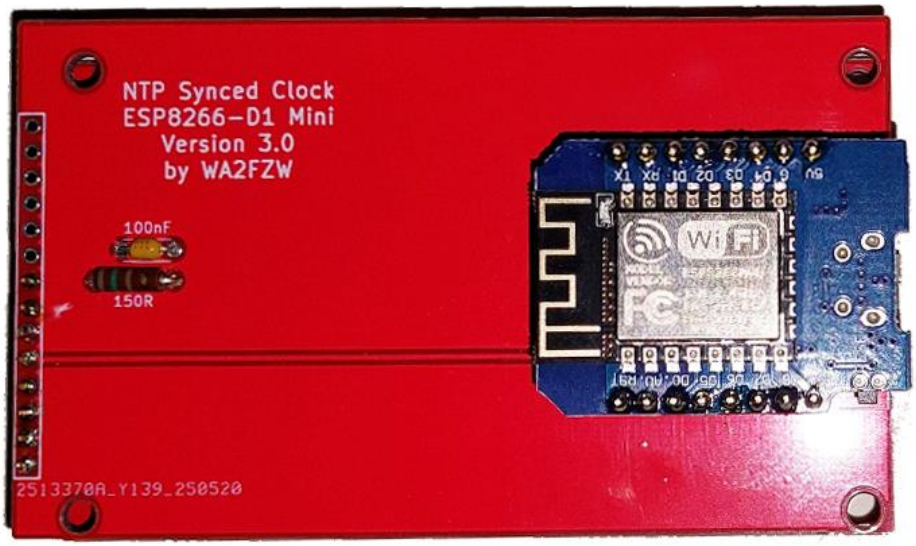
Here's the schematic for the Version 3.0 ESP8266 D1 Mini board:



Here's the PCB layout:



And here’s the assembled circuit board:

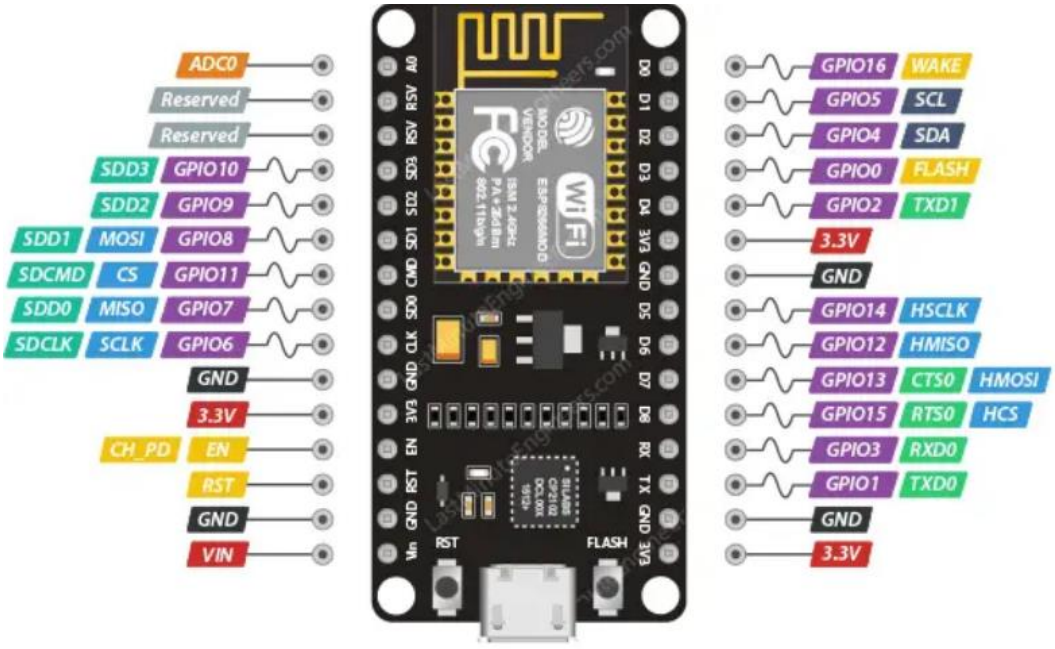


Similarities and Differences

All three circuits are essentially identical with the exception of the 10uF capacitor used on the ESP32 version to allow software to be loaded without manual intervention. It is not needed on the ESP8266 versions.

The ESP8266 modules are a bit strange in how the pins are designated. Whereas on the ESP32 DEVKIT board, the pins are labeled with the actual GPIO pin numbers, on the ESP8266 modules the pins have labels like ‘D0’, ‘D1’, etc. Those numbers are *NOT* the GPIO pin numbers!

The following pictures show the pin mapping for the ESP8266 NodeMCU module and the ESP8266 D1 Mini modules that the PCBs are designed to use:



The mappings for the pins used for the clock are the same for both ESP8266 modules.

Original Versions

The Gerber files for Bruce's ESP8266 version of the clock along with the Gerber files and documentation for my original (Version 1.1) ESP32 version (with the voltage regulator) are still available on Bruce's Github page.

This version of the software will run on either of those versions.

Not a Builder?

If you'd like to have the clock, but aren't into building things, the software can also be run on what's commonly known as the 'Cheap Yellow Display'.

Some notes of caution about the Cheap Yellow Displays, however. They are not all the same. I spent a couple of weeks working with Glenn (VK3PE) trying to figure out why some of his worked and some didn't.

His CYDs had 2 issues; on some of his displays. 3/4 of the screen was displayed correctly and 1/4 of the screen was rotated 90°. On some, the colors were wrong (inverted).

The 'ESP32 Cheap Yellow Display.h' file I provided on Github for the TFT_eSPI library has the options to solve those problems and appropriate comments explaining the options to try should your display have the same problems.

There might be other bogus versions out there with other problems. My CYD displays came from Amazon and those worked correctly without the options.

The Software

As mentioned in the *Introduction*, the software works equally well on either the ESP32 or the ESP8266 processors. There are a couple of places in the software where connections to the 'hamqsl.com' website have to be handled differently, but the Arduino IDE compiler handles those differences transparently to the user, provided the proper board type is selected in the IDE's board manager. The proper selections are: 'ESP32 DEV Module' or 'Generic ESP8266 Module'.

The software will not only work on my versions but on the ones built by the California QRP Club and W8BH's versions.

GPIO Pin Assignments

There are no GPIO pin assignments in the software, but rather those are defined in the 'User_Setup.h' file that is included in the TFT eSPI library which handles the display.

There a couple of ways to set the library up appropriately. The important things that must be set in the library are the GPIO pins used for the display and the selection of the proper display.

After installing the library (using the 'Library Manager' option under the IDE's 'Tools' menu), you will need to edit the 'User_Setup.h' header file in the TFT_eSPI library folder. Modify the definitions as shown here for the processor being used.

For the ESP32 DEVKIT:

```
// Only define one driver, the other ones must be commented out
 #define ILI9341_DRIVER
```

The GPIO pins I used on the original ESP32 PCB and kept the same on the Version 3.0 PCB are different that the defaults in the TFT_eSPI library 'User_Setup.h' header file. Edit that file in the TFT_eSPI folder and set the following GPIO pin assignments for the ESP32 version:

```
// For ESP32 Dev board (only tested with ILI9341 display)
// The hardware SPI can be mapped to any pins

     #define TFT_MISO 19
     #define TFT_MOSI 23
     #define TFT_SCLK 18
     #define TFT_CS    5
     #define TFT_DC    2
     #define TFT_RST  15
//   #define TFT_RST  -1
```

Near the end of the file, you will also see:

```
#define SPI_FREQUENCY  27000000
// #define SPI_FREQUENCY  40000000
```

The bus frequency should work just fine at 40000000 so you can try changing that.

The GPIO pin assignments if you are building either of the ESP8266 versions the ESP8266 in the 'User_Setup.h' file should be as follows:

```
// ###### EDIT THE PIN NUMBERS IN THE LINES FOLLOWING TO SUIT YOUR ESP8266 SETUP ######

// For NodeMCU - use pin numbers in the form PIN_Dn where Dn is the NodeMCU or D1 Mini module pin designation as described above.

#define TFT_MISO  PIN_D6
#define TFT_MOSI  PIN_D7
#define TFT_SCLK  PIN_D5
#define TFT_CS    PIN_D8
#define TFT_DC    PIN_D3
#define TFT_RST   PIN_D4
```

There is a way to have multiple setup files in the 'User_Setups' folder in the TFT_eSPI library directory. This can be done by modifying the 'User_Setup_Select.h' file in the TFT_eSPI directory. That file has the instructions in the comments on how to set that up.

Taking this approach greatly simplifies changing configurations for multiple projects. I currently have seven different setup files for different projects.

The 'User_Setup.h' file for the Cheap Yellow Display is the same as that for the ESP32 DEVKIT except the GPIO pin assignments need to be:

```
#define TFT_MISO 12   // Read data from the display (not used)
#define TFT_MOSI 13   // Data going to the display
#define TFT_SCLK 14   // Data clocking
#define TFT_CS   15   // Chip select control pin
```

```
#define TFT_DC    2   // Data Command control pin
#define TFT_RST  -1   // Display RESET connected to ESP32 RST
```

The following definitions also need to be in the file:

```
#define TFT_BL   21             // LED back-light control pin
#define TFT_BACKLIGHT_ON HIGH   // Level to turn ON back-light
```

Note the issues described above with the Cheap Yellow Displays.

The properly configured user setup files for all versions are available on Github. Those files can be placed in the 'User_Setups' folder in the TFT_eSPI library folder and selected via changes in the 'User_Setup_Select'h' file. Instructions on how to set it up this way are in the 'User_Setup_Delect'h' file provided with the TFT_eSPI library.

Definitions in the UserSettings.h Header file

The first thing in the 'UserSettings.h' file (in the project directory) is a list of available WiFi networks each of which is identified by its SSID (Service Set Identifier). Each also has a password. Here's what it looks like in the software as distributed:

```
wifiData ssid_pwd [] =
{
      "SSID_1", "PWD_1",
//    "SSID_2", "PWD_2",
//    "SSID_3", "PWD_3",
//    "SSID_4", "PWD_4",
//    "SSID_5", "PWD_5",
};
```

You must define at least one network or the clock won't work at all!

Simply edit them to change the 'SSID_n" and 'PWD_n' strings to valid information and if more than one remove the '//' (comment indicator) at the beginning of the lines containing valid networks; for example:

```
wifiData ssid_pwd [] =
{
      "My_House", "12345",
```

```
      "My_Garage", "54321",
//    "SSID_3", "PWD_3",
//    "SSID_4", "PWD_4",
//    "SSID_5", "PWD_5",
};
```

You can also add more if you like! Just add more lines like those already there.

In Version 3.1, I added the ability to display a rotating sequence of different local timezones. The instructions for how to change the list are in the ‘UserSettings.h’ file.

The rest of the ‘UserSettings’ file contains a number of definitions that allow the user to customize the display. Some are optional and some are required; those that you must set correctly are identified with an asterisk:

timeZones	This array contains the list of local timezones to be displayed. Instructions on how to modify the list are in the comments.
tzCount	Defines the number of entries in the ‘timeZones’ list. It is computed using the ‘ELEMENTS’ macro.
TZ_INTERVAL	Defines how long each timezone will be displayed. As distributed, it is set to 5 seconds. It can be changed to any number between 1 and 30, but should be a number that divides evenly into 60.
TITLE	The specified string will be displayed in the local time header. It can be anything you like as long as it fits.
BAUDRATE *	The baud rate for the serial monitor, which can be used for debugging if necessary.
SCREEN_ORIENTATION *	The display needs to operate in landscape mode; the value must be ‘1’ or ‘3’. Which one depends on which way you mounted the display in your enclosure.

LOCAL_FORMAT_12HR	When ‘true’, the local time is displayed as AM or PM. When ‘false’ the local time is displayed in 24 hour format.
UTC_FORMAT_12HR	When ‘true’, the UTC time is displayed as AM or PM. When ‘false’ the UTC time is displayed in 24 hour format.
DISPLAY_AMPM	Only applies when 12 hour format is selected for one or both time displays. When ‘true’ shows ‘AM’ or ‘PM’ along with the time.
HOUR_LEADING_ZERO	When ‘true’ a leading zero is displayed when the hour is a single digit; i.e., ‘01:00’ vs. ‘ 1:00’.
DATE_LEADING_ZERO	When ‘true’ a leading zero is displayed when the date is a single digit; i.e., ‘Feb 07’ vs. ‘Feb 7’.
DATE_ABOVE_MONTH	When ‘true’ the date is displayed above the month; i.e., ‘12 Feb’ vs. ‘Feb 12’.
PRINTED_TIME	Can be set to ‘0’, ‘1’ or ‘2’ as explained in the comments in the header file to display local time, UTC time or no time on the serial monitor.
TIMECOLOR DATECOLOR LABEL_FGCOLOR LABEL_BGCOLOR	These set the colors used for things on the display. Feel free to play with them if you like.
COLOR_NORMAL COLOR_MEDIUM COLOR_HIGH	These color definitions are used in displaying some of the solar data items based on defined levels. I may add some others if I decide that some of the solar data items should have more than three levels.

MEDIUM_K HIGH_K	The NOAA website shows bar graphs for the ‘A’ and ‘K’ indices. When ‘K’ is 4 or greater, the bar will be yellow and when ‘K’ is 5 or greater the bar will be red. The program is set up to comply with those colors, but you can change them.
MEDIUM_K HIGH_K	The NOAA website shows bar graphs for the ‘A’ and ‘K’ indices. When ‘A’ is 20 or greater, the bar will be yellow and when ‘A’ is 30 or greater the bar will be red. The program is set up to comply with those colors, but you can change them.
MEDIUM_SFI HIGH_SFI	The settings for these breakpoints are arbitrary. Most of my operations are VHF and UHF. When the SFI is high, there is a good chance of TEP (Trans Equatorial Propagation) or F2 propagation on 6 meters.
SHOW_SFI SHOW_GMF SHOW_S2N SHOW_AUR SHOW_SSN DATA_ITEMS	These allow the user to select which items of the solar data should be displayed and in which order. The comments in the file explain how to set them up. The comments also explain how to add other items but note that requires some changes within the actual code.
CYCLE_TIME	This sets how many seconds each displayed solar data item will remain on the screen. The number has to be something that evenly divides into 60 or one item will get a shorter time. Good choices are 2, 3, 4, 5, 6 and 10. If you set it to zero, nothing will get displayed.

There are some other definitions in the '.ino' file; you shouldn't change those.

More specific information about the meanings of the solar data values can be found on the *Glossary* page of the 'hamqsl.com' website.

The Certificate.h File

This file (in the project directory) contains the SSL certificate for 'hamqsl.com'. Should the website update the certificate, this will need to be updated as well. Other than that, don't touch it!

I assume that if it changes, we will have to get the new one from Paul (N0NBH).

Function Descriptions

The following descriptions give a brief overview of what each function does in the overall scheme of things. Comments in the code give a more thorough description of how they actually work.

The list is in order of appearance in the code.

Setup

Setup calls a series of other functions that setup the list of the solar data items to be displayed, start up the serial monitor, establish the WiFi and NTP connections and paint the startup screen and fixed parts of the running display.

loop

The loop function runs forever. It gets time updates from the NTP server and updates the times on the screen every second. It also calls *ShowNextData* to display the next selected solar data every 'CYCLE_TIME' seconds.

BuildDataItemList

This function looks at the ‘SHOW_xxx’ definitions in the ‘UserSettings.h’ header file and adds pointers to the functions that actually display the selected items to the ‘dataItems’ array.

Detailed instructions as to how to set the list up are in the comments in the ‘UserSettings.h’ file.

ShowSplash

This function displays screen showing the program title and author credits.

StartupScreen

This function displays the mostly empty startup screen which shows the user selected ‘TITLE’ at the top.

The blank areas will be filled in with the WiFi and NTP connection status.

ShowConnectionProgress

This function establishes the WiFi and NTP connections and shows the status of each on the startup screen.

In Version 3.0 of the software, we allow the user to establish a list of more than one WiFi network as described above.

If there is more than one entry, the function will cycle through the list until a WiFi connection is established. If there is only one entry, that network will be tried until a connection is established. It will try each network 10 times before moving onto the next one in the list.

The connection statuses will be displayed on the screen.

If the user fails to have any networks in the list, the error message: "**NO WIFI NETWORKS DEFINED!**" will flash on the screen forever.

NewDualScreen

This paints the static parts of the display; time header blocks and ‘TITLE’.

UpdateDisplay

This function sequences through the various functions to update all of the information on the clock.

ShowClockStatus

This function checks the WiFi and NTP server connection statuses and modifies the indicator at the right side of the local time header block as necessary.

If NTP updates have been received in the last hour, the background of the WiFi indicator is green; if the NTP data is more than one hour old, the indicator will be orange and if the data is more than 24 hours old, it turns red.

This was modified in Version 3.0. The original code displayed the negative of the WiFi RSSI (Received Signal Strength Indicator). The RSSI is measured in dBm, so by displaying the negative value, the numbers were all positive which was very misleading as one would expect a higher number to be better which was the exact opposite.

The actual RSSI in dBm is now shown in the indicator.

If the WiFi connection is lost, the startup screen is re-displayed and the error message: "**LOST WIFI CONNECTION!**" will flash on the screen for 10 seconds. After that, the program will execute a software re-boot and try to re-connect. That was the easiest way to recover.

If you have more than one WiFi network defined, the program will cycle through them until it finds one that works,

ShowAMPM

If the 'DISPLAY_AMPM' definition in the 'UserSettings.h' file is set to 'true', 'AM' or 'PM' will be displayed to the right of the time if the corresponding time is in 12 hour mode as defined by the settings of 'LOCAL_FORMAT_12HR' and 'UTC_FORMAT_12HR' in the 'UserSettings.h' file.

ShowTime

Updates the times.

ShowDate

Updates the dates.

ShowTimeZone

Updates the timezones.

ShowTimeDate

This function calls the above functions that update all the time and date information.

PrintTime

This function can be used to display either the UTC or local time on the serial monitor output depending on the setting of the 'PRINTED_TIME' symbol in the 'UserSettings.h' file.

GetSolarData

This function downloads the latest solar data from ‘hamqsl.com’.

Doing some testing, it looks like the 'AUR', ‘BZ’ and 'SSN' values are updated hourly at about the top of the hour and the ‘SFI’, ‘A’, ‘K’, ‘GMF’ and ‘S2N’ values are only updated every three hours at about the half hour. I don't know about other data items that are not currently being (optionally) reported in the current version of the code.

Paul (N0NBH) has a note on the ‘hamqsl.com’ website requesting that users not try to access the web data more than once an hour. I checked with Paul to see if it would be ok to query the site every half hour and he indicated that even every 15 minutes would not be a problem.

Since the data only seems to change twice per hour, we're only going to poll the solar data shortly after the top of the hour and shortly after the half hour. We're going to do that slightly randomly using a modification of an algorithm provided by Robert (AI6P).

The algorithm generates a pseudo random number based on the second at which the function was first invoked. That number +2 (and +32) become the minutes at which we will query the solar data.

I further modified the function so that if it fails to connect (after 5 retries in rapid succession), it will continue to try every 5 minutes until the connection is established. Once the connection is re-established, the polling will revert to the normal scheme.

If the connection fails, the ‘xmlData’ is set to the string “Missing”. That accomplishes two things; first since the ‘xmlData’ is not a null string, we won’t retry everytime the time changes. Secondly, since there is no valid data in ‘xmlData;, all the displayed information will show as ‘??’ indicating there is no valid data. How this works is described in detail in the comments in the code.

GetXmlData

GetXmlData is a poor man's xml tag extraction function added by Robert (AI6P) rather than pulling in an entire XML library which really wasn't necessary for this project.

It works by searching for the beginning and terminating XML tags that surround the data item. If the tags are not found, the data will be displayed as '??'. This generally happens when the program is not able to connect to the 'hamqsl.com' website.

ShowNextData

ShowNextData cycles through the list of pointers to the functions that display the selected items from the 'xmlData' received from 'hamqsl.com' every 'CYCLE_TIME' seconds.

Instructions on how to establish the list can be found in the 'UserSettings.h' file.

ShowSFI

If selected for display, *ShowSFI* displays the Solar Flux Index and 'A' and 'K' readings in the UTC time header block.

The displayed information is color coded based on the settings of a number of symbols in the 'UserSettings.h' file. The explanations of how to set the colors up is in that file.

ShowGMF

If selected for display, *ShowGMF* displays the Geomagnetic Field status. The display data is not currently color coded.

ShowS2N

If selected for display, *ShowS2N* displays much noise (in S-units) is being generated by interaction between the solar wind and the geomagnetic activity.

ShowAUR

ShowAUR displays the ‘AUR’ and ‘BZ’ values.

‘AUR’ is calculated from the current hemispheric power value (0-150 GW) to give the old reported scaled factor value from 0 to 10++. Indicates how strong the F-Layer ionization is in the Polar Regions. Higher values cause auroral events (including northern/southern lights) to move to lower latitude.

When the ‘BZ’ is negative, it cancels out earth’s magnetic field, which increases the impact of solar particles in the ionosphere, which in turn increases the possibility of auroral activity.

ShowSSN

The ‘SSN’ is the current number of sunspots observed.

ClearSolarData

This function simply erases any previously displayed solar data in preparation for displaying the next item.

Known Bugs and Glitches

The value being displayed for ‘AUR’ may or may not be correct. The values from ‘hamqsl.com’ do not match the values shown on ‘DxMaps.com’.

Paul (N0NBH) is looking into this.

Possible Enhancements

None that I can think of for now!

Suggestion Box

I welcome any suggestions for further improvements. Please feel free to email me at WA2FZW@ARRL.net.

Bill of Materials

Here is a list of the parts you will need and in many cases, links to where you can get the less common parts.

There are Gerbers for three different PCBs available on Github depending on which processor you choose. Other than the PCB and processor, the other parts are common to all three versions.

The PCB		Choose the correct PCB for the processor you intend to use. The Gerber files are available on Github.
U1	Processor	Pick the one you want! All are available from Amazon. ESP32 DEVKIT-32D ESP8266 NodeMCU ESP8266 D1 Mini
U2	2.8” ILI9341 TFT Display	Available from Amazon or possibly on eBay
R1	150 Ohm 1/4W	
C1	10uF 25V Electrolytic	PCB assumes 0.1” lead spacing Only used for the ESP32 version
C2	100nF	
J1	2-pin male pin header	Only used for the ESP32 version *BOOT* connection
	14-pin female header	Connects display to the PCB (you really only need 8 pins)
	4-pin male and female headers	Connects display to the PCB (mechanical support only)