Documentation for HotTimer project

Jack Gilding (maker@backroad.com.au)

Version

Version	Date	Comments
04	2025-10-17	To accompany v0.7 loaded to Github

Table of Contents

Overview

The hardware and software in this project is designed to control the timing of a resistive storage hot water heater to minimise cost. It can be configured in two modes:

- Combined with solar PV it can be used to heat water at times when surplus solar generation is likely to be available.
- In situations where there is not solar PV installed it is designed to heat water only when cheap power is available via a time of use tariff.

In addition the software displays the current tariff so that other loads can be manually controlled to make use of cheaper times and avoid more expensive times.

The software is currently hard-coded for Tasmanian residential time of use tariff (Aurora tariff 93) but can be configured for other time of use tariffs.

Electricity tariffs in Australia that are time based operate on standard time (ie ignore daylight saving time (DST)). This changes the apparent time at which tariff change. During daylight saving time the main time is show in DST in yellow but the standard time is also displayed in smaller text.

When used in conjunction with solar PV a boost function is available so that water temperature can be boosted if the solar PV is not sufficient. The boost timer will display any time the boost button is pressed, but boost has no effect if the hot water circuit is already powered, either because it is solar boost time or (if set for no solar) during off peak times.

Figure 1: Prototype mounted under bench

Instructions for users

The green LED indicates that power is on.

The red LED indicates that power is being provided to the hot water service. Whether water is actually being heated will be determined by the thermostat in the hot water service.

Date and time are displayed – if the time is in yellow, it is currently daylight saving time, the standard time is displayed in small print.

Without solar

Power will be supplied to the hot water service during off-peak times.

You can maximise savings by using other appliances (eg washing machine, dryer) during off-peak times.

With solar

Power will be supplied to the hot water service during the programmed solar times.

At other times, if the water temperature is too low, you can press the Boost button to turn on the water heater. Each time you press Boost (press black button to right) a set amount of time (default 10 minutes) will be added to the boost time remaining. Press Cancel (black button to left) to cancel the boost.

Setting the time and date

The real time clock (RTC) module in the device keeps pretty accurate time. It will rarely need to be adjusted. If the device forgets the date and time when disconnected the battery in the RTC is probably flat. Replace it with a CR2032 button battery (3V).

Press the navigation switch inwards to enter the mode for setting the date and time.

> Pressing the navigation button up or down will cycle between settings (year, month, date, hour (0-23), minutes and seconds.
>
> Pressing the navigation button left will reduce whichever setting is highlighted, pressing it right will increase the current setting.
>
> Note the limitation below on adjusting seconds.
>
> The day of the week is displayed but cannot be adjusted, it is determined by the date.

Press the navigation switch inwards again to return to normal operation.

Limitations

The device controls power to a hot water service (via a contactor). The device indicates (red LED) whether power is supplied to the contactor, but whether the hot water is actually heating will be controlled by the thermostat on the hot water service. It is not possible to see whether the hot water is heating unless you have separate consumption monitoring.

The real time clock cannot be set to 29 February on leap years.

Switching between solar and without solar modes requires changing the software, it cannot be changed by the user. Also the solar and any additional boost times cannot be set by the user.

Due to software and hardware limitations the device will only work between the years 2020 and 2099.

When adjusting the data and time, the displayed data will not be updated while waiting for the navigation button to be pressed. However the RTC is 'ticking' in the background. As soon as a selection is made, all the settings will be updated to the current time (plus any increase or decrease selected). This will be most obvious if setting the seconds, where, depending on how long it is since the last button press, selecting **more** may jump forward more than a second, and selecting **less** may not seem to be effective.

Do not change the month to a shorter month without checking that the date is still valid.

Notes on the code

Apologies for any non-standard coding, I am self-taught on this.

The naming conventions I used for variables are:

- constants are in caps.
- most variables are global and are declared at the start of the code
- variables that are local to a function start with `f_` and are declared in the function.

In order to correctly calculate the tariff it is necessary to know the day of the week. I could not get the `.dow` function for this to work, so as a workaround I converted the date and time from the RTC to the DateTime variable type and then used the weekday() function in the TimeLib.h library.

Full code

The full code is available at https://github.com/JackGilding/HotTimer.

Settings that require code changes

Set the constant `USE_SOLAR` to activate solar times. When this is set to false, the relay will be on (and hence power to the hotwater cylinder) whenever it is off-peak time.

The constant `RESET_RTC` is used to determine whether to reset the real time clock. Add the current date and time to lines in the `setup()` loop and upload the program once. Set the `RESET_RTC` to `false` and re-run the program so that the time is not reset each time the software is run. The battery in the RTC should keep track of the time while the unit is unpowered.

Now that a setting menu has been added to the software it is not necessary to reprogram the unit to reset the date and time.

Enter one or both solar on and off times. Times should be set in standard (not DST). An on and off must be in the same day (ie it will not work if you try to turn on at 11pm (`hh`=23) and off at 2am (`hh`=02)).

Set the on and off to the same time if you don't want to use one of the two periods.

A typical configuration is on during peak solar time (eg 10am to 4pm). The second period is typically if you think there is a need for an additional late evening or early morning boost (avoid peak times).

There is some code for displaying a shoulder period, but this is not fully implemented as the Tasmanian residential tariff 93 has only peak and off-peak bands.

Notes on coding

Updating all displays every time the second changes resulted in excessive flickering of the display. This is the reason only the display of seconds is updated when the second changes. Many other aspects of the display are updated when the minute changes (even though some things only change when the hour, date, or the start or finish of daylight saving changes).

Hardware and software chosen

Overview of options

The original build was on a Mega with Wi-Fi board (XC4421). This used a Jaycar XC4630 screen (320x240) with an SD card slot and touchscreen capability. The intention was to use the Wi-Fi to adjust the clock and possibly also get data from the Solar Analytics API. The larger screen was to display water tank temperatures. I gave up on this because the Arduino Create Editor stopped recognising the board, programming the Wi-Fi was difficult and the combined cost of all the components was getting too high for the functionality.

I decided to start again with a Uno board. I aimed for the cheapest components. See comments below on individual components. The smaller box is a very tight fit (especially if you decide to include a DC-DC convertor – see notes below) and requires additional soldering. If building from scratch you may find it easier to use the bigger jiffy box and components that can be connected via fly leads rather than soldered wires.

Uno clone from Temu

These were not recognised by the Arduino Create online IDE. The chips on the Temu Uno board do not seem to have part numbers so I tried this advice to install the CH340 driver, without success. The Uno clones probably work fine, but I gave up trying to use them for now.

Contactor

The system operates by feeding 12V to a contactor (ie relay) in the main switchboard which in turn controls the supply of power to the hot water heater. We used the Finder 22 Series contactor which is DIN rail mounted and can be manually switched to be either on or off if the relay function is not being used.

The cable for the 12V input to the contactor should be run to a location where the control unit will be visible and accessible.

The contactor must be installed by a licenced electrician, but once installed allows the control system to be modified and updated without requiring an electrician.

Colour LCD display

A wide variety of colour LCD screens are available. We chose a 128x160 pixels screen from Temu as the best balance of cost and functionality.

Note that the Temu unit does not have pins for fly leads and requires soldering, although two sets of pins are supplied if you do not want to solder directly to the board.

The colour functions in TFT.h seem to accept a single RGB565 hexadecimal input rather than the three separate red, green and blue integer values listed in the documentation. We copied constant values for various colours as shown in the table below.

Some of these colours displayed differently on the Temu 128x160 screen than the colours displayed on a previously used 128x128 pixel screen. It seems that the Temu unit swaps the red and blue values when interpreting the 4 digit hexadecimal value.

Constant name	**Value used on 128x128 screen**	**Value used on Temu 128x160 screen**
BLACK	0x0000 (0,0,0)	0x0000 (unchanged)
BLUE	0x001F (0,0,31)	0xF800
RED	0xF800 (31,0,0)	0x001F
GREEN	0x07E0	0x07E0 (unchanged)
YELLOW	0xFFE0 (31,63,0)	0x07FF (0, 63,31)
WHITE	0xFFFF (31,63,31)	0xFFFF (unchanged)
ORANGE	0xFB60 (31,27,0)	0x037F (0,27,31)

The text functions in TFT.h use characters that are all the same width. Using .textsize() allows the size to be changed. At textsize(1) characters are 6 pixels wide and 10 pixels high. This includes space between characters and between lines. At textsize(2) characters are 12 pixels wide and 20 pixels high. Larger sizes are possible with similar multipliers but are not much use on a 128x160 screen. Allowing a few pixels on either side allows 20 characters (20*6=120) to a line at textsize(1) and 10 characters at textsize(2) (10*12=120).

Wiring of 128x160 display

The wire colours are arbitrary and mainly for my own use.

1.8" Temu 128x160	Uno	Wire colour
GND	GND	green
VDD	3.3V	yellow
SCL	13	brown
SDA	11	white
RST	8	grey
DC	9	purple
CS	10	blue
BLK		not connected

Real time clock

To keep track of the time we originally added a real time clock (RTC) module (Jaycar XC4450). Unfortunately this is was discontinued and we tried the Jaycar XC9044 but it is expensive and the battery is spot welded so not easily replaced. We settled on the Temu AT24C32 IIC, however note that you will need to buy CR2032 batteries separately.

The Temu RTC is much larger than the Jaycar XC9044 (see photo below) so if you are trying to build into the smaller jiffy box it might be best to use the Jaycar RTC. The Temu RTC board also has an EEPROM on it, so this may be useful for future improvements, eg recording and altering solar boost time via a menu.

Whatever RTC module you buy, check that it has the DS3231 chip which is highly accurate and is supported by the DS3231.h library.

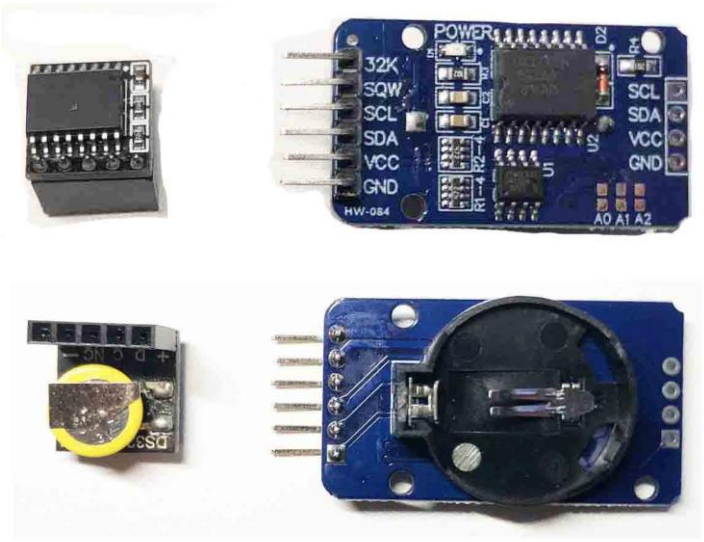


Figure 2: real time clocks: Jaycar XC9044 (left),Temu AT24C32 IIC (right)

The following table shows how the components are connected to the Uno.

Jaycar XC9044	Temu AT24C32 IIC	Uno	Wire colour
NC			not connected
-	GND	GND	black
+	VCC	3.3V	red/orange
D	SDA	A4	yellow
C	SCL	A5	blue
	SQW		not connected
	32K		not connected

12V supply

These Temu 12V power supplies are cheap but make sure you order the 12V version not the 24V version. Also make sure anything you use has passed Australian safety standards. You may have something sitting around, or you can source one from the local tip shop. Make sure it is 12V (many computer power supplies are 19V) and is capable of at least 1 Amp.

DC to DC converter

Based on this advice that it was inadvisable to run an Arduino for long periods with a 12V supply, I added a DC to DC step down voltage regulator (Jaycar XC4515, an identical unit "LM2596 DC-DC Buck Converter" is available cheaper from Temu).

Note the instruction in the data sheet: “With new modules, the pot might be at its highest setting. You will have to turn the potentiometer around 10 times counter-clockwise before you see the voltage change.”

I set the output voltage to 6V.

The 12V input is fed straight to the relay without being reduced.

Navigation switch

I tried the Adafruit version of the 5-way navigation switch because you can add on a nice rubber cap. I bought the Adafruit products from Core Electronics because shipping a direct order from USA is expensive and complicated.

The pins on the small switch do not easily fit into standard perf board (2.54mm pitch). Also the switch is not labelled so you need to test which way up to mount it before soldering wires to the pins. I drilled a hole, then used a Stanley knife to cut four corners. The switch was then glued into the hole with Araldite, being careful not to block the switch action. I mounted the switch slightly forward of the front surface of the jiffy box so that the rubber cap was free to move.

This very small and cheap 5-way switch is difficult to use and mount, so I tried the larger board mounted 5 way Tactile Navigation Button Module from PhippsElectronics.com in NSW (2 for 15.95). Identical modules are available more cheaply in China (eg from makerfabs.com) but there are minimum quantities and shipping costs to consider.

The module version is a little more robust and has the advantage that it can be connected with socket leads, but it is still a challenge to mount so the button is accessible and free to move in all four directions. I drilled a 12mm hole and cut some 3mm nylon spacers to reduce the setback from the jiffy box cover. This also required shortening some 3mm nylon bolts.

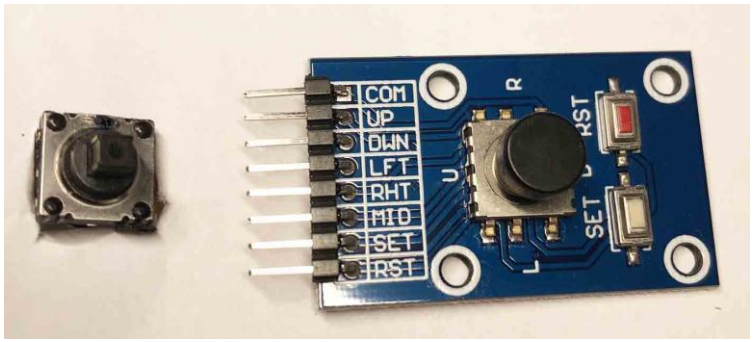


Figure 3: Navigation switches, Adafruit ADA504 (left), 5 Way Tactile Navigation Button Module (right)

Adafruit ADA504 (from front)

Up=PREV=purple=3		push=MENU =orange=2
Left=LESS=green=5		Right=MORE=yellow=6
Down=NEXT=white=4		common=GND=black

5 Way Tactile Navigation Button Module

Pin	Uno	Wire colour
COM (GND)	GND	black
UP (PREV)	3	purple
DWN (NEXT)	4	white
LFT (LESS)	5	green
RHT (MORE)	6	yellow
MID (MENU)	2	orange
SET		not connected
RST		not connected

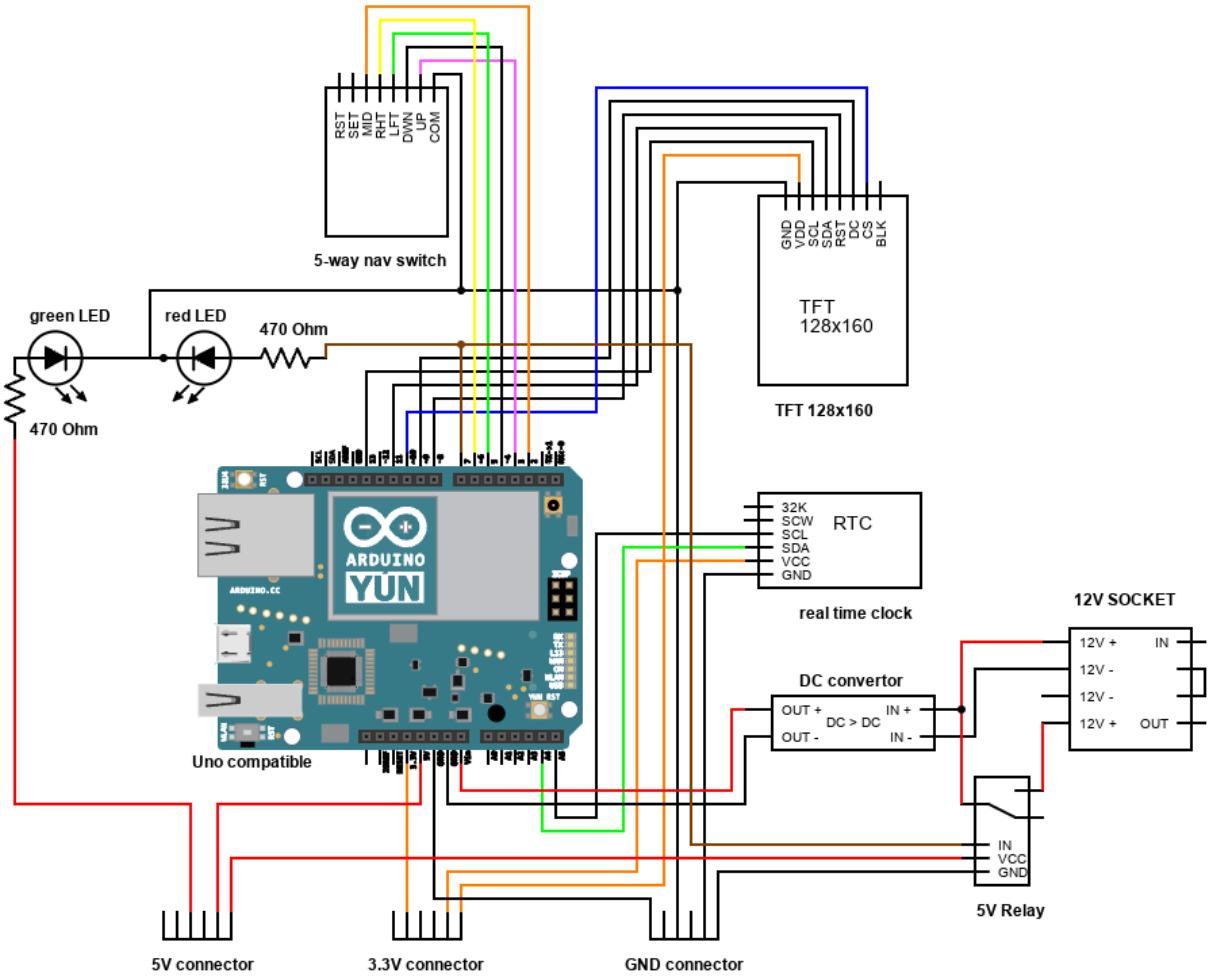
Relay

I inadvertently bought Temu relays that work inversely to the ones I bought from Jaycar, ie connecting signal to ground rather than 5V activates the relay. This could be changed in the code, but as the relay has normally open and normally closed contacts, I just changed the terminal used. The switch contacts (NO and NC in the Jaycar version) are labelled in Chinese characters so trial and error is the easiest way to find the desired contact for the 12V feed to the contactor in the switchboard.

12V plug and socket

I use these to make it easy to remove the control unit and take it to the bench for updates. After investigating various options, the cheapest solution seemed to be the Jaycar HM3104 Header and HM3124 Terminal Block Socket. I glued the header to the outside of the box and use the terminal block for the cables from the 12V supply and the lead to the contactor. I connect the in and out negative terminals on the terminal block. It would also be possible to use the three pole versions (HM3103 and HM3123) since the negative terminal is common to both the in and out circuits.

Circuit



Circuit diagram shows arrangement of components in the prototype rather than a simplified circuit diagram. Top third of diagram is lid components.

Diagram created in Schemeit. Some wire colours do not match documentation.

Parts list

Component / Australian / Cheapest	Australian	Cheapest[1]	Prototype
Arduino compatible Uno board Jaycar XC4410 Temu 3 x UNO R3 for AU$27.15 (but I couldn't get to work)	$38.95	$9.05	$38.95
TFT screen I originally used the bigger Jaycar XC4630 (320x240) screen The 128x160 pixels screen from Temu cost me $9.94	$32.95	$9.94	$9.94
Real Time Clock module Jaycar for Raspberry Pi XC9044 Temu AT24C32 IIC 3 for AU$10.71 + CR2032 battery	$19.95	$4.20	$4.20
DC to DC convertor Jaycar XC4515 Temu LM2596 DC-DC Buck Converter 5 for AU$8.51	$7.95	$1.70	$1.70
5V Relay Board Jaycar XC4419 Temu Heavy-Duty 1-Channel Relay Module Board 5 for AU$9.28	$5.45	$2.00	$2.00
5-way navigation switch Phipps 5-way Tactile Navigation Button Module Adafruit ADA504 via Core Electronics	$8.00	$4.15	$8.00
Jiffy Box - Black - 158 x 95 x 53mm Jaycar HB6011	$6.50	$6.50	$6.50
Plug and socket for 12V connections Jaycar HM3104 Header ($2.15) and HM3124 Terminal Block ($4.95)	$7.10	$7.10	$7.10
	$126.85	**$44.64**	**$78.39**

[1] These are cheapest unit costs and do not account for need for bulk purchase or shipping costs. Where I have used local parts for the cheapest build I have included the Australian price in the Cheapest column. You may be able to source these parts more cheaply.

Additional components

You may already have these. If not some of these purchases will cover multiple projects.

Component / Australian / Cheapest	Australian	Cheapest	Prototype
Jumper leads Jaycar WC6027 Jumper Lead Mixed Pack 100 pieces Temu 120pcs Multicolored Silicone Dupont Wire	$18.95	$8.14	$18.95
230V to 12V DC power supply Tip shop Temu 12V 1A DC Power Adapter	$13.00	$8.73	$8.73
3mm Red LED Jaycar ZD0100	$0.30	$0.30	$0.30
3mm Green LED Jaycar ZD0120	$0.30	$0.30	$0.30
470 Ω resistor for LEDs (2 of) Jaycar RR2766	$0.68	$0.68	$0.68
M2 (2mm) spacers, bolts and nuts White Nylon Screw and Mounting Kit -M2 Core Electronics $17.45 Temu 350 piece set black nylon M2	$17.45	$12.99	$17.45
M3 (3mm) spacers and bolts Jaycar HP0924 $9.95 for 25 and Jaycar HP0140 $2.95 for 10 Temu 350 piece set black nylon M3	$12.90	$11.83	$12.90
	$63.58	**$42.97**	**$59.31**

The Build

I built the prototype so that it could easily be assembled and disassembled, and parts changed. As a result, I used plug and socket leads. I also made the components in the lid of jiffy box so that the lid could be removed. To minimise the number of wires between the lid and the base, the two LEDs, the navigation switch and the TFT display all connect to a single ground wire.

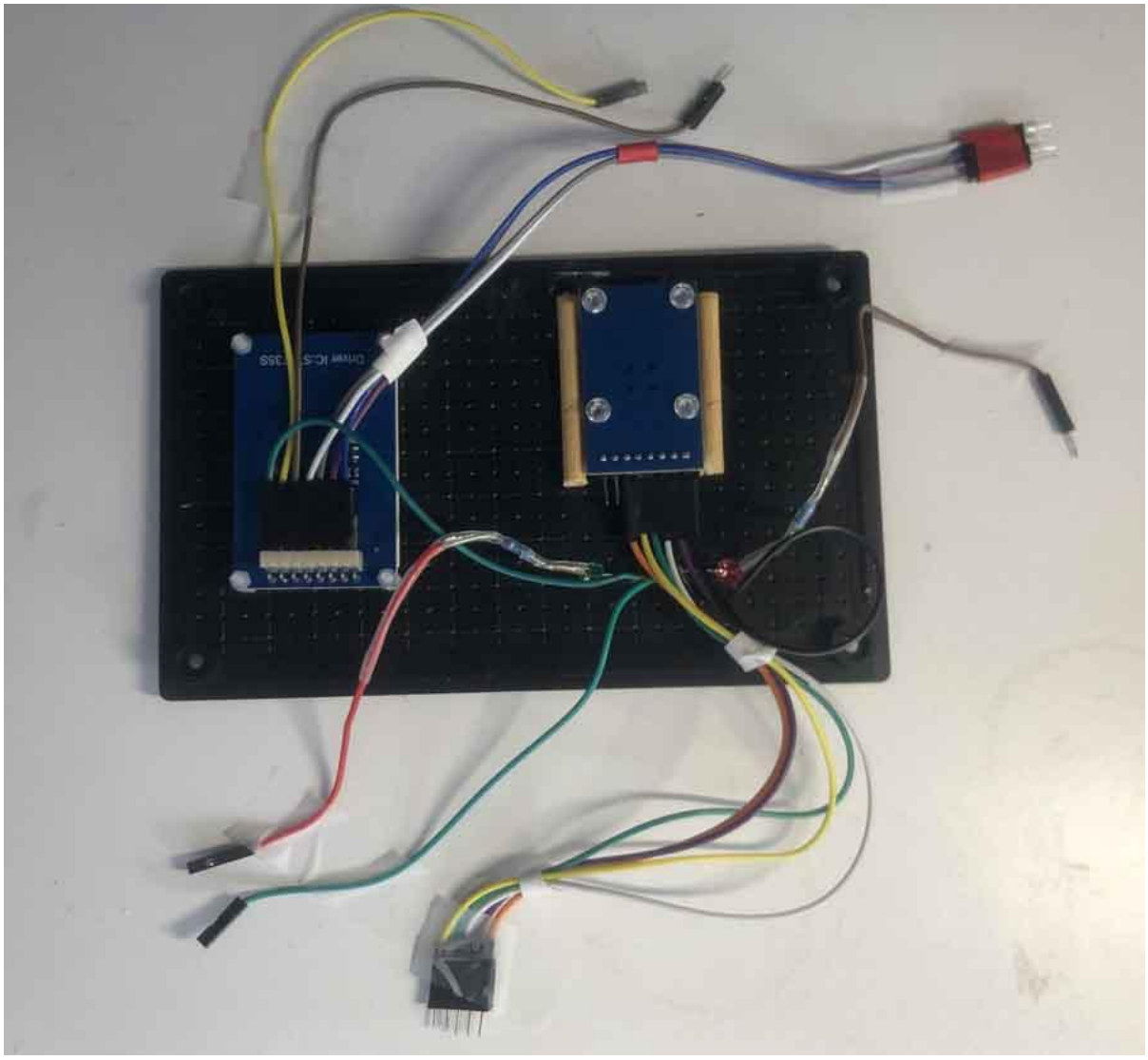


Figure 4: Lid components, showing common GND cable (green) and pins taped in the order they plug into the Uno

You might want to solder some connections to make the build more reliable.

I mounted the main components to model ply (3 mm thick) using Jaycar HP0924 tapped nylon hex spacers and Jaycar HP0140 3x12mm nylon screws. Drilling the baseboard with a 6.5 mm drill bit allows a tight fit and by not pressing the spacers all the way in they stand the circuit boards slightly above the baseboard. If glueing in the hex spacers make sure glue does not get inside the threaded spacer.

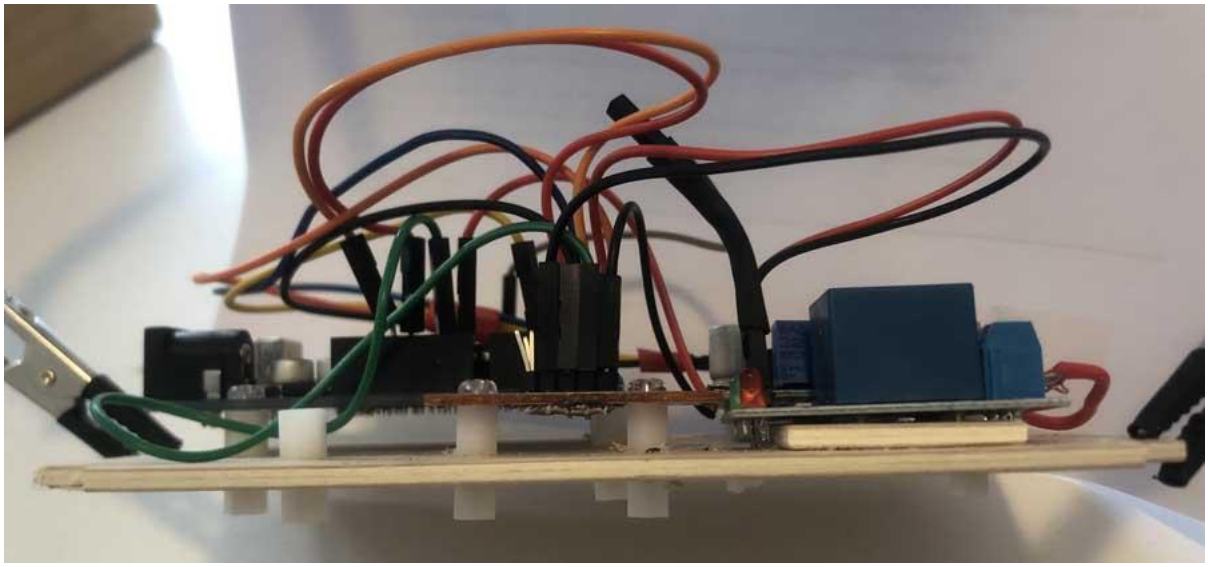


Figure 5: Side view of baseboard showing nylon spacers above and below baseboard. Note that relay does not have mounting holes and is glued to an extra slice of ply.

Unfortunately the TFT screen has 2mm mounting holes so you will need M2 screws and spacers as well. You might choose a different mounting method.

The DC plugs and sockets allow the unit to be quickly removed from its location and taken to the workbench for re-programming.

I did successfully build the unit in the smaller (130 x 68 x 44mm) jiffy box but it was a very tight squeeze and required soldering a lot of wires to curved pins in order to reduce the height of wires into the Uno. I also used the smaller 5-way switch. If you really want to use the smaller box, consider using the smaller RTC module and dispensing with the DC to DC convertor.

I would recommend using the larger (158 x 95 x 53mm) jiffy box which allows more room for larger components and wires.

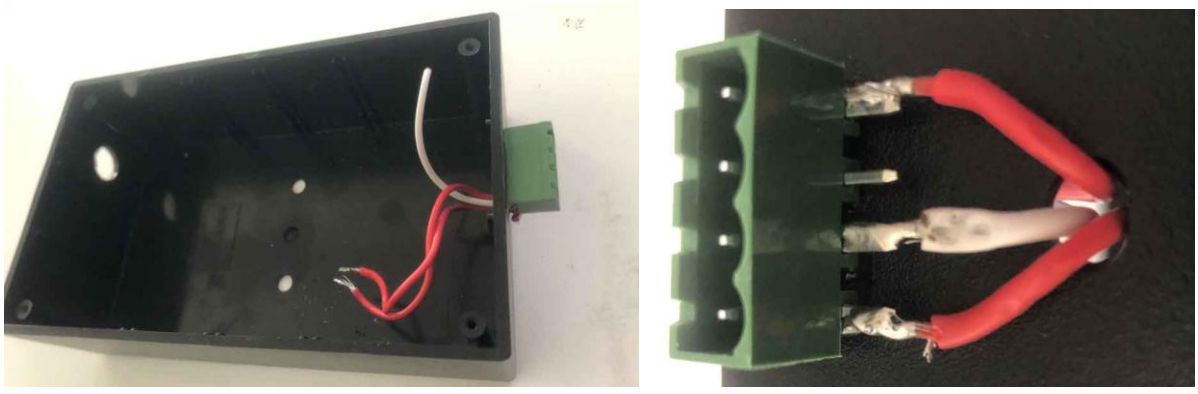


Figure 6: Jiffy box base showing hole for USB and glueing of Pluggable Header

Construction order

- Create openings in lid: 28x36mm rectangle for display, holes for LEDs and navigation switch
- Mount the components in the lid
- Decide how the jiffy box will be mounted (I used a 75x75mm angle bracket with holes 41mm apart mounted under a bench).
- Decide how the baseboard will be fixed inside the jiffy box (I used two metal thread screws from the back of the jiffy box into the diagonal corners of the baseboard.)
- Drill the base of the jiffy box with holes for USB cable and 12V wires, plus mounting holes.

- Glue to Pluggable Header for 12V connections to the outside of the jiffy box.
- Fix the baseboard inside the jiffy box.
- Complete the wiring.

Limitations and challenges

Backlight in the display module

I could not find any good documentation on the use of the backlight socket (BLK) on the 128x128 display. Without it connected the backlight was on full which was too bright. I tried using a trim pot as a voltage divider between 3.3v, the BLK socket and GND. In the end by trial and error the best result was obtained using a 10K trim pot between BLK and GND (ie no connection to 3.3V required).

Although the Temu 128x160 display has a backlight pin this does not seem to need to be used.

Hard coding of Aurora Energy Tasmania's tariff 93

The software is written to work specifically with Aurora Energy's tariff 93.

Hard coding of Tasmanian Daylight Saving Times

Dates and times for daylight saving time changes are hard coded in the software for Tasmania. The adjustment is made by adding an hour during DST. The date will not display between 11pm and midnight during DST (it should show the hour as zero, not 24 and roll over the day, but this is hard to calculate).

Credits and copyright licence

Thanks to David Prentice for TFT code and assistance with troubleshooting.

Thanks to the DIYode Garden Time project for the nifty formula for checking if a day is the first Sunday of the month.

Further developments

Possible future developments include:

- Having the option to display the time in 12 hour format.
- Arranging the RTC menu with all values arranged vertically so that the interaction between the navigation switch and the display is more intuitive.
- Allowing user configuration of whether solar is present, solar on and off times and 12/24 hour format selection. (This would require some form of non-volatile memory, eg an SD card or the EEPROM on the RTC.)
- Monitoring the temperature in the hot water tank.
- Using the tank temperature to automatically boost the tank if the temperature gets too low.
- Reading and displaying available surplus solar generation using the Solar Analytics API. This would require WiFi access.