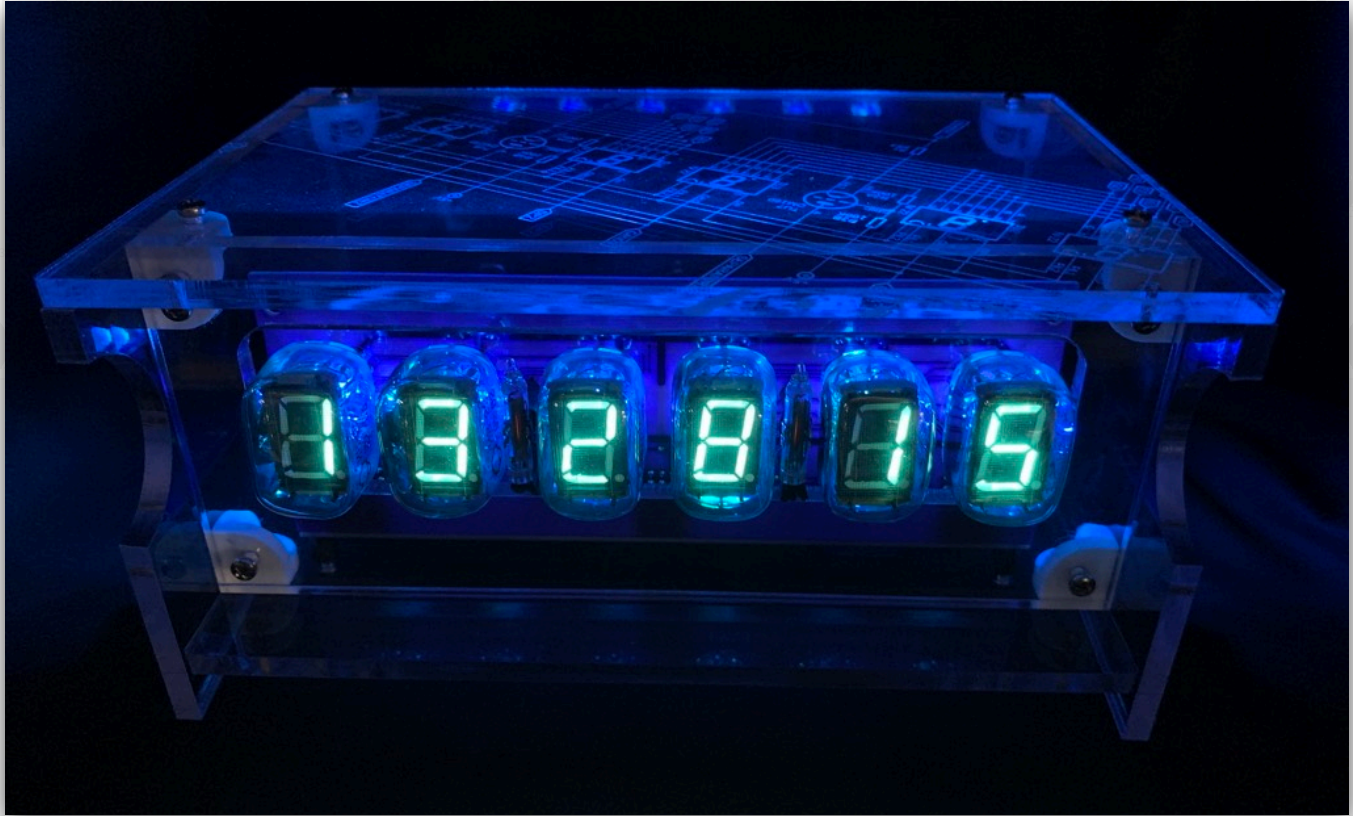


VFD Tube Clock with ESP32 DevKit-C Assembly Manual



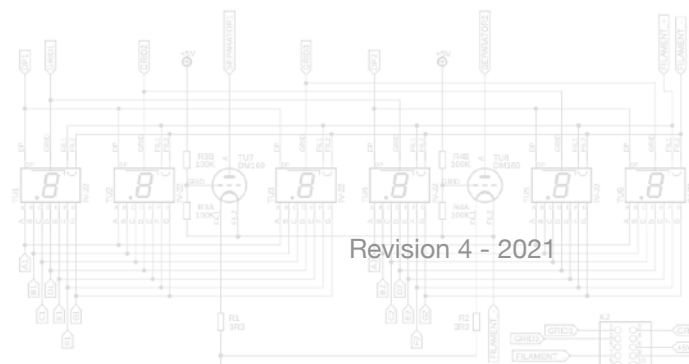
This electronics kit is designed for hobbyists with intermediate experience in assembling electronics and is most definitely not for beginners. It requires knowledge of soldering, programming using the Arduino platform and a good grasp of basic electronics. If this is not you, then please do not attempt to build the kit unless you have qualified assistance.¹ The clock uses a rather old fashioned display tube called a VFD, short for Vacuum Fluorescent Display², either the Russian IV-22 or Chinese YS27-3 varieties.³

Please take your time - it will take approximately four hours to complete this kit if done correctly. It helps to have a clean, tidy work area and quality tools to complete the process.

¹ We don't want you to be disappointed.

² https://en.wikipedia.org/wiki/Vacuum_fluorescent_display

³ The latter is quite a rare tube.



Assemble the parts in the order as stated in the instructions. Read and understand each step before you perform each operation for the best chance of success. If you have any questions at any point, feel free to contact us via email.⁴

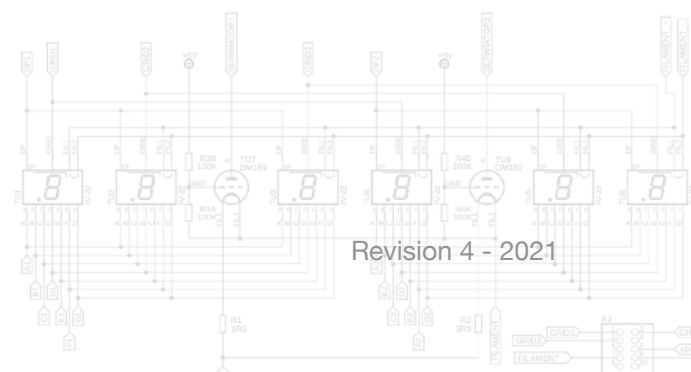
The following tools and materials are required to assemble the clock:

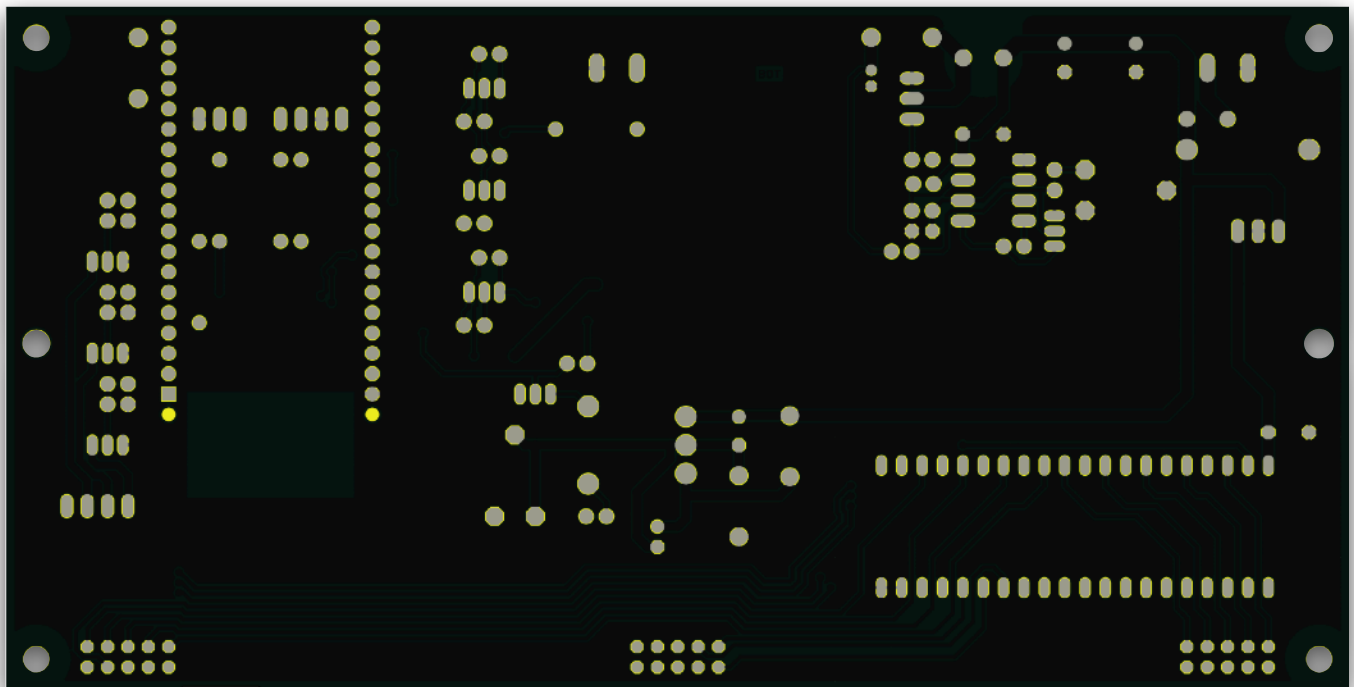
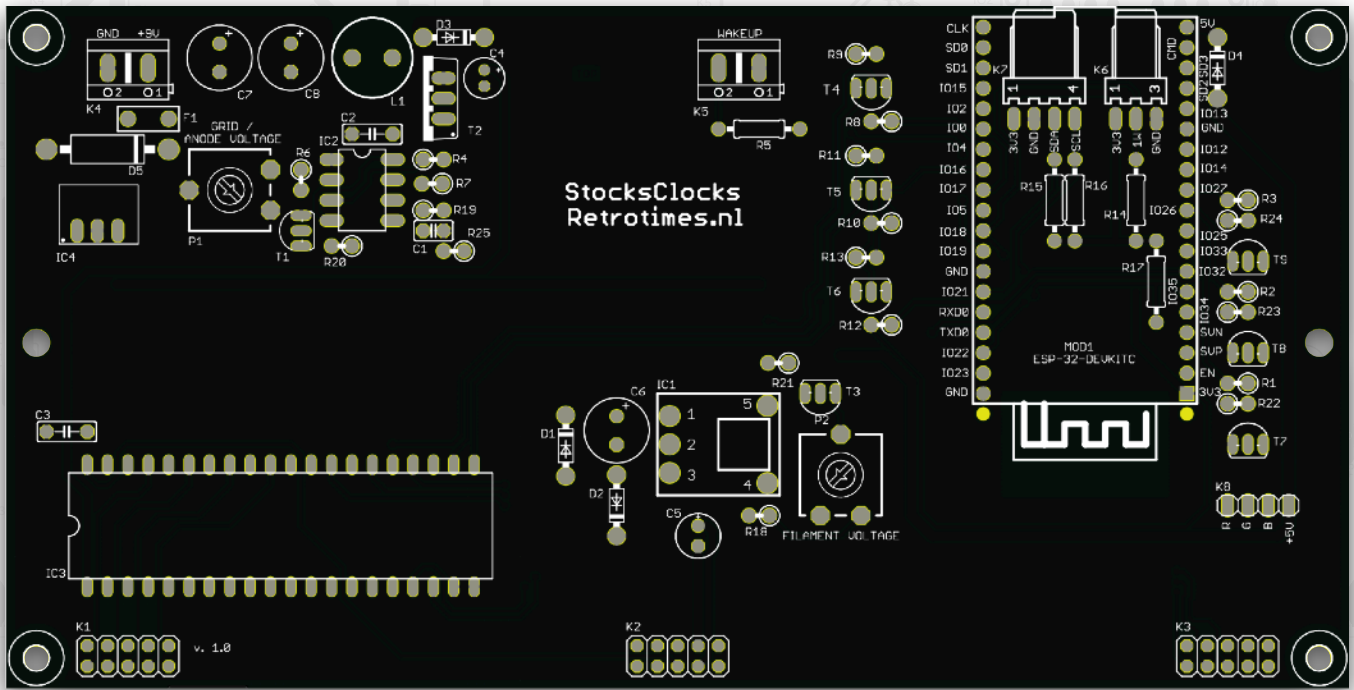
- A good quality soldering iron (25-40W) with a small chisel style tip (2-3 mm).
- Thin solder wire with no-clean flux. Do not use any flux or grease. Do not use lead-free solder.
- A set of small screwdrivers.
- A small wire cutter for electronics.
- Wire strippers.
- Long nose pliers.
- A hot air soldering station / heat gun / lighter (or alternative).
- A multimeter.
- A computer with the Arduino IDE and Arduino Core for ESP32 installed.
- Solderwick (*just in case....*)

It is a good idea to read over the whole manual first to get a grasp of what is involved in the build process. Of note, it is imperative that every resistor is measured using a multimeter *prior* to being soldered into the boards. Tracing incorrect resistor placement can be quite challenging....⁵

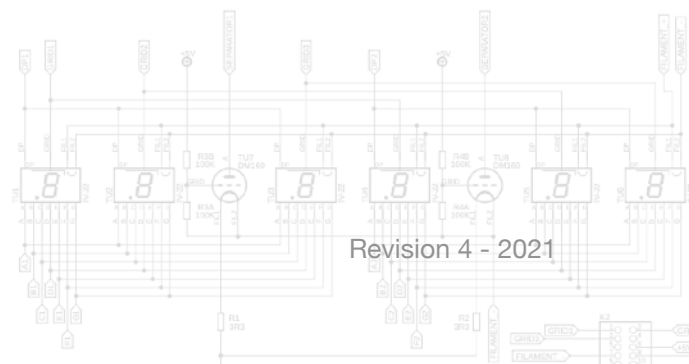
⁴ stocksclocks@gmail.com

⁵ Ask me how I know.... :)

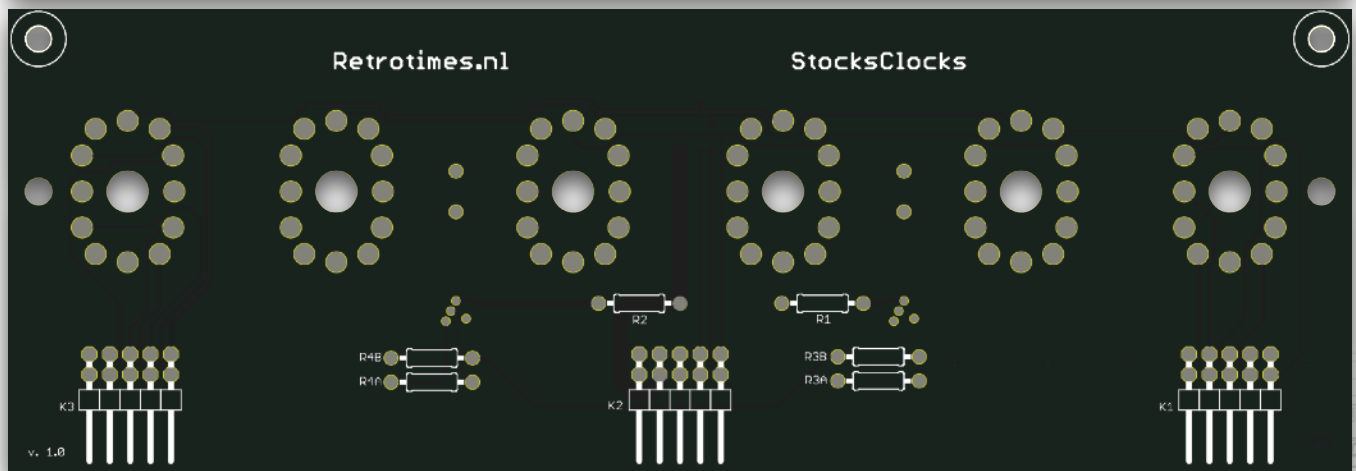
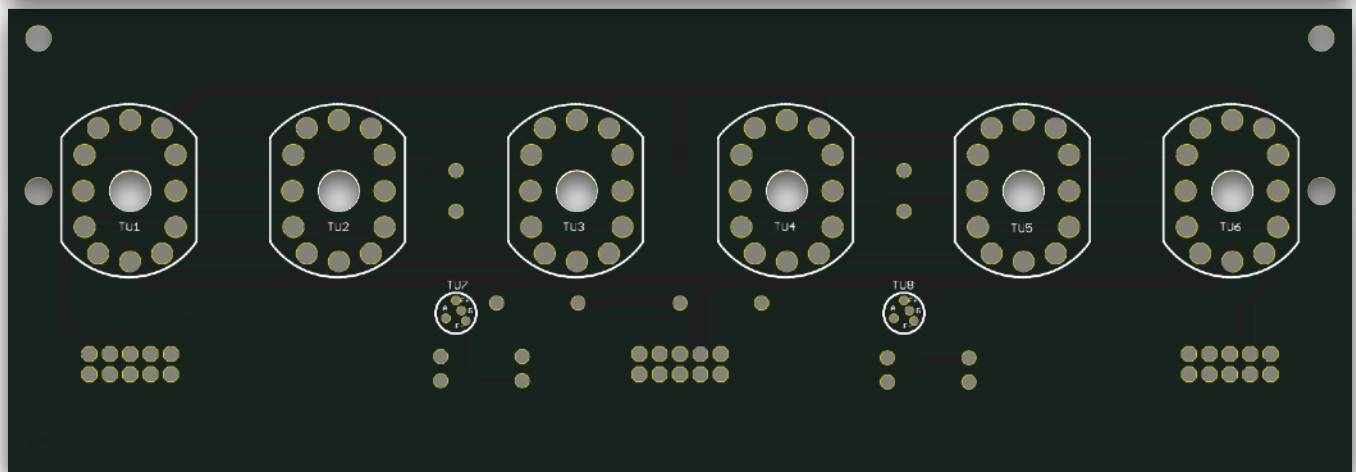
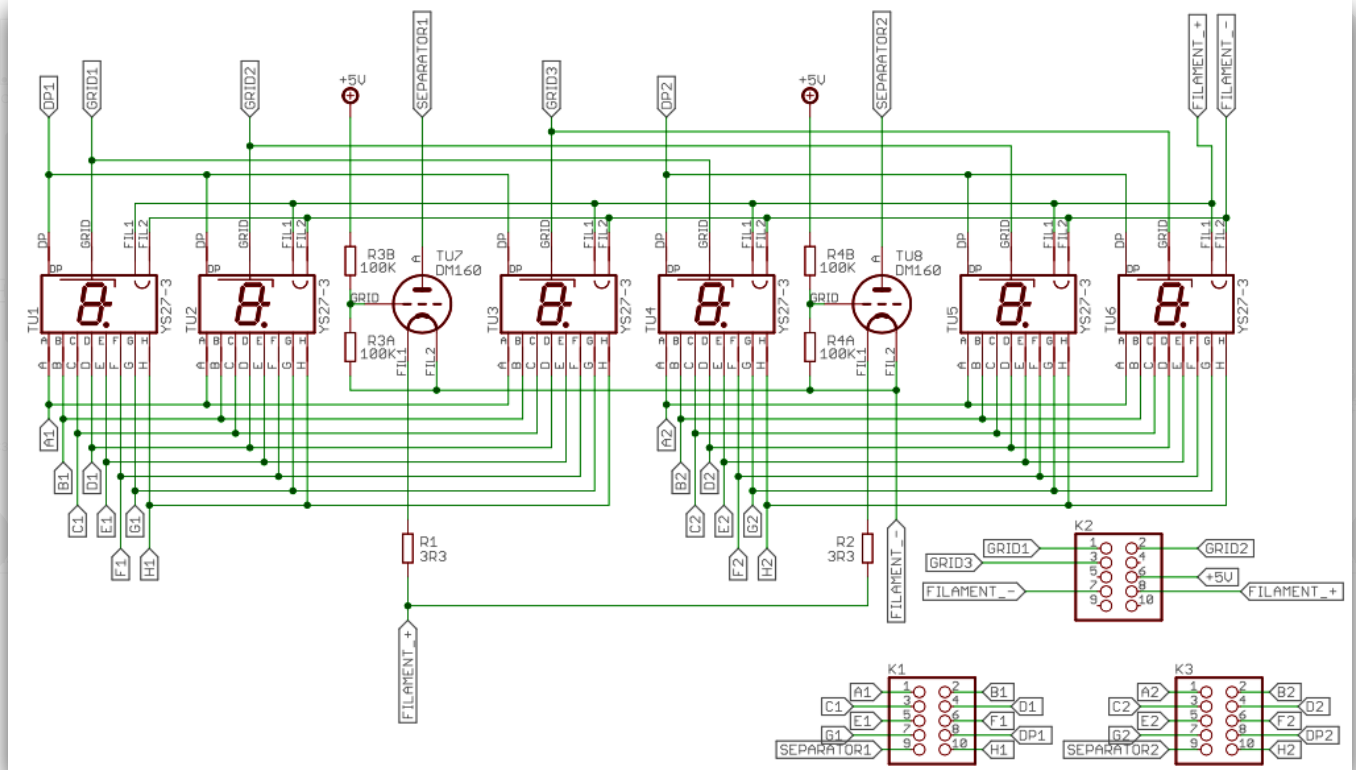




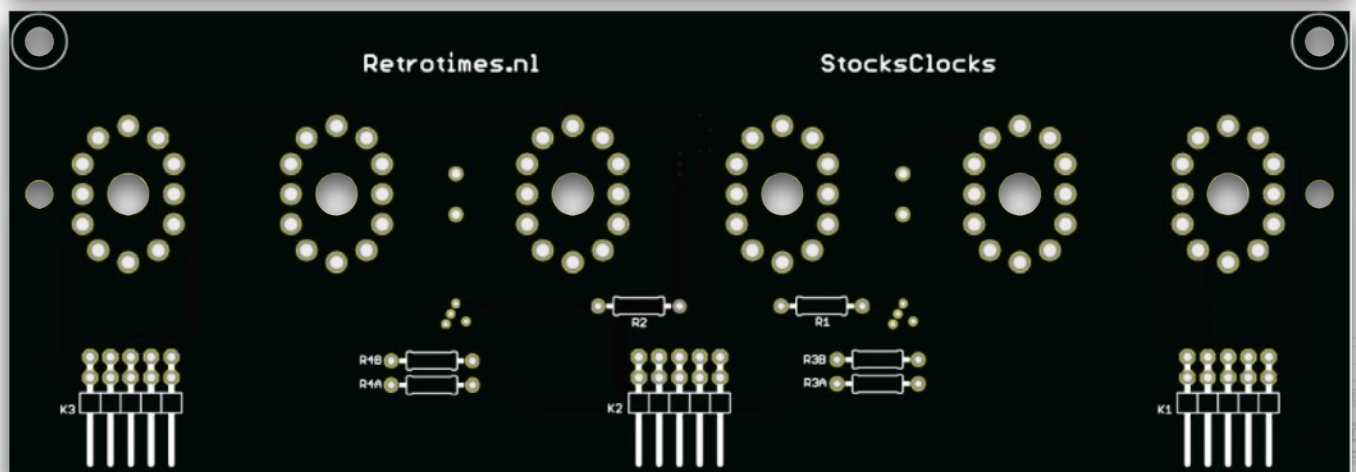
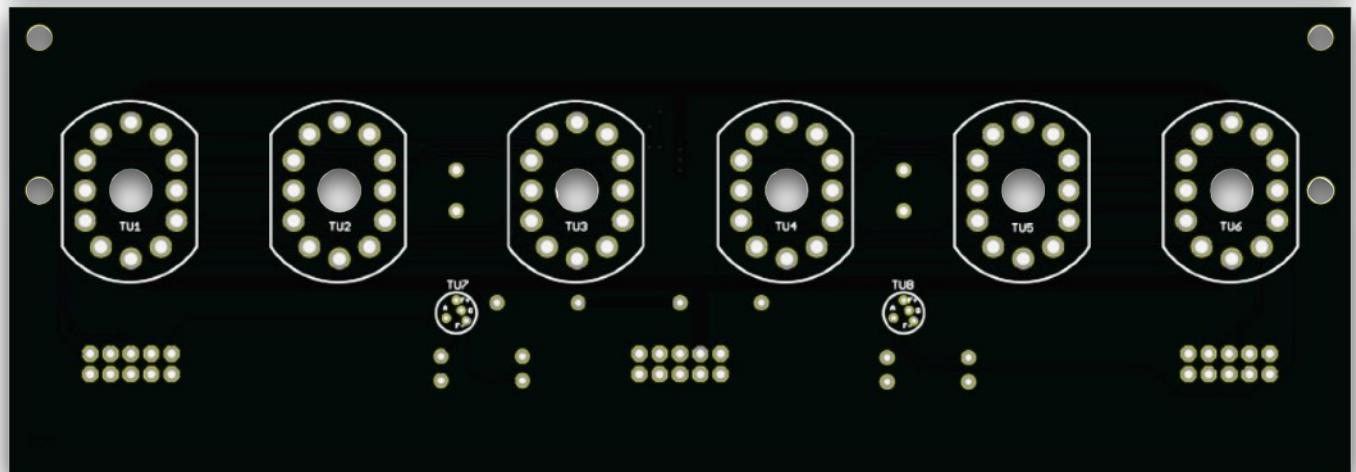
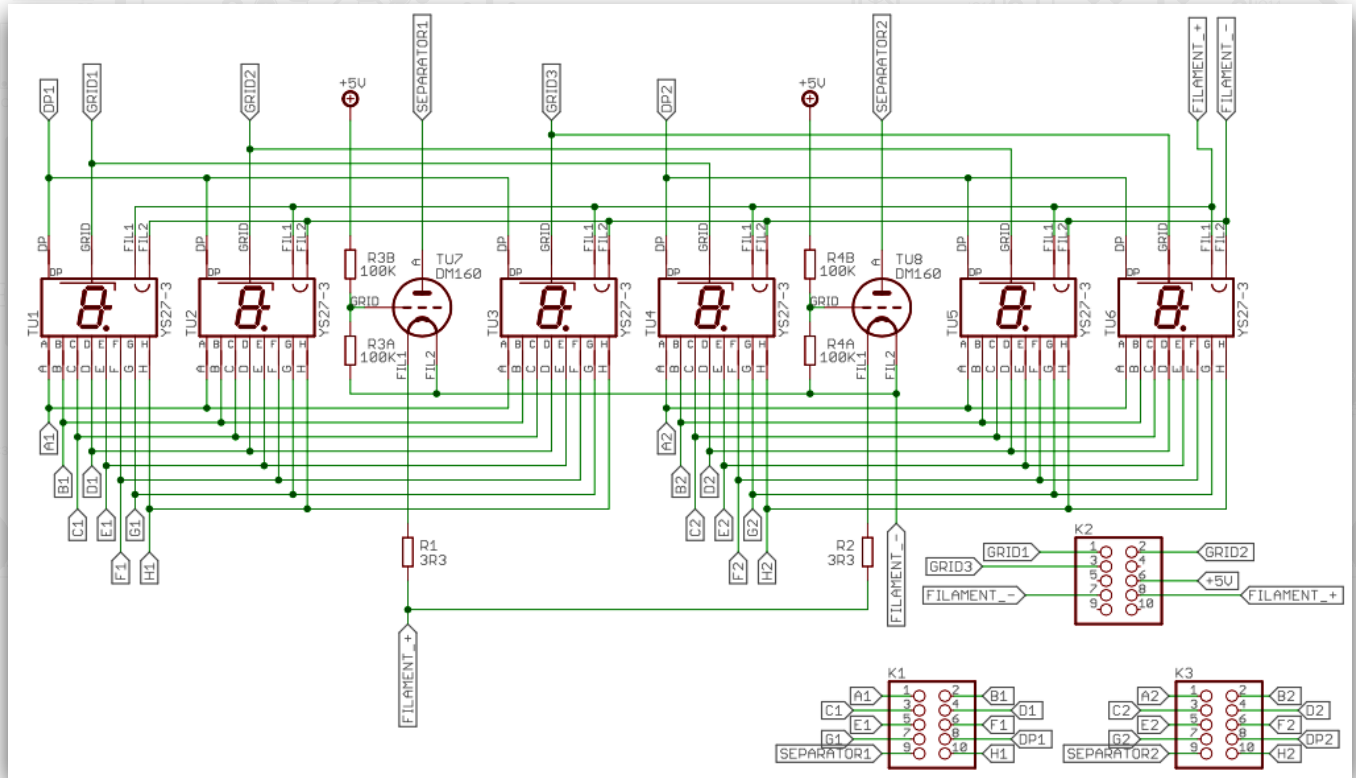
Main Controller Board Top and Bottom



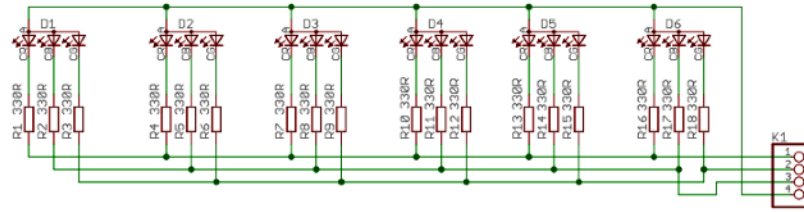
Display PCB (IV22)



Display PCB (YS27-3)



Backlight PCB



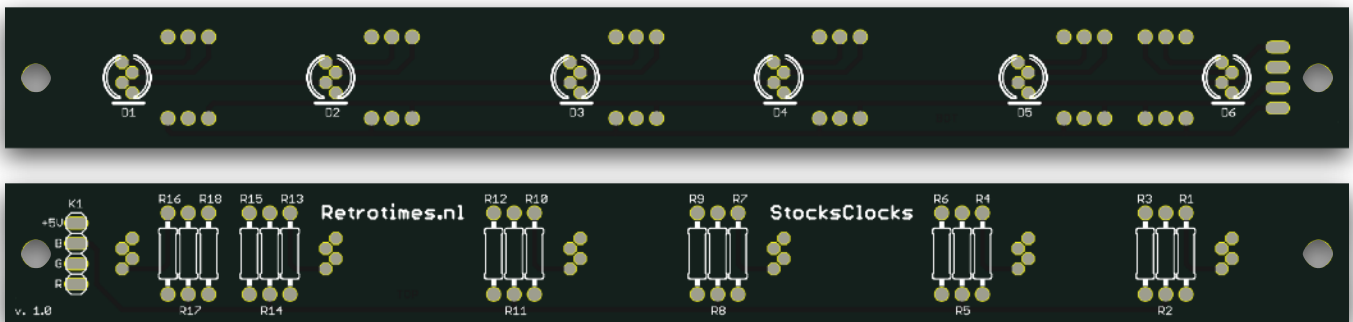
TITLE: vfd_clock_backlight_v10_ns_production

Document Number:

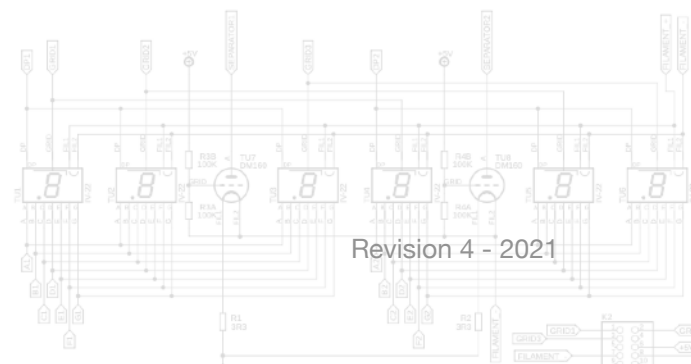
REV:

Date: 16/10/2020 0:14:14

Sheet: 1/1



Backlighting PCB Front (top) and Back (bottom)



Assembling the Display PCB

Assembly instructions for DM160 tubes or equivalent (as supplied with the kit):

- Mount resistors R1 and R2, 3.3 Ω .
- Mount resistors R3A and R4A, 100K (leave R3B and R4B open).

Assembly instructions for Russian IV-15 tubes (not supplied with the kit):

- Mount resistors R1 and R2, 6.8 - 10 Ω .
- Mount resistors R3B and R4B, 100K (leave R3A and R4A open). We didn't try this out in real life, some experimentation with the value of R1 and R2 may be necessary. Further in this manual we assume that you use DM160 tubes. Values and results may differ somewhat when using IV-15 tubes.

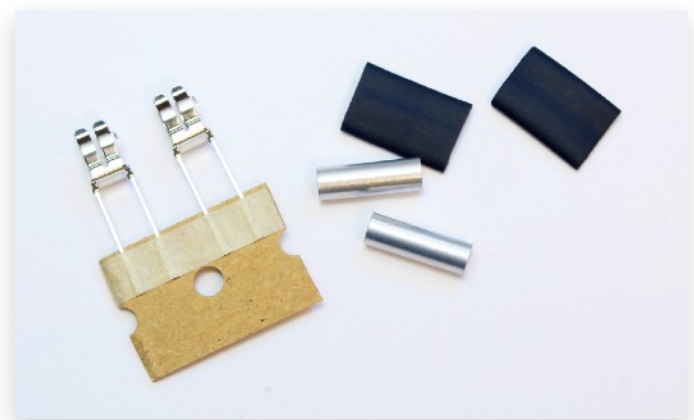
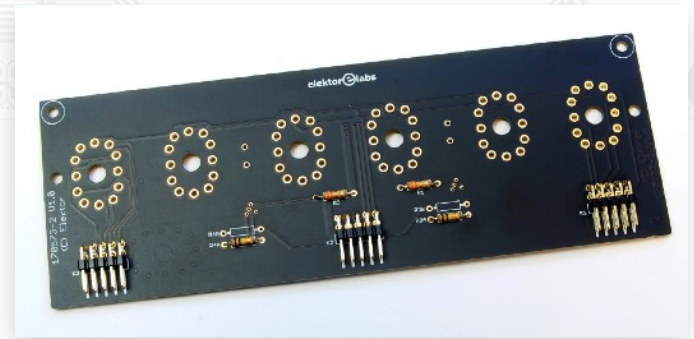
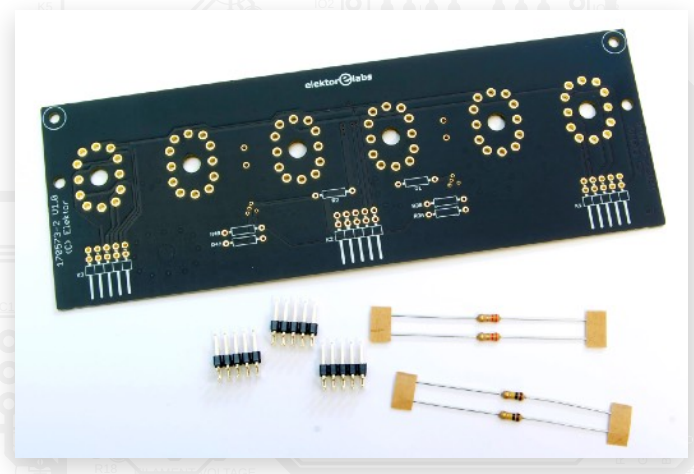
"DM160 tube" also refers to equivalent tubes like the CV5412, CV6094 and 6977 and possibly other compatible types.

Now mount the angled pin headers K1, K2 and K3.

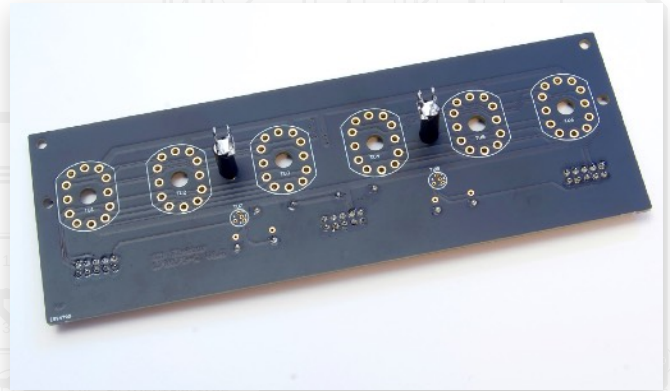
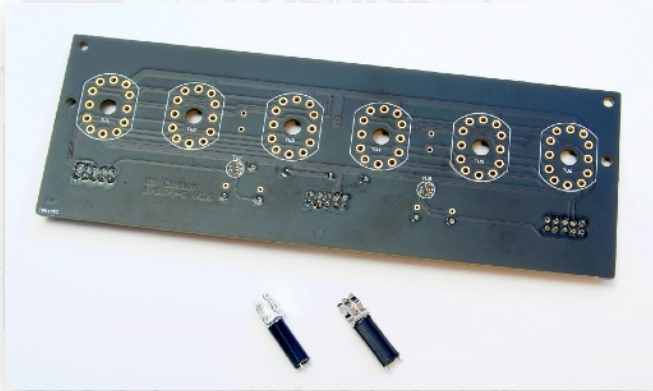
Cut two pieces of the 6.4 mm heat shrink tube with the same length as the 14 mm aluminum spacers.

Carefully remove the fuse holders from the paper strip without cutting the leads. You will notice that the leads are somewhat wider at the ends. Trim the ends longitudinally with a sharp wire cutter so the leads have a uniform width.

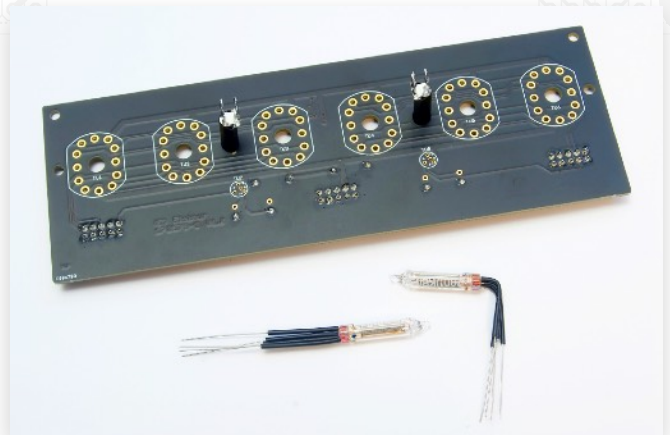
Place an aluminum spacer between the leads of the fuse holder and slide the heat shrink tube over the aluminum spacer and the leads of the fuse holder. Crimp the heatshrink tube and carefully remove the small metal tab from the fuseholder itself with a wire cutter so the DM160 tube will fit.



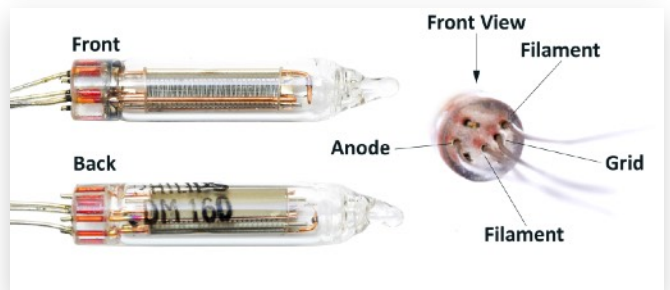
Mount the fuseholder assemblies to the front side of the display PCB.



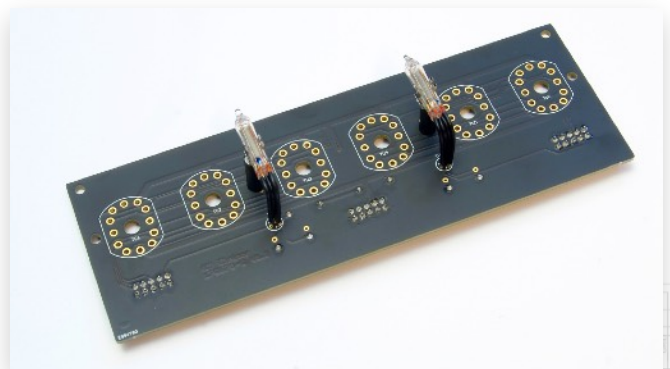
Cut eight 22 mm length pieces of the 1.6 mm heat shrink tube. Slide them over a thin piece of wire, a lead of a resistor or something similar and shrink them using a heat source.



When you look at the bottom side of the DM160 tube you will see four wires. Three of them are in a single row and close to each other. The fourth wire is somewhat separate from the other three. This is also reflected on the display PCB to make mounting of the tubes easier.



Slide the pre-shrunk heat shrink tubes over the leads of the DM160 tubes (TU7 and TU8) and bend the leads at an angle of 90 degrees (at a distance of approximately 5 mm of the tube bottom). Slide the leads through the holes of the PCB and press the tubes into the fuse holders. Make sure the front of the tubes is facing forwards and finally solder and clip the leads.

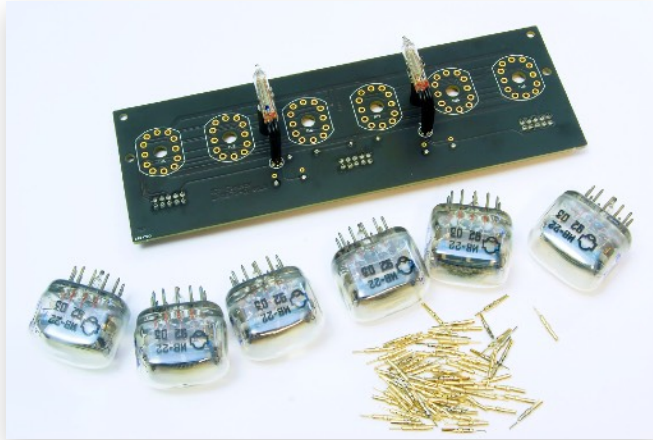


Measure the resistance between pin 7 and 8 of K2 with a multimeter. The resistance should be around 6 ... 6.5 Ω or slightly more depending on the resistance of the test leads. This is an

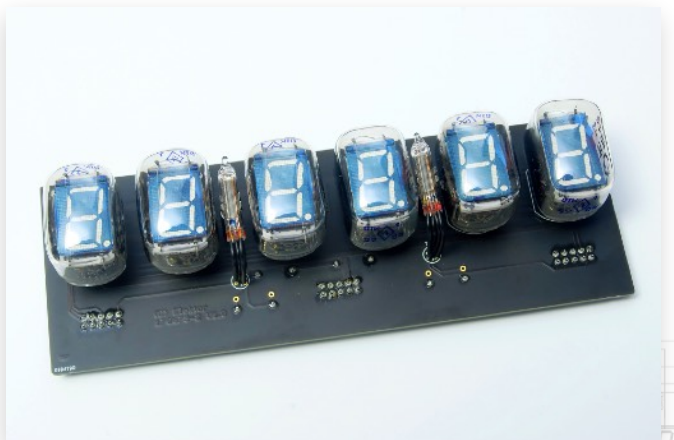
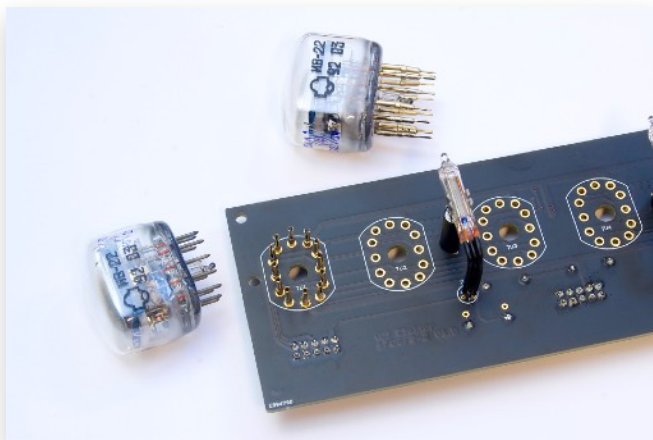
indication that the DM160 tubes are mounted correctly.

Have the IV-22 tubes and tube socket pins ready for the next step. Before starting the assembly you can put the IV-22 tubes temporarily on the PCB to see in which order they look best. IV-22 tubes have quite some tolerances, especially concerning mechanical dimensions and background color of the 7-segment digit.

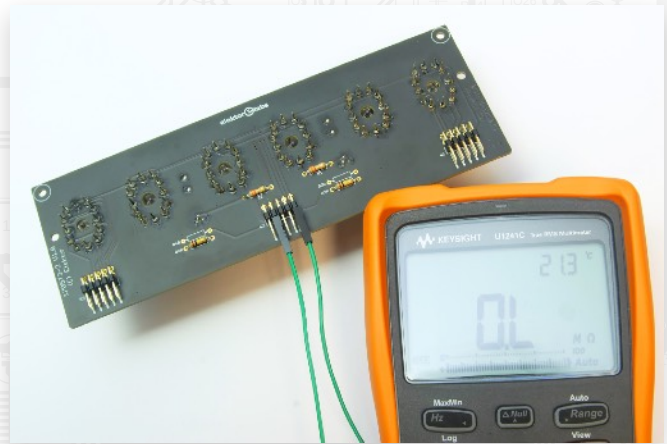
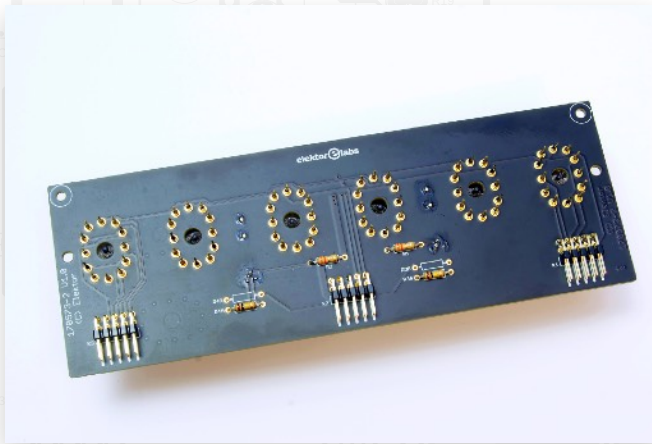
Take a look at the tubes. Normally the getter on the side of the tube should look dark with a silvery shine. Tubes supplied with the kit are always checked prior to shipping, but if you have a stock of your own tubes, then check the getters. If any of them are white, then air has entered the tube and you should discard the tube as alas it will not work anymore.



Pinch the tube socket pins between your fingers so they will be more tight. In order to mount the tubes, you can choose between two options. You can either slide the tube socket pins over the pins of the tube and push the tube with socket pins through the holes of the PCB or fit the tube socket pins to the PCB and push the tubes in the socket pins. If you choose the latter option, make sure the pins of the tubes are correctly seated into the socket pins.

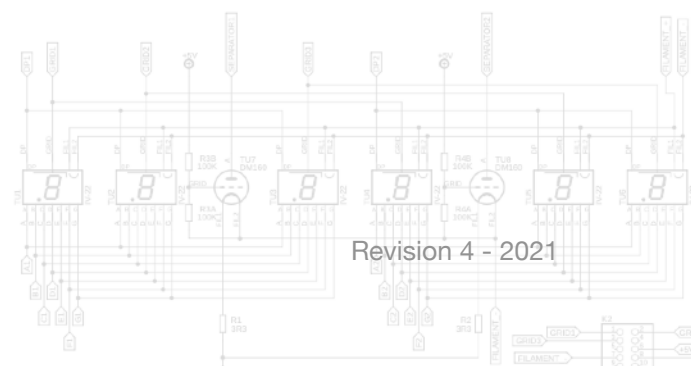


When all the tubes are in place, turn over the PCB and solder the tube socket pins from the backside. Do not use too much solder and keep soldering time short.



Finally measure the resistance between pin 7 or 8 of K2 (filament connections) and all the other pins on K1, K2 and K3. If everything is ok, you shouldn't be able to measure any resistance (open circuit) between any of the pins and the filament connections. If you measure a resistance one of the IV-22 tubes has a production fault or is mounted upside down.

The display PCB is now finished!



Assembling the Main Controller PCB

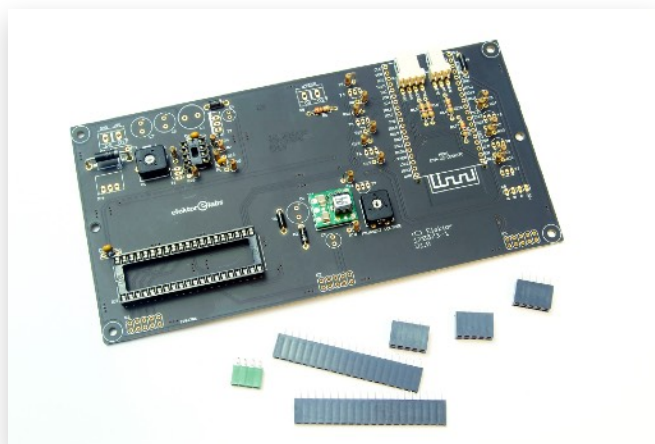
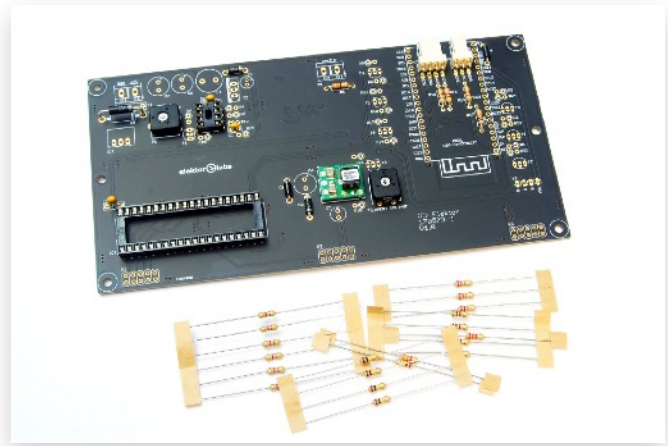
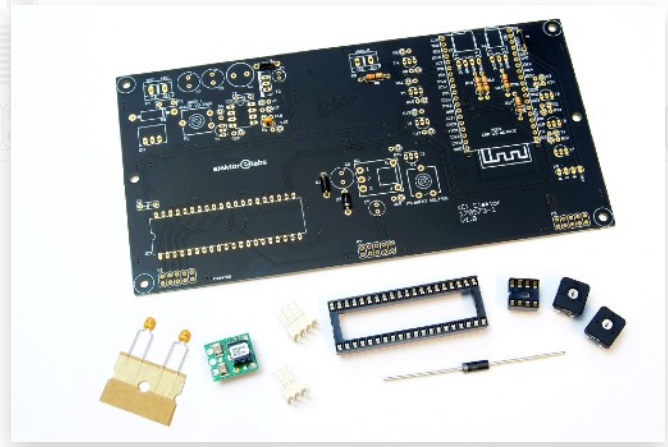
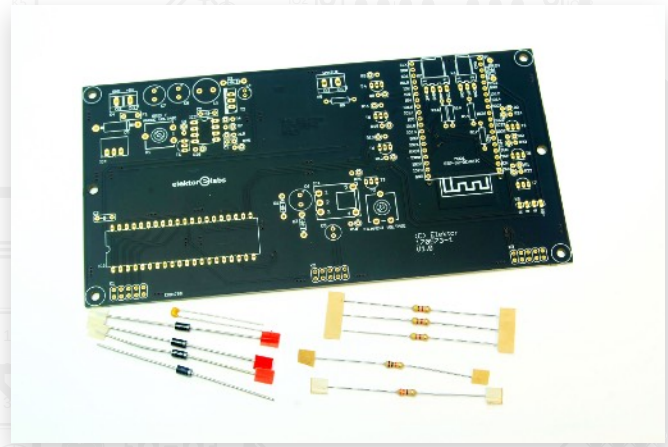
It is always good practice to mount the components from low profile to high profile. Start with the horizontally mounted resistors R5 (330 Ω), R14, R15 and R16 (4K7) and R17 (10K). Then mount the diodes D1, D2 (1N4007), D3 (MUR160G) and D4 (1N5819). Finally solder C1 (330 pF).

Mount the IC sockets for IC2 (8p) and IC3 (40p). Install the DC/DC step-down converter module IC1 (PTH08080WAH) and keep the soldering time as short as possible to avoid damaging the component.

The trim potentiometers P1 (4K7) and P2 (47K) and capacitors C2, C3 (100 nF) should be installed next. The trim potentiometers should be adjusted to their middle positions.

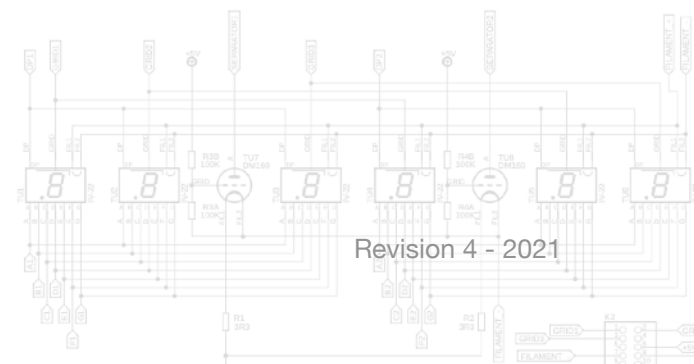
Mount the angled locking headers K6 (3p) and K7 (4p). Finally solder zener diode D5 (1N5347BG). Make sure this is installed with a few millimeters between the PCB and the body of the diode as this component may get very hot during fault conditions (over-voltage, wrong polarity).

Mount the resistors R1 to R4 (100 Ω), R6 (2K2), R7 (3K3), R8 to R13 (4K7), R18 to R20 (27K), R21 to R24 (100K) and R25 (220K) in an upright position. You will note that there is a circle on the silkscreen to denote where the body of the resistor should be installed. This isn't critical, but will lead to a cleaner and more professional



looking installation.

Mount the socket headers K1 ... K3 (2x5p), K8 (1x4p) and MOD1 (ESP32-devkitC, 2 off 1x20p).

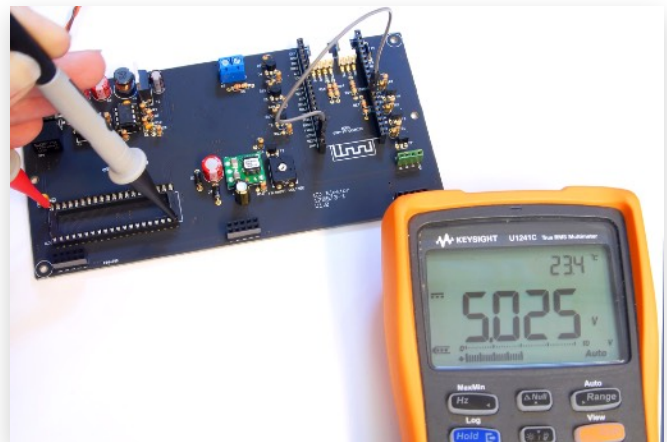
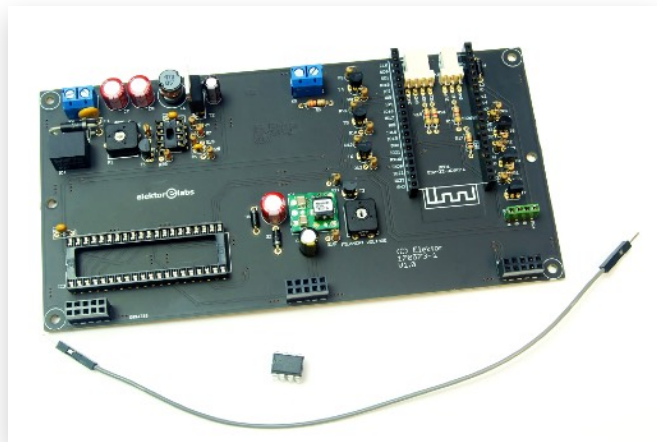
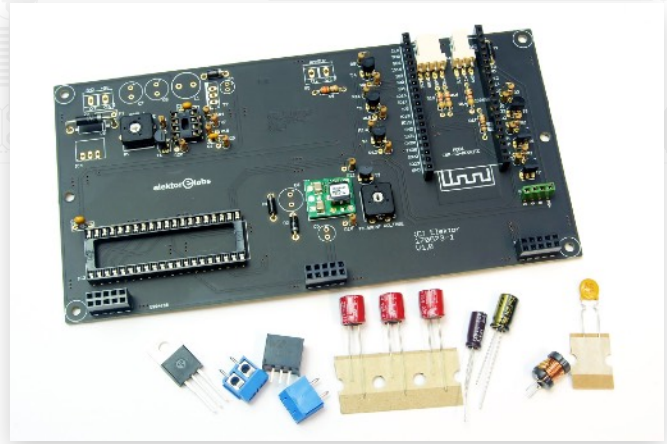
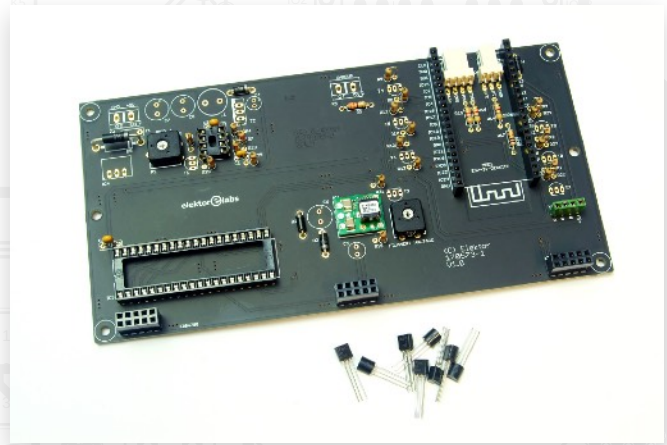


Insert the transistors with a TO92 style case T1 (BC574B) and T3 ... T9 (2N7000). Mount them so the top of their casings are flush with the top of the socket headers. This makes things easier later on.

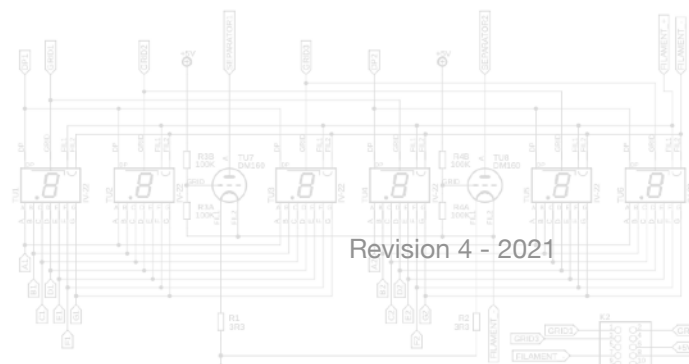
Finally mount the larger components. Start with C6 ... C8 (220 μ F). Then mount the terminal blocks K4 and K5. Solder the 5V DC/DC step-down converter IC4 (Würth 173019578). Mount polyfuse F1 (500 mA), L1 (47 μ H 1.2A), C4 (4.7 μ F 100 V) and C6 (100 μ F).

Finally, solder MOSFET power transistor T2 (IPP200N15N3 G). Please note there may be a substitute for this transistor in our kits with another type MOSFET at our discretion⁶. This will in no way alter the functionality of the clock.

Place IC2 (ICM7555) in its socket and connect GND and pin 8 (IO33) on the ESP32-devkitC socket with a piece of jumper wire. Power up the board using a 9 V power supply capable of delivering at least 500 mA. Measure the voltage between pin 20 (-) and pin 40 (+) of the IC socket of IC3. The voltage should be approximately 5 V.



⁶ Usually due to component availability.

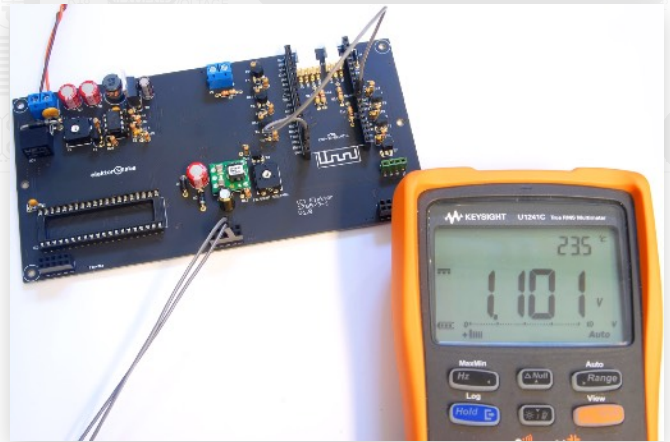
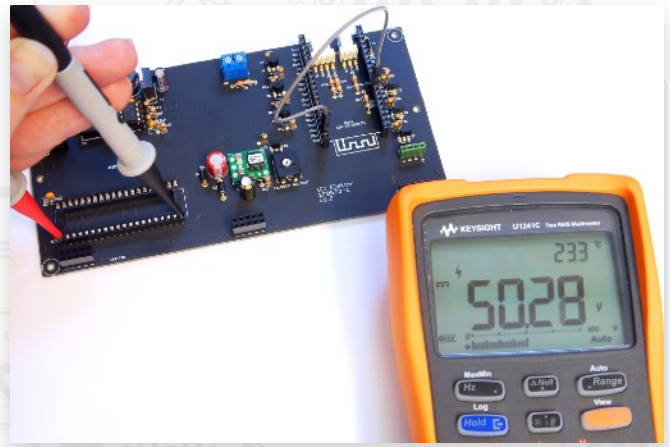


Now measure the grid/anode voltage between pin 1 (+) and pin 20 (-) of the same IC socket. Adjust this voltage to approximately 50 V using trim potentiometer P1.

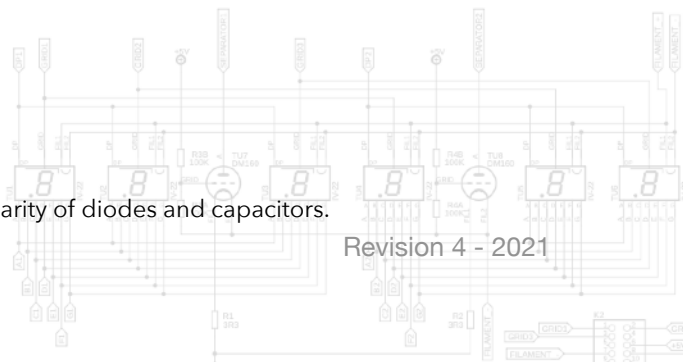
Finally measure the voltage between pin 2 (-) and pin 3 (+) of IC1 (PTH08080WAH). You can also measure this voltage on K2. Do not measure this voltage in reference to ground (GND)! Now adjust the voltage to approximately 1.1 V using trim potentiometer P2.

Do not continue assembling the clock if the voltages are not correct but look for assembling faults instead.⁷

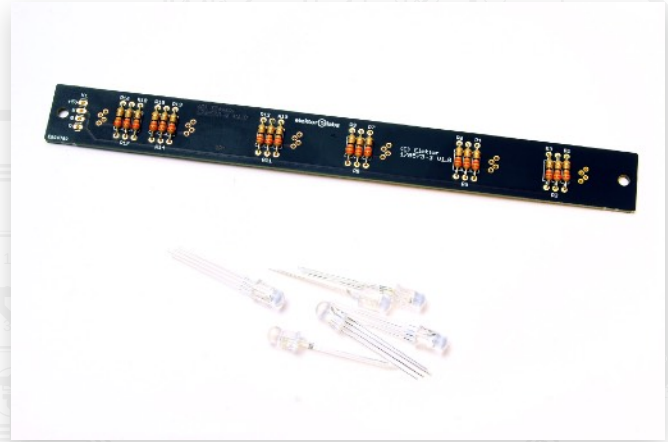
If everything is ok, disconnect the power supply.



⁷ Common errors include soldering bridges, cold joints and incorrect polarity of diodes and capacitors.



Assembling the RGB Backlight PCB



Solder the 330 Ω resistors R1 ... R18 on the backside of the PCB. Keeping them all in the same orientation is always an aesthetically pleasing practice.



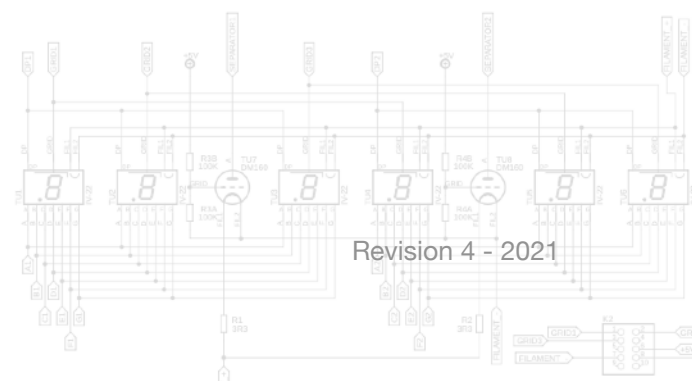
Insert the RGB LEDs on the front side of the PCB. Pay attention to the polarity, the flat side of the LED goes where the PCB symbol is marked with a dash. Push the LEDs through the holes until they don't go any further. When soldering them, be careful not to overheat the LEDs. Solder one lead of each LED first and check the alignment, then solder the remaining leads.

Make up a small connector cable using 2x4 pin headers, 4 pieces of colored wire (white, red, green and blue) and 8 pieces of 3.2mm diameter heat shrink tubing cut to 1 cm lengths. Use the heat shrink to insulate the solder joints. The cable should look like that depicted in the next photo.



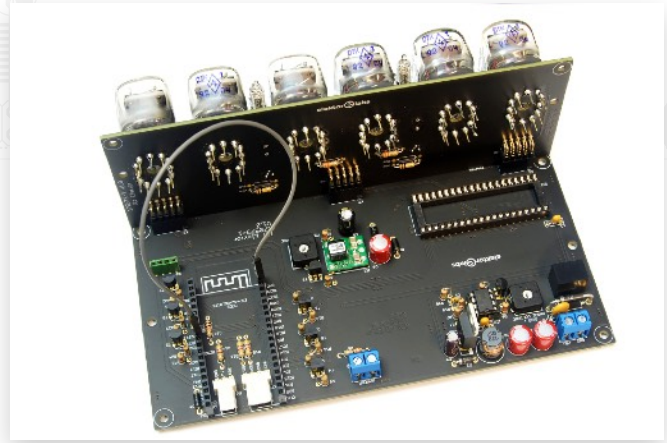


Solder 1 side to the RGB backlight PCB, so that the wires protrude from the backside. The white wire goes to the solder pad marked with +5 V. If you wish, you can test the LEDs using a 5 V (lab) power supply. This completes the RGB backlight PCB assembly. Take a break.



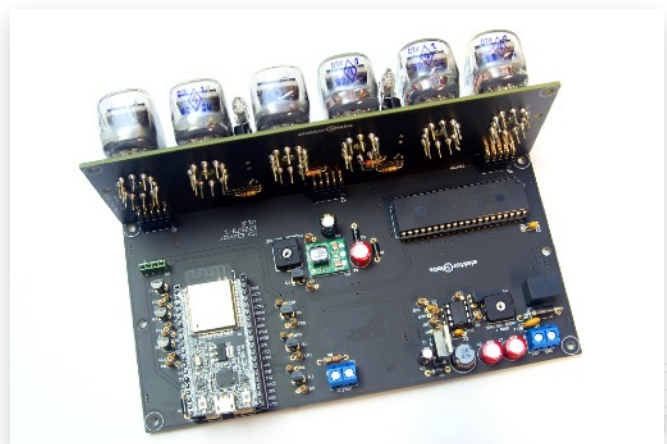
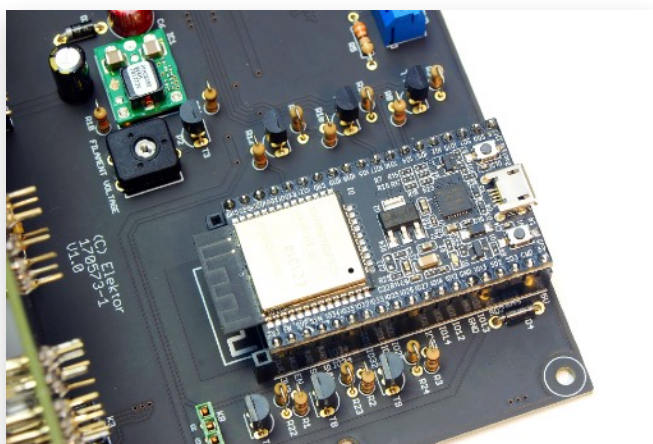
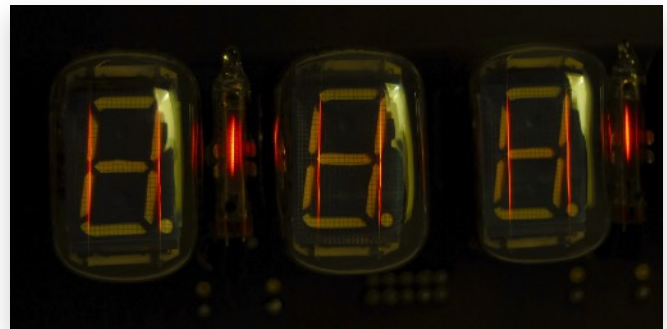
Final Assembly Steps...

Connect the display PCB to the main PCB. Power up the main PCB and take a look at the VFD tubes in a dark room. The filaments of the tubes should glow dimly with a dark red color. If the filament of an IV-22 tube does not glow or very faint compared to the other tubes, wriggle the tube a bit back and forth in the socket pins until the filament glows properly. Sometimes there is a bad contact between the IV-22 tube pins and the socket pins on the PCB due to a thin layer of



oxidation. Given the very low filament voltage of 1.1 V, this can sometimes be problematic.

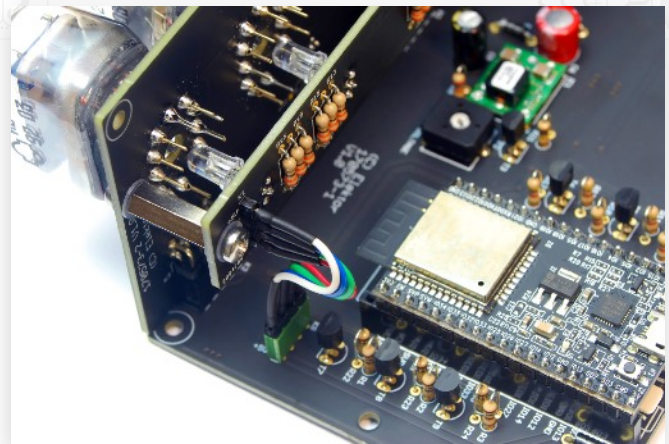
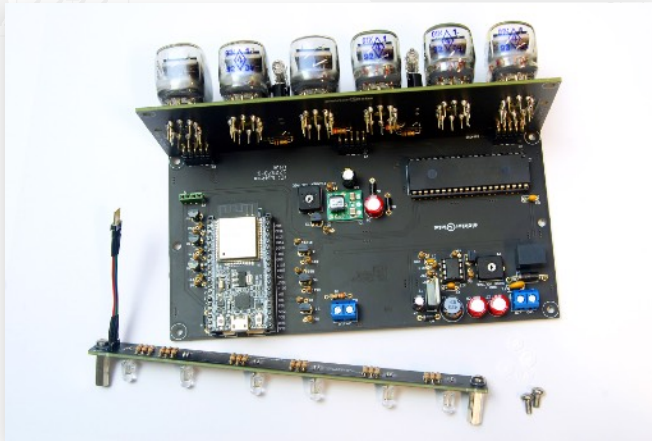
Place IC3, HV518P in its socket. Remove the jumper wire between GND and pin 8 (IO33) of the ESP32-devkitC headers. Fit the ESP32-devkitC module into the socket headers. The socket headers are 20 way while the pin headers on the ESP32-devkitC have only 19 pins, so be



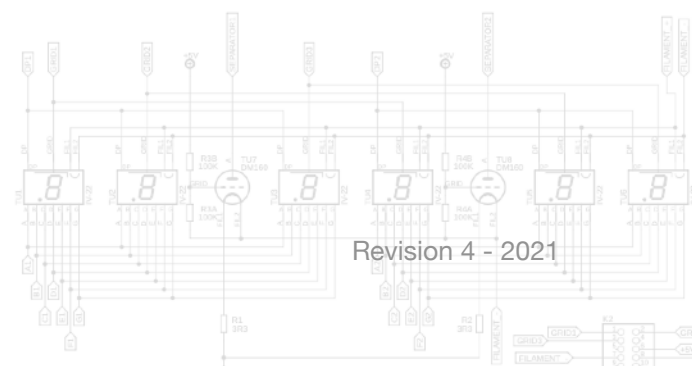
very careful to fit the ESP32-devkitC correctly into the socket headers. 20 way socket headers are a standard part and 19 way socket headers are difficult to get or even require custom manufacturing which is very costly.

Mount two 15 mm hex standoffs to the RGB backlight PCB using 2 M3 x 6 machine screws and 4 M3 nylon washers.

Attach the RGB backlight PCB to the display PCB using 2 M3 x 6 machine screws and 4 M3 nylon washers. Connect the RGB backlight PCB cable to K8 on the mainboard. Be sure to use the right orientation (note the connection illustrated in the photo below).



That completes the hardware construction of the clock. Now onto the software!



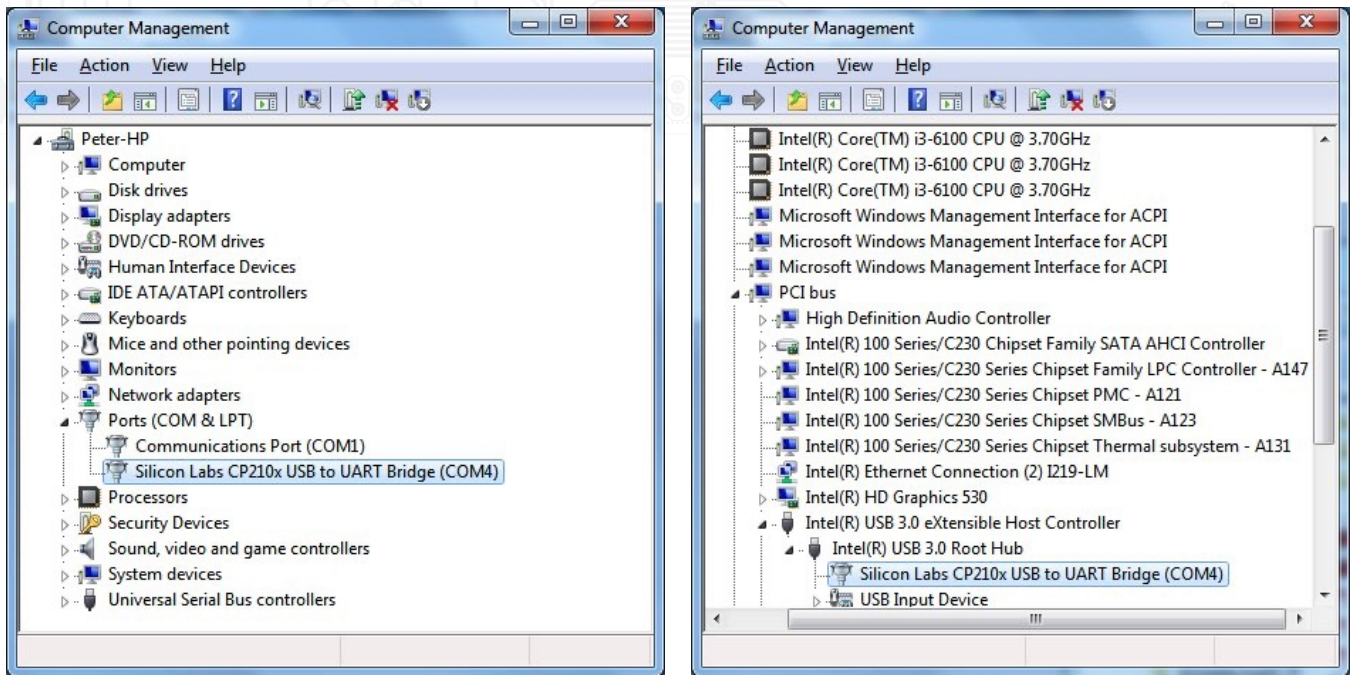
Setting up the Software

This section is for the most part aimed at using the Arduino IDE in Windows⁸. You should have no problem when using another version of Windows or any other operating system that's supported by the Arduino IDE.

Serial Interface

Power up the clock and connect the ESP32-devkitC with a PC using a USB cable. The ESP32-devkitC incorporates a USB-to-serial function that enables the PC to access the clock as a serial interface.

If you're using Linux, the system will create an entry in the device directory structure e.g. `/dev/ttyUSB0`. Run command `lsusb` to obtain a list of all connected USB devices.



If you're using Windows, the operating system will assign a COM-port. A serial port is typically named COM<x> where <x> is a number between 1 and 256. You can observe the COM-port in the device manager. When the devices are displayed by type, the COM-port appears under the "Ports (COM & LPT)" branch. You can select to view devices by connection which renders a tree structure.

Clock Software

The clock software is an Arduino sketch. We used Arduino IDE v1.8.7 for developing and testing the software. The source code is made up of .ino files. You will hopefully find many useful remarks and comment blocks in these source files to aid in understanding the code. Aside from the IDE, you will need to install the Arduino core for the ESP32. This package adds support for ESP32 to the Arduino IDE. Currently, this package can be obtained from GitHub:

<https://github.com/espressif/arduino-esp32>

⁸ At the time this was written, Windows 7 was in use.

This link will show you the main page of the software package. The page contains installation instructions. At the time of writing the Arduino core for the ESP32 version 1.0.0 was marked as a stable release.

There are two ways of putting the clock software in the ESP32-devkitC:

- Use esptool to upload the binary files.
- Upload from within the Arduino IDE.

Either way you will need to install the **Arduino IDE** and the **Arduino core for the ESP32**.

To upload the binary files using esptool, first put all binary files in a directory, and then invoke the following command from the command prompt in that directory:

```
bin> path-to-esp32-tools\esptool.exe --chip esp32 --port COM4 --baud 921600 --before default_reset --after hard_reset write_flash -z --flash_mode dio --flash_freq 80m --flash_size detect 0xe000 boot_app0.bin 0x1000 bootloader_qio_80m.bin 0x10000 vfd_clock_wifi.ino.bin 0x8000 vfd_clock_wifi.ino.partitions.bin
```

Replace **COM4** with the correct serial path. Replace **bin** with the path to the directory that contains the binary files. Replace **path-to-esp32-tools** with the path to the tools directory in the Arduino core for the ESP32, for example:

```
C:\Users\Peter\AppData\Local\Arduino15\packages\esp32\tools\esptool\2.3.1\esptool.exe
```

The esptool will output information like this:

```
esptool.py v2.1
Connecting.....
Chip is ESP32D0WDQ6 (revision 0)
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 921600
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 8192 bytes to 47...
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.0 seconds (effective 4096.0
kbit/s)...
Hash of data verified.
Flash params set to 0x022f
Compressed 12384 bytes to 8181...
Wrote 12384 bytes (8181 compressed) at 0x00001000 in 0.1 seconds (effective 792
6 kbit/s)...
Hash of data verified.
Compressed 508016 bytes to 328749...
Wrote 508016 bytes (328749 compressed) at 0x00010000 in 5.7 seconds (effective
17.7 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 122...
Wrote 3072 bytes (122 compressed) at 0x00008000 in 0.0 seconds (effective 1638.
kbit/s)...
Hash of data verified.

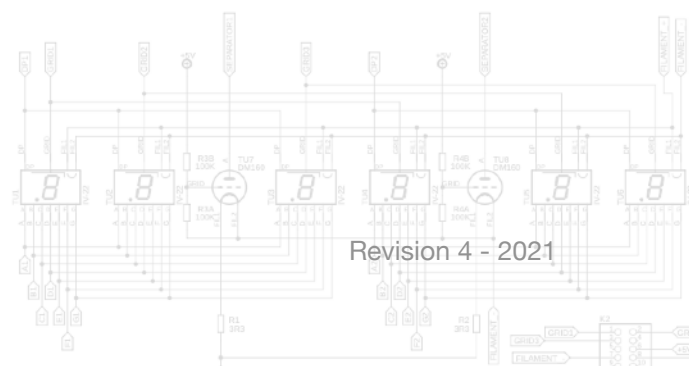
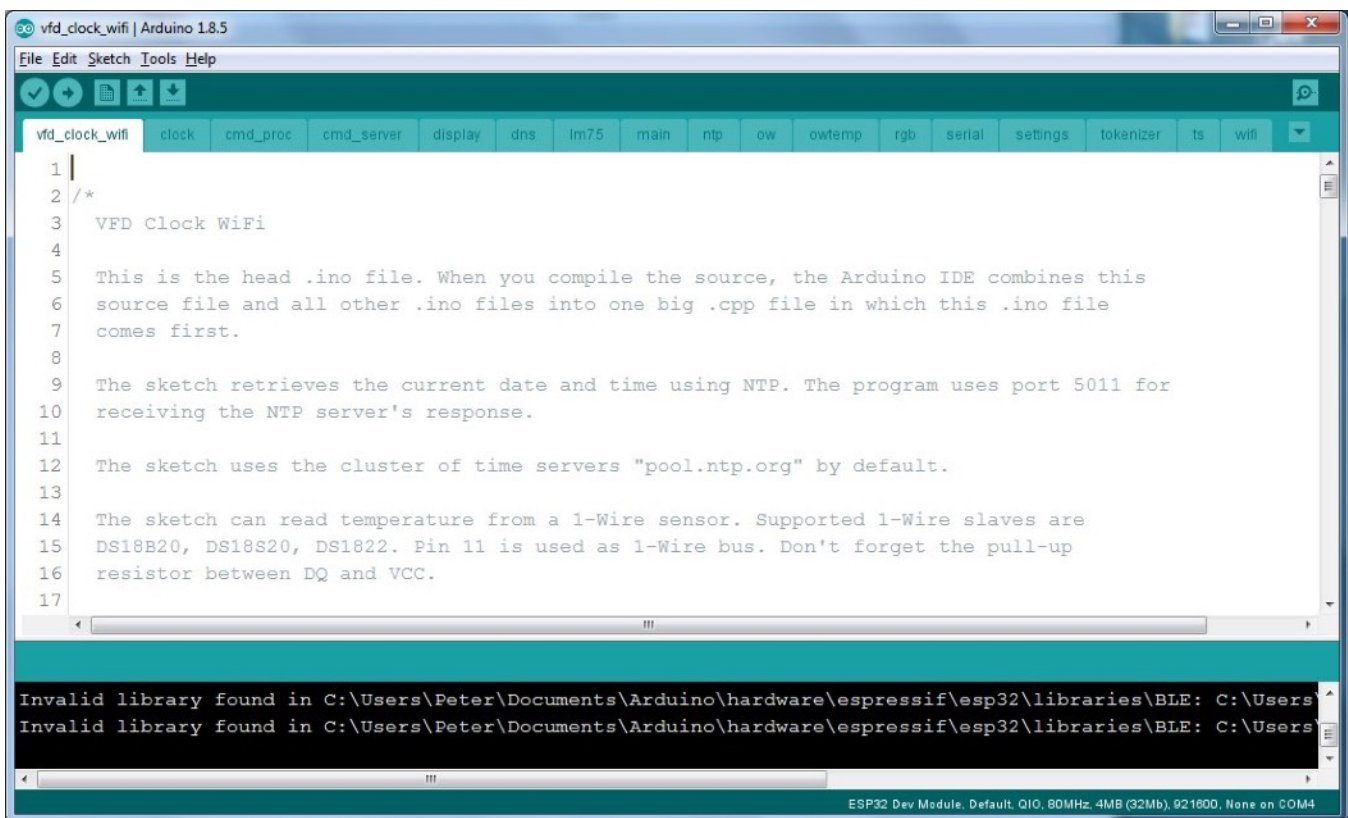
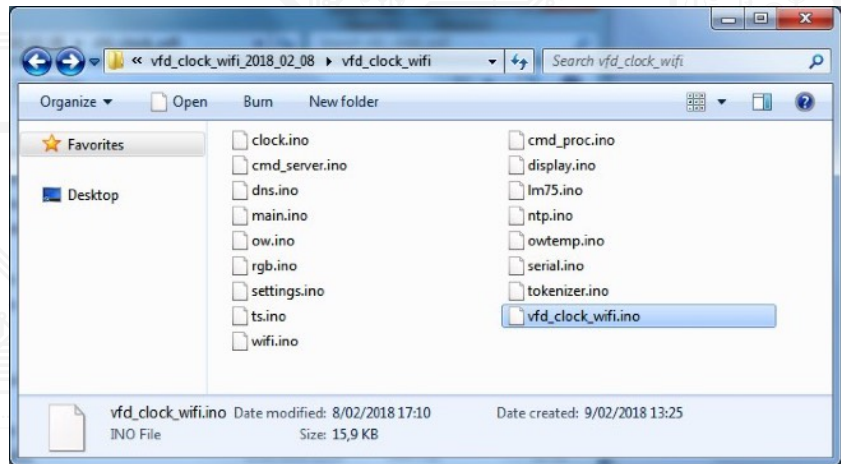
Leaving...
Hard resetting...
```

After this, the clock is ready to be set up as described further.

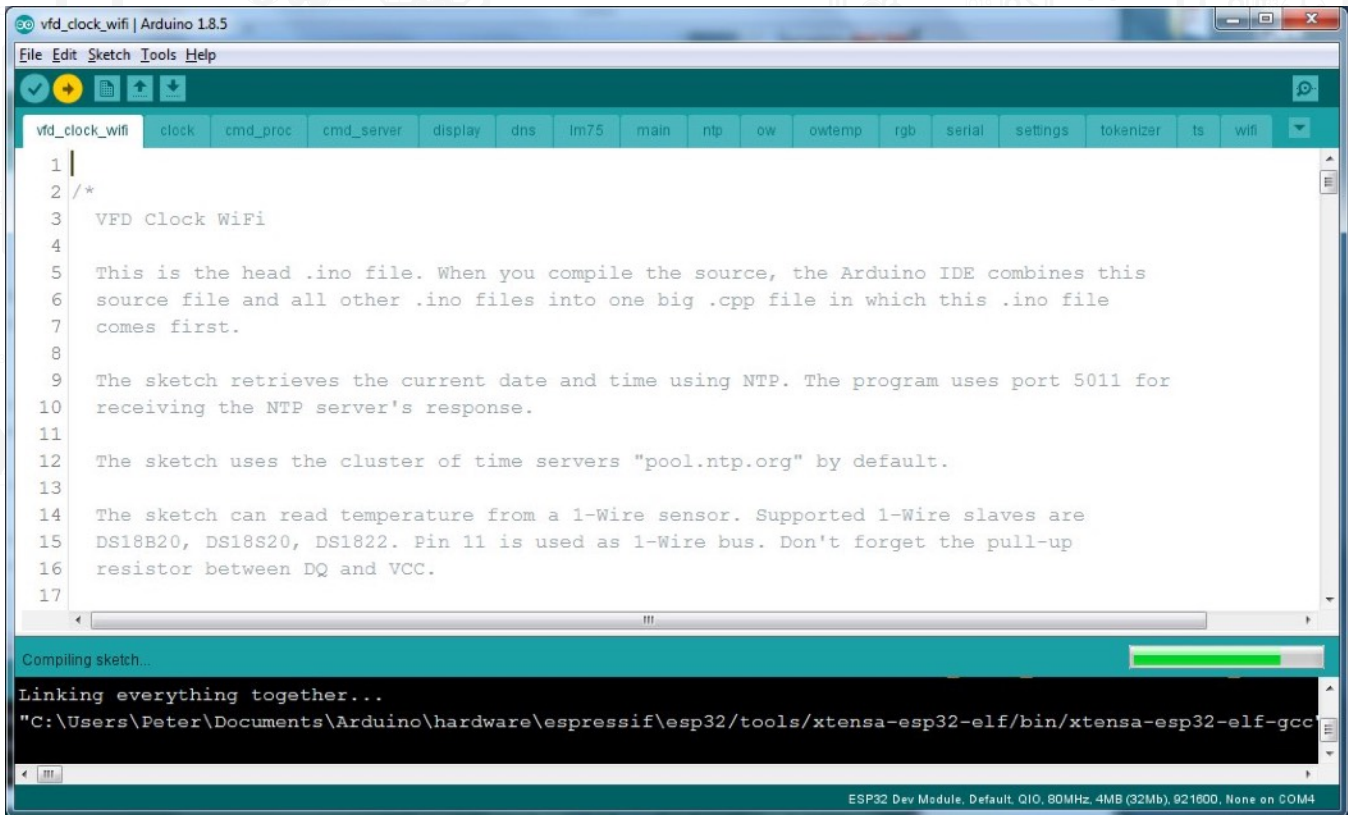
The other way of uploading the clock software to the ESP32-devkitC module is using the Arduino IDE.

The source code consists of multiple .ino source files. The main file is called **vfd_clock_wifi.ino**, it has the same name (without .ino) as the containing folder.

Open **vfd_clock_wifi.ino** in the Arduino IDE to open up the Sketch. The IDE shows all files in tabs, starting with the main file and followed by the other files in the folder sorted alphabetically.



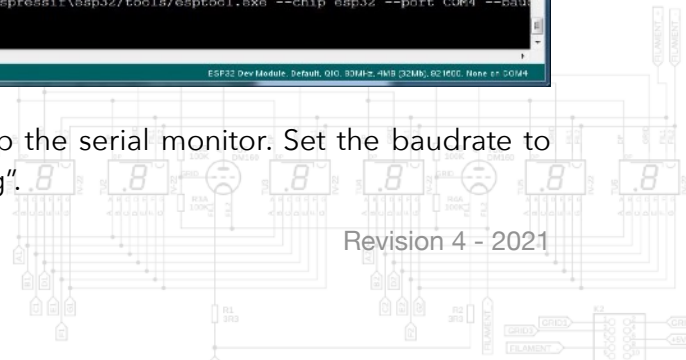
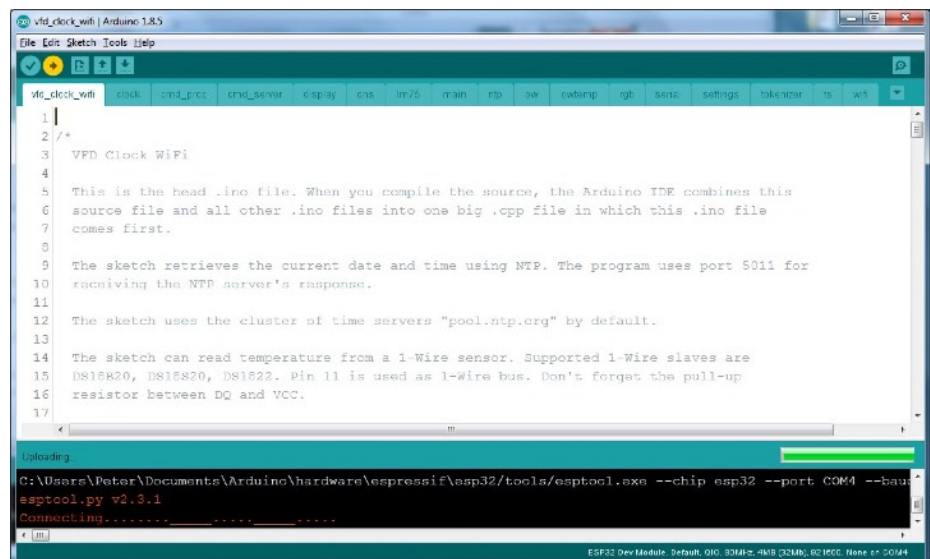
Before you start building the sources, you have to set up the IDE for use with the ESP32-devkitC module. In the Tools menu, select board type "ESP32 Dev Module" and select the COM-port assigned to your clock.



Now you can build and upload the software. In the Arduino IDE, this process is simply called "Upload". Select menu Sketch -> Upload, or press CTRL-U to proceed.

If you get compiler warnings or the build process stops because of warnings, select menu File -> Preferences -> Settings tab and set "Compiler Warnings" to "None".

After the IDE has successfully uploaded the binary to the ESP32-devkitC, the clock sends text to the serial interface. You can observe this information in the IDE's serial monitor. Select menu Tools -> Serial Monitor, or press CTRL-SHIFT-M to open up the serial monitor. Set the baudrate to 115200 and select something different from "no line ending".

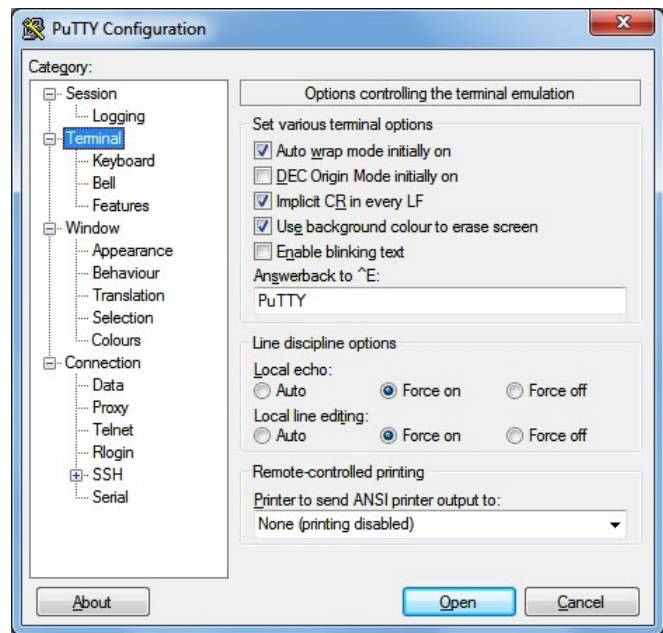
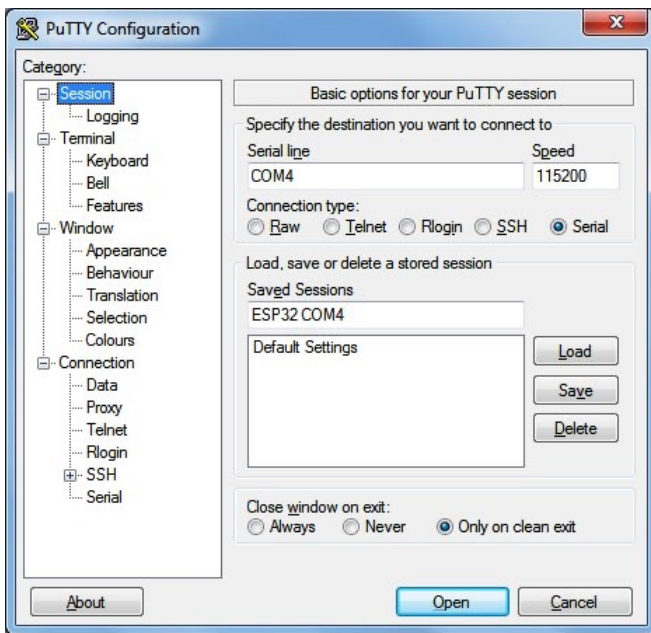
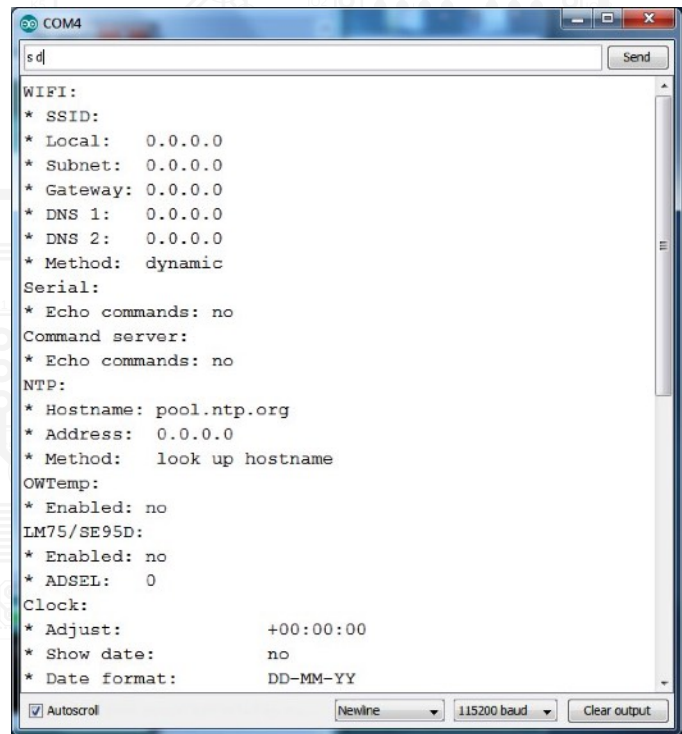


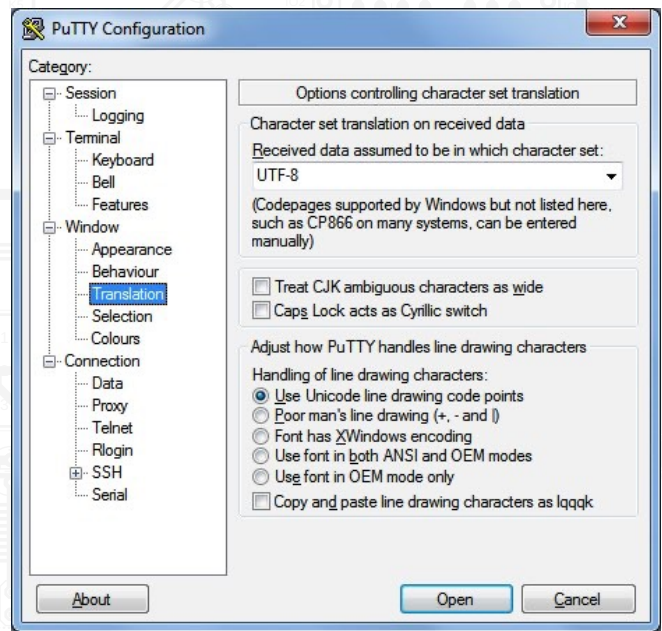
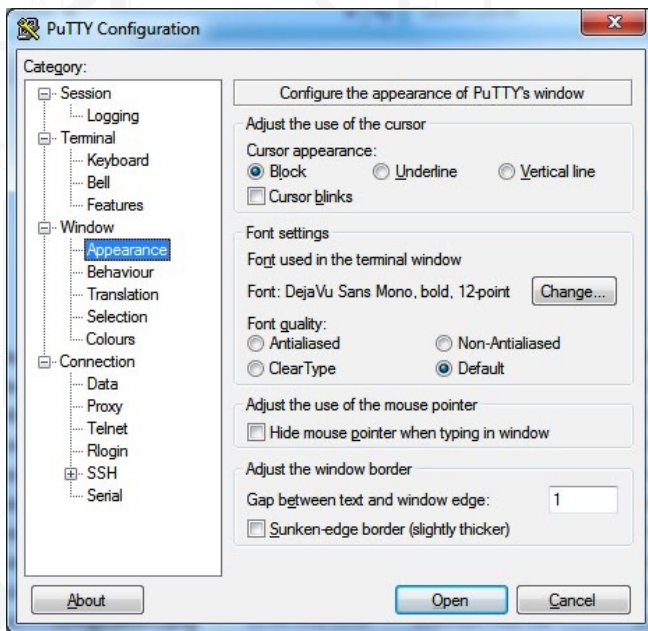
IMPORTANT! Each time you open the serial interface, the board will reset. This behavior is specified by Arduino and occurs with all Arduino-compatible boards, independent of operating system and serial monitor program (Arduino IDE's serial monitor, PuTTY, netcat, ...). So beware, when you change settings, then close and reopen the serial interface before writing the settings to non-volatile memory (a.k.a. EEPROM), the changes will be lost.

You can send commands to the clock from the serial monitor. For example, command `s d` (a short version of settings dump) reports a list of all current settings.

At this point, no settings are set, unless your ESP32-devkitC was already set up for the clock. To fill in the settings, you have to send a series of commands. You can manually type each command in the serial monitor. This will take time, so let's use a better method.

You can put the commands in a text file and send its contents to the serial port. Since the serial monitor in the Arduino IDE isn't capable of sending a file or pasting multiple lines from the clipboard, we're going to use another serial monitor. A good choice is PuTTY in Windows and Linux. Another good choice is netcat in Linux.



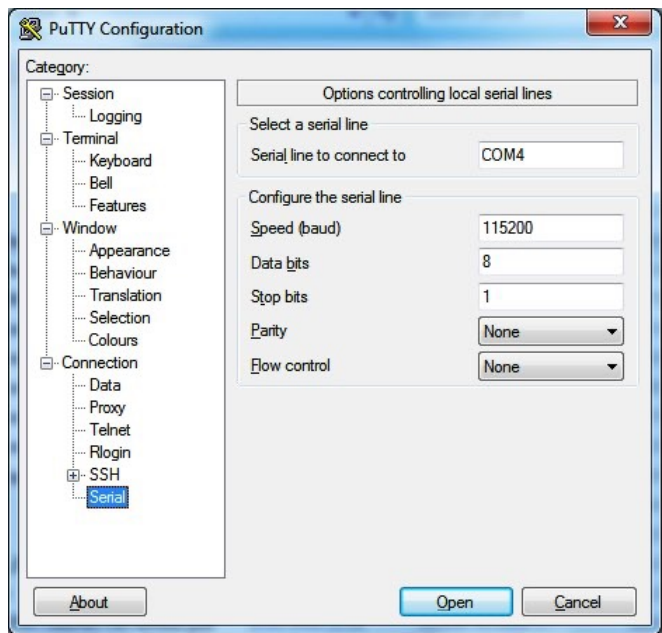


The most important PuTTY settings are:

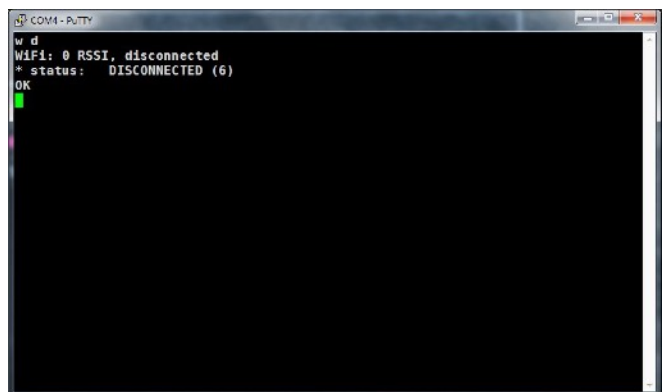
- Select the COM-port assigned to the clock.
- Select baudrate 115200, 8 data bits, 1 stop bit, no parity, no flow control.
- Turn on implicit CR in every LF.
- Set local echo to "Force on".
- Set local line editing to "Force on".
- If your screen has a high resolution, you may want to select a larger font.
- Use the UTF-8 character set.

You can save these settings in a named session.

Click "Open" to make a connection with the clock.



Once connected, you can send commands to the clock, like **w d** (short for **wifi dump**).



Let's create a text file that contains the necessary commands for setting up the clock. Many schemes are possible; the commands presented here create one possible setup.

You can find a list of commands in a later section of this manual.

```

File Edit Format View Help
wifi set ssid "your ssid"
wifi set password "your password"
wifi set local 192.168.1.141
wifi set subnet 255.255.255.0
wifi set gateway 192.168.1.254
wifi set dns 1 192.168.1.254
wifi set dns 2 0.0.0.0
wifi set method static

serial set echocmd yes

cs set echocmd no

ntp set hostname "pool.ntp.org"
ntp set addr 0.0.0.0
ntp set method hostname

owtemp set enabled yes

lm75 set enabled no
lm75 set adsel 0

clock set adjust + 01 00 00
clock set showdate yes

File Edit Format View Help
clock set dateformat dmy
clock set datesepmode dmy
clock set dateformat dmy
clock set datesepmode off
clock set hourformat 24
clock set hourlz yes
clock set timesepmode blink
clock set showtemp yes
clock set tempformat celsius

rgb set grad id 1
rgb set solid id 17
rgb set sleep raw 0 0 0
rgb set method grad
rgb set grad apply

main set wificonnect yes
main set ntprequest yes
main set ntprefresh 01 00 00
main set cmdsrvstart yes
main set sleeptime 01 00 00
main set wakeuptime 06 30 00
main set sleepmode yes

settings write
  
```

```

File Edit Format View Help
clock set dateformat dmy
clock set datesepmode off
clock set hourformat 24
clock set hourlz yes
clock set timesepmode blink
clock set showtemp yes
clock set tempformat celsius

rgb set grad id 1
rgb set solid id 17
rgb set sleep raw 0 0 0
rgb set method grad
rgb set grad apply

main set wificonnect yes
main set ntprequest yes
main set ntprefresh 01 00 00
main set cmdsrvstart yes
main set sleeptime 01 00 00
main set wakeuptime 06 30 00
main set sleepmode yes

settings write
  
```

```

COM4 - PuTTY
OK
rgb set method grad
OK
rgb set grad apply
OK

main set wificonnect yes
OK
main set ntprequest yes
OK
main set ntprefresh 01 00 00
OK
main set cmdsrvstart yes
OK
main set sleeptime 01 00 00
OK
main set wakeuptime 06 30 00
OK
main set sleepmode yes
OK

settings write
OK
  
```

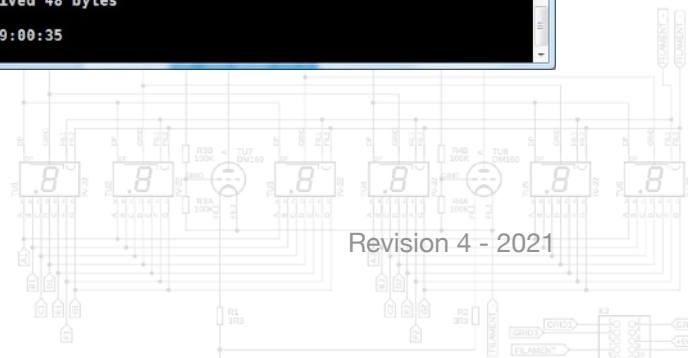
When the text file is done, you can transfer it to the serial port in the serial monitor. In the case of PuTTY, select all text in the editor and copy it to the clipboard. CTRL-C will do the trick (if you're using a PC). In PuTTY, press the right mouse button to paste the text from the clipboard. The settings will take effect as soon as the clock receives them. If the settings are correct, the clock connects to your local wireless network, queries the date and time with the specified NTP server, and shows the time on the tubes.

We have to store the settings in non-volatile memory to preserve them when the clock is powered down or reset. We included command settings write in our text file so all settings are immediately written to non-volatile storage.

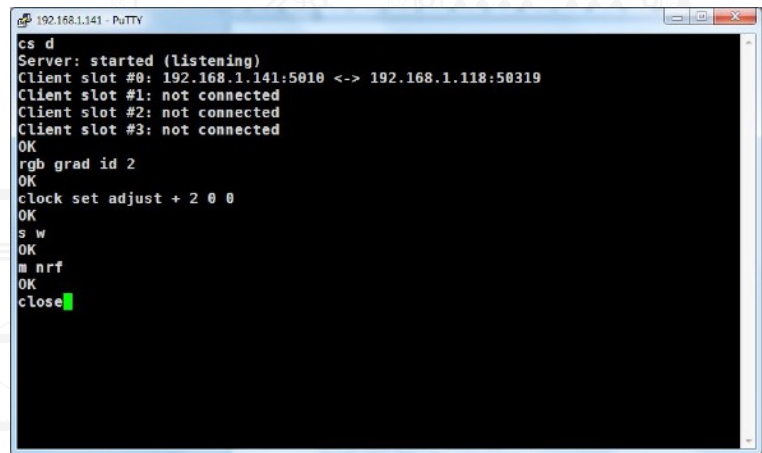
```

COM4 - PuTTY
main set cmdsrvstart yes
OK
main set sleeptime 01 00 00
OK
main set wakeuptime 06 30 00
OK
main set sleepmode yes
OK

settings write
OK
WiFi event 7 - SYSTEM EVENT_STA_GOT_IP
WiFi status 3 - CONNECTED
WiFi got IP: 192.168.1.141
WiFi event 7 - SYSTEM EVENT_STA_GOT_IP
WiFi status 3 - CONNECTED
WiFi got IP: 192.168.1.141
NTP Start
DNS Host By Name Init: err -5, pending
DNS Host By Name CB: pool.ntp.org -> IP address 158.43.128.33
NTP query: packet received 48 bytes
* time_t: 1524337235
* UTC: 2018-04-21 19:00:35
  
```



The clock software allows a network client to connect using port 5010. We can use PuTTY, netcat, or similar terminal program to accomplish this. Once the terminal program has established a connection with the clock, you can send commands the same way you do with the serial monitor. Note that unlike opening the serial interface, establishing a connection over the wireless network won't reset the ESP32-devkitC module.



```
192.168.1.141 - PuTTY
cs d
Server: started (listening)
Client slot #0: 192.168.1.141:5010 <-> 192.168.1.118:50319
Client slot #1: not connected
Client slot #2: not connected
Client slot #3: not connected
OK
rgb grad id 2
OK
clock set adjust + 2 0 0
OK
s w
OK
m nrf
OK
close
```

Here are the contents of settings.txt. You can copy the text from this document in your own editor and adjust the settings to your liking.

```
wifi set ssid "your ssid"
wifi set password "your password"
wifi set local 192.168.1.141
wifi set subnet 255.255.255.0
wifi set gateway 192.168.1.254
wifi set dns 1 192.168.1.254
wifi set dns 2 0.0.0.0
wifi set method static
wifi set hostname "your hostname"

serial set echocmd yes

cs set echocmd no

ntp set hostname "pool.ntp.org"
ntp set addr 0.0.0.0
ntp set method hostname

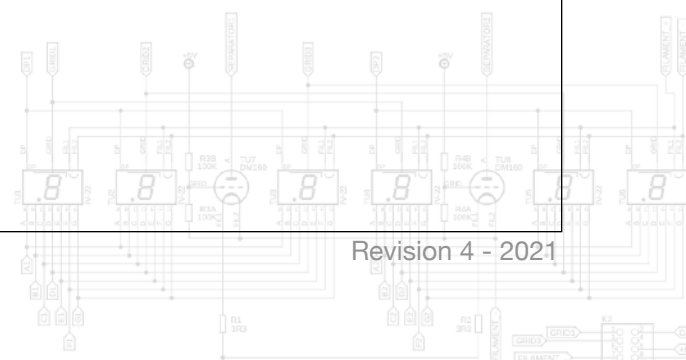
owtemp set enabled yes

lm75 set enabled no
lm75 set adsel 0

clock set adjust + 01 00 00
clock set showdate yes
clock set dateformat dmy
clock set datesepmode off
clock set hourformat 24
clock set hourlz yes
clock set timesepmode blink
clock set showtemp yes
clock set tempformat celsius

rgb set grad id 1
rgb set solid id 17
rgb set sleep raw 0 0 0
rgb set method grad
rgb set grad apply

main set wificonnect yes
main set ntprequest yes
main set ntprefresh 01 00 00
main set cmdsrvstart yes
main set sleeptime 01 00 00
```




```
main set wakeup 06 30 00
main set sleepmode yes

settings write
```

Commands

The clock software accepts an extensive list of commands. There are several groups of commands, each starting with a specific keyword e.g. **wifi**. Many commands can be abbreviated. For example, **w set m st** is short for **wifi set method static**.

Commands that change a setting have a 2nd keyword called **set** followed by the actual setting e.g. **wifi set method dynamic**. Settings are always changed in RAM. Command **s d** reports all current settings in RAM. To make the settings permanent, issue command **s w** (short for **settings write**).

Note that the **settings** command group doesn't change actual settings. These commands manage the settings as a whole.

The commands are documented and implemented in source file **cmd_proc.ino**.

Here's a list of recurring parameters. Parameters that are specific to a command are explained with the command.

Recurring parameters:

```
IPV4AD  IPv4 address, for example: 192.168.1.1
YEAR    1970..2225
MONTH   1..12
DAY     1..31
HOUR    0..23
MINUTE  0..59
SECOND  0..59
RED     0..4095
GREEN   0..4095
BLUE    0..4095
YN      yes|y,no|n
```

Here is a list of all of the available commands.

close	Close the command interface (network connection only)
--------------	---

echocmd ec YN	Enable or disable echoing of commands on the command interface
----------------------	--

sys	System
L reset	System reset

wifi w	Wifi
L start s	Start WiFi
L stop p	Stop WiFi
L dump d	Dump information
L set	Settings
L ssid "..."	Specify the SSID, network name (max. 32 characters)
L password "..."	Specify the password (max. 63 characters)

L local IPV4AD	Static IPv4 local address
L subnet IPV4AD	Static IPv4 subnet mask
L gateway IPV4AD	Static IPv4 gateway address
L dns NUMBER IPV4AD	Static IPv4 DNS address (NUMBER=0..1)
L method m	Method of assigning IPv4 address
L dynamic dyn	Use DHCP
L static st	Use static address
L hostname "..."	Specify the hostname (max. 63 characters)

serial ser	Serial interface
L set	Settings
L echocmd ec YN	Enable or disable echoing of commands on the command interface

cmdsrv cs	Command server
L start s	Start the command server
L stop p	Stop the command server
L dump d	Dump information
L close c	
L all	Close all client connections
L INDEX	Close specific client connection (INDEX=0..3)
L set	Settings
L echocmd ec YN	Enable or disable echoing of commands on the command interface

ntp	NTP
L start s	Start NTP request
L set	Settings
L method m	Method of determining IPv4 address of NTP server
L hostname h	Use hostname for looking up IPv4 address
L addr a	Use IPv4 address
L hostname "..."	Hostname of NTP server
L addr IPV4AD	IPv4 address of NTP server

owtemp	1-Wire temperature sensor
L set	Settings
L enabled ena YN	Enable or disable temperature sensor

lm75 se95d	I ² C temperature sensor
L set	Settings
L enabled ena YN	Enable or disable temperature sensor
L adsel N	Specify A[2..0] pin configuration of the chip (N=0..7)

clock c	Clock
L write w YEAR MONTH DAY HOUR MINUTE SECOND	Write date and time
L read r	Read current date and time
L suspend	Suspend the clock
L resume	Resume the clock
L set	Settings
L adjust [N] +/- HOUR MINUTE SECOND	Adjustment applied on time received from NTP server (multiple adjustments, N=1..4, default is 1)
L selectadjust sa N	Select adjustment (N=1..4)
L showdate sd YN	Show date y/n
L dateformat df dmy mdy ymd	Date format: dmy=day-month-year, mdy=month-day-year, ymd=year-month-day
L datesepmode dsm on off	Mode of separator tubes when showing date: on=always on off=always off
L hourformat hf 12 24	Hour format: 12=12-hour format, 24=24-hour format
L hourlz hlz YN	Show hour with leading zero y/n
L timesepmode tsm blink ampm on off	

```

Mode of separator tubes when showing time:
  blink=blink ones per second,
  ampm=indicate AM/PM,
  on=always on,
  off=always off
L showtemp|st YN          Show temperature y/n
L tempformat|tf celsius|c|fahrenheit|f      Temperature format

```

```

display|d                Display driver
L suspend                Suspend the display driver
L resume                 Resume the display driver

rgb                       RGB driver
L solid|s               Select solid RGB colors
  L id N                 Select scheme from list (N=0..17)
  L raw RED GREEN BLUE  Specify RGB values
L grad|s                Select animated gradient colors
  L id N                 Select scheme from list (N=0..5)
L set                    Settings
  L method|m            Color method
    L solid|s           Use solid RGB colors
    L grad|g            Use animated gradient colors
  L solid|s             Solid RGB colors
    L id N              Select scheme from list (N=0..17)
    L raw RED GREEN BLUE Specify RGB values
    L use               Copy active solid RGB colors to settings
    L apply             Apply solid RGB colors in settings as active
  L grad|g              Animated gradient colors
    L id N              Select scheme from list (N=0..5)
    L use               Copy active gradient colors to settings
    L apply             Apply gradient colors in settings as active
  L sleep               Solid RGB colors shown during sleep mode
    L id N              Select scheme from list (N=0..17)
    L raw RED GREEN BLUE Specify RGB values
    L use               Copy active solid RGB colors to settings
    L apply             Apply solid RGB colors in settings as active during
                        sleep mode

```

```

main|m                    Main function
L ntprefresh|nrf         Perform an NTP request and refresh clock date and time
L set                    Settings
  L wificonnect|wc YN   Automatically connect to wireless network y/n
  L ntprequest|nrq YN   Periodically perform an NTP request y/n
  L ntprefresh|nrf HOURS MINUTE SECOND
                        NTP refresh period (HOURS=0..255)
  L cmdsrvstart|css YN  Automatically start command server y/n
  L sleeptime|st HOUR MINUTE SECOND
                        Time to go to sleep
  L wakeupuptime|wut HOUR MINUTE SECOND
                        Time to wake up
  L sleepmode|sm YN     Enable or disable sleep mode

```

```

settings|s               Settings
L write|w                Write settings from RAM to EEPROM
L read|r                 Read settings from EEPROM to RAM, verify, apply
                        default settings if invalid
L reset                  Reset settings in RAM
L clear                  Clear settings in EEPROM
L dump|d                 Dump information

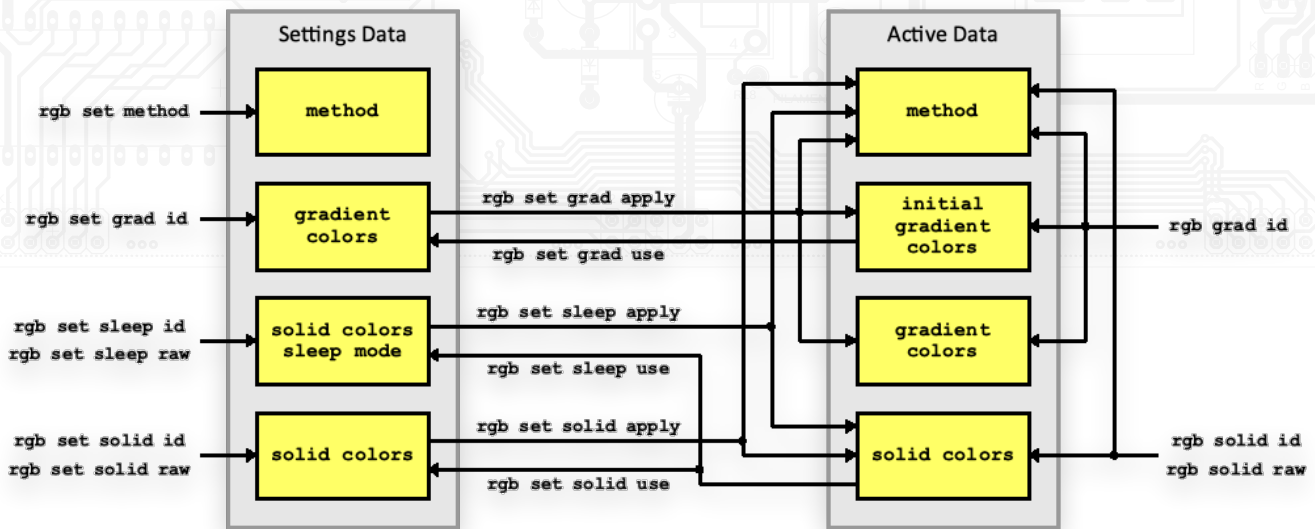
```

The number of **rgb** commands is extensive. This is because the software manages two sets of RGB data, one for settings and one for activity (colors actually being shown). An early version of the

software had one data set, but this design led to confusion when one was updating settings while fiddling with the RGB colors. So it was decided to split up the RGB data into two data sets.

The diagram below shows how the various **rgb** commands influence the data sets. Each command induces a data flow through the associated arrow(s).

The active set stores a copy of the initial gradient colors since the colors and thus the data values are constantly changing when the animated gradient color method is active.



Time Adjustments

The clock receives date and time from an NTP server as Coordinated Universal Time (UTC). The NTP protocol supplies no information about time zones or daylight saving time (DST). To display the time correctly for your physical location, the clock software adds or subtracts a time offset to or from the displayed time and date. This is called time adjustment.

The data set for time adjustment consist of hours, minutes, seconds and a flag that indicates addition or subtraction.

The clock software maintains four time adjustment data sets. There are two commands for managing time adjustment, **clock set adjust** and **clock set selectadjust**. The former sets up a time adjustment data set, the latter selects the active data set.

In addition, the push button on the back panel can be used for controlling the active adjustment. When you hold down the button for a couple of seconds, the tubes display the active adjustment, a value ranging from 1 to 4. By holding down the button a bit longer, the value cycles through its range until you release the button. Once the value changes, a new time adjustment comes into effect, and as soon as the button is released the clock settings are written to non-volatile memory (all settings are written, it's the same action as command **settings write**).

To adjust for a timezone, one data set is required. Daylight saving time requires an additional data set. For example, when your timezone is UTC+1, you can adjust the displayed time with these commands:

```
clock set adjust 1 + 1 0 0  
clock set selectadjust 1
```

To support daylight saving time in this timezone, use a second data set. For example, if your location is timezone UTC+1 and it's the winter months, issues these commands:

```
clock set adjust 1 + 1 0 0  
clock set adjust 2 + 2 0 0  
clock set selectadjust 1
```

When daylight saving time starts, you can hold the push button to change to data set number two. When daylight saving time ends, hold the push button once again to change back to data set number one.

