AUTODESK
Instructables

ClockSketch V7 - Part I

By parallyze in CircuitsClocks

Introduction: ClockSketch V7 - Part I

V7
part I

This Instructable is about v7 of the sketch running on many of the things I've published. As many identical questions scattered across various thin gs are a bit less than ideal I've decided to upload v7 as a thing on its own.

December 2023 Update

All sketches have been updated to v7.4. There was only one change/bug fix:

If WiFi+ NTP + RTC were enabled but the configured ntp server did not respond while resyncing, the time on the RTC could be overwritten with a completely wrong time. This did not happen on the initial sync.

Fixed this, so time on the RTC should now only be overwritten in case a valid ntp response is received.

February 2023 Update

All sketches have been updated to v7.3. Due to a change in the RTC library I'm using I had to fix some things in all sketches. If you had problems using RTC by Makuna v2.3.6 (released a few weeks ago) when compiling, this has been fixed.

Added another check if the RTC connected/configured is running or not. If not, the sketch will try to start the RTC and tell you about this in the Ser ial Monitor.

If using NTP and a RTC it will now sync at 3:01am local time instead of 0:00 UTC.

Fixed a bug where NTP without a RTC would fail if DEBUG was additionally disabled.

Removed a few orphaned brackets and unused variables, cleaned up a bit.

June 2022 Update

Occassionally some people using ESP8266 MCUs end up with heavy flickering/glitching on the LED strip. This is caused by a combination of WS 2812B types and the "bitbanged output" used in the sketch. While trying to figure this out I came across a few things you might want to have a loo k at: https://www.reddit.com/r/FastLED/comments/viem4p/my_quest_for_flicker_some_oddities/

May 2022 Update

Uploaded v7.2 of all sketches for 7 segment models, this one adds optional fading effects. Enable them by uncommenting "FADING" on top of the sketch, also check the options further down inside the sketch (roughly lines #192, where the other options like brightness levels are configured).

Also added Lazy Grid Clock v2 / Grid Clock v2, both will fade all changing pixels, no differentiation between dots/digits.

March 2022 Update

Added v7.1 of all sketches. When using autoDST setting date/time could be a bit tricky, depending on what time the clock is set in combination wit h how much offset there is to UTC. This could lead to a wrong date written to the RTC. This was visible in the serial output (1 day off) - but if one missed this.... anyways, added a workaround. Also autoDST will now display the adjusted date in the serial output, something like this should be more obvious now.

24.09.2021 Update / Info

Please note that there's some problems using esp8266/v3 and FastLED in certain configurations. That's not a problem of this sketch and there's n ot much I can do about it. There's a lot of threads about this on github(fastled/esp core), here's a few examples:

https://github.com/esp8266/Arduino/issues/8054

https://github.com/FastLED/FastLED/issues/1264

I am using esp8266 core v2.7.1 and didn't encounter these issues. So if you're using v3 (check using the boards manager inside the Arduino IDE) a rollback to 2.x might solve issues with flickering leds, especially if it's only the first one. Updates down below (Step 12, "Basic Troubleshooting").

This Instructable will also add some notes to ldr/rtc/nodemcu things, so I recommend reading this before building one of my designs.

I will upload sketches for many of my designs after adapting/testing v7 on them. The ones that get v7 will also be updated accordingly but redirect to this Instructable for downloads.

Also planned is "Part II" of this Instructable - a few words about customizing the sketch and/or adapting it to other strip layouts. One of the goals w hen doing v7 was making the sketch even easier to adapt to different led strip layouts.

I've chosen to add animated GIFs instead of short segments of youtube videos. This is also a great way to avoid off topic comments on videos.... ^^

While resolution/fps are low the purpose is to illustrate some aspects, not showing 4k high def button presses.

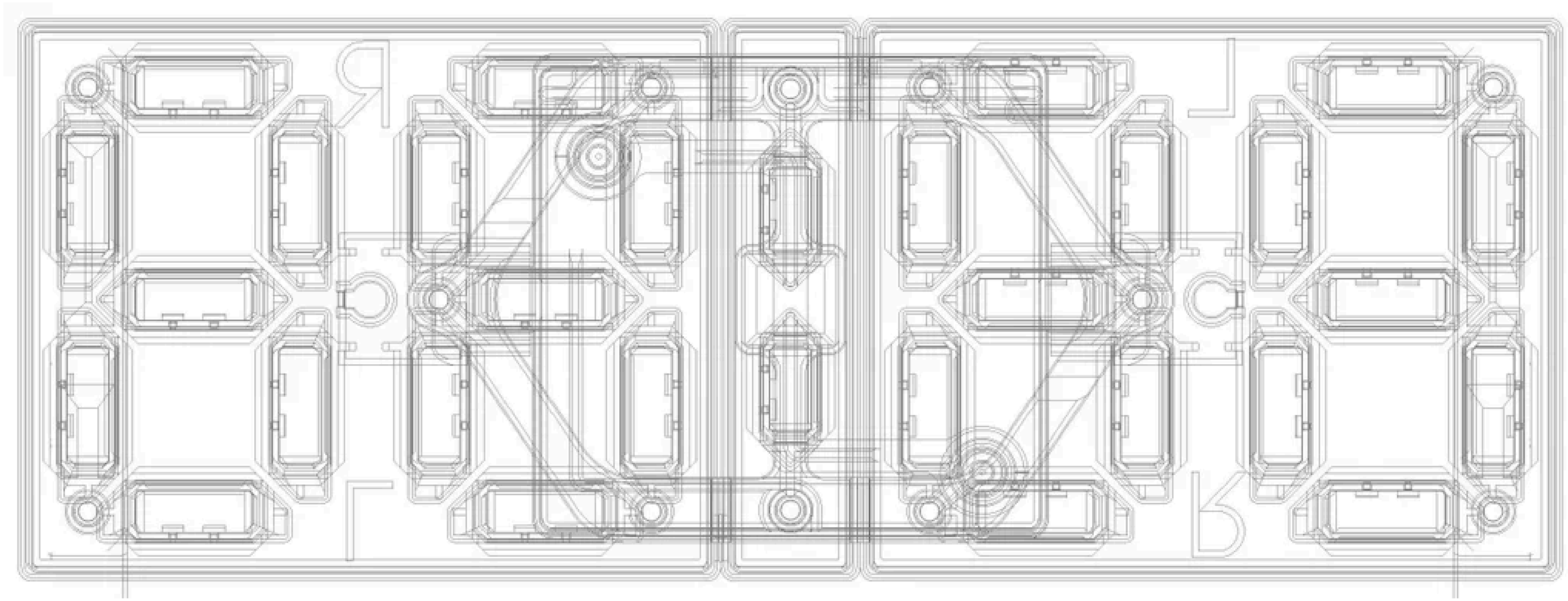
Note:

While working on this Instructable the "Full Preview" stopped working. So I'll be working/editing on this a few times after publishing, there might be a few small changes...

Supplies

For this Instructable one of the supported/tested clocks is required.

Step 1: A Few Words About V7 / Features



Up to v6 there's been many changes over the past years to my sketch. After designing various led displays/clocks with unique led routing inside I always had to somehow adapt the sketch to this. After adding the optional support for nodeMCU/ESP8266 the whole thing got a bit cluttered, so v 7 is a complete rework. Technically this means there hasn't been a v6, it was just a "pre-release"... ^^

While reaching ~610 lines before this one is much bigger at ~2000 lines. This doesn't matter for the ones simply using it on my designs, everythin g is pre-configured to work right away on the configurations shown in the instructions.

Many of the things changed in this sketch won't be really visible to many - but there's been a lot of changes to how the clocks did work so far.

Features:

12h/24h mode, selectable

autoDST - set summer/winter time (DST) automatically depending on configurable rules autoBrightness - adjust brightness automatically by using a photoresistor/LDR

nightMode - switches to a single color at very low light levels, avoids color flickering

Various pre-configured color palettes

Different color modes (per digit, per led, per segment...)

Complete control using 2 buttons or serial input

Supports DS1302 (SPI), DS1307 (I2C), DS3231 (I2C) RTCs

WiFi/WPS, manual SSID/PW, NTP support

Examples for custom things, like time dependent coloring / temperature readouts

Extended test mode to check led configuration step by step

Tested on:

Arduino UNO/Nano/Pro Mini and nodeMCU/EPS8266*, DS3231, WS2812B

Notes:

While I've tested this on nodeMCU/ESP8266 I still don't recommend them for simple things like my clock/grid designs for various reasons. But as most people who don't check the requirements seemingly completely ignore any warning before proceeding I won't cover this topic here...

Changes:

There's lots of minor changes. While the buttons did work as intended I wasn't really happy with the handling inside the main loop. If one wanted t o enter setup and/or change settings using long button presses the clock would always cycle through brightness/color settings.

Button readouts are now improved regarding that. Single button presses will only be registered by the main loop when released, long button press es will stop repeating as soon as an action is executed. Additional buttons should be easier to add because of the way the return values are handl ed.

One of the biggest changes is how time is handled inside the sketch. For a while I thought on a clock like this it'd be enough to sync system time t o the rtc regulary. So depending on the quality/drift on the microcontroller used one could see the dots either lighting up or being blank for a bit lon ger than a second. This was caused by the difference between system and rtc time.

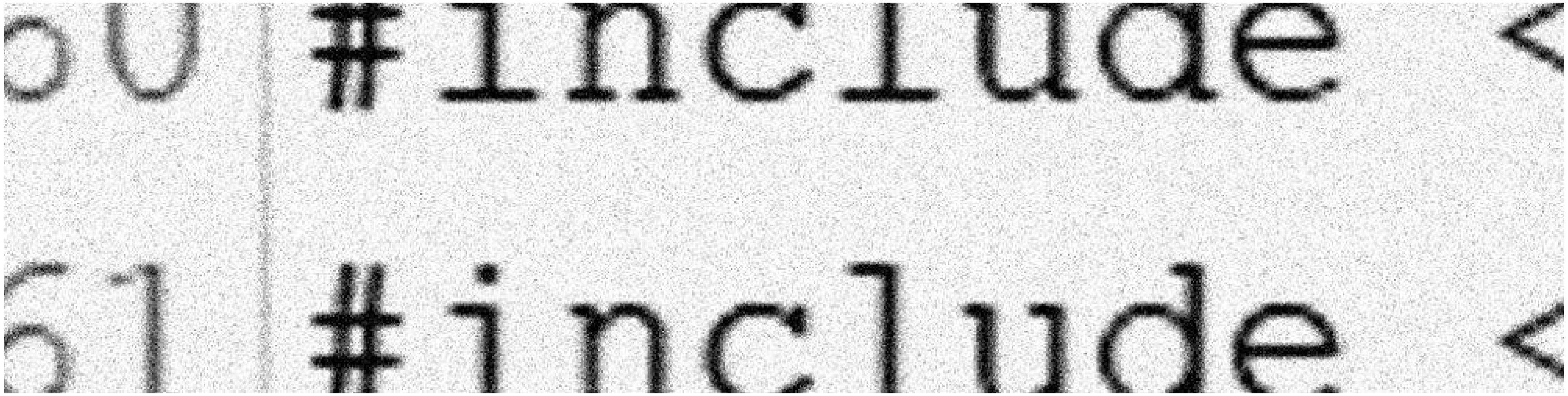
As it turned out those cheap boards can drift quite a lot more depending on the load the sketch is causing. While testing I've had Arduinos which were running ahead by multiple seconds after less than 10 minutes of testing. So the sketch will now ignore system time in many places and run ri ght off of the rtc which is read out every 50ms.

Lots of changes have been done to the three most important functions: Displaying a segment, displaying digits/symbols and their arrays. While pr evious sketches had definitions for led strip lengths/amount of leds it's handled completely different in v7. These changes allow much more advan ced configuration options and should be easier to understand when using this sketch on custom clocks. Also this does allow for mixing different la youts, like small 7 segment displays for HH, bigger ones for MM and a grid for SS. More of this will be explained in Part II, as it's not something co nsidered "basic usage and options"...

Coloring everything is done differently, also. When doing display updates the leds inside the array only get "flagged" using a single color. Before o utputting the data to the leds the coloring will take place, independently of other stuff. This allows for displaying custom values while keeping color s consistent. Also this makes adding more coloring modes easier.

Debug input/output has seen some improvements, all button presses (long/short) can be sent using numbers in the serial monitor. In theory this al lows for building the clocks without buttons and can be extended to remote control them (in the end all you have to do is send single numbers usi ng the serial connection...).

Step 2: Configuration / Libraries



```
#include <
#include <
```

Pre-configured sketches will always be using the most basic features/options as mentioned in the corresponding instructions. So they're set up fo r:

- Arduino

- DS3231 RTC

- WS2812B leds

Libraries used in this sketch:

"FastLED" by Daniel Garcia

"Time" by Michael Margolis (Maintainer "Paul Stoffregen" in newer builds of the Arduino IDE)

"Rtc by Makuna" by Michael C. Miller

Update: Added a video showing the installation of all of the above required libraries

Depending on options set there might be other libraries required:

"Timezone" by Jack Christensen

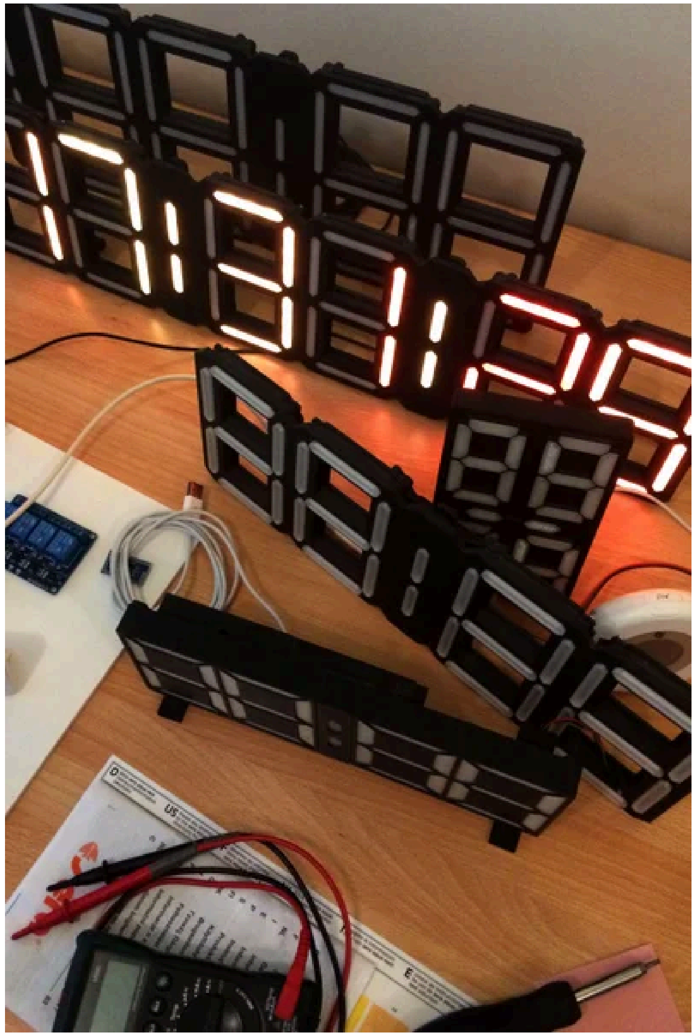
"NTPClient" by Fabrice Weinberg

Libraries used in my sketches are always available using the Library Manager inside the Arduino IDE. So please use that, don't copy/paste files a cross multiple Arduino IDE installations, clone from github, copy/paste github links inside the sketch(!!!), rename includes to match installed librari es...

If you ignore all the requirements I mention in my instructions, like a properly set up Arduino IDE, installing libraries, knowing the difference betwe en compiling and uploading a sketch - go ahead, but don't expect me to react to comments/messages every time...

Also note this Instructable is about v7, basic usage/configuration and the sketches provided for various things. This is NOT about "you building clo ck 4 led in segment!11 led do how sketch work make do?!?! show temperature!!?"... wait for Part II.

Step 3: Tested Sketches



Sketches for the various models I've tested so far using v7 are attached to this step.

7 Segment Clocks

Retro 7 Segment Clock - The Final One(s) - 3 LEDs/segment (4 and 6 digits) https://www.instructables.com/Retro-7-Segment-Clock-the-Final-Ones/

(ClockSketch_v7-7SCv3-TFO-3.ino)

XL - 4 LEDs/segment (4 and 6 digits)

(ClockSketch_v7-7SCv3-TFO-4.ino)

Retro 7 Segment Clock - Regular Edition (4 and 6 digits)

https://www.thingiverse.com/thing:3014572

(ClockSketch_v7-7SCv2-RE.ino)

Retro 7 Segment Clock - Small Edition (4 digits only)

https://www.thingiverse.com/thing:3095715

(ClockSketch_v7-7SCv2-SE.ino)

Retro 7 Segment Clock - XL Edition (4 and 6 digits)

https://www.thingiverse.com/thing:3136688

(ClockSketch_v7-7SCv2-XL.ino)

Lazy 7 - Quick Build Edition (4 digits)

https://www.instructables.com/Lazy-7-Quick-Build-E...

(ClockSketch_v7-L7-QBE.ino)

S7ripClock - Basic Edition (4 digits)

https://www.instructables.com/S7ripClock-Basic-Edi...

(ClockSketch_v7-S7ripClock-BE.ino)

7 Segment Clock - Small Printers Editions (4 and 6 digits)

https://www.instructables.com/7-Segment-Clock-Smal...

(ClockSketch_v7-SPE.ino)

7 Segment Clock - Tiny Edition (4 digits)

https://www.thingiverse.com/thing:3865571

(ClockSketch_v7-TE.ino)

Lazy 7 / Mini (4 digits)

https://www.instructables.com/Lazy-7-One/

(ClockSketch_v7-L7-Mini.ino)

Lazy 7 / One (4 and 6 digits)

https://www.instructables.com/Lazy-7-One/

(ClockSketch_v7-L7-One.ino)

Grid Clocks

Lazy Grid Clock v2

https://www.instructables.com/Lazy-Grid-Clock-V2/

(ClockSketch_v7-LGCv2.ino)

Grid Clock v2

https://www.thingiverse.com/thing:3433644

(ClockSketch_v7-GCv2.ino)

Lazy Mini Grid v1

https://www.instructables.com/Lazy-Mini-Grid/

(ClockSketch_v7-LMGv1.ino)

Word Clocks

Lazy Words v1

https://www.instructables.com/Lazy-Words-Single-LED-Strip-Word-Clocks/

(ClockSketch_v7-Lazy_Words.ino)

Notes:

The 6 digit version of Lazy 7 / One is extremely low on RAM when using an Arduino, mainly caused by the massive amount of 388 leds. On the 6 digit version you will only be able to run it successfully if you comment out "#define DEBUG" on top of the sketch. RAM will be just enough to ena ble autoDST and autoBrightness, using a temperature sensor with a required library or other stuff might not be possible to add!

The sketches for Lazy Grid Clock v2 and Grid Clock v2 are quick'n'dirty modifications. There's only the same two color modes as in v6 and the C USTOMHELPER for custom 7 segment layouts is missing. GCv2 configuration for 18/27 pixels is selected by setting LED_DIGITS to 4 or 6, like o n the 7 segment clocks.

Step 4: Usage Instructions

While the clock is running you can use the buttons and/or serial input (numbers) to do the following:

Button A -> Switch brightness levels (7 using serial input)

Button B -> Switch color palette (8)

Long press Button A -> Switch coloring mode (4)

Long press Button B -> Switch 12h/24h mode (5)

Long press Button A + Button B -> Enter date/time setup (6)

Short press A+B is currently not used (9).

When switching color palettes/color modes the clock will display the newly selected colors/pattern for 3 seconds. This can be disabled using the b asic options.

Note:
All selected options will be written to the eeprom. I do not recommend setting things like 12h/24h display mode inside the sketch, use the buttons/ serial input to select and write the desired setting to the eeprom.

The sketch will always use the push button connected to d3 as buttonA, the one connected to d4 is buttonB. As it's a personal preference how to put them inside the electronics case (A-B left-right looking from the front or back?) I don't tell people which one to put into either the left or right sp ace... so I simply can't tell you if the button "on the left" is A or B on the clock you built... ^^

Step 5: Time Setup

I suggest selecting 12h/24h mode before setting time. The sketch will now indicate the mode selected so there should be no confusion if switching 12h/24h at 3am. Press and hold button B to switch between 12h/24h mode.

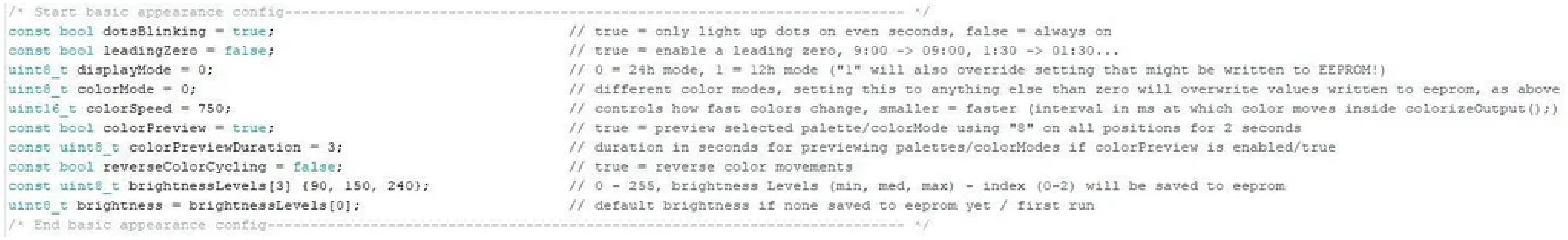
Press and hold A + B until the display gets cleared.

Button A will increase the current value, button B will proceed to the next setting.

In 12h mode a single dot (upper ones on 6 digit configurations) will indicate AM times, both dots indicate PM. This is only while in setup, not later on with the clock running.

There's some animated GIFs showing setup in 24h mode, in 12h mode (note the single dot at 4am, clock is finally set to 4pm) and switching 12h/24h mode.

Step 6: Basic Options / Note About LEDs and Power Limits



```
/* Start basic appearance config------------------------------------------------------------------------------ */
const bool dotsBlinking = true;                          // true = only light up dots on even seconds, false = always on
const bool leadingZero = false;                          // true = enable a leading zero, 9:00 -> 09:00, 1:30 -> 01:30...
uint8_t displayMode = 0;                                 // 0 = 24h mode, 1 = 12h mode ("1" will also override setting that might be written to EEPROM!)
uint8_t colorMode = 0;                                   // different color modes, setting this to anything else than zero will overwrite values written to eeprom, as above
uint16_t colorSpeed = 750;                               // controls how fast colors change, smaller = faster (interval in ms at which color moves inside colorizeOutput();)
const bool colorPreview = true;                          // true = preview selected palette/colorMode using "8" on all positions for 2 seconds
const uint8_t colorPreviewDuration = 3;                  // duration in seconds for previewing palettes/colorModes if colorPreview is enabled/true
const bool reverseColorCycling = false;                  // true = reverse color movements
const uint8_t brightnessLevels[3] {90, 150, 240};        // 0 - 255, brightness Levels (min, med, max) - index (0-2) will be saved to eeprom
uint8_t brightness = brightnessLevels[0];                // default brightness if none saved to eeprom yet / first run
/* End basic appearance config------------------------------------------------------------------------------ */
```

There's a few parameters which can be used to quickly modify the display options in certain ways. These can be set inside the sketch (around line ~170). In case you'd like to enable leading zeros in front of the hours like 9:00 -> 09:00, change the dots from blinking to always on or if you simpl y want to change the three brightness levels, this is the place to look at.

If you're building a 6 digit version of one of my designs you will have to tell the sketch about this. Look for "LED_DIGITS", around line 250:

```
#define LED_DIGITS 4                                        // 4 or 6 digits, HH:MM or HH:MM:SS
```

By default all the sketches have 4 digits configured. The ones supporting 6 (depending on model, mentioned in the instructions) can be set to 6 he re.

There is another thing you might need to adjust, depending on how many leds are used and the requirements (specific colors/brightness), the led power limit.

WS2812B leds can draw quite some power, up to 60mA each according to the data sheets. I've written about this before, so I won't repeat everyth ing again.

All the sketches I upload with my designs are using a power limit (based on the FastLED library).

While previous releases had power limits somewhat adjusted to the amount of leds (usually in the range of ~500-800mA), v7 will now have a 500 mA limit by default on all preconfigured sketches.

Many people simply ignore everything they're told about power consumption and how to not connect leds to an Arduino. So from now on the sketc h is "500mA usb port"-safe.

This is not really a problem for designs like the "Small Printers Edition" or "S7ripClock", using < 70 leds each. Depending on the selected brightne ss setting they're usually in a range of 300mA-450mA.

But if you're raising the brightness levels and/or are using different colors/white a lot you might have to increase the power limit. Or simply if you're using a design like the Retro 7 Segment Clock with 4/6 digits, which will also hit the limit quite easily (99/151 leds total).

There's an easy way to see if the clock is hitting the power limit:

Select a palette with bright tones (like the blue one included) and switch to the highest brightness setting. Set time to something that will light up a s many segments as possible, like "23:58", not "1:11". If all the digits dim a bit down when the dots are lit -> you're very likely hitting the power limi t.

If you need to raise the limit, this can be found around line 250 inside the sketch:

```
#define LED_PWR_LIMIT 500         // 500mA - Power limit in mA (voltage is set in setup() to 5v)
```

Note:
Make sure you don't set this higher than the wiring (and power supply) you're using does allow for!

Step 7: AutoDST

Enabling autoDST will make the clock change time according to what's called "Time Change Rules".
Enabling autoDST is as easy as uncommenting "#define autoDST" inside the sketch:

```
// autoDST - uncomment to enable automatic DST switching, check Time Change Rules below!
#define AUTODST
```

The moment the sketch is uploaded with autoDST enabled the clock will use UTC time internally. Also it will store UTC time to the rtc, only the dis played values (time/setup) will be adjusted to local time. So whenever you're switching between autoDST on/off you'll have to set time again.

Also autoDST will enable setting the year/month/day in setup, this is required to make autoDST work at all.

There's two sets of time change rules inside the sketch:

```
  //-------------------------------------------------
  /* US */
  // TimeChangeRule tcr1 = {"tcr1", First, Sun, Nov, 2, -360};
     /* utc -6h, valid from first sunday of november at 2am */
  // TimeChangeRule tcr2 = {"tcr2", Second, Sun, Mar, 2, -300};
     /* utc -5h, valid from second sunday of march at 2am */
  //-------------------------------------------------
  /* Europe */
  TimeChangeRule tcr1 = {"tcr1", Last, Sun, Oct, 3, 60};
     /* standard/winter time, valid from last sunday of october at 3am, UTC + 1 hour */
  TimeChangeRule tcr2 = {"tcr2", Last, Sun, Mar, 2, 120};
     /* daylight/summer time, valid from last sunday of march at 2am, UTC + 2 hours */
  //-------------------------------------------------
  Timezone myTimeZone(tcr1, tcr2);
```

Adjust/add rules as you need them - but make sure only two of them are active/uncommented, so avoid something like this:

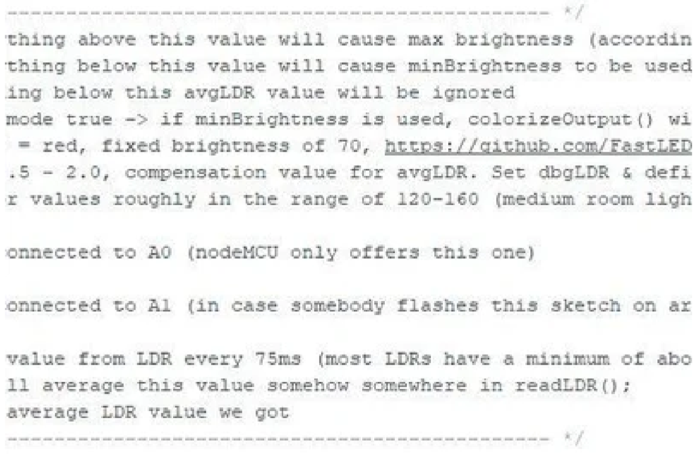
```
//-------------------------------------------------
  /* US */
  TimeChangeRule tcr1 = {"tcr1", First, Sun, Nov, 2, -360};
     /* utc -6h, valid from first sunday of november at 2am */
  TimeChangeRule tcr2 = {"tcr2", Second, Sun, Mar, 2, -300};
     /* utc -5h, valid from second sunday of march at 2am */
  //-------------------------------------------------
  /* Europe */
```

```
TimeChangeRule tcr1 = {"tcr1", Last, Sun, Oct, 3, 60};
   /* standard/winter time, valid from last sunday of october at 3am, UTC + 1 hour */
TimeChangeRule tcr2 = {"tcr2", Last, Sun, Mar, 2, 120};
   /* daylight/summer time, valid from last sunday of march at 2am, UTC + 2 hours */
//-----------------------------------------------
Timezone myTimeZone(tcr1, tcr2);
```

(this will throw an error because you're using tcr1 and tcr2 twice)

If you need help on configuring these rules, have a look at the instructions from the timezone library here: https://github.com/JChristensen/Timezone

Step 8: AutoBrightness / LDR

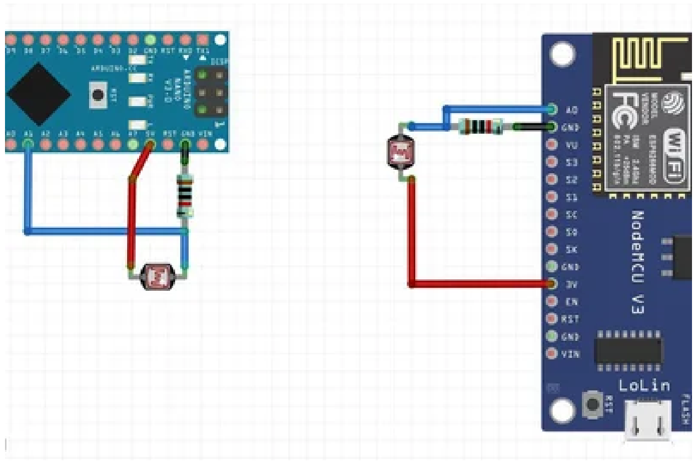


```
----------------------------------------------- */
thing above this value will cause max brightness (accordin
thing below this value will cause minBrightness to be used
ing below this avgLDR value will be ignored
mode true -> if minBrightness is used, colorizeOutput() wi
 = red, fixed brightness of 70, https://github.com/FastLED
.5 - 2.0, compensation value for avgLDR. Set dbgLDR & defi
r values roughly in the range of 120-160 (medium room ligh

onnected to A0 (nodeMCU only offers this one)

onnected to A1 (in case somebody flashes this sketch on ar

value from LDR every 75ms (most LDRs have a minimum of abo
ll average this value somehow somewhere in readLDR();
average LDR value we got
----------------------------------------------- */
```



If there's an LDR connected (A1 on Arduinos in my designs, A0 on nodeMCUs) it can be used to adjust the clocks brightness setting according to the environment.

Uncomment #define autoBrightness inside the sketch to enable this:

```
// autoBrightness - uncomment to enable automatic brightness adjustments by using a photoresistor
#define AUTOBRIGHTNESS
```

There's additional options to set a bit further below in the sketch. The animated GIF does show "nightMode = true" and switches to red at very low levels. Using primary colors at very low levels avoids strange color effects (big "steps", like banding in gradients).

Switching brightness levels is still possible using an LDR, the color used for "nightMode" will be at a fixed brightness setting for all three levels.

Depending on the LDR used (I've been using GL5516 ones and 10kOhm resistors so far) you might want to set "dbgLDR" to true to output the val ues from your LDR to the serial monitor. The range can be adjusted by setting "factorLDR".

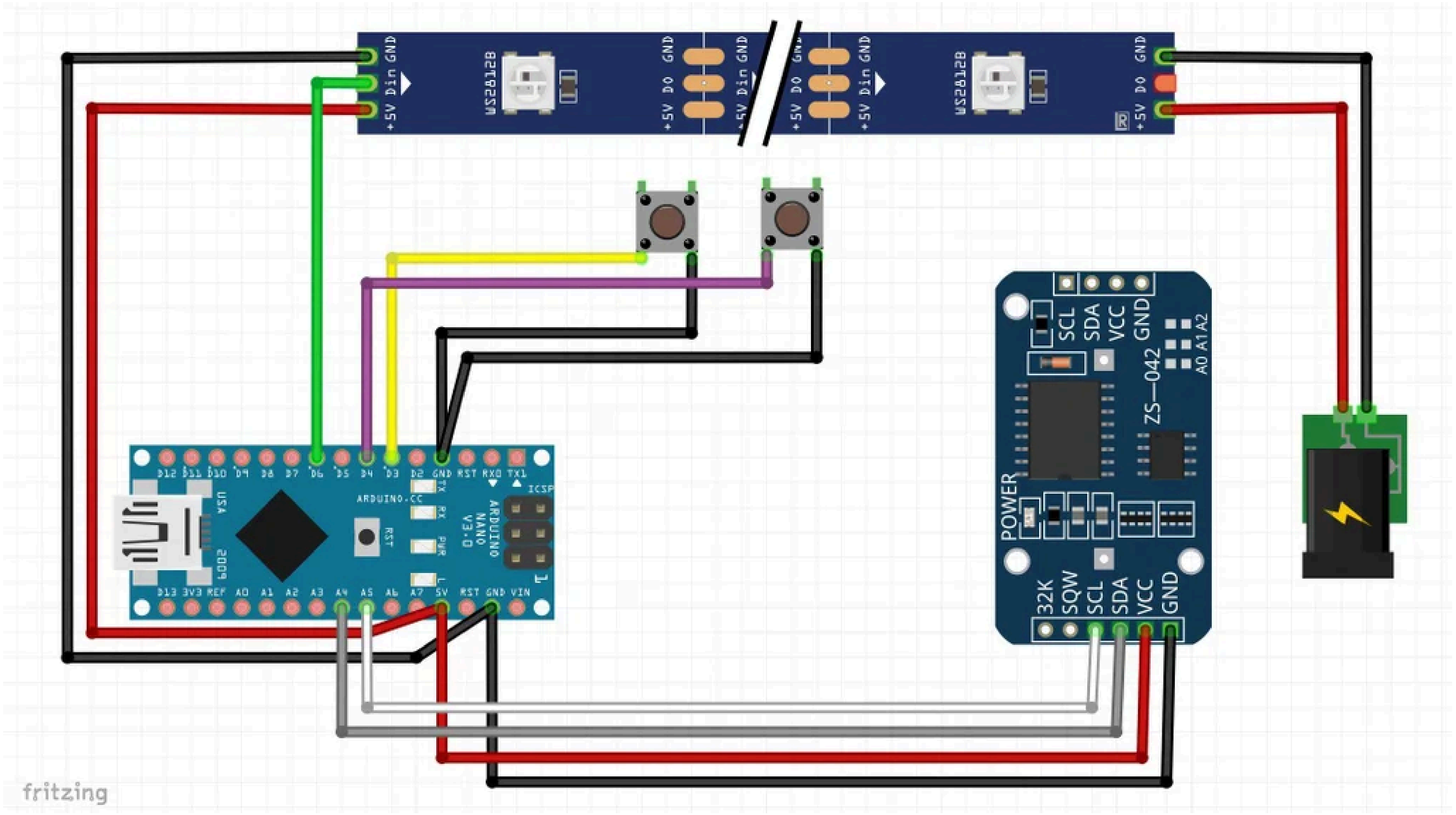
Note:
People have been using one of my older instructions pointing to the Arduino Playground as a reference for connecting the LDR. While that is a go od place to start, there's one thing to consider:
If using a nodeMCU/ESP8266 the LDR must not be powered by 5V like on Arduinos. Use the 3V pin on nodeMCUs for this!

The analog input on an ESP8266 will only handle a max of ~1.0 Volts. nodeMCUs and other boards already have resistors included to allow for 3. 3v, so there's less confusion with the logic levels.

Be careful here as there's only one analog input available on nodeMCUs and other ESP8266 based boards.

There's a "schematic" added showing how I'd recommend connecting the LDR on Arduino vs. nodeMCU.

Step 9: Electronics



So far the basic functions have been taken care of, let's add a few words about the electronics used. While the led strips are routed in different wa ys, sometimes a single piece and other times multiple pieces, the rest is always the same:

- Arduino Nano/Pro Mini (AtMega 328, 5V, 16 MHz)
- WS2812B LEDs connected to d6
- 2 push buttons connected to d3/d4
- ds3231 rtc using i2c (pins a4+a5)

If you have a look at the "schematic" in this step you'll notice it is actually not very complicated. At least it should, because I always tell people the se are not projects with complete beginners in mind...

Personally I'm often using Arduino Pro Minis. So the first schematic I once added had everything shown using a Pro Mini. This resulted in quite so me mails, comments and messages about people having problems programming them. Or not even knowing they do not come with a usb port.

That's a great example of what I mean when I say "basic knowledge" in the introductions to my things/instructables.

For beginners I strongly recommend using the Nano, not the Pro Mini. And no, please don't send me screenshots of your shopping carts on Ali Ba ngpress and other sites, I simply will not react to that.

Sketch v7 will not work on AtMega168 chips any longer. While it's possible to stay below the 1kb RAM limitation (on some models, depends on le d count) the sketch requires more than the 16k flash available on them.

All previous sketches were using the DS3232RTC library by Jack Christensen. So I always told people to use either a DS3231 (recommended) or DS1307.

But for some reason people send me messages about their clock not working. And after a while it turns out they just bought $some rtc, often a DS 1302.

Believe it or not - but different types of rtcs happen to come with different interfaces. SPI vs. i2c in this case. So if I say "DS3231 or DS1307" I sim ply do NOT mean "DS1302".

Sketch v7 does support DS1302 modules in theory. I simply have not tested this because I won't buy one. As far as I can tell the DS3231 is better in many ways and there's lots of reports about DS1302 boards draining the battery if not powered. So if you happen to have a DS1302 at hand it should work. But you will need to check the pin assignments and wiring for yourself (DS1302 requires different connections than a DS3231).

The thing you have to watch out for are "bad DS3231 chips". While some sellers advertise (and show images of) the DS3231SN chip, you often e nd up with DS3231M chips. They're missing features of the SN versions, temperature compensation for example. There's nothing I can do about t his. But you can complain wherever you bought them and request the proper ones.

So far I'm using the ZS-042 modules most of the time. There's smaller modules like the "DS3231 for Pi" ones - but I had packs of 5 delivered with up to 4 DS3231M-chips which aren't really good...

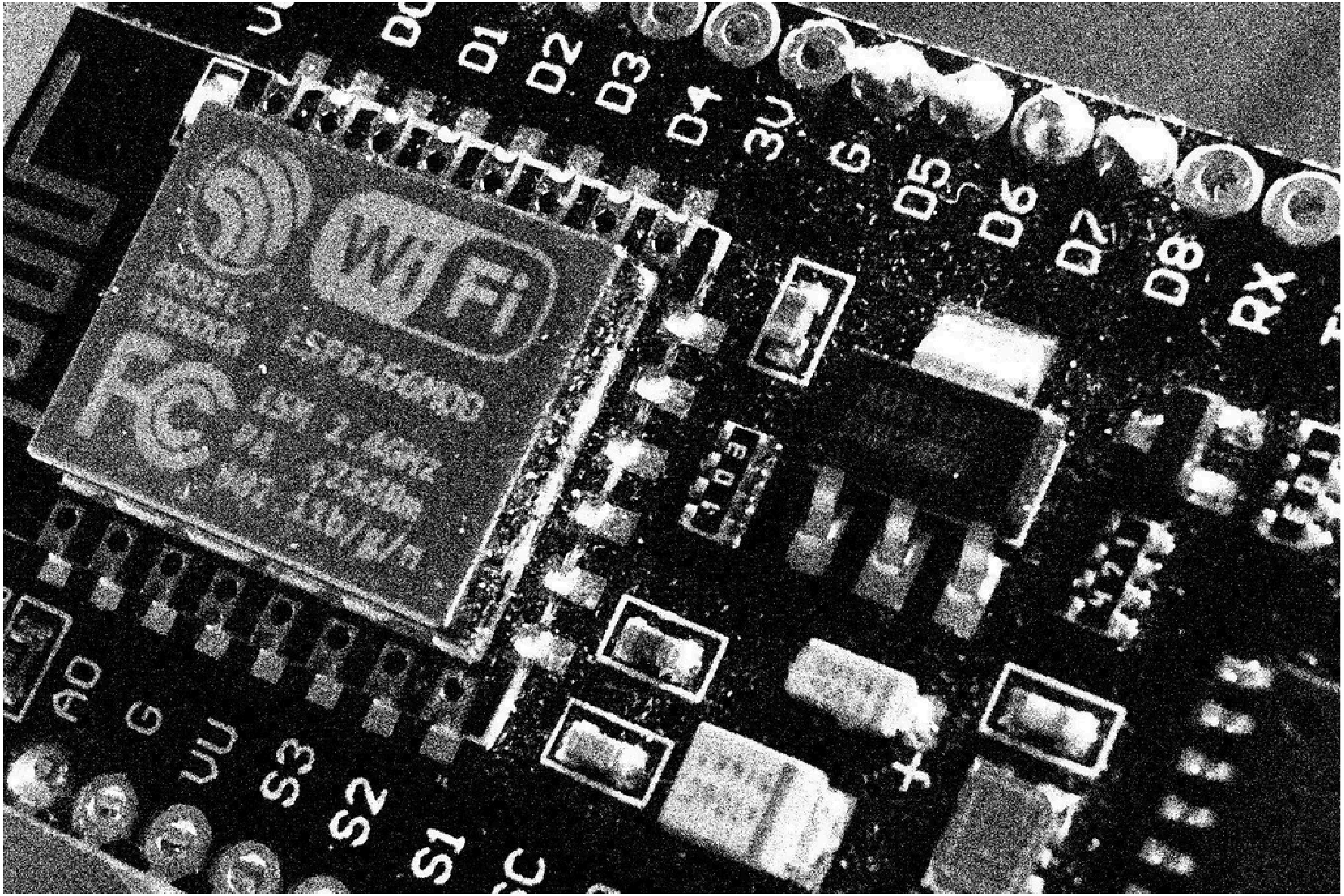
What you should know about the ZS-042 is their charging circuit. Many of these boards come with an active charging circuit. Simply ignoring this, connecting the module to 5v and putting in a cheap battery will kill the battery within a short amount of time. I always check for this and remove co mponents on the pcb according to this - but there is an easy workaround if you are using the Arduino Nano:

Connect the rtc to 3v instead of 5v on the Arduino. That's it.

When doing the schematics for the instructions I simply wasn't thinking of that - Pro Minis don't offer 3.3v like the Nano and I check the modules a nyways...

Good read on that topic: https://www.onetransistor.eu/2019/07/zs042-ds3231-...

Step 10: NodeMCU / ESP8266


D1
D2
D3
D4
3V
G
D5
D6
D7
D8
RX
WiFi
ESP8266MOD
A0
G
VU
S3
S2
S1

If you want to compile this for nodeMCU/ESP8266, make sure you've selected the proper board type inside the Arduino IDE.

Uncomment "#define NODEMCU" on top of the sketch:

```
// nodeMCU - uncomment to compile this sketch for nodeMCU 1.0 / ESP8266, select proper board
// type inside the IDE! This mode is NOT supported and only experimental!
#define NODEMCU
```

If using a nodeMCU the sketch will default to these pins:

LEDs:

```
#define LED_PIN 12                          // led data in connected to GPIO_12 (d6/nodeMCU)
```

Buttons:

```
const uint8_t buttonA = 13;         // momentary push button, 1 pin to gnd, 1 pin to d7 / GPIO_13
const uint8_t buttonB = 14;         // momentary push button, 1 pin to gnd, 1 pin to d5 / GPIO_14
```

RTC/i2c is using d1/d2, spi whatever you've set it to.

I've tested all the sketches running on a nodeMCU 1.0, still I don't recommend/support them. There's various reasons for this, most important of th em all: It's simply not required most of the time.

Another important thing: As people are often very ignorant and don't like to read much, they simply don't care about differences between Arduino s/nodeMCUs... and this can lead to trouble. One major thing to keep in mind is that WS2812 leds operate on 5v logic levels while nodeMCU/ESP 8266 boards use 3.3v logic levels.

Most of the time it will work right away - but it's not guaranteed. Also this depends on the type of board you're using. As far as I can tell the Wemo s D1 Mini seems to be more prone to problems here... but that's just my very limited experience regarding that.

This can be very specific to the combination of microcontroller / leds (ws2812, ws2811, ws2801), so be warned...

Here's something you might want to have a look at regarding logic levels:

https://learn.adafruit.com/neopixel-levelshifter

https://hackaday.com/2017/01/20/cheating-at-5v-ws2...

If you experience flickering you should always make sure the data line is fine. If using a resistor, place it close to the led strip, not close to the microcontroller. And keep the data line as short as possible - I've seen pictures where people run them through multiple breadboards before finally going to the led strip 3-5ft away...

Also it helps to have a really close look at the effect you're seeing... sometimes there's random leds going on/off, others showing wrong colors. This is very likely a problem with a badly shielded/bad soldered data line.

Another type of flicker I've seen is some kind of "high frequency overshoot". While the time is displayed as it should there's always some additional leds flickering at very high speeds, following the route of the led strip. I've only seen this two times happening on ESP8266 based boards, both times colors were right, just the additional segments flickering on/off. The problem disappeared the moment a simple "delay(1);" was added to the main loop.

Sketch v7 does handle led updates in a different way, so if the high update/refresh rate was causing this, this should not happen any longer.

The only way so far I was able to get any flickering across the leds on my nodeMCU was to insert ~3ft of additional wire to the data line running through multiple crimped dupont connectors and pull them out half way....

Note:

If you experience flickering using WS2812B on ESP8266 and checked twice there's nothing wrong with power and/or data lines, you might have gotten a WS2812B with slightly different timing specs - from looking at the LEDs you simply cannot tell what exact timings they require:

https://www.reddit.com/r/FastLED/comments/viem4p/my_quest_for_flicker_some_oddities/

Step 11: WiFi and NTP

...this covers the last two things in Part I, WiFi support and using a NTP server for time requests.

Enable WiFi support by uncommenting "#define USEWIFI" on top of the sketch:

```
// useWiFi - enable WiFi support! If no WPS support is available on a router check settings
// further down, set useWPS to false and enter ssid/password there
#define USEWIFI
```

To connect the clock to your network there's two ways:

1. WPS, WiFi Protected Setup (default)

The clock will try to connect to your network when powered on. So if the controller used is already connected and you didn't reset the wifi settings while flashing the sketch, there's nothing to do.

To connect to a network using WPS, simply press A + B and hold them for a few seconds after initiating WPS on your router/access point. The clo ck will try a few times to connect and if successful, will display "SSid".

2. Using SSID/password

If your router/ap doesn't offer WPS you can enter the required information in the sketch. Look for this part and enter the proper ssid/password ther e. Also make sure you set useWPS to false.

```
/* Start WiFi config/parameters------------------------------------------------------------- */
#ifdef USEWIFI
  const bool useWPS = true;          // set to false to disable WPS and use credentials below
  const char* wifiSSID = "maWhyFhy";
  const char* wifiPWD = "5up3r1337r0xX0r!";
#endif
/* End WiFi config/parameters--------------------------------------------------------------- */
```

NTP support

If you'd like to get time from a ntp server, enable this by uncommenting "#define USENTP" inside the sketch:

```
// useNTP - enable NTPClient, requires NODEMCU and USEWIFI. This will also enforce AUTODST.
// Configure a ntp server further down below!
#define USENTP
```

Add a server for your requests a bit further below:

```
/* Start NTP config/parameters----------------------------------------------------------------
   Using NTP will enforce autoDST, so check autoDST/time zone settings below!                  */
#ifdef USENTP
  //#define NTPHOST "europe.pool.ntp.org"
  #define NTPHOST "192.168.2.1"
  #ifndef AUTODST
    #define AUTODST
  #endif
#endif
/* End NTP config/parameters------------------------------------------------------------------ */
```

(Make sure there's only one NTPHOST defined, the commented one is an example, I was using a local ntp service)

I strongly recommend using a local ntp server if available. And I absolutely don't recommend building a clock without an rtc, polling public ntp serv ers 50 times a day.

Also note that autoDST will be enabled when using ntp. Many people seem to think the internet does everything for you... it doesn't. NTP (usually) will give you UTC time. And as long as you don't live in a place where that's the correct time through the whole year, you will have to define offset s. This is done using the time change rules for autoDST.

~~The clock will try to resync to the ntp server each day at 0:00 utc.~~

Changed to 3:01am local time from 0:00 utc in version 7.3.

Note:

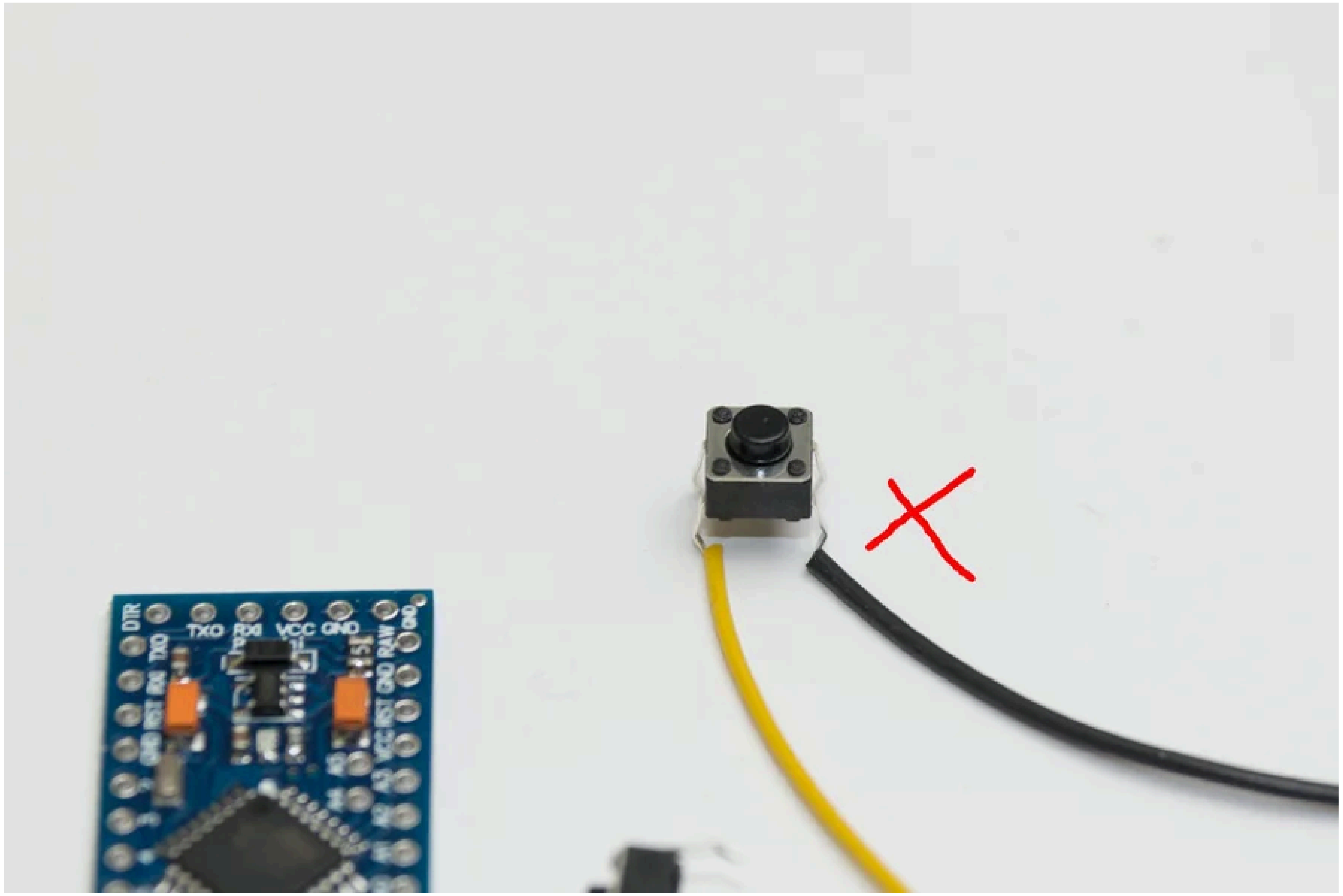
Using WiFi/ntp isn't something I'd recommend. The idea behind my designs/clocks is to have something that's usable without requiring the interne t. Also using wifi/ntp doesn't really give any benefit here, the clocks I have running on DS3231 modules drift as low as 2-7 seconds/year.

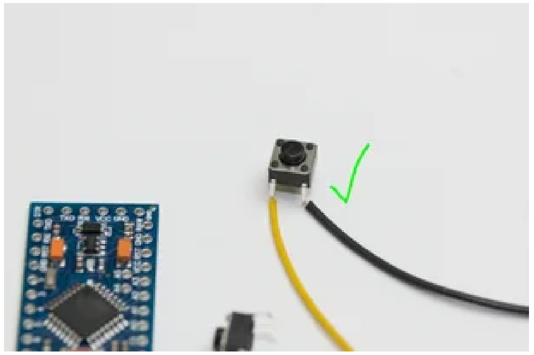
So while simply not using a rtc and using ntp with dozens requests a day is technically possible, it doesn't seem very "smart" to me. The moment your connection drops the clock will loose it's time source. The moment you change router/ap/ssid settings you have to reflash (unless WPS is su pported and used). You can't use the clock when there's no WiFi available.

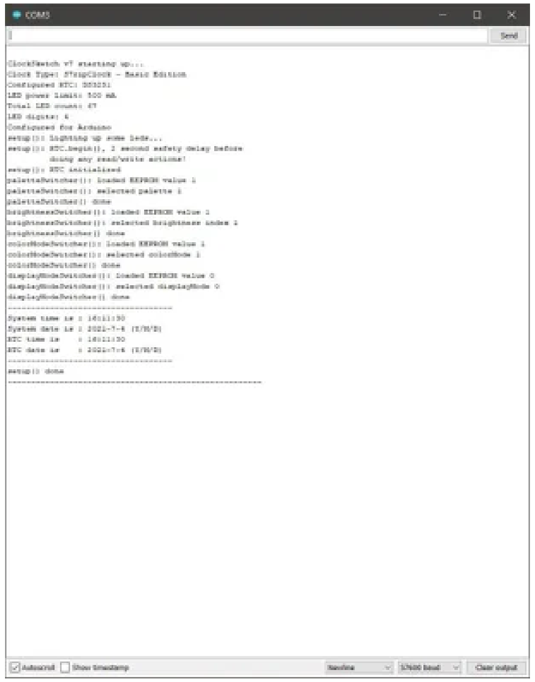
And if you wanted to give the clock away as a present... you'd have to ask for ssid/pw to flash it. Everytime the one who got the clock changes so mething on his network....

Actually I could go on for days talking about the downsides of this... I don't really want to, but I need something I can point people at if they ignore d my recommendations. ;)

Step 12: Basic Troubleshooting







```
ClockSketch v7 starting up...
Clock Type: StripClock - Basic Edition
Configured RTC: DS3231
LED power limit: 500 mA
Total LED count: 47
LED digits: 6
Configured for Arduino
setup(): Lighting up some leds...
setup(): RTC.begin(), 2 second safety delay before
         doing any read/write actions!
setup(): RTC initialized
paletteSwitcher(): loaded EEPROM value 1
paletteSwitcher(): selected palette 1
paletteSwitcher() done
brightnessSwitcher(): loaded EEPROM value 1
brightnessSwitcher(): selected brightness index 1
brightnessSwitcher() done
colorModeSwitcher(): loaded EEPROM value 1
colorModeSwitcher(): selected colorMode 1
colorModeSwitcher() done
displayModeSwitcher(): loaded EEPROM value 0
displayModeSwitcher(): selected displayMode 0
displayModeSwitcher() done
----------------------------------------
System time is : 16:11:30
System date is : 2021-7-6 (Y/M/D)
RTC time is    : 16:11:30
RTC date is    : 2021-7-6 (Y/M/D)
----------------------------------------
setup() done
--------------------------------------------------------------
```

Some tips if you should encounter trouble on a project:

1. Use the Serial Monitor

The clock will output quite some amount of information, so please have a look at it.

Here's what the output looks like:

```
ClockSketch v7 starting up...
Clock Type: Retro 7 Segment Clock v2 - Regular Edition
Configured RTC: DS3231
LED power limit: 500 mA
Total LED count: 99
LED digits: 4
Configured for Arduino
setup(): Lighting up some leds...
setup(): RTC.begin(), 2 second safety delay before
         doing any read/write actions!
setup(): RTC initialized
paletteSwitcher(): loaded EEPROM value 0
paletteSwitcher(): selected palette 0
paletteSwitcher() done
brightnessSwitcher(): loaded EEPROM value 1
brightnessSwitcher(): selected brightness index 1
brightnessSwitcher() done
colorModeSwitcher(): loaded EEPROM value 0
colorModeSwitcher(): selected colorMode 0
colorModeSwitcher() done
displayModeSwitcher(): loaded EEPROM value 0
displayModeSwitcher(): selected displayMode 0
displayModeSwitcher() done
-----------------------------------
System time is : 00:00:05
System date is : 2000-1-1 (Y/M/D)
RTC time is    : 00:00:05
RTC date is    : 2000-1-1 (Y/M/D)
-----------------------------------
setup() done------------------------------------------------------
```

That's running with a fresh rtc. The clock will output the current time/rtc time to the serial monitor every 20 seconds. You can check if the correct date/time have been stored quite easily by looking at the output. This one has autoDST enabled and no date/time set yet:

```
-----------------------------------
System time is : 00:00:20
System date is : 2000-1-1 (Y/M/D)
RTC time is    : 00:00:20
RTC date is    : 2000-1-1 (Y/M/D)
autoDST time is: 01:00:20
-----------------------------------
```

The displayed offset of +1 hour fits my autoDST settings (germany, middle european time).

This is what the output looks like after setting date/time:

```
----------------------------------
System time is : 18:38:40
System date is : 2021-7-8 (Y/M/D)
RTC time is    : 18:38:40
RTC date is    : 2021-7-8 (Y/M/D)
autoDST time is: 20:38:40
-----------------------------------
```

utc +2 hours, middle european summer time.

2. Checking the leds/assignments

If you're having trouble with displaying the digits or parts of the led strip not working, try uncommenting "#define CUSTOMHELPER" on top of the sketch:

```
// customHelper will start some kind of assistant when adapting this sketch to other led layouts, this
// tests all the steps neccessary to run it on almost any led strip configuration.
#define CUSTOMHELPER
```

After uploading the sketch the clock will run some test patterns and tell you about them in the serial monitor. The first test will simply light all up le ds one by one, so if there's something wrong it should be visible pretty easily.

The other tests are more with custom clocks/led strip layouts in mind, those will be covered in part II.

3. Buttons

Sometimes people inadvertently connect the push buttons the wrong way. They do come with 4 pins and no obvious sign/marking on what side to use them. On previous sketches this would simply lead to the following behaviour:

The clock starts up, immediately blanks the screen (because buttons A + B are pressed) and stays there. While this can be seen easily in the seri al monitor, the clock itself would look rather "dead". A quick look at the serial monitor tells you if there's a button pressed/shorted. Sketch v7 will lig ht up some leds anyways before entering setup - so it should be more obvious if there's something wrong with the buttons.

4. RTC

If you're having trouble setting date/time on a rtc or the clock doesn't display anything except setup/wps connection status or color previews when switching palettes/modes, check the rtc module. Time not showing up every 20 seconds in the serial monitor is also a strong indicator of somethin g going wrong with the communications to/from the rtc.

I recommend using the examples supplied with the "Rtc by Makuna" library, like "DS3231_Simple" to test basic functionality, not uploading the wh ole clock sketch each time (or "SetSerial", if using old releases and the DS3232RTC library).

One of the things happening frequently seems to be switching the a4/a5 pins on Arduinos.

5. Versions / Libraries / Board Configs

Please note that there's some problems using esp8266/v3 and FastLED in certain configurations. That's not a problem of this sketch and there's not much I can do about it. There's a lot of threads about this on github(fastled/esp core), here's a few examples:

https://github.com/esp8266/Arduino/issues/8054

https://github.com/FastLED/FastLED/issues/1264

I am using esp8266 core v2.7.1 and didn't encounter these issues. So if you're using v3 (check using the boards manager inside the Arduino IDE) a rollback to 2.x might solve issues with flickering leds, especially if it's only the first one.

Update: You might want to check out FastLED v3.5.0, according to the changelog things might be working a bit smoother in combination with esp8266 core v3, from the FastLED changelog:

> Greatly improved ESP32 and ESP8266 support