

# **ZÁVĚREČNÁ STUDIJNÍ PRÁCE**

## **dokumentace**

### **7 segment led display**

Jakub Knappe

**Obor:** 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  
se zaměřením na počítačové sítě a programování

**Třída:** IT4

**Školní rok:** 2020/2021

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě      31. 12. 2020

---

podpis autora práce

## **ANOTACE**

Cílem mého projektu, bylo vytvořit led displej složený ze sedmi segmentů, jehož hlavní funkcí by mělo být zobrazení aktuálního času. Čas je získáván z internetu a prezentován jak ve formě rozsvícených ledek, tak i ve formě zobrazení na HTML stránce. Součástí projektu by měly být také funkce jako je změna barvy, jas displeje. Tyto funkce lze najít na HTML stránce a jednoduše s nimi pracovat. Na stránce lze také najít hodnoty vlhkosti a teploty, které jsou získávány ze senzoru DHT11. Hodnoty jsou zapsány do jednoduché tabulky, součástí této tabulky jsou také tlačítka, které změni režim a zobrazí tak dané hodnoty na displej.

## OBSAH

<b>ÚVOD.....</b>	<b>5</b>
<b>1 VÝROBA A VÝVOJ.....</b>	<b>6</b>
<b>2 VYUŽITÉ TECHNOLOGIE .....</b>	<b>7</b>
<b>3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY.....</b>	<b>8</b>
3.1 NTP SERVER .....	8
3.2 PRÁCE S LED PÁSKEM .....	8
3.3 INDEXY .....	9
3.4 BITOVÉ ČÍSLO.....	10
3.5 ASYNCHRONNÍ SERVER .....	10
3.6 VYUŽITÍ KNIHOVNY ESPASYNCWIFIMANAGER.....	11
<b>4 HTML STRÁNKA .....</b>	<b>13</b>
4.1 VÝPIS ČASU NA HTML STRÁNKU .....	13
4.2 JAS .....	13
4.3 BARVY .....	14
<b>5 SENZOR DHT11.....</b>	<b>15</b>
5.1 REŽIMY DISPLEJE .....	15
<b>6 VÝSLEDKY ŘEŠENÍ, VÝSTUPY, UŽIVATELSKÝ MANUÁL.....</b>	<b>16</b>
<b>ZÁVĚR .....</b>	<b>17</b>
<b>SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ.....</b>	<b>19</b>
<b>SEZNAM PŘÍLOH.....</b>	<b>20</b>

## ÚVOD

Cílem této práce bylo vytvořit funkční sedmi segmentový displej, na kterém se bude zobrazovat aktuální čas. Čas je získáván z internetu, přesněji z NTP serveru (Network Time Protocol server).

Displej se celkově skládá z 30 ledek. Dvacet osm z nich slouží pro zobrazení čísel, zbylé dvě jako oddělovač číslic (hodin, minut). Výhodou mých používaných ledek je ta, že jsou adresovatelné. To znamená, že můžu určit, která dioda se má rozsvítit. K tomu abych dokázal ovládat jednotlivé ledky, bylo zapotřebí přidělit každé diodě pořadové číslo/index. Z těchto indexů pak bylo možné vytvořit tzv. bitové číslo, které nám určilo pořadí ledek, které se má rozsvítit.

Další výhodou je, že u nich lze měnit barva a jas. Jako výchozí barvu jsem si zvolil červenou. Tuto barvu ale můžeme jednoduše změnit na HTML stránce, kterou vytváří samotná vývojová deska. Stránka dále nabízí nastavení jasu, můžeme zde najít také výpis času přesný na vteřiny.

Na stránce se také nachází tabulka s názvem Senzor DHT11. Tato tabulka je pojmenovaná podle senzoru, který používám pro získávání hodnot teploty a vlhkosti. Hodnoty senzoru jsou vypisovány do tabulky. U každé hodnoty můžeme najít tlačítko SHOW. Toto tlačítko slouží k přepínání režimů hodin, tedy ke změně obsahu displeje. V mém projektu zatím využívám pouze tři režimů, a to jsou hodiny, teplota a vlhkost.

## 1 VÝROBA A VÝVOJ

Mezi první kroky mého projektu patřilo vybrání vhodných součástek. Jelikož jsem do té doby neměl tolik zkušeností se sestavováním integrovaných obvodů, součástky jsem před samotným nákupem pečlivě zkoumal.

Základ mého projektu tvořila vývojová deska ESP8266 WeMos Mini D1 obsahující wifi modul. Dále následoval nákup led pásku WS2812b a dalších potřebných součástek. Dobu čekání na součástky jsem využil ke zprovoznění frameworku Platformio ve Visual Studio Codu. Začal jsem si také hledat užitečné knihovny, který bych ve svém projektu mohl v budoucnu použít.

Ještě před propojením jednotlivých součástek jsem se rozhodl pracovat s vývojovou deskou. Začal jsem programovat jednoduché funkce, abych se s deskou trochu seznámil a ověřil si její funkčnost. Po propojení součástek nastala programovací část, která se většinou týkala prací s diodami, nebo tvorbou jednoduché webové aplikace. Ke konci projektu jsem se spíše soustředil na doladování kódu než na přidávání nových funkcí. K vytvoření schématu mého projektu jsem použil program KiCAD.

## 2 VYUŽITÉ TECHNOLOGIE

- Vývojová deska ESP8266 WeMos Mini D1
- Led pásek WS2812b
- Senzor teploty a vlhkosti DHT11
- Redukce USB Micro pro nepájivé pole
- 330  $\Omega$  resistor
- 220  $\Omega$  resistor
- Napájecí adaptér 5V/3A
- Visual Studio Code
- Platformio
- KiCAD
- Bootstrap 4

## 3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

### 3.1 NTP server

Ještě před propojením čipu s diodami jsem se rozhodl napsat kód pro získání času z internetu. Využil jsem k tomu knihovnu *arduino-libraries/NTPClient@^3.1.0*. Mezi první krok také patřilo připojení se k domovské Wifi síti. Bylo nutné zadat správnou SSID a heslo k síti. Aby kód vypisoval správný čas, bylo ještě potřeba zjistit v jakém GMT časovém pásmu se nacházíme. Pro Českou republiku platí pásmo GMT +1, v letním čase platí GMT +2. Pomocí příkazu *Serial.println(formattedTime)*; jsem pak do konzole vypsals současný čas. Sice bylo pěkné, že se mi do konzole vypisoval celý čas. Já však potřeboval rozdělit čas na jednotlivé hodiny a minuty. K tomu mi posloužily funkce *timeClient.getHours* a *timeClient.getMinutes*, ke kterým jsem si deklaroval proměnné *hours*, *minutes* do kterých jsem čas ukládal.

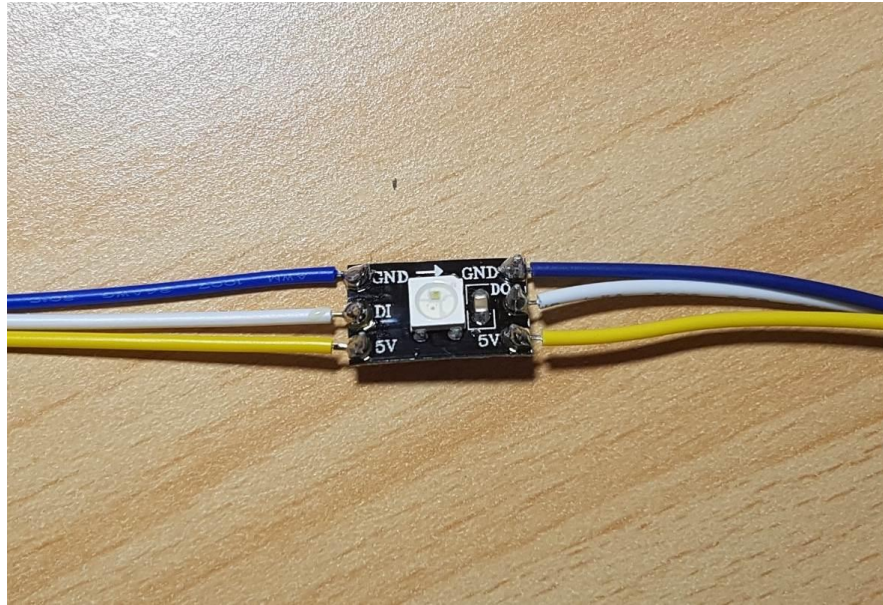
### 3.2 Práce s LED páskem

Ledky použité v tomto projektu jsem koupil ve formě pásku. Můj koupený pásek obsahoval celkem 30 diod. Dvacet osm z jich jsem použil k sestavení čtyř sedmi segmentů, zbylé dva jsem použil jako tečky, které oddělovaly první dva segmenty od druhých. Pásek jsem si nejdříve rozdělil na jednotlivé diody. Vstupní napětí pásku je 5 voltů. Pásek obsahuje 3 piny z každé strany, 5V, data pin a GND. Ledky jsem vzájemně propojoval pomocí propojovacích vodičů. Abych se v projektu lépe orientoval, určil jsem si pro každý typ pinu určitou barvu vodiče. Například napětí 5 voltů jsem měl propojené vodiči se žlutou barvou. Data pinům jsem dal bílou barvu a GND jsem dal modrou barvu.

Když už jsem měl jednotlivé ledky správně propojené, stačilo je jen připojit na správné piny k desce. Jelikož deska nedokázala napájet tak velké množství ledek, použil jsem napájení 5V/3A. Po-té už přišla na řadu programovací část. Nejdříve jsem si na internetu musel najít knihovnu podporující mé ledky. V mém případě se jednalo o knihovnu *fastled/FastLED@^3.3.3*.

Tento odkaz jsem vložil do platformio.ini a pak už jen stačilo knihovnu inicializovat v souboru main.cpp. Pro zkontrolování, zda všechny ledky správně fungují jsem použil jednoduché funkce.

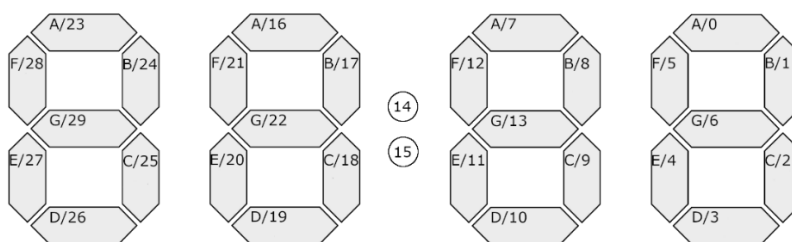




Obrázek č. 1 Ukázka ledky WS2812b

### 3.3 Indexy

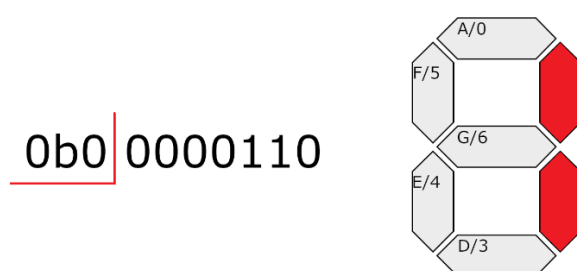
Abych vůbec dokázal čísla na displej zobrazit, musel jsem si jednotlivé ledky očíslovat, určit jim index. Jako pomůcku jsem si vytvořil soubor `indexes.png`, který můžete najít na mém githubu v repozitáři `zaverecny_projekt_poznamky`. Na tomto obrázku můžeme vidět, jak by měl displej vypadat, u každé ledky se nachází číslo/index. Čísla indexů jsem přiděloval podle postupného zapojení ledek. Tedy první dioda od desky je přidělena číslo 0 a poslední 29. Tyto čísla/indexy mi později sloužili k vytvoření tzv. bitových čísel.



Obrázek č. 2 Indexy

### 3.4 Bitové číslo

Jak už jsem naznačil, soubor `indexes.png` nyní využijeme pro tvorbu bitového čísla. Bitové číslo se celkově skládá z předpony „0b0“ a sedmi čísel v podobě jedniček a nul. Jednička v tomto případě znamená stav ON, tedy že dioda svítí. Nula naopak, že je dioda zhaslá. Nyní už jen stačilo si představit, jak by měly čísla vypadat a podle toho sestavit jednotlivá bitová čísla. K vytvoření těchto čísel jsem si opět pomohl vytvořením souboru `numbers.xlsx`. V tomto souboru jsem měl všechny čísla, znaky na jednom místě, později mi usnadňoval práci s vytvářením nových znaků např. °C.



Obrázek č. 3 Ukázka bitového čísla

V souboru `main.cpp` bylo potřeba vytvořit funkce *displaytime* a *updateclock*. Funkce *displaytime* obsahuje dva parametry, `index` a `numbers`. Parametr `index` určuje pořadí první ledky, od které se má začít počítat. Druhý parametr, parametr `numbers`, pak obsahuje samotné číslo, které se má vypsát. Ve funkci *updateclock* jsou proměnné s aktuálním časem, je zde vyvolaná funkce *displayTime*. Aby vše fungovalo jak má, bylo nutné funkci *updateclock* vyvolat ve *void loop*.

### 3.5 Asynchronní server

Jako další technologii, kterou jsem ve svém projektu využil je asynchronní web server. Do souboru `platformio.ini` jsem přidal knihovny *ESPAsync WebServer* a *ESPAsyncTCP*, podporující asynchronní server. Tyto knihovny jsem musel inicializovat v souboru `main.cpp`. Poté už jen stačilo, přidal do toho souboru port, případně si vytvořit proměnné pro SSID a heslo k síti na kterou chceme být připojeni. Do svého kódu jsem dále přidal výpis IP adresy, na které poslouchá má vývojová deska. Abych se ujistil, že mi kód funguje správně, vytvořil

jsem si ve funkci *void setup* HTTP request, který mi vypsal text „test“ na IP adrese vypsané v konzoli. K IP adrese bylo potřeba ještě přidat „/html“.

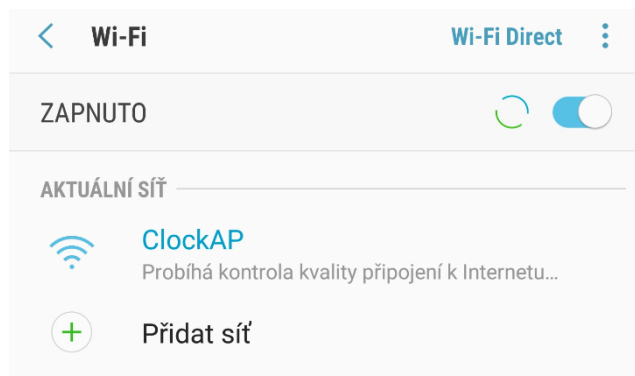
```
server.on("/html", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(200, "text/plain", "test");
});
```

Tento příklad pouze vypsal jednoduchý text. Abych mohl vypsat html stránku, musel jsem ji mít někde uloženou. K uložení stránky jsem použil souborový systém SPIFFS, který ukládá informace do interní paměti desky. Opět bylo potřeba přidat knihovnu, podporující tento souborový systém. V mém projektu bylo nutné ještě vytvořit adresář data obsahující soubor index.html. Pomocí dalšího HTTP requestu jsem poslal HTML soubor na IP adresu.

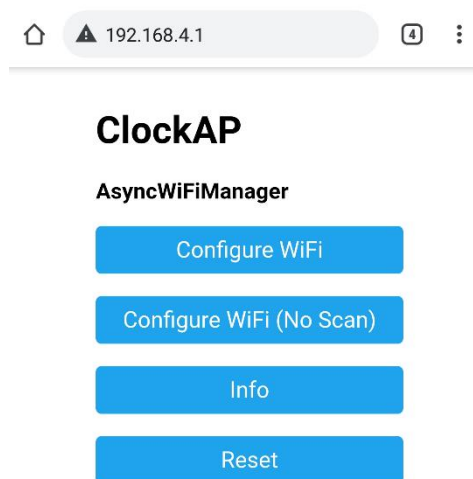
```
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/index.html", String());
});
```

### 3.6 Využití knihovny ESPAsyncWiFiManager

Pomocí této knihovny jsem se dokázal připojit na libovolnou wifi síť bez jakékoliv změny v souboru main.cpp. Opět bylo zapotřebí přidání knihovny do příslušných souborů. Knihovna obsahuje mnoho typů wifi manageru. Já si zvolil typ AutoConnect. Tento typ funguje tak, že deska vytvoří přístupový bod (access point) nazvaný ClockAP. Po připojení na tento bod se nám na adrese 192.168.4.1 vytvoří konfigurační portál. Portál zobrazí všechny dostupné WI-FI sítě s možností se na kteroukoliv připojit. V případě, že jsem se už dříve na jednu z WI-FI sítí připojili, přístupový bod ClockAP se nám vůbec neobjeví a deska se rovnou připojí k síti.



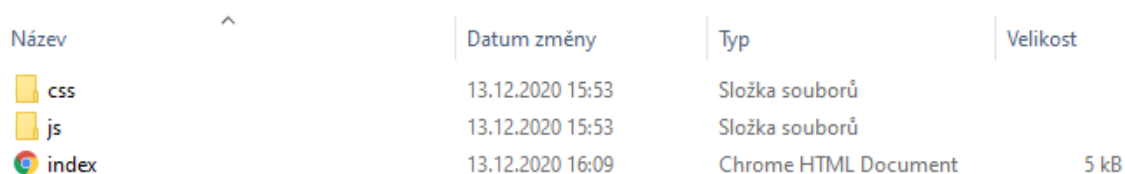
Obrázek č. 4 AsyncWifiManager



Obrázek č. 5 Konfigurační portál

## 4 HTML STRÁNKA

Hlavní funkcí mé HTML stránky bylo ovládání displeje. Přesněji ovládání jasu, barvy a změny režimů. Na stránce jsem využil kaskádové styly (CSS), jazyk JavaScript společně s javascriptovou knihovnou jQuery a strukturou JSON. Díky souborového systému SPIFFS je stránka přímou součástí projektu a je uložena v adresáři data pod názvem index.html. Na stránce dále využívám framework Bootstrap 4, který mi usnadnil práci především v oblasti responzivity stránky.



Název	Datum změny	Typ	Velikost
css	13.12.2020 15:53	Složka souborů	
js	13.12.2020 15:53	Složka souborů	
index	13.12.2020 16:09	Chrome HTML Document	5 kB

Obrázek č. 6 Soubory adresáře data

### 4.1 Výpis času na HTML stránku

Když už jsem si dokázal vypsat aktuální čas do konzole, tak nebyl tak moc velký problém vypsat jej i na HTML stránku. Pro výpis na stránku jsem si v souboru main.cpp vytvořil funkci *getTime*. Funkce uložila současný čas do proměnné datového typu String a následně jej vrátila. K tomu, aby byl čas poslaný na stránku jsem využil HTTP request GET. Po-té jsem si na internetu našel aktualizací script, který by na stránce aktualizoval můj čas. Vložil jsem jej do souboru main.js a nastavil jsem aktualizaci na každou sekundu.

### 4.2 Jas

Jako první funkce, která přímo pracovala s mými diodami byl jas. Na webu w3schools.com jsem si našel jednoduchý příklad použití rozsahového posuvníku. S drobnými úpravami jsem tento posuvník použil na mé stránce. U tohoto rozsahového posuvníku šel měnit minimální, maximální rozsah. Jako minimální hodnotu jsem si zvolil 10 a maximální 255. V souboru main.js jsem si vytvořil funkci *setBrightness*. Tato funkce se aktivovala po stisknutí tlačítka

těsně pod rozsahovou lištou. Obsahem této funkce bylo přečtení hodnoty z posuvníku, uložení ní do proměnné, vypsání hodnoty do konzole stránky a poslání na vývojovou desku pomocí HTTP requestu POST. Data na desku ESP jsou poslána ve formátu JSON. Dále jsem vytvořil již zmíněný request POST společně s proměnnou jas. Aby se změna jasu vůbec provedla, bylo ještě potřeba vyvolat funkci *FastLED.setBrightness*.



Obrázek č. 7 Ukázka posuvníku

### 4.3 Barvy

Další funkce přidaná do mého projektu, je funkce pro nastavení barvy. K tomu jsem využil knihovnu jQuery MiniColors. Tato knihovna nabízí spoustu palet pro vybírání barev. Na stránkách knihovny se nacházel odkaz ke stažení. Obsahem tohoto odkazu byly soubory .js, .css a .png obrázek. Tyto soubory jsem vložil do svého projektu do adresáře data. V index.html pak stačilo přidat příklad palety nacházející se na stránce. V souboru main.js jsem ještě musel určit typ palety. V mém případě se jedná o typ inline, který mi vrací barvu v podobě RGB. Po-té už následoval velice podobný postup, jako v případě jasu. Akorát místo hodnoty jasu jsem vracel hodnoty RGB.



Obrázek č. 8 Paleta pro výběr barev

## 5 SENZOR DHT11

Ve svém projektu jsem dále využil teplotní senzor DHT11. Tento senzor získává hodnoty teploty a vlhkosti v místnosti. DHT11 má celkově 4 vývody. První z nich je vstupní napětí, další je datový pin, třetí pin je NC (no connected) a poslední je GND (zem). Z doporučení jsem mezi vstupní napětí a datový pin zapojil 220  $\Omega$  rezistor. Ze začátku jsem měl se senzorem trochu problém. Deska nedokázala správně přečíst hodnoty senzoru. Problém byl způsoben díky špatné kombinaci pinu a knihovny. Po vyřešení problému jsem si hodnoty uložil do proměnných a následně poslal na stránku. Na stránce jsem vytvořil tabulku s názvem DHT11. Tato tabulka obsahovala hodnoty teploty a vlhkosti v současném čase díky aktualizacím scriptů umístěných v souboru main.js. Součástí tabulky byly také tlačítka Show, které měnily režim displeje.

DHT11 Senzor		
temperature	23 °C	Show
humidity	34 %	Show

Obrázek č. 9 Tabulka senzoru DHT11

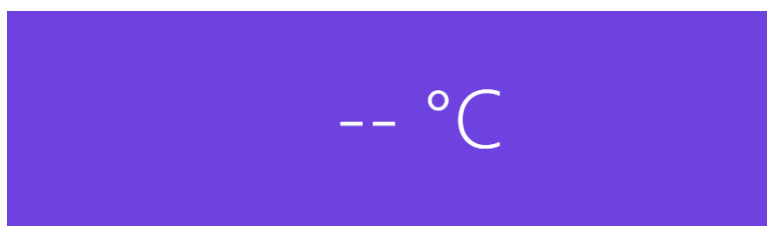
### 5.1 Režimy displeje

S přidáním senzoru do projektu mě napadlo vypisovat i jiné hodnoty na displej než jen hodnoty času. K tomu abych dokázal měnit hodnoty na displeji, jsem si vytvořil tzv. režimy. Režimy lze měnit pomocí tlačítek nacházejících se na HTML stránce. Tyto tlačítka vrací hodnotu, podle toho, ke kterému režimu patří. Výchozí hodnotou režimu je 0. Displej v tomto režimu zobrazuje aktuální čas. Zbylé dva režimy zobrazují hodnoty získané ze senzoru.

## 6 VÝSLEDKY ŘEŠENÍ, VÝSTUPY, UŽIVATELSKÝ MANUÁL

Displej v současné době funguje bez chyby. Barvy i jas fungují dobře. Jedinou věcí, kterou jsem si všiml je, že diody neumí správně zobrazit hnědou barvu. Já většinou používám jiné barvy, takže mi to tolik nevadí. Pro displej jsem stále nestihl vymyslet a vytvořit kryt, který by displeji (diodám) dal určitý tvar. Asi je to jedna z nejdůležitějších věcí, kterou bych měl v budoucnu vyřešit. Sice diody jsou vzájemně propojeny vodiči, ale v této podobě se s nimi špatně manipuluje.

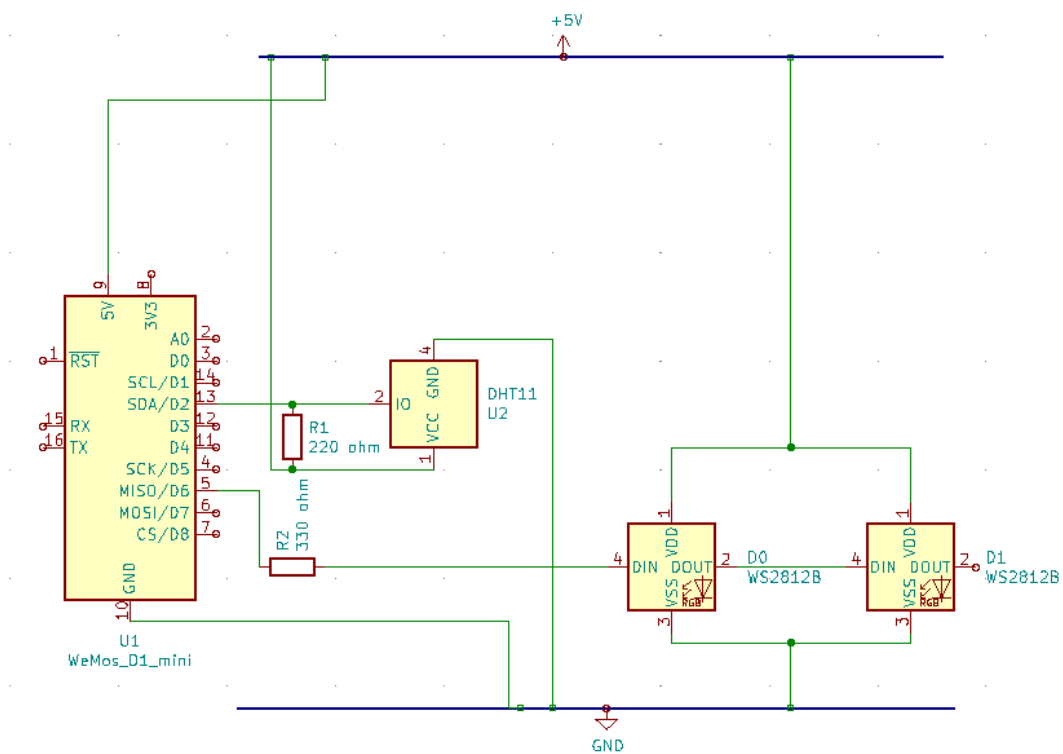
Ke konci projektu jsem zaznamenal, že senzor DHT11 někdy nedokázal poslat informace na vývojovou desku. Sice se jednalo, pouze o pár vteřin výpadku, ale mě to vypisovalo nesmyslná čísla na HTML stránce. Abych zabránil výpisu nesmyslných čísel vytvořil jsem v souboru main.js funkce, které zjišťují, zda senzor hodnoty poslal, případně místo nich vypíše „-“.



Obrázek č. 10 Ukázka výpisu pomlček

Webová aplikace funguje plynule a změny se na displeji projeví okamžitě. Snažil jsem se spíš soustředit na funkčnost stránky než na její vzhled. Hodnoty ze stránky jsou posílány na desku ESP ve formátu JSON za použití HTTP requestů GET a POST. V záhlaví stránky se mění hodnoty, podle toho, v jaké režimu se displej nachází. Na stránce se také mění barvy stylu. Barva se mění tam, kde je třída s názvem *change*. Tato třída mění barvu podle barvy diod.





Obrázek č. 11 Schéma projektu

## **ZÁVĚR**

Cílem projektu bylo vytvořit funkční led displej, který by zobrazoval čas. V současné době je tento displej plně funkční a dokáže zobrazit i jiné hodnoty než jen čas. Navíc se dá ovládat pomocí jednoduché webové aplikace.

Existuje spousta věcí, které se dají do projektu ještě přidat. Momentálně mě napadá přidání nového barevného režimu, který by plynule měnil barvy diod. Velice zajímavou věcí, kterou bych chtěl v budoucnu do projektu přidat je také OTA update. Jediná věc, která mě celkem mrzí je, že se mi nepovedlo vytvoření krytu pro diody. Momentálně jsou diody propojeny s vývojovou deskou skrz nepájivé pole a manipulace s nimi je celkem obtížná.

GitHub: [https://github.com/01knappe/zaverecny\\_projekt](https://github.com/01knappe/zaverecny_projekt)

## SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

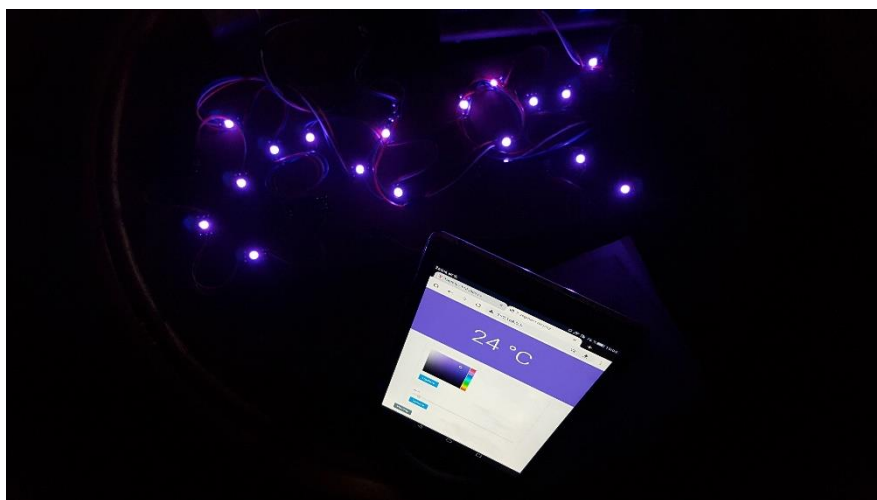
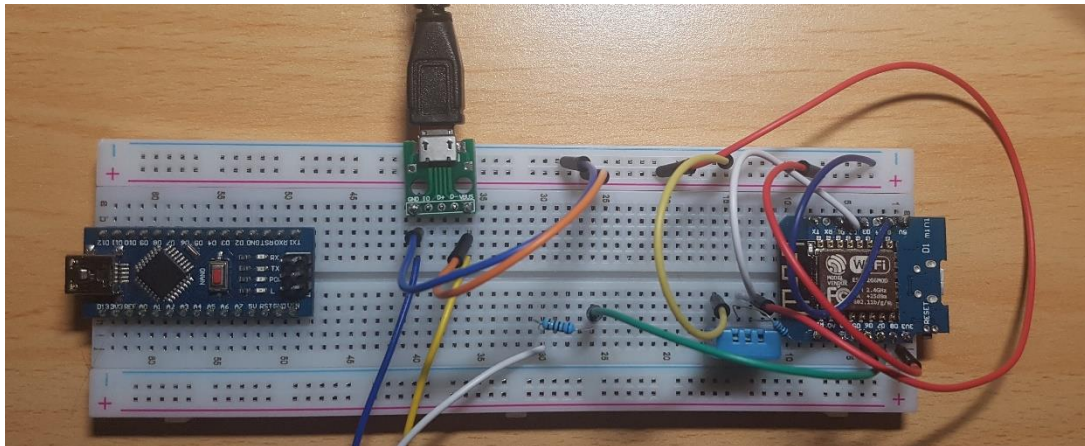
- [0] 3D Print a Sleek 7-Segment LED Clock [online]. [cit. 2020-12-31]. Dostupné z: <https://all3dp.com/weekend-project-3d-print-a-sleek-7-segment-led-clock/>
- [1] 7-segment-clock [online]. [cit. 2020-12-31]. Dostupné z: <https://github.com/leoclee/7-segment-clock/>
- [2] 7-Segment-Digital-Clock [online]. [cit. 2020-12-31]. Dostupné z: <https://github.com/leonvandenbeukel/7-Segment-Digital-Clock/>
- [3] 7-Segment-Digital-Clock-V2 [online]. [cit. 2020-12-31]. Dostupné z: <https://github.com/leonvandenbeukel/7-Segment-Digital-Clock-V2/>
- [4] esp8266/Arduino [online]. [cit. 2020-12-31]. Dostupné z: <https://github.com/esp8266/Arduino/tree/master/libraries/esp8266/examples/>
- [5] WS2812B Addressable RGB LEDs [online]. [cit. 2020-12-31]. Dostupné z: <https://www.electronicshub.org/ws2812b-addressable-leds/>
- [6] ESP8266 Web Server using SPIFFS (SPI Flash File System) – NodeMCU [online]. [cit. 2020-12-31]. Dostupné z: <https://randomnerdtutorials.com/esp8266-web-server-spiffs-nodemcu/>
- [7] DHT11/DHT22 Sensor [online]. [cit. 2020-12-31]. Dostupné z: <https://randomnerdtutorials.com/complete-guide-for-dht11dht22-humidity-and-temperature-sensor-with-arduino/><https://randomnerdtutorials.com/esp8266-web-server-spiffs-nodemcu/>
- [8] Range Sliders [online]. [cit. 2020-12-31]. Dostupné z: [https://www.w3schools.com/howto/howto\\_js\\_rangeslider.asp](https://www.w3schools.com/howto/howto_js_rangeslider.asp)
- [9] Bootstrap 4 Tutorial [online]. [cit. 2020-12-31]. Dostupné z: <https://www.w3schools.com/bootstrap4/default.asp>
- [10] jQuery MiniColors [online]. [cit. 2020-12-31]. Dostupné z: <https://labs.abeautifulsite.net/jquery-minicolors/>
- [11] ESPAsyncWebServer [online]. [cit. 2020-12-31]. Dostupné z: <https://github.com/me-no-dev/ESPAsyncWebServer>
- [12] ESPAsyncWiFiManager [online]. [cit. 2020-12-31]. Dostupné z: <https://github.com/alanswx/ESPAsyncWiFiManager>

## **SEZNAM PŘÍLOH**

č. 1 Fotodokumentace

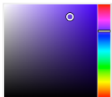
č. 2 Seznam obrázků

## Příloha č. 1: Fotodokumentace





15:51:57

  
[Update](#)

Brightness  
  
[Update](#)

[Clock](#)

---

**DHT11 Sensor**

temperature	24 °C	<a href="#">Show</a>
humidity	34 %	<a href="#">Show</a>

## **Příloha č. 2: Seznam obrázků**

Obrázek č. 1 Ukázka diody ws2312b

Obrázek č. 1 Ukázka ledky WS2812b

Obrázek č. 2 Indexy

Obrázek č. 3 Ukázka bitového čísla

Obrázek č. 4 AsyncWifiManager

Obrázek č. 5 Konfigurační portál

Obrázek č. 6 Soubory adresáře data

Obrázek č. 7 Ukázka posuvníku

Obrázek č. 8 Paleta pro výběr barev

Obrázek č. 9 Tabulka senzoru DHT11

Obrázek č. 10 Ukázka výpisu pomlček

Obrázek č. 11 Schéma projektu