SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE
dokumentace

Hodiny MAX7219

Daniel Peterek

Obor:	18-20-M/01 INFORMAČNÍ TECHNOLOGIE se zaměřením na počítačové sítě a programování
Třída:	IT4
Školní rok:	2020/2021

Poděkování

Děkuji panu učiteli Ing. Petru Grussmanovi za pomoc s výběrem projektu, následné konzultace a rady, panu učiteli Mgr. Marcelu Godovskému za pomoc se součástkami a pájení a panu učiteli Mgr. Markovi Lučnému za pomoc s dokumentací.

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. 2017

podpis autora práce

Anotace

Hodiny MAX7219 je zařízení, které vypisuje aktuální čas, datum a text na červenou LED matici o velikosti 32x8 LEDek. Matice je řízena mikroprocesorem WeMos s čipem ESP8266, který umožňuje připojení na WiFi. Čip získává aktuální čas a datum z výchozího serveru. Softwarový program je napsán v jazyce Arduino, webová stránka v HTML s trochou programování v JavaScriptu. Hodiny jsou rozděleny do dvou módů – čas a text. Módy hodin mohou být měněny na webovém serveru.

Klíčová slova

Hodiny, matice, displej, mikroprocesor, čas, datum, text

OBSAH

ÚVOD

Původním projektem měl být zvukový MIDI kontrolér. Tento projekt byl příliš snadný, jelikož využíval jiné další programy a programování bylo na pár řádků a opakovalo se. K současnému projektu jsem dospěl po konzultaci s panem učitelem, který mi doporučil projekt s ESP čipem, takže jsem se rozhodl zvolit projekt s hodinami.

Jelikož se jedná o můj první větší projekt, nekladl jsem si velké cíle. Cílem mého projektu bylo vytvořit hodiny, které budou získávat čas z internetových serverů. Hodiny se následně vypíšou na LED matici a budou obnovovat čas každou sekundu.

Jako prvotním cílem jsem usiloval o vypsání samotných hodin, následně přišly nápady na vylepšení a rozšíření těchto hodin. Snažil jsem se o vypsání aktuálního data získaného ze serveru a také textu, který je zadán na webové stránce.

V této dokumentaci se budu věnovat postupu výroby hardwaru, dále vybraným součástkám a způsobu zapojení, použitým technologiím, principu fungování knihoven a následnému vyobrazení na LED matici a způsobu fungování na straně serveru. Zhodnotím, zda jsem splnil své cíle a zmíním i případné vylepšení do budoucna.

1 VÝROBA HARDWARU

Prvním důležitým krokem projektu bylo zajištění určitých potřebných součástek. Nejdůležitější součástkou byl určitě mikroprocesor. Zvolil jsem mikroprocesor s integrovaným WiFi čipem firmy WeMos, konkrétně WeMos D1 Mini, který měl k dispozici pan učitel. Další potřebnou součástkou byl displej. Vybral jsem soustavu čtyř modulů 8x8 LED matic MAX7219, zapojených do kaskády,

Po výběru součástek jsem musel otestovat funkčnost zařízení. Matici jsem zapojil s využitím drátků do nepájivého pole. Zde jsem se setkal s prvním problémem. Matici jsem zapojil do pinů 3V3, GND, D5, D7 a D8. Po připojení USB kabelu se celá matice rozsvítila a svítila velmi oslnivě. To způsobilo přehřívání mikroprocesoru po pár sekundách.

Po této chybě jsem si všiml pinu 5V na desce mikroprocesoru. Přepojil jsem drátek z pinu 3V3 na 5V. Po připojení USB se mikroprocesor stále přehříval. Mikroprocesor jsem trochu potrápil, ale vypojil jsem ho pokaždé včas, takže nedošlo k trvalému zničení mikroprocesoru, ale chybělo k tomu málo.

Jelikož LED matice potřebuje napájecí napětí 5 V, bylo potřeba podpořit napájení z mikroprocesoru, které posílá pouze 3,3 V.

Pro tento účel jsem vybral univerzální spínaný síťový napájecí adaptér s výstupním napětím 3 až 12 V a výstupním proudem 2,25 A. Výstupní napětí lze měnit otočným přepínačem. Adaptér je nastaven pro výstup 5 V.

Poté následovala výroba základní desky. Zvolil jsem možnost připájet součástky na univerzální pájivé pole. Drátky byly náchylné k roztržení a při manipulaci se mohly snadno vypojit, takže součástky jsou napájeny na desku o velikosti 6,5x5 centimetrů. Na desce jsou napojeny konektory, tím pádem jsou části (mikroprocesor a matice) snadno odnímatelné a mohou být nahrazeny.

Po zapojení adaptéru a USB kabelu se matice rozsvítila, tentokrát už méně jasně než při prvním pokusu. Mikroprocesor se ani po pár minutách svitu nepřehříval, takže napájecí adaptér fungoval perfektně.

Po těchto krocích bylo načase začít s programováním. K otestování funkčnosti displeje jsem zvolil program ArduinoIDE. Už hned zezačátku mi program nevyhovoval, byl nepřehledný, špatně se zde přidávaly knihovny, které nebyly součástí programu. Tak jsem přešel k programu Visual Studio od společnosti Microsoft a zde využil rozšíření PlatformIO, které splňovalo mé požadavky.

2 VYUŽITÉ TECHNOLOGIE

2.1 Využitý hardware

2.1.1 Seznam součástek

- Vývojová deska WeMos D1 mini
- LED matice MAX7219
- Napájecí konektor DC-005 5,5/2,1 mm

2.1.2 WeMos D1 Mini

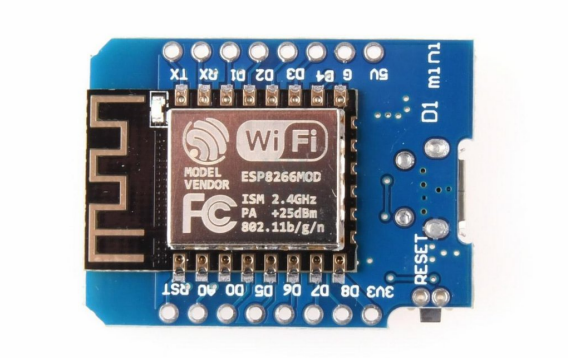
Základ projektu tvoří vývojová deska WeMos D1 mini (obrázek č. 1), která je založená na oblíbeném Wi-Fi modulu ESP8266. Je kompatibilní s Arduino platformou.

Tento mikroprocesor obsahuje 16 pinů, z nichž devět pinů je digitálních GPIO a jeden analogový pin. Všechny digitální piny pracují s napětím 3,3 V. Další piny slouží jako napájení.

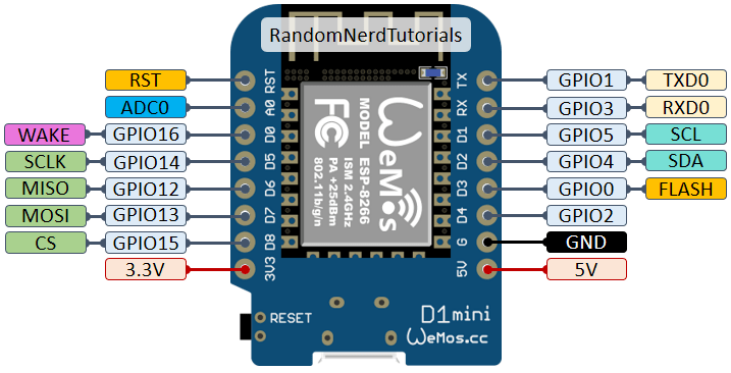
Pro tento projekt jsem využil digitální piny D2, D5 a D7.

Parametry:

- Napájení 3,3 V
- Flash paměť 4 MB
- Frekvence 80 / 160 MHz



Obrázek č. 1 WeMos D1 mini



Obrázek č. 2 Pinout WeMos D1 mini

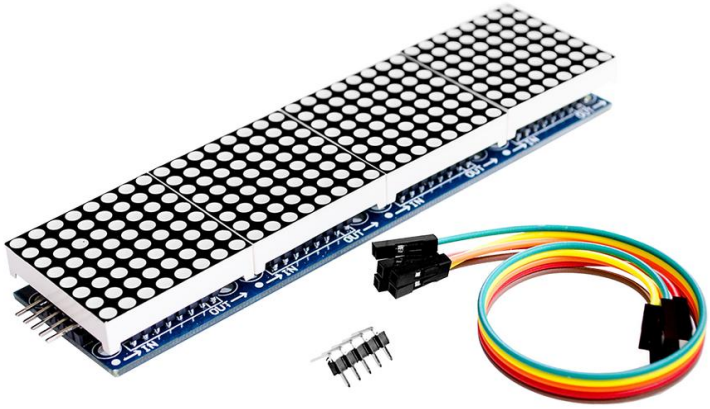
2.1.3 LED matice MAX7219

Displej projektu tvoří čtyři moduly LED matic MAX7219 (obrázek č. 2) o velikosti 8x8 LEDek zapojené do kaskády, dohromady 32x8 LEDek. Díky zapojení do kaskády je možné displej snadno rozšiřovat.

Displej svítí červenými LED diody. Průměr jedné LED diody je 3 mm.

Matice pracuje s napájecím napětím 5 V, proudový odběr je přímo úměrný počtu modulů.

Pro připojení potřebujeme jen pět vodičů, dva pro napájení (GND a VCC) a tři pro řízení (DIN, CS a CLK).



Obrázek č. 3 Displej MAX7219

2.2 Využitý software

2.2.1 Visual Studio Code

Programovací prostředí od firmy Microsoft. Podporuje programování v mnoha programovacích jazycích. Můžeme zde vytvářet různé programy. Konkrétně jsem zde využil rozšíření PlatformIO. PlatformIO IDE je integrované vývojové prostředí, které je postaveno na textovém editoru Atom.

2.3 Využité programovací jazyky

2.3.1 Jazyk Arduino

Jazyk Arduino má podobu frameworku v jazyce C++. Obsahuje dvě základní funkce. Funkce setup(), která slouží k inicializaci proměnných a nastavení potřebných hodnot a funkce loop(), která je hlavní, stále se opakující funkcí programu.

2.3.2 HTML 5

HTML 5 je značkovací jazyk, kterým vytváříme obsah a strukturu stránky. Při tvorbě uživatelského rozhraní jsem použil framework Bootstrap 3, který mi usnadnil práci.

2.3.3 JavaScript

JavaScript je programovací skriptovací jazyk, který se používá v internetových stránkách k modifikaci obsahu. Zapisuje se přímo do HTML kódu. JavaScript je závislý na prohlížeči.

3 ZPŮSOBY ŘEŠENÍ

3.1 Řešení softwarové části displeje

3.1.1 Definice knihoven

Setkal jsem se s problémem definicí knihoven. PlatformIO umožňuje definovat knihovny v souboru platformio.ini. Knihovny lze nainstalovat příkazem nebo lze ručně vložit odkaz ke knihovně. Já jsem knihovny nainstaloval pomocí příkazu z oficiálních stránek, ale žádná z knihoven nefungovala správně a můj kód vůbec nefungoval. Tento problém jsem vyřešil vložením odkazů knihoven z githubu.

3.1.2 Knihovna displeje MAX72xx

Během testování výpisu textu na displeji jsem se setkal s dalším problémem. Při výběru knihovny jsem zvolil ne příliš aktuální knihovnu. Po nahrání kódu do mikroprocesoru se na matici vypsal testovací text, ale místo jednoho souvislého textu psaný vodorovně se vypsali čtyři texty vypsané svisle. To zapříčinila neaktualizovaná knihovna, která měla podporu pouze pro jeden modul.

Tento problém jsem vyřešil zvolením kombinací knihoven MD_MAX72XX a MD_Parola od uživatele MajicDesigns. Tyto knihovny podporují velkou řadu displejů a implementují spoustu funkcí, mezi které patří vycentrování textu vpravo, uprostřed nebo vlevo v poli zobrazení, posouvání textu, různé efekty, ovládání parametrů zobrazení a rychlosti animace, při větším počtu modulů lze definovat více virtuálních displejů - zón (displej lze rozdělit na dvě části, každá část může zobrazovat jiný text), nahrazení jednotlivých znaků, definovat fonty a podpora dvojité výšky (dvě matice na sobě).

```
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
#define CS_PIN 4
#define DATA_PIN 13
#define CLK_PIN 14
```

První řádek označuje definici typu matice, FC16_HW je označení pro kaskádový typ matice. MAX_DEVICES označuje počet modulů (4 moduly 8x8). Ostatní řádky definují piny.

```
MD_Parola DotMatrix = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN,
MAX_DEVICES);
```

Spojení SPI. SPI je synchronní datový protokol používaný mikrokontroléry pro rychlou komunikaci s jedním nebo více periferními zařízeními na krátké vzdálenosti. Zde mám určený název DotMatrix, který nese informaci o typu hardwaru, počtu modulů a využitých pinů.

```
uint8_t scrollSpeed = 50;
textEffect_t scrollEffect = PA_SCROLL_LEFT;
textPosition_t scrollAlign = PA_LEFT;
uint16_t scrollPause = 3000;
```

Parametry displeje. První řádek určuje rychlost displeje, druhý řádek určuje efekt displeje, zde je určené posouvání doleva, třetí řádek definuje pozici textu – vlevo a poslední řádek definuje, na jak dlouho se text pozastaví po provedení animace v milisekundách.

```
DotMatrix.begin();
DotMatrix.setIntensity(0);
```

Vyvolání, zahájení displeje a nastavení jasu displeje, dá se také vložit mezi parametry.

```
DotMatrix.displayText("ESP", scrollAlign, scrollSpeed, scrollPause,
                      scrollEffect,scrollEffect);
```

Výpis textu na displeji, využívá naše definované parametry.

3.1.3 Získání data a času

K získání data a času jsem využil kombinací knihoven NTPClient a WifiUdp. NTPClient získává informace, tedy čas a datum z určeného NTP serveru.

```
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);
```

Tyhle dva řádky definují klienta NTP, aby získával datum a čas ze serveru NTP, je nastaven výchozí server.

```
  timeClient.begin();
  timeClient.setTimeOffset(3600);
```

Vyvolání NTP klientu a následná metoda pro úpravu času pro naše časové pásmo v milisekundách. GMT +1 = 3600

```
while (!timeClient.update()) {
    timeClient.forceUpdate();
}
```

Díky těmto řádkům dostaneme platný datum a čas.

```
formattedDate = timeClient.getFormattedDate();
Serial.println(formattedDate);
```

Převedení data a času na čitelný formát. Datum a čas je vypsán v konzoli v následujícím formátu: `2020-12-31T16:00:38Z`

```
void showDate(){
    year = formattedDate.substring(0, 4);
    month = formattedDate.substring(5, 7);
    date = formattedDate.substring(8, 10);
}
void showTime(){
    hour = formattedDate.substring(11, 13);
    minute = formattedDate.substring(14, 16);
    second = formattedDate.substring(17, 19);
}
```

Převedení formátu data a času pomocí metody substring, která extrahuje znaky z řetězce mezi dvěma zadanými indexy a vrátí nový podřetězec. V našem případě se z řetězce `2020-12-31T16:00:38Z` vrátí podřetězec 2020 12 31 a 16 00 38.

```
dateStamp =  date + "." + month + year;
dateStamp.toCharArray(dateBuffer, dateStamp.length()+1);
```

Převod dne, měsíce a roku do jednoho řetězce.

```
enum {TIME, DATE, TEXT};
boolean displayMode = TIME;
```

Módy hodin jsou rozděleny na čas, datum a text. Výchozím módem je čas.

```
if (second.toInt() == 0) {
      displayMode = DATE;
      DotMatrix.displayClear();
      DotMatrix.displayText(dateBuffer, scrollAlign, scrollSpeed,
            scrollPause, scrollEffect, scrollEffect);
}
else if (second.toInt() % 2) {
      timeStamp = hour + ":" + minute;
}
else {
      timeStamp = hour + " " + minute;
}
```

První podmínka určuje, že každou minutu (pokud počet sekund se bude rovnat nule) se mód přepne na datum, který se následně vypíše na displeji s určenými parametry. Další podmínky znázorňují sekundy, pokud je sekunda lichá, vypíše dvojtečku. Pokud sekunda není lichá, nevypíše dvojtečku.

```
DotMatrix.setTextAlignment(PA_CENTER);
```
Text je zarovnán uprostřed matice. Čas bude uprostřed.

```
if(mode == 0){
      displayMode = TIME;
      DotMatrix.print(timeStamp);
}
```
Pokud je mód nastaven na nulu (mód času), vypíše se čas.

3.2 Řešení webové aplikace

3.2.1 HTML stránka

Webovou stránku jsem vytvořil v HTML jazyce pomocí rozšíření Bootstrap 3, které umožňuje vytvořit jednoduchou, srozumitelnou šablonu stránky v HTML verzi 5. Následně jsem vytvořil nadpis, jednoduchý formulář a dvě tlačítka. Jedno pro odeslání formuláře a druhé pro přepnutí hodin. Tlačítka využívají funkcí JavaScriptu.

```
<h1 class="text-center" style="font
family:verdana;">ESP8266 MAX7219 hodiny</h1>
        <div class="text-center">
            <p>Napiš text pro zobrazení na matici</p>
            <form id="input">
                <input id="text" type="text" name="input1">
          <button type="button" class="btn btn-success"
onclick="showText()">Ukaž</button>
            </form>
        </div>
        <div class="text-center">
            <button type="button" class="btn btn-info btn-lg"
onclick="showClock()">Hodiny</button>
        </div>
```
Obrázek č. 4 Obrázek kódu html

3.2.2 Princip fungování webové stránky

K vytvoření webového serveru jsem využil kombinací knihoven ESPAsyncWebServer a ESPAsyncTCP. Knihovny poskytují snadný způsob, jak vytvořit asynchronní webový server, což znamená, že dokáže zpracovat více než jedno připojení současně.

```
AsyncWebServer server(80);
```
Vytvoří objekt AsyncWebServer s názvem server na výchozím portu HTTP 80.

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

Knihovna ESPAsyncWebServer nám umožňuje konfigurovat trasy, kde server dostává požadavky HTTP a spouští funkce, které jsou přijaté na této trase. K tomu použije metodu on() na objektu serveru.

Když server obdrží požadavek na kořenovou adresu „/“ URL, odešle klientovi soubor index.html.

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

V souboru HTML jsme odkazovali na soubor JS, klient požádá o soubor JS a následně se soubor JS odešle klientovi.

```
server.on("/clock", HTTP_POST, [](AsyncWebServerRequest *request){
    mode = 0;
    displayMode = TIME;
    request->send(200, "text/json", "{\"result\":\"ok\"}");
});
```

Server odesílá požadavek HTTP, následně změní mód na čas (na nulu).

```
server.on("/text", HTTP_POST, [](AsyncWebServerRequest *request) {
    text = request->arg("text").c_str();
    text = utf8ascii(text);
    Serial.println(text);
    mode = 1;
    displayMode = TEXT;
    request->send_P(200, "text/json", "{\"result\":\"ok\"}");
});
```

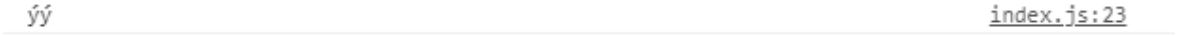
Server odesílá požadavek HTTP, vypíše zadaný text do konzole a změní mód na text (na jedničku).

3.3 Převod UTF-8 na ASCII

Nastal zde problém při vypsání znaků s háčky a čárkami. Odeslaný formulář vypsal text do konzole správně, tedy s háčky a čárkami. Arduino podporuje UTF-8, což mu umožňuje

zpracovat tisíce znaků. Matice bohužel háčky a čárky nevypsala, vypisuje pouze znaky z tabulky ASCII od pozice 32 do pozice 127.

Obrázek č. 5 Text ve formuláři



Obrázek č. 6 Text v konzoli

Obrázek č. 7 Zobrazení textu „ýý“ na displeji

Tento problém jsem vyřešil pomocí UTF-8 dekodéru, který převedl UTF-8 znaky do rozšířeného ASCII. Bohužel, převodník převádí znaky do rozšířeného ASCII kódování Windows-1252. Win-1252 je znaková sada používaná pro západoevropské jazyky. To znamená, že nevypisuje znaky s háčky a například kroužkované u.

Windows-1252

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
8_	U+20AC € 128	129	U+201A ‚ 130	U+0192 ƒ 131	U+201E „ 132	U+2026 … 133	U+2020 † 134	U+2021 ‡ 135	U+02C6 ˆ 136	U+2030 ‰ 137	U+0160 Š 138	U+2039 ‹ 139	U+0152 Œ 140	141	U+017D Ž 142	143
9_	144	U+2018 ‘ 145	U+2019 ’ 146	U+201C “ 147	U+201D ” 148	U+2022 • 149	U+2013 – 150	U+2014 — 151	U+02DC ˜ 152	U+2122 ™ 153	U+0161 š 154	U+203A › 155	U+0153 œ 156	157	U+017E ž 158	U+0178 Ÿ 159
A_	U+00A0 NBSP 160	U+00A1 ¡ 161	U+00A2 ¢ 162	U+00A3 £ 163	U+00A4 ¤ 164	U+00A5 ¥ 165	U+00A6 ¦ 166	U+00A7 § 167	U+00A8 ¨ 168	U+00A9 © 169	U+00AA ª 170	U+00AB « 171	U+00AC ¬ 172	U+00AD 173	U+00AE ® 174	U+00AF ¯ 175
B_	U+00B0 ° 176	U+00B1 ± 177	U+00B2 ² 178	U+00B3 ³ 179	U+00B4 ´ 180	U+00B5 µ 181	U+00B6 ¶ 182	U+00B7 · 183	U+00B8 ¸ 184	U+00B9 ¹ 185	U+00BA º 186	U+00BB » 187	U+00BC ¼ 188	U+00BD ½ 189	U+00BE ¾ 190	U+00BF ¿ 191
C_	U+00C0 À 192	U+00C1 Á 193	U+00C2 Â 194	U+00C3 Ã 195	U+00C4 Ä 196	U+00C5 Å 197	U+00C6 Æ 198	U+00C7 Ç 199	U+00C8 È 200	U+00C9 É 201	U+00CA Ê 202	U+00CB Ë 203	U+00CC Ì 204	U+00CD Í 205	U+00CE Î 206	U+00CF Ï 207
D_	U+00D0 Ð 208	U+00D1 Ñ 209	U+00D2 Ò 210	U+00D3 Ó 211	U+00D4 Ô 212	U+00D5 Õ 213	U+00D6 Ö 214	U+00D7 × 215	U+00D8 Ø 216	U+00D9 Ù 217	U+00DA Ú 218	U+00DB Û 219	U+00DC Ü 220	U+00DD Ý 221	U+00DE Þ 222	U+00DF ß 223
E_	U+00E0 à 224	U+00E1 á 225	U+00E2 â 226	U+00E3 ã 227	U+00E4 ä 228	U+00E5 å 229	U+00E6 æ 230	U+00E7 ç 231	U+00E8 è 232	U+00E9 é 233	U+00EA ê 234	U+00EB ë 235	U+00EC ì 236	U+00ED í 237	U+00EE î 238	U+00EF ï 239
F_	U+00F0 ð 240	U+00F1 ñ 241	U+00F2 ò 242	U+00F3 ó 243	U+00F4 ô 244	U+00F5 õ 245	U+00F6 ö 246	U+00F7 ÷ 247	U+00F8 ø 248	U+00F9 ù 249	U+00FA ú 250	U+00FB û 251	U+00FC ü 252	U+00FD ý 253	U+00FE þ 254	U+00FF ÿ 255

Obrázek č. 8 ASCII tabulka znakové sady Windows-1252

Napsal jsem funkci v JavaScriptu, která částečně nahrazuje znaky. Částečně proto, protože nejsou zde definované všechny znaky, které matice nedokáže vypsat.

```
var diakritika = "ščřžěňďťůĚŠČŘŽĚŇĎŤŮ";
var noDiakritika = "scrzendtuESCRZENDTU";
```

Seznam znaků, které nejsou součástí kódování Win-1252, většinou se jedná o znaky s háčky. Tyto znaky jsou nahrazeny znaky bez háčků.

```
function convertDiakritika(str){
    var result = "";
    for(var i=0;i<str.length;i++){
        var c = diakritika.search(str.charAt(i));
        if(c >= 0){
            result+=noDiakritika.charAt(c);
        }
        else{
            result+=str.charAt(i);
        }
    }
    return result;
}
```

Pokud je v zadaném textu nalezena diakritika, je nahrazena znakem bez diakritiky.

Obrázek č. 9 Text ve formuláři

Obrázek č. 10 Nahrazený text v konzoli

Obrázek č. 11 Zobrazení textu „řčšýýů“ na displeji

4 VÝSLEDKY ŘEŠENÍ

4.1 Podoba hardwarového zařízení

Výsledkem hardwarového zařízení jsou funkční aktuální s datem a výpisem textu.

Výpis času je nastaven ve formátu 16:00, čas je zarovnán uprostřed a blikání dvojtečky značí sekundy.

Obrázek č. 12 Ukázka hodin

Výpis data je nastaven ve formátu 31. Pro 2020. Měsíc je převeden z čísla na zkratku měsíce. Datum se ukazuje každou minutu tehdy, kdy je počet sekund na 0 (16:01:00 = vypíše se datum). Cyklus se stále opakuje.

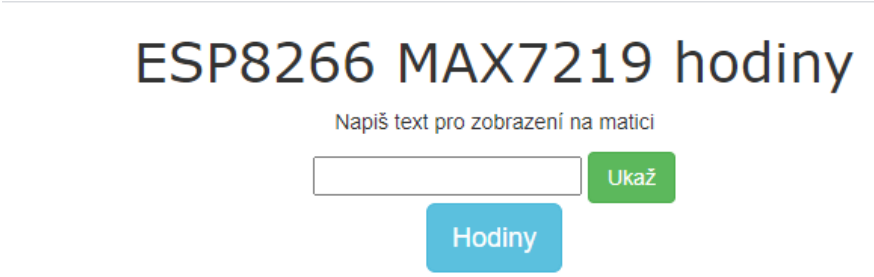
Obrázek č. 13 Ukázka zobrazení data, část měsíce a rok

Poté, co je zadán text na webové stránce se text vypíše na matici. Text se posouvá zleva doprava a opakuje se dokola, dokud není text změněn nebo není změněn mód na hodiny.

4.2 Podoba webové aplikace

Webovská aplikace není nijak složitá, obsahuje pouze název projektu, textové pole s tlačítkem pro odeslání textu a tlačítko hodin.

Do textového pole lze napsat velké množství znaků, klidně celé věty. Po odeslání (zelené tlačítko) se ve webovské konzoli vypíše celý text, který je odeslán do mikroprocesoru a následně se vypíše na displeji. Text se opakuje, dokud není pozměněn nebo nebyl stisknutý mód hodin (modré tlačítko).



Obrázek č. 14 Ukázka webové stránky s formulářem

Obrázek č. 15 Zadaný text ve formuláři

Obrázek č. 16 Ukázka zobrazení zadaného textu z formuláře

Závěr

Cílem tohoto projektu bylo vytvoření funkčních hodin. Hlavní cíl byl splněn, což je vypsání aktuálního času a data na LED matici. Povedlo se mi projekt rozšířit o funkční webserver, kde lze měnit módy hodin a také napsat text pro vypsání na matici.
Projekt lze stále vylepšovat a rozšiřovat, lze přidávat různé senzory jako například senzor vlhkosti vzduchu, teploměr, které by následné hodnoty vypsaly na matici. Dále matice dokáže vypsat další různé notifikace z různých serverů. Rád bych také displej rozšířil o další matice, které rozdělím na zóny, aby ukazovaly více informací najednou. Tato varianta je o trochu dražší záležitostí, proto jsem ji neuplatnil v mém projektu.

https://github.com/danielpeterek/projekt_v2

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] *Získání data a času – NTP klient*
https://randomnerdtutorials.com/esp32-ntp-client-date-time-arduino-ide/

[2] *Knihovna MD PAROLA*
https://majicdesigns.github.io/MD_Parola/

[3] *Knihovna MD PAROLA*
https://arduinoplusplus.wordpress.com/category/parola/

[4] *ESPAsyncWebServer s použitím SPIFFS*
https://randomnerdtutorials.com/esp32-web-server-spiffs-spi-flash-file-system/

[5] *Převod UTF-8 na rozšířené ASCII*
https://playground.arduino.cc/Main/Utf8ascii/

[6] *WeMos D1 mini dokumentace*
https://www.wemos.cc/en/latest/d1/d1_mini.html

[7] *WeMos D1 mini pinout*
https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/