Vyšší odborná škola obalové techniky a Střední škola, Štětí, Kostelní 134, příspěvková organizace

Multimediální ovladač

MATURITNÍ PRÁCE

Studijní program: 18-20-M/01 Informační technologie

Studijní obor: Internet věcí

Autor: Tomáš Kulík

Třída: 4.I

Vedoucí práce: Bc. Jakub Škrabánek

Štětí, Duben 2024

Prohlášení

Prohlašuji, že jsem maturitní práci *Multimediální ovladač* vypracoval samostatně za použití v práci uvedených pramenů a literatury.

Ve Štětí dne 12. dubna 2024 .....

Podpis studenta

Poděkování

Tímto bych chtěl poděkovat mému vedoucímiu práce za mnoho rad a poprou při vypracovávání této maturitní práce

Obsah

Seznam obrázků

Anotace

Nápad na můj projekt jsem dostal když jsem v autě potřeboval přepínat písničky bez toho abych musel používat telefon, jednoduše a rychle a stisknutím pouhého tlačítka. Také jsem si vybral tento projekt právě kvůli tomu že jsem si chtěl vyzkoušet jak funguje Spotify API

Cíl projektu byl vytvořit funkční prototyp

1. Teoretická část část

1.1 popis využitých technologii

1.1.1 Wiring

Wiring je programovací jazyk vytvořený pro programování mikrokontroléru bez specifických znalostí hardware. V současné době je nejznámější jako součást open-source platformy Arduino, kde má podobu frameworku v jazyce C++. Wiring vznikl pro vývojový kit podobný Arduinu a vychází z dalšího open-source projektu Processing. Pro programování v jazyce Wiring se nejčastěji používá integrované vývojové prostředí Arduino IDE, k dispozici jsou ale i další vývojová prostředí jako Arduino Eclipse. Wiring vyžaduje mikrokontrolér se zaváděcím programem, typicky desku Arduino osazenou čipy ATmega. Prvotním autorem jazyka je Hernando Barragán, který ho definoval ve své diplomové práci na italském institutu IDII (Interaction Design Institute Ivrea) jako součát prototypovacích nástrojů pro elektroniku a programování

1.1.2 Arduino IDE

Arduino IDE (integrated development environment – vývojové prostředí) je tedy program vytvořený přímo pro Arduino, který může být spuštěn jak ve Windows, tak v macOS a Linuxu. Tato vzájemná kompatibilita je způsobena právě kvůli samotné kódové struktuře IDE, je totiž napsaná v Javě. Mezi funkce integrovaného kódovacího editoru patří: kopírování a vkládání textu, vyhledávání a přepisování textu, automatické odsazení, doplňování závorek a zvýrazňování textu. Dále zde můžeme najít místo pro vypisování zpráv (chyby, stav nahrávání kódu...), konzoli, lištu pro práci s kódem a lištu pro nastavení IDE.

Je navrženo tak, aby podporovalo širokou škálu desek Arduino, od klasických verzí, jako je Arduino Uno, až po pokročilejší varianty, jako je Arduino Mega nebo Arduino Nano. Jako open-source software umožňuje Arduino IDE komunitě přispívat k jeho vývoji a rozšíření, což přispívá k jeho neustálému vylepšování a modernizaci.

1.1.3 LaTeX

LaTeX je profesionální systém pro sazbu dokumentů, často používaný pro psaní vědeckých a technických textů. Jeho klíčové vlastnosti zahrnují vysokou typografickou kvalitu, pod-

poru matematiky, konzistentní formátování a možnost přizpůsobení vzhledu dokumentů. Dokumenty jsou psány v textových souborech a LaTeX je multiplatformní a open-source. Je oblíbeným nástrojem mezi vědci a akademiky díky své flexibilitě a profesionálnímu vzhledu.

Konkrétně já jsem pro práci s LaTeXem využil OverLeaf, což je online LaTeX editor umožňující psaní přímo ve webovém prohlížeči. S OverLeafem můžete pracovat v reálném čase, využívat šablony, spravovat verze a exportovat dokumenty do různých formátů.

1.1.4 HTTP

HTTP (Hypertext Transfer Protocol) je internetový protokol určený pro komunikaci s WWW servery. Slouží pro přenos hypertextových dokumentů ve formátu HTML, XML, i jiných typů souborů. Používá obvykle port TCP/80, verze 1.1 protokolu je definována v RFC 2616. Společně s elektronickou poštou je HTTP nejvíce používaným protokolem, který se zasloužil o obrovský rozmach internetu.

Je však používán i pro přenos dalších informací. Pomocí rozšíření MIME umí přenášet jakýkoli soubor (podobně jako e-mail), používá se společně s formátem XML pro tzv. webové služby (spouštění vzdálených aplikací) a pomocí aplikačních bran zpřístupňuje i další protokoly, jako je např. FTP nebo SMTP.

HTTP používá jako některé další aplikace tzv. jednotný lokátor prostředků (URL, Uniform Resource Locator), který specifikuje jednoznačné umístění nějakého zdroje v Internetu.

Samotný protokol HTTP neumožňuje šifrování ani zabezpečení integrity dat. Pro zabezpečení HTTP se často používá TLS spojení nad TCP. Toto použití je označováno jako HTTPS.

1.1.5 API, Spotify API

API (zkratka pro application programming interface) označuje v informatice rozhraní pro programování aplikací. Tento termín používá softwarové inženýrství. Jde o sbírku procedur, funkcí, tříd či protokolů nějaké knihovny (ale třeba i jiného programu nebo jádra operačního systému), které může programátor využívat. API určuje, jakým způsobem jsou funkce knihovny volány ze zdrojového kódu programu.

Spotify API poskytuje rozhraní pro programování aplikací, které umožňuje vývojářům integrovat bohaté funkce Spotify do svých aplikací a služeb. S tímto API mohou vývojáři přistupovat k rozsáhlému katalogu hudby Spotify a provádět různé operace, jako je vyhledávání

skladeb, získávání informací o umělcích, albech a playlistech, správa uživatelských playlistů, řízení přehrávání hudby a mnoho dalšího.

Díky Spotify API mohou vývojáři vytvářet inovativní hudební aplikace, jako jsou doporučovací služby, hudební vizualizace, aplikace pro organizaci hudebních událostí a mnoho dalšího. API také umožňuje vytvářet personalizované hudební zážitky pro uživatele, které reagují na jejich preference a chování.

1.1.6 Spotify

Spotify je služba nabízející streamování či podcasting hudby od vybraných vydavatelství jako jsou Sony, EMI, Warner Music Group, Universal a další. Původně vznikla ve Švédsku, odkud se postupně rozšířila do některých zemí Evropy, Severní a Jižní Ameriky a Oceánie. Po dalších expanzích v letech 2020 a 2021 je služba dostupná v téměř 180 zemích světa.[1][2] V březnu 2023 měla 527 milionů uživatelů, z toho jich bylo 210 milionů platících.

1.1.7 WiFi

Pod pojmem Wi-Fi se skrývá bezdrátové propojení koncových zařízení, které používáme každý den, jako je například počítač, tablet, chytrý telefon. Následně probíhá připojení k počítačové síti, která je buď lokální nebo internetová. Zkratka Wi-Fi znamená Wireless Fidelity, tedy bezdrátová věrnost. Hlavním znakem Wi-Fi připojení je bezdrátová funkce, která přenáší signál. Nejedná se o připojení k internetu, jak se velmi často užívá, ale o zajištění bezdrátové komunikace. Kýžené připojení na internet je tedy zajišťováno jinými způsoby, například prostřednictvím routeru (je nutné správně umístit), modemu nebo přístupového bodu.

Základním „kamenem“ Wi-Fi je vysílač, který vytváří okolo sebe oblast signálu. V momentě, kdy se v dané oblasti vyskytuje zařízení, které je správně nastaveno a je schopné se na vysílač napojit, může být použito pro přístup k síti.

2. Praktická část

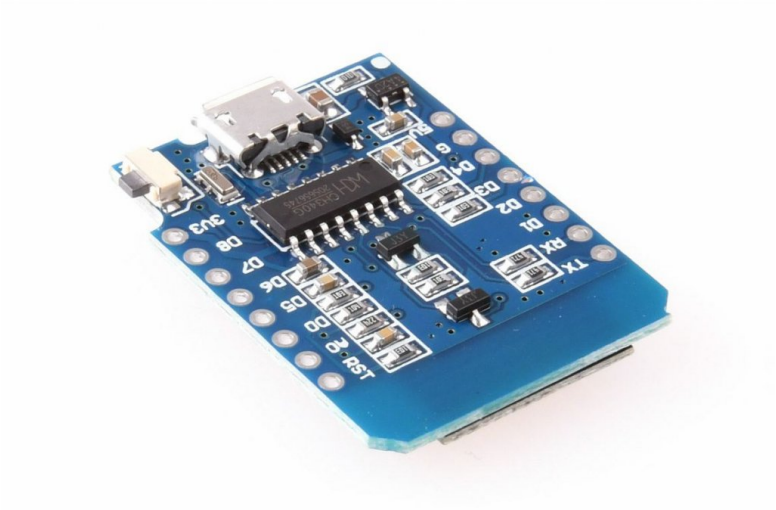
2.1 Popis konkrétní techniky

2.1.1 WeMos D1 Mini ESP8266 WiFi modul

Vývojová deska WeMos D1 Mini je kompaktní a výkonná platforma pro vývoj IoT a projektů s vestavěným Wi-Fi modulem. Založená je na čipu ESP8266, který nabízí vestavěné Wi-Fi připojení, což umožňuje snadné propojení s internetem a komunikaci s dalšími zařízeními v síti. WeMos D1 Mini má kompaktní velikost, což z ní dělá ideální volbu pro projekty, kde je důležitý malý formát. Navzdory svým rozměrům však nabízí dostatečně mnoho pinů pro připojení periferních zařízení.

WeMos D1 Mini je kompatibilní s Arduino IDE, což je oblíbené vývojové prostředí mezi elektronickými nadšenci. Díky této kompatibilitě je snadné programovat WeMos D1 Mini pomocí jazyka Arduino a využívat obrovské množství knihoven a zdrojových kódů dostupných online. To z ní dělá ideální volbu pro začátečníky i pokročilé vývojáře.

Jednou z klíčových vlastností WeMos D1 Mini je jeho vestavěná Wi-Fi konektivita. Tato vlastnost umožňuje vzdálené řízení a monitorování zařízení prostřednictvím internetu, což je základní požadavek pro mnoho projektů IoT.



Obrázek 2.1: Wemos

2.1.2 Potenciometr

Potenciometr v projektu slouží k měnění hlasitosti zvuku.

Obrázek 2.2: Potenciometr

2.1.3 Tlačítka 6x6x5mm

Tlačítka v projektu slouží k ovládání funkcí jako je zastavování skladby. Vybral jsem tyto tlačítka protože mi vyhovoval jejich rozměr.

Obrázek 2.3: Tlačítka

2.2 Popis kódu

```
#define WIFI_SSID "YOUR_WIFI_NAME"
#define PASSWORD "YOUR_WIFI_PASSWORD"

#define CLIENT_ID "YOUR_CLIENT_ID"
#define CLIENT_SECRET "YOUR_CLIENT_SECRET"
#define REDIRECT_URI "http://YOUR_ESP_IP/callback"
```

Obrázek 2.4: Přihlášení

Tato část kódu obsahuje definice konstant pro připojení k Wi-Fi síti a autentizaci u Spotify API. Obsahuje názvy Wi-Fi sítě a heslo pro připojení, stejně jako identifikační číslo a tajný klíč pro autentizaci u Spotify API. Taktéž zahrnuje URI adresa, na kterou bude Spotify API přesměrovávat při autentizaci uživatele.

```
bool getTrackInfo(){
    String url = "https://api.spotify.com/v1/me/player/currently-playing";
    https.useHTTP10(true);
    https.begin(*client,url);
    String auth = "Bearer " + String(accessToken);
    https.addHeader("Authorization",auth);
    int httpResponseCode = https.GET();
    bool success = false;
    String songId = "";
    bool refresh = false;
    if (httpResponseCode == 200) {
                //
```

Obrázek 2.5: Informace o skladbě

Tato část kódu implementuje funkci getTrackInfo(), která získává informace o aktuálně přehrávané skladbě ze Spotify API.

```
bool toggleLiked(String songId){
    String url = "https://api.spotify.com/v1/me/tracks/contains?ids="+songId;
    https.begin(*client,url);
    String auth = "Bearer " + String(accessToken);
    https.addHeader("Authorization",auth);
    https.addHeader("Content-Type","application/json");
    int httpResponseCode = https.GET();
    bool success = false;
    // Check if the request was successful
    if (httpResponseCode == 200) {
        String response = https.getString();
        https.end();
        if(response == "[ true ]"){
            currentSong.isLiked = false;
            dislikeSong(songId);
        }else{
            currentSong.isLiked = true;
            likeSong(songId);
        }
        drawScreen(false,true);
        Serial.println(response);
        success = true;
    } else {
        Serial.print("Chyba při přidání písničky do oblíbených: ");
        Serial.println(httpResponseCode);
        String response = https.getString();
        Serial.println(response);
        https.end();
    }
```

Obrázek 2.6: Přidání/odebrání z oblíbených

Tato část kódu používá funkci ‘toggleLiked(String songId)‘, která přepíná oblíbenost určité skladby na Spotify.

```
bool togglePlay(){
    String url = "https://api.spotify.com/v1/me/player/" + String(isPlaying ? "pause" : "play");
    isPlaying = !isPlaying;
    https.begin(*client,url);
    String auth = "Bearer " + String(accessToken);
    https.addHeader("Authorization",auth);
    int httpResponseCode = https.PUT("");
    bool success = false;
    // Check if the request was successful
    if (httpResponseCode == 204) {
        // String response = https.getString();
        Serial.println((isPlaying ? "Playing" : "Pausing"));
        success = true;
    } else {
        Serial.print("Error pausing or playing: ");
        Serial.println(httpResponseCode);
        String response = https.getString();
        Serial.println(response);
    }
```

Obrázek 2.7: Pozastavení/spuštění

Tato část kódu implementuje funkci togglePlay(), která přepíná mezi přehráváním a pozastavením přehrávání aktuální skladby na Spotify.

```
bool skipForward(){
    String url = "https://api.spotify.com/v1/me/player/next";
    https.begin(*client,url);
    String auth = "Bearer " + String(accessToken);
    https.addHeader("Authorization",auth);
    int httpResponseCode = https.POST("");
    bool success = false;
    // Check if the request was successful
    if (httpResponseCode == 204) {
        // String response = https.getString();
        Serial.println("skipping forward");
        success = true;
    } else {
        Serial.print("Error skipping forward: ");
        Serial.println(httpResponseCode);
        String response = https.getString();
        Serial.println(response);
    }
}
bool skipBack(){
    String url = "https://api.spotify.com/v1/me/player/previous";
    https.begin(*client,url);
    String auth = "Bearer " + String(accessToken);
    https.addHeader("Authorization",auth);
    int httpResponseCode = https.POST("");
    bool success = false;
    // Check if the request was successful
    if (httpResponseCode == 204) {
        // String response = https.getString();
        Serial.println("skipping backward");
        success = true;
    } else {
        Serial.print("Error skipping backward: ");
        Serial.println(httpResponseCode);
        String response = https.getString();
        Serial.println(response);
```

Obrázek 2.8: Dopředu/dozadu

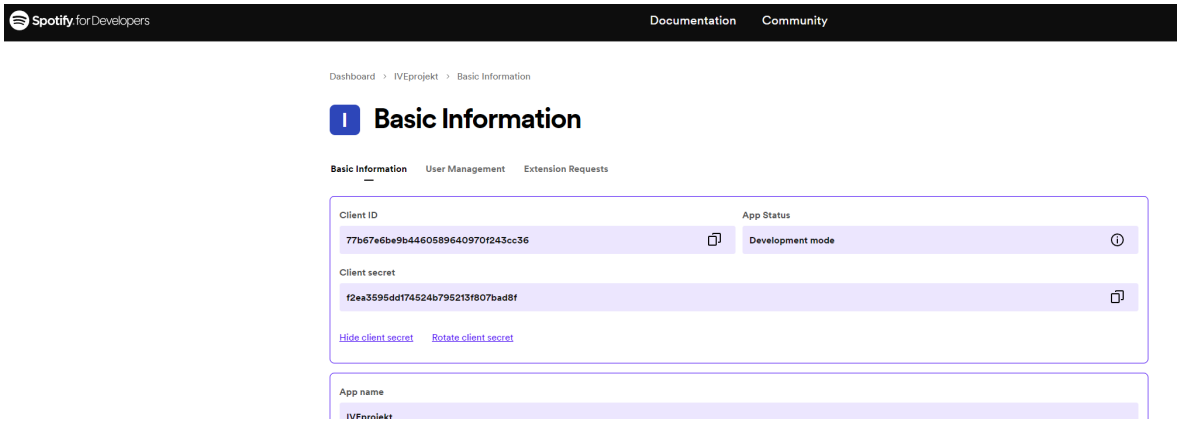
Tato část kódu slouží k posunu skladby na Spotify. Metoda skipForward() posouvá skladby vpřed, skipBack() posouvá skladby zpět.

```
for(int i = 0; i < 4; i ++){
    int reading = digitalRead(buttonPins[i]);
    if( reading != buttonStates[i]){
        if(millis() - debounceTimes[i] > debounceDelay){
            buttonStates[i] = reading;
            if(reading == LOW){
                switch (i)
                {
                case 0:
                    spotifyConnection.togglePlay();
                    break;
                case 1:
                    spotifyConnection.toggleLiked(spotifyConnection.currentSong.Id);
                    break;
                case 2:
                    spotifyConnection.skipForward();
                    break;
                case 3:
                    spotifyConnection.skipBack();
                    break;

                default:
                    break;
                }
            }
            debounceTimes[i] = millis();
        }
    }
}

int volRequest = map(analogRead(A0),0,1023,0,100);
if(abs(volRequest - spotifyConnection.currVol) > 2){
    spotifyConnection.adjustVolume(volRequest);
}
```

Obrázek 2.9: Tlačítka

Tato část kódu čte stav tlačítek a provádí odpovídající akce v závislosti na jejich stavu. Každé tlačítko má přiřazenou specifickou funkci které jsem představil předešlých ukázkách kódu.

2.3 Postup



Obrázek 2.10: Ukázka z Developerského prostředí Spotify

Nejdříve jsem se zaregistroval jako vývojář na platformě Spotify, abych získal přístup k jejich API a získal své vlastní identifikační údaje klienta (Client ID). Tento proces byl nezbytný pro autorizaci mého projektu a komunikaci s Spotify službami.

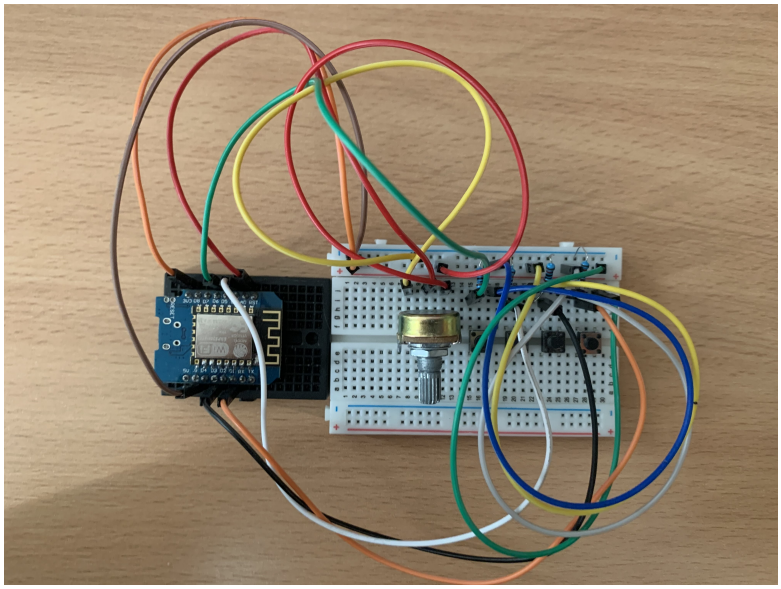
Poté jsem se zaměřil na zkoumání dokumentace Spotify API. Spotify poskytuje bohatě dokumentované API s mnoha užitečnými funkcemi a možnostmi. Získal důležité informace o tom, jak pracovat s API, jaké funkce jsou k dispozici a jak je použít.

S těmito znalostmi jsem se pustil do nákupu potřebných součástek pro realizaci projektu. Pro tento konkrétní projekt jsem se rozhodl použít Wemos D1 Mini ESP8266, který je kompaktní a cenově dostupným řešením pro IoT projekty a hlavně má v již v sobě zabudovaný Wi-Fi modul. Potřeboval jsem také tlačítka a potenciometr pro ovládání přehrávání a hlasitosti.

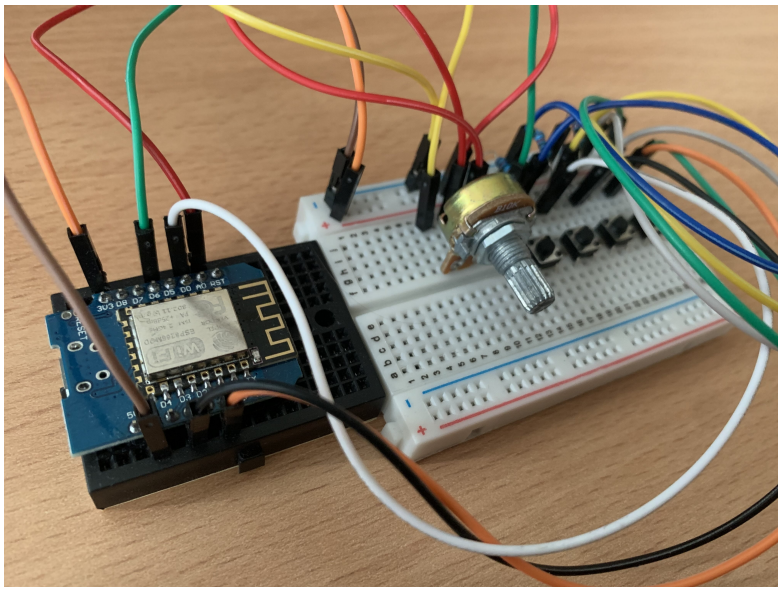
Po přípravě všech potřebných součástek jsem mohl začít s fyzickou realizací projektu a implementací kódu. Díky znalostem získaným ze studia dokumentace jsem byl schopen efektivně naprogramovat zařízení tak, aby se připojovalo k Spotify API, získávalo informace o aktuálně přehrávané skladbě a umožňovalo ovládání přehrávání pomocí tlačítek.

Po vytvoření přvního funkčního prototypu jsem se musel naučit s LaTeXem a zpracovat dokumentaci, bylo to náročné, ale zvládl jsem to

2.4 Fotky prototypu



Obrázek 2.11: Schéma 1



Obrázek 2.12: Schéma 2

3. Závěr

3.0.1 Budoucnost projektu

Display

Do budoucna mám v plánu rozšířit funkcionality projektu o možnost zobrazování aktuálně přehrávané písničky přímo na displeji. Tato funkce by zahrnovala zobrazení názvu písně, obrázku alba a důležitě také aktuální pozici v přehrávané písničce.

Pro tuto funkcionalitu plánuji využít 1.8"128x160 TFT displej ST7735. Tento displej nabízí dostatečné rozlišení a širokou barevnou škálu, což umožní zobrazit informace o přehrávané písničce v dostatečné kvalitě.

Implementace této funkcionality bude vyžadovat úpravu stávajícího kódu tak, aby byl schopen komunikovat s displejem a zobrazovat informace o aktuálně přehrávané písničce. Dále bude třeba získat data o názvu písně, obrázku alba a pozici v písničce z Spotify API a přenést je do formátu, který bude možné zobrazit na displeji.

Tímto rozšířením projektu bych chtěl zvýšit jeho užitečnost a přehlednost pro uživatele, a poskytnout tak ještě pohodlnější způsob ovládání a sledování aktuálně přehrávané hudby.

Krabička

Plánuji vymodelovat a vytisknout krabičku. Ale předtím budu muset trošku optimalizovat zapojení, protože to současné není zcela nejšťastnější a mohlo by dojít k problémům s umístěním všech komponentů do krabičky.

3.0.2 Vlastní poznatky

Přiznám se, že práce s Spotify API byla pro mě původně velkou výzvou. Avšak i přes prvotní obtíže jsem byl schopen získat potřebné znalosti a začlenit Spotify API do mého projektu.

Pokud jde o LaTeX a prostředí OverLeaf, překvapivě mi to velmi vyhovuje. Přestože jsem s LaTeXem předtím neměl mnoho zkušeností a zprave jsem se bál ho využít, rychle jsem se naučil s ním pracovat. Plánuji pokračovat v jeho používání i do budoucna.

Co se týče samotného projektu, jsem s prvním funkčním prototypem celkem spokojen. Je pravda, že je stále prostor pro zlepšení, především co se týče estetiky, ale momentálně jsem s dosaženými výsledky spokojen.

4. Zdroje

https://en.wikipedia.org/wiki/Arduino

https://www.arduino.cc/

https://www.overleaf.com/

https://www.overleaf.com/learn

https://wikisofia.cz/wiki/Wi-Fi

https://cs.wikipedia.org/wiki/Wi-Fi

https://www.laskakit.cz/wemos-d1-mini-esp8266-wifi-modul/

https://www.wemos.cc/

https://cs.wikipedia.org/wiki/API

https://cs.wikipedia.org/wiki/Wiring_(programovací_jazyk)

https://developer.spotify.com/documentation/web-api

https://cs.wikipedia.org/wiki/Spotify

https://en.wikipedia.org/wiki/LaTeX