Gymnázium, Praha 6, Arabská 14

Programování

# Maturitní Práce

Ondřej Kuban

duben 2020

Gymnázium, Praha 6, Arabská 14

# MATURITNÍ PRÁCE

Předmět: Programování

Téma: Inteligentní ovladač ambientního osvětlení

**Autor:** Ondřej Kuban

**Třída:** 4.E

**Školní rok:** 2019/2020

**Vedoucí práce:** Mgr. Jan Lána

**Třídní učitel:** Mgr. Jana Urzová, Ph.D.

# Prohlášení

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne ..... .....

# Anotace

Ve své maturitní práci jsem se zabýval vývojem zařízení pro ovládání adresovatelných LED diod a mobilní aplikací pro operační systém Android. Práce měla tři části. První částí byl hardware, který bylo nutno vymyslet a následně sestavit tak aby umožňoval funkce které byly v zadání. Druhou částí byl vývoj softwaru zařízení, které ovládá LED diody, komunikuje za pomocí Bluetooth a Wi-Fi a zpracovává audio signál v reálném čase. Pro vývoj softwaru byl použit jazyk C++. Třetí částí byl vývoj mobilní aplikace pro operační sytém Android v programovacím jazyce Kotlin, pomocí které je uživateli umožněno konfigurovat a ovládat zařízení.

# Abstract

In my graduation thesis, I dealt with the development of devices for controlling addressable LEDs and mobile applications for the Android operating system. The work had three parts. The first part was the hardware, which had to be invented and then assembled to allow the functions that were in the assignment. The second part was the software development of a device that controls LEDs, communicates via Bluetooth and Wi-Fi and processes the audio signal in real time. C ++ language was used for software development. The third part was the development of a mobile application for the Android operating system in the Kotlin programming language, which allows the user to configure and control the device.

# Zadání

Zadáním práce bylo vytvořit mobilní nebo počítačovou aplikaci s možností zvolení barvy nebo barevného schéma. Aplikace by měla umožňovat ovládání pomocí Google asistenta a mít možnost nastavení rutin neboli budíků. Řídící jednotkou zařízení mělo být ESP32 na kterém by měla probíhat audio vizualizace pomocí Fourierovy transformace a automatická regulace jasu založená na okolním osvětlení.

# 1. Úvod

V této maturitní práci jsem se zabýval vývojem zařízení, které ovládá adresovatelný LED pásek a vývojem mobilní aplikace určené k ovládání toho zařízení. Mobilní aplikace umožňuje uživateli jednoduchým způsobem upravovat vlastnosti LED pásku, například světelný efekt nebo barvu. Jedním z efektů je vizualizace zvuku, kterou zajišťuje algoritmus rychlé Fourierovy transformace a grafický ekvalizér MSGEQ7. Uživatel si může taktéž nastavit budík a nastavit mu příslušný efekt a čas spuštění. Komunikaci mezi zařízením zařizuje lokální Wi-Fi síť a Bluetooth.

Hlavní motivací vytvoření toho projektu bylo mé nadšení tvorby barevného osvětlení v mém pokoji. Tento projekt jsem začal před dvěma lety, kdy jsem se rozhodl, že chci barevné osvětlení mého pokoje, které bude reagovat na hudbu. Jelikož jsem projekt zanedbával, zdálo se jako dobrý nápad projekt povýšit na maturitní práci a plně se mu věnovat a dokončit ho.

# 2. Technologie

V této kapitole jsou popsány použité technologie, které byly použity při vývoji mobilní aplikace a zařízení.

## 2.1. Vývojářské nástroje

Tato podkapitola se věnuje vývojářským a designerským nástrojům, které byly využity při vývoji.

### 2.1.1. Mobilní Aplikace

Vývoj mobilní aplikace probíhal převážně ve vývojářském prostředí Android Studio, které je vyvíjeno firmou Google. Toto prostředí bylo zvoleno, protože poskytuje největší podporu vývoje aplikací s operačním systémem Android. Vývoj probíhal v programovacím jazyce Kotlin, který je upřednostňován k vývoji aplikací před Javou. Grafika aplikace byla primárně vytvořena pomocí aplikací Adobe Illustrator a Adobe After Effects a následně přepsána do jazyka XML.

### 2.1.2. Zařízení

Visual Studio Code a Arduino IDE byly vývojářskými prostředími pro vývoj softwaru pro zařízení. Arduino IDE bylo použito pro vývoj testovacích programů a pro datovou vizualizaci pomocí vestavěného plotru. Ve Visual Studiu Code byl vyvíjen finální program za pomocí rozšíření PlatformIO.

## 2.2. Technologie

Tato kapitola popisuje technologie a knihovny které byly použity při vývoji.

### 2.2.1. Material Design

Material design je knihovna obsahující nové nebo vylepšené UI komponenty. Tato knihovna byla využita pro ucelení grafického výstupu aplikace a pro širší možnosti úprav chování a grafiky jednotlivých komponent. Cílem této knihovny je sjednotit grafiku uživatelského prostředí na více platformách a umožnit vývojářům a grafikům flexibilitu při tvorbě uživatelského prostředí.

### 2.2.2. Lottie

Knihovna Lottie vyvíjena firmou Airbnb umožňuje jednoduchou tvorbu animací napříč všemi platformami. Animace vytvořené v Adobe After Effects exportované ve formátu JSON umí knihovna Lottie zobrazovat. Výhodou této knihovny je jednoduchost vytváření animací a jejich jednoduché zobrazování uživateli. Jelikož jsou animace vektorové, nedochází ke zhoršení kvality při zvětšení a nezabírají velké množství paměti, jak by to bylo u animací poskládaných z jednotlivých snímků.

### 2.2.3. UDP

Jako komunikační protokol byl zvolen protokol UDP neboli User Datagram Protocol. Tento protokol oproti protokolu TCP nezaručuje doručení dat, protože nemá potvrzování o přijetí ani časové limity. Protokol UDP nezaručuje také doručení dat ve stejném pořadí, v jakém byly odeslány. Oproti TCP je ale jednodušší a nezatěžuje tak zbytečně mikrokontroler.

### 2.2.4. PlatformIO

Platform IO je nástroj na vývoj softwaru pro mikrokontrolery. Umožňuje kompilace a nahrání kódu do mikrokontroleru skrz uživatelské prostředí Visual Studio Code. Tento nástroj umožňuje rychlejší vývoj softwaru díky kontrole syntaxe a automatického doplňování kódu.

# 3. Mobilní Aplikace

Mobilní aplikace zajišťuje veškerou interakci uživatele se zařízením. Aplikace umožňuje přepínat mezi šesti jednotlivými typy efektů. U některých efektů může uživatel nastavovat odstín, saturaci, jas a rychlost. Uživatel má možnost přidání budíků, u kterých lze nastavit čas a den spuštění, název, efekt a trvání efektu.

## 3.1. Main Activity

Hlavní aktivita aplikace. Načítá uloženou IP adresu zařízení pomocí SharedPrefernces.

Hlavní aktivita se stará o zobrazování hlavních fragmentů jako je HomeFragment, PresetsFragment. Tyto fragmenty jsou zobrazovány pomocí view pageru a pager adaptéru. Pager adaptér umožňuje přesun mezi jednotlivými fragmenty pomocí gest.

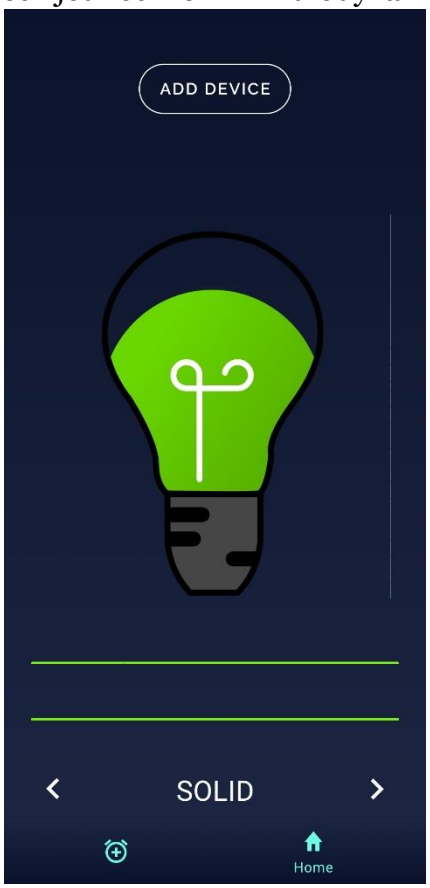
## 3.2. Fragmenty

### 3.2.1. Home

*HomeFragment.kt* zobrazuje uživateli hlavní obrazovku, na které může měnit odstín, saturaci a jas LED diod. Může přepínat mezi jednotlivými efekty a měnit rychlost jejich přehrávání. Uživatel zde může přidat nové zařízení pomocí tlačítka „ADD DEVICE“.

Na obrazovce jsou celkem čtyři posuvníky, pomocí kterých lze měnit barvu jas a rychlost efektů světla. V horní části obrazovky je posuvník pro ovládání rychlosti, na pravé části obrazovky posuvník pro ovládání jasu. Posuvníky v dolní části obrazovky umožňují změnu barvy. Horní posuvník mění odstín světla a zároveň mění svou barvu podle zvoleného odstínu. Dolní posuvník mění saturaci a jeho barva je stejná jako barva posuvníku pro odstín s odpovídající saturací. Každý z posuvníků má nastavenou tloušťku na jeden pixel, aby při používání aplikace nepůsobila dojmem přeplácanosti. Při kliknutí na posuvník se jeho tloušťka zvětší.

V dolní části obrazovky se nachází menu s efekty. Přepínat mezi jednotlivými efekty lze šipkami. Celkem je v aplikaci šest různých efektů. Efekt „Solid“ nastaví jednotnou barvu pro všechny LED diody. „Pulsing“ efekt postupně snižuje a zvyšuje jas všech LED diod. Efekt „Twinkle“ náhodně rozsvěcí jednotlivé LED diody a postupně je zhasíná. „Phasing“ mění barvu všech LED diod a prolíná všemi odstíny. Efekt „Rainbow“ vytváří pohybující se duhu na celém LED pásku. V neposlední řadě efekt „Music“ využívá analýzy zvuku v zařízení a zobrazuje efekty korespondující se zvukem.



*Obr. 1 Hlavní obrazovka*

Uprostřed obrazovky se zobrazuje animace žárovky. Animaci zprostředkovává knihovna Lottie. Spolu se změnou jasu se mění i žárovka, při maximálním jasu je žárovka úplná a s postupně snižujícím se jasem mizí vlákno a světlo žárovky. Změnou barvy se mění i barva světla žárovky. Změnu barvy zajišťuje metoda *lightBulbColor* která nastaví barevný filtr pro jednotlivé vrstvy animace.

### 3.2.2. Wifi List

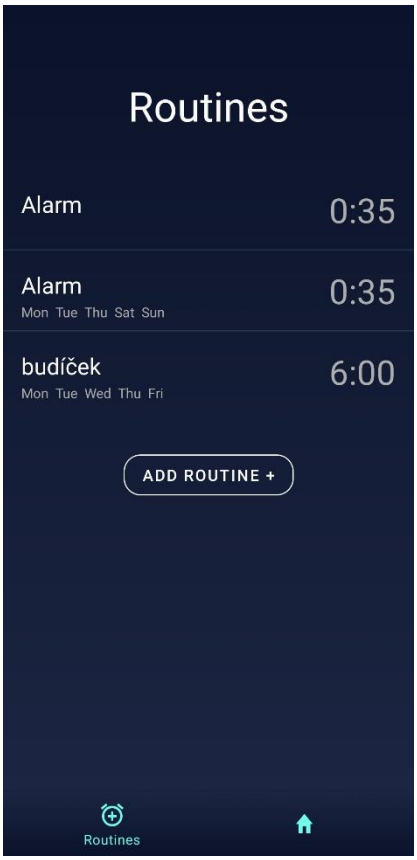
Wifi List fragment zobrazuje v listView dostupné Wi-Fi sítě. Uživateli se zobrazí síla signálu, název sítě, MAC adresu sítě a jestli je síť otevřená nebo uzamčená. Uživatel vybere síť, ke které chce zařízení připojit a aplikace ho přesměruje do Wifi Login fragmentu.

### 3.2.3. Wifi Login

Tento fragment slouží k zadání hesla k Wi-Fi síti. Po zadání hesla se heslo spolu s jménem sítě odešle pomocí Bluetooth do zařízení, které se k síti lze připojit. Pokud nelze je uživatel nucen pokusit se připojit k jiné síti nebo zkusit znovu zadat heslo. Pokud se zařízení úspěšně připojilo pošle zpět do aplikace jeho IP adresu.

### 3.2.4. Presets

Presets fragment zobrazuje nastavené budík pomocí listView. U každého budíku je zobrazeno jméno, čas a dny aktivace a možnost deaktivace. Přidání nového budíku je možné tlačítkem „Add Routine“ které zobrazí fragment pro nastavení budíku po jehož dokončení se jeho parametry odešlou do zařízení.




*Obr. 2 Presets Obrazovka*

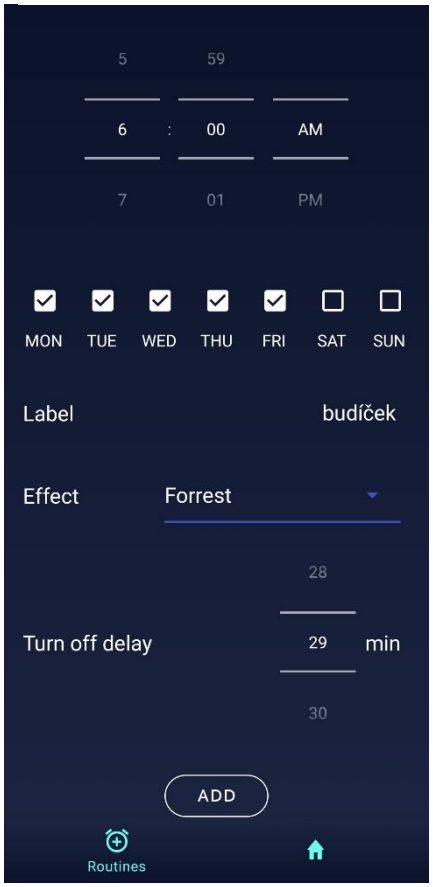
## 3.3. Adaptéry

### 3.3.1. Routines Adapter

Adaptér pro seznam budíků v Presets Fragmentu. Adaptér zpracovává data z Routine objektu do jedné položky.

### 3.3.2. Wifi Adapter

Adaptér pro list Wi-Fi sítí ve WifiList fragmentu. Adaptér ze vstupních dat nastaví položku a přidělí ji příslušnou ikonu sítě která je volena na základě síly signálu v decibelech a zabezpečení.



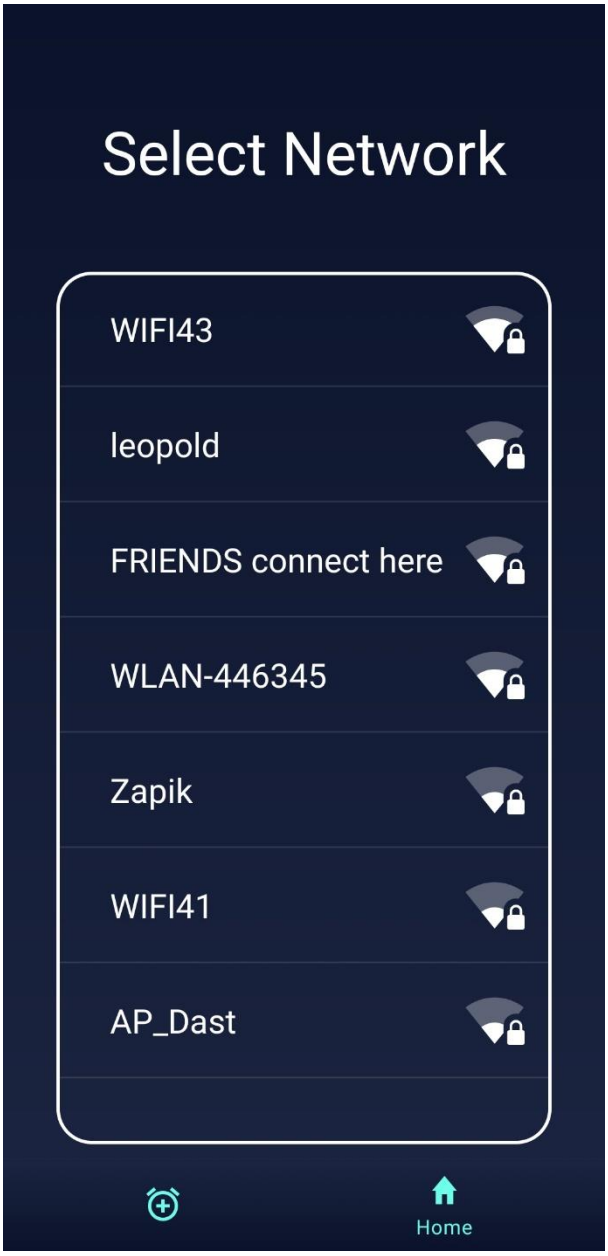

*Obr. 3 Vytváření budíku*

## 3.4. Komunikace

### 3.4.1. Bluetooth

Bluetooth komunikace se skládá ze dvou tříd, z BluetoothSerice a CommunicationThread. BluetoothService při přidávání nového zařízení zahájí vyhledávání Bluetooth zařízení v okolí s příslušným názvem. Pokud je zařízení nalezeno, aplikace zjistí, zdali je zařízení již spárováno, pokud ano naváže se komunikace pomocí CommunicationThreadu. Pokud zařízení spárované není, aplikace zařízení spáruje a následně se naváže komunikace.

Communication Thread spustí nekonečnou smyčku která naslouchá, zda nepřišla data ze zařízení. Pokud ano data se zpracují a ověří (viz. kapitola 4.2.3).



*Obr. 4 Obrazovka pro zvolení sítě*

### 3.4.2. Socket Client

Třída SocketClient zpracovává všechny odeslané zprávy pomocí UDP protokolu. Každá odeslaná zpráva se zpracuje dle ASCII protokolu (viz. kapitola 4.2.3). Pokud tělo paketu obsahuje více než jeden údaj jsou údaje odděleny rovnítkem. Následně se spočítá kontrolní součet a data se odešlou do všech připojených zařízení.

# 4. Zařízení

## 4.1. Hardware

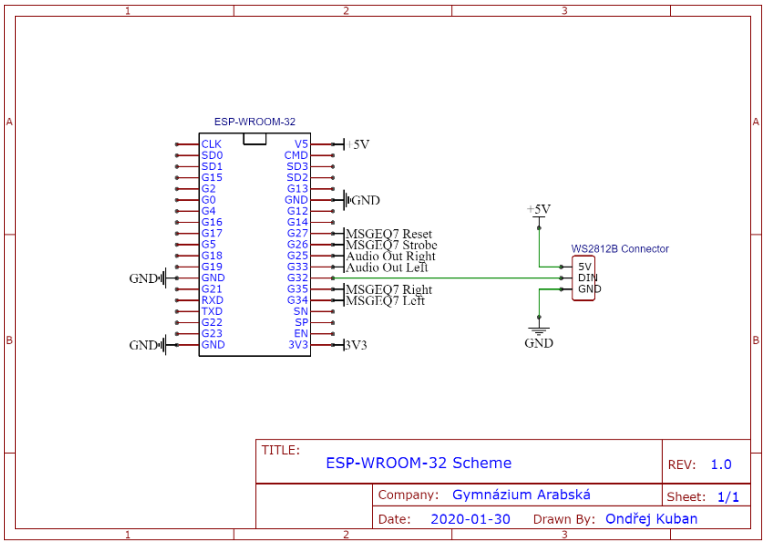
Hardwarovou část tohoto projektu tvoří mikrokontroler ESP32, grafický ekvalizér MSGEQ7, obvod pro automatické řízení hlasitosti, LED pásek WS2812B a pěti voltový zdroj o výkonu 100 wattů.

*Obr. 5 ESP-WROOM-32*

### 4.1.1. Mikrokontroler ESP32

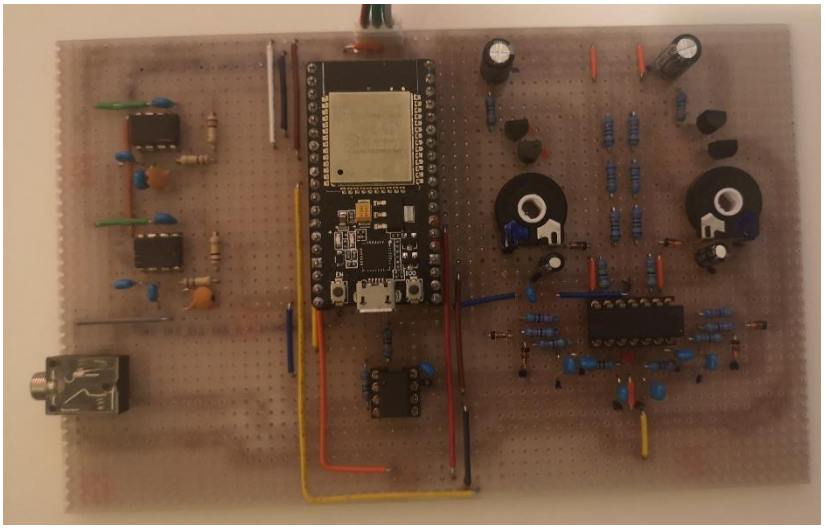
Pro cílové zařízení byl zvolen mikrokontroler ESP32, verze ESP-WROOM-32, pro jeho výkon a možnost komunikace pomocí Wi-Fi nebo Bluetooth.

Tento mikrokontroler je open-source a jeho výrobce Espressif poskytuje veškerá schémata a rozložení desky plošných spojů zdarma. Mikrokontroler je schopný operovat v teplotách od – 40 C do 85 C a je tak naddimenzovaný pro použití v této práci. Disponuje 520 kb statickou RAM pamětí a 16 MB flash pamětí. Procesor operuje dvěma výpočetními jádry s frekvencí 240MHz. Mikrokontroler má šestnáct 12bitových ADC (Analog to Digital Converter), při zapnuté Wi-Fi jsou ADC 0-3 nefunkční. ESP32 pracuje na napětí 3.3 V, je tedy schopno dosáhnout maximálního výstupního napětí 3.3 V. Disponuje také 10 kapacitními vstupy pro kapacitní senzory, stejně jako u ADC jsou kapacitní vstupy 0-3 nefunkční při zapnuté Wi-Fi.

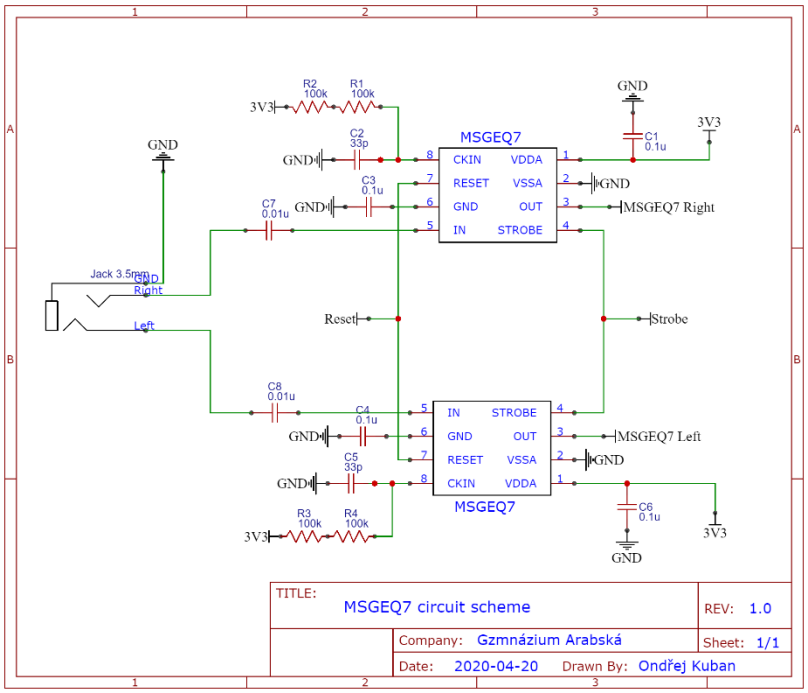



*Obr. 6 Schéma zapojení mikrokontroleru*

### 4.1.2. Grafický ekvalizér MSGEQ7



*Obr. 7 Sestavené deska zařízení*




*Obr. 8 Schéma zapojení MSGEQ7*

MSGEQ7 je čip se sestavou sedmi pásmových propustí, špičkových detektorů a multiplexoru. Pásmové propusti jsou nastaveny na frekvence 63 Hz, 160 Hz, 400 Hz, 1kHz, 2.5kHz, 6.25 kHz a 16kHz. Čip může být napájen 2.7 V až 5.5 V. V zařízení je napájen ze zdroje 3.3 V mikrokontroleru i přes to že při 5 V poskytuje nejlepší výkon. Klidový odběr čipu je méně než 1 mA a přispívá tak k nízké spotřebě celého zařízení.

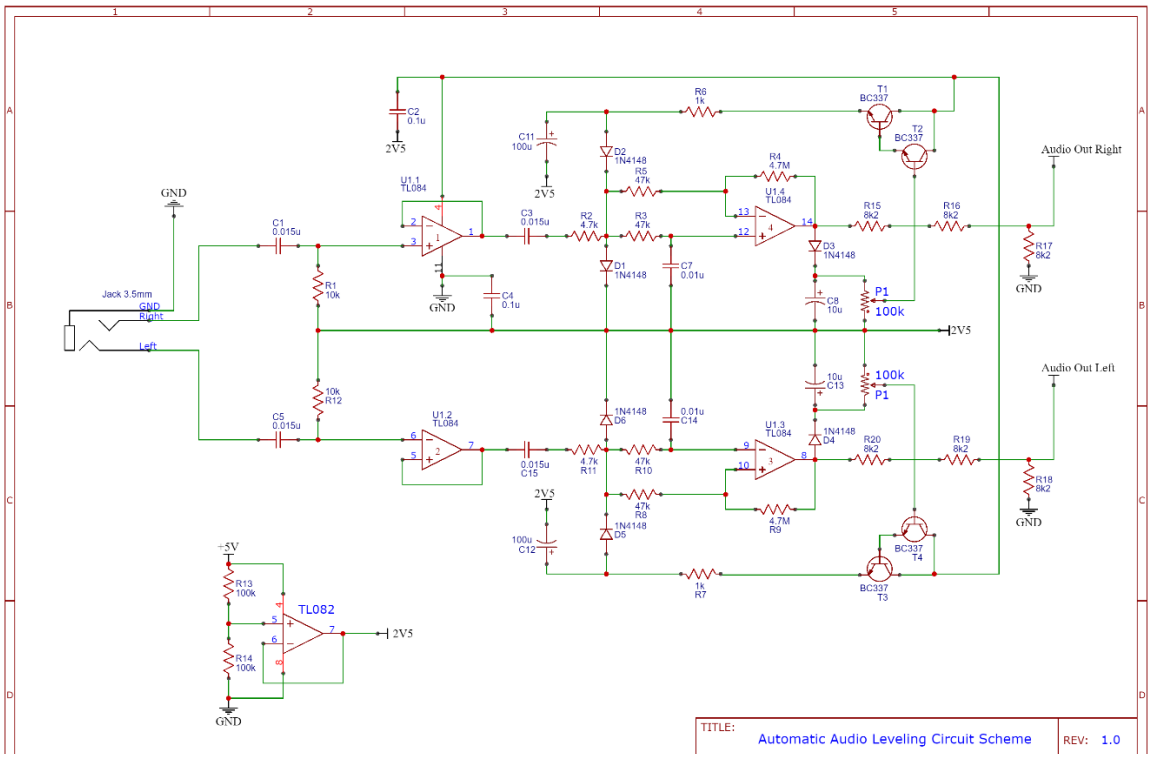
Čip odstraňuje šum z audio vstupu pomocí anti-alias filtru a následně je signál zpracován sedmi pásmovými filtry a samostatnými špičkovými detektory. Signál špičkových detektorů je přepínán multiplexorem do analogového výstupu měřeného ADC vstupem mikrokontroleru. Multiplexor je řízen čítačem a jeho vstupem “strobe”. Pro synchronizaci je použit vstup “reset”.

Doporučené zapojení tohoto čipu se skládá ze dvou 22KΩ určené pro spojení levého a pravého audiosignálu a jednoho 200kΩ rezistoru, čtyřech keramických kondenzátorů o hodnotách 0.1,0.01 mikro Faradů a 33 piko Faradů. Toto zapojení bylo upraveno použitím dvou 100kΩ rezistorů místo jednoho 200kΩ. V konkrétním zapojení rezistory 22kOhm nejsou zapojeny vzhledem k použití dvojice MSGQ7 po každý kanál zvlášť.

### 4.1.3. Obvod automatického řízení hlasitosti

V zadání této práce byl úkol použít pro audio vizualizaci Fourierovu transformaci. Při testování audio signálu bylo zjištěno, že úroveň signálu je úměrná hlasitosti. Vzhledem k proměnlivé úrovni signálu z počítače nebo mobilu bylo nutné tento signál upravit na konstantní úroveň. Pro ustálení amplitudy signálu byl použit obvod pro automatické řízení hlasitosti.

Tento obvod není doporučován pro Hi-Fi, ale pro analýzu audio signálu byl dostačující. V obvodu je použit integrovaný čip TL082, avšak v zařízení je použit operační zesilovač TL084, který disponuje 4 jádry naproti TL082, které disponuje pouze jádry dvěma. Obvod je určen pro napájení souměrným napětím +12V a –12V. Vzhledem k tomu, že v zařízení je k dispozici pouze nesouměrné napájení 5V, bylo




*Obr. 9 Schéma automatického řízení hlasitosti*

nutné pomocí jednoho z jader TL082 vytvořit virtuální souměrné napájení se středem 2.5V stejnosměrných. Mikrokontroler je schopen měřit ADC převodníkem max. 3.3V, proto je výstup kompresního OZ opatřen odporovým děličem.

## 4.2. Software

### 4.2.1. Programové smyčky

Jak již bylo zmíněno mikrokontroler ESP32 má dvě výpočetní jádra a na každém z nich lze provádět operace nezávislé na sobě. Jádro číslo jedna bylo využito pro zpracovávání audio signálu, aby nebylo ovlivněno ostatními funkcemi zařízení. Na prvním jádře je také zpracovávána Bluetooth komunikace, jelikož audio vizualizace neprobíhá simultánně s Bluetooth komunikací. V druhém jádru byla vytvořena sekundární programová smyčka na aktualizaci LED diod a času pro budíky.

### 4.2.2. Ovládání LED

Ovládání adresovatelných LED diod zajišťují metody v souborech *Light.cpp* a *Light.h*. K ovládání je nutná knihovna FastLED která komunikuje s diodami.

Ve funkci *setupLight* se přidají LED diody a inicializují se pole pro jejich ovládání. Jelikož je knihovna schopna zpracovat pouze pole typu RGB, které umožňuje nastavování barev pouze mícháním třech základních barev, je nutné vytvořit pole se stejnou velikostí typu HSV které umožňuje jednoduší nastavování barev pomocí složek odstínu, saturace a jasu.

Funkce *updateLight* je volána v sekundární programové smyčce. Funkce volá efekt, který je aktuálně zvolen. Volání efektů je omezeno pouze na určitý počet volání za sekundu, tento počet je variabilní a jeho změnou se změní rychlost přehrávaných efektů.

Funkce *setEffect* nastaví číselné označení efektu a u vybraných efektů přednastaví barvy pro efekt.

Ve funkci *setMusicPeak* se přepočítávají výstupní data Fourierovy transformace na data vhodnější pro následnou audio vizualizaci.

Následně tento soubor obsahuje několik menších metod, které zastupují jednotlivé efekty a zpracovávají barvy pro jednotlivé LED diody.

### 4.2.3. Třída Packet

Třída Packet zpracovává přijatá data ve formě ASCII protokolu a ověřuje je.

ASCII protokol se skládá ze tří částí, z hlavičky, těla a kontrolního součtu. Začátek paketu označuje znak ASCII s hodnotou 1 což je kód pro „Start of Header“ neboli začátek hlavičky. Poté obsahuje paket hlavičku, která je ukončena znakem s hodnotou 2 tedy „Start of Text“, v češtině začátek textu. Následuje tělo, které může obsahovat jednu nebo více položek. Tělo paketu je ukončeno kódem 3 což je „End of Text“ neboli konec textu za kterým následuje kontrolní součet ukončený hodnotou 4 „End of Transmission“ neboli konec přenosu.

Metoda *setData* zpracovává přijatá data do jednotlivých částí a ověřuje jejich správnost pomocí kontrolního součtu.

Pro nastavení hlavičky a těla paketu pro odeslání se používají metody *setHead* a *setBody*. Pro odeslání paketu ve správném formátu je potřeba použít funkci *buildPacket* která převede data do formátu ASCII protokolu a vypočítá kontrolní součet.

Výpočet kontrolního součtu je vypočten v metodě *calculateCheckSum*. Tato metoda dostane jako parametr data a délku dat ze kterých vypočítá kontrolní součet. Kontrolní součet se počítá od začátku paketu až po konec textu (znak s kódem 3, „End of Text“). Pro výpočet kontrolního součtu byl použit Fletcherův algoritmus. Fletcherův algoritmus patří k jednodušším kontrolním součtům ale také k efektivním. I malá změna v paketu se projeví ve finálním výsledku. Kontrolní součet má dva různé výsledky. První výsledek je pouze součet všech hodnot všech znaků v paketu a druhý výsledek je součtem všech prvních mezivýsledků při každém znaku. Každý z výsledků je poté vymodulován 255 a přepočten do šestnáctkové soustavy, výsledný kontrolní součet je tak tvořen čtyřmi znaky.

V první verzi této třídy byl místo pole pro znaky použit string a ukládání dat probíhalo přidáváním jednotlivých znaků. Tato operace zabírala tolik času, že se

projevovalo zpoždění na zobrazování barev na LED diodách a působila „sekavě“. Po přepsání všech funkcí tak aby podporovaly pole pro znaky toto „sekavé“ chování zmizelo. Následně bylo zjištěno že přidávání znaků do stringu může být až dvacetkrát pomalejší než načítání znaků do pole.

### 4.2.1. Wi-Fi komunikace

Zařízení komunikuje převážně přes Wi-Fi síť pomocí UDP protokolů. Zařízení se připojí na Wi-Fi síť pomocí SSID a hesla které bylo získáno od uživatele. Při prvním připojení se heslo a SSID uloží do vnitřní paměti mikrokontroleru. Před připojováním je nejprve odeslán paket o začátku připojování k síti přes Bluetooth. Pokud pokus o připojení patnáctkrát selže, je přes Bluetooth odeslána informace o selhání s kódovým označením 2. Když se připojení k síti naváže je odeslána informace o úspěšném připojení a následně je odeslána lokální IP adresa zařízení. Nakonec je spuštěn UDP server, který přijímá data a zpracovává je.

### 4.2.2. Bluetooth komunikace

Zařízení využívá svého vestavěného Bluetooth modulu pro prvotní nastavení zařízení. Při spuštění se inicializuje Bluetooth a nastaví se jméno zařízení.

V sekundární programové smyčce se volá metoda *checkBluetooth* která zpracovává přijatá data. Při přijetí kódu 1(„Start of header“) se začnou načítat data do zásobníku jehož plnění je ukončeno kódem 4(„End of Transmission“). Po naplnění zásobníku je zásobník předán třídě Packet pro zpracování.

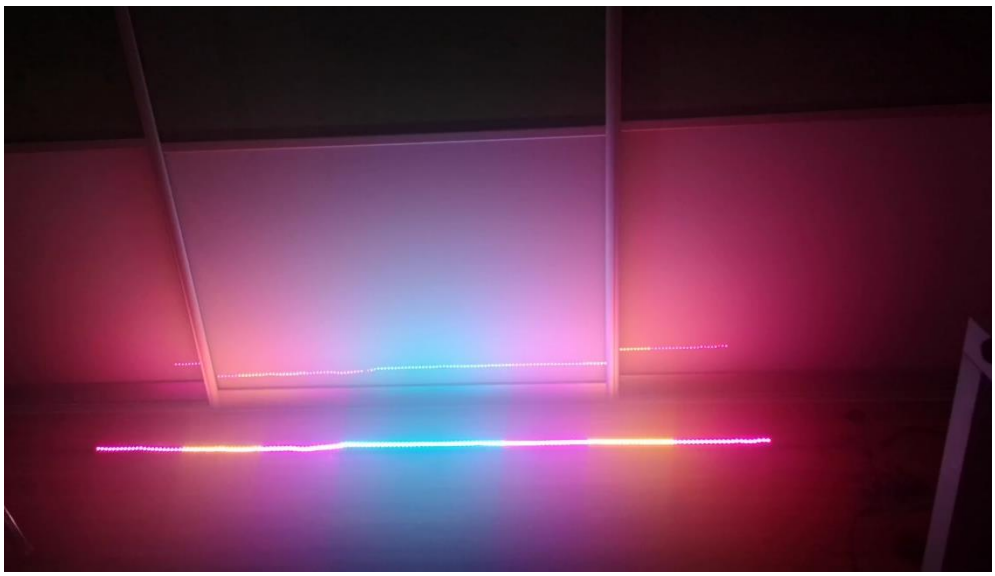
Zpracovaný paket je předán funkci *executeCommand* která vyhodnotí hlavičku paketu. Při prvním spuštění musí nejdříve přijít ověřovací paket který obsahuje ověřovací kód. Po ověření přijde paket pro připojení k Wi-Fi síti.

Připojení k síti vyhodnocuje metoda *connectToWifi* která rozdělí tělo paketu na dvě části. První část těla obsahuje SSID Wi-Fi sítě a druhá část heslo. SSID a heslo je předáno funkci *wifiSetup*.

### 4.2.3. Analýza zvuku

Signál z obvodu pro automatické řízení hlasitosti je zpracováván rychlou Fourierovou transformací. Pro Fourierovu transformaci byla použita vzhledem ke své jednoduchosti knihovna *arduinoFFT*. Pro transformaci je důležitá konstantní frekvence načítání vzorků počet vzorků. Frekvence načítání vzorků musí být dvojnásobná, nežli je zjišťovaná frekvence. Jelikož má většina tónů v hudebních nahrávkách frekvenci od 30Hz do 10kHz byla vzorkovací frekvence nastavena na 20kHz. Po testování byl počet vzorků nastaven na 512 vzorků, načtení tohoto počtu vzorků bylo dostatečně rychlé na to, aby uživatel nezpozoroval zpoždění oproti zvuku.

V hlavní programové smyčce se načtou všechny vzorky a následně se odstraní stejnosměrná složka signálu ze vzorků. Následně se rychlou Fourierovou transformací vypočítají výsledky, které se uloží do pole *vReal*. Metoda *MajorPeak* nalezne nejvíce zastoupenou frekvenci a jeho sílu předá metodě *setMusicPeak*. Metoda přepočítá sílu přes logaritmus, jelikož se síly signálu liší se zvyšující frekvencí a vynásobí tisícem. Od této hodnoty je odečítána konstanta 3900 která byla určena měřeními při vypnuté hudbě, tím se zamezí nechtěnému blikání.



*Obr. 10 rozsvícené LED diody při audivizualizaci*

### 4.2.4. Budíky

Jeden z úkolů maturitní práce bylo vytvoření sytému budíků. Sytém kdy uživatel nastaví čas, ve který se rozsvítí světlo zvoleným způsobem. Pro funkčnost tohoto konceptu musí být zařízení připojené na internet, odkud získává aktuální čas.

Pro uložení budíků bylo potřeba vytvořit strukturu budíku v tomto případě třídu *Alarm,* která obsahuje informace o čase, aktivovaných dnech, způsobu rozsvícení, času, po kterém se světlo vypne a interním identifikátoru. Pro všechny tyto informace byl použit 8bitový unsigned integer vzhledem k tomu, že žádná z hodnot nepřesáhne hodnotu vyšší než 255. Pro informaci se dny, ve kterých se má budík spustit byl použit 8bitový integer také. Každý bit reprezentuje jeden den a o jeho stavu rozhoduje jeho hodnota. Pro hodnotu jedna je budík aktivován a pro hodnotu nula je budík deaktivovaný. Tento způsob uchovávání informace byl zvolen, protože předávání polí mezi metodami není vždy komfortní a toto řešení předává pouze jednu hodnotu.

Pro automatickou synchronizaci času byla použita knihovna *NTPClient*. Při spuštění zařízení se klient spustí a nastaví se časový posun. V sekundární programové smyčce je čas aktualizován a upraven formát dnů (knihovna začíná týden v neděli). Následně se otestuje, zda je v daný den nějaký z budíků aktivovaný. Bitová hodnota pro daný den se získává bitovým posunem a následným vymaskováním.

### 4.2.5. Paměť zařízení a ukládání dat

V případě že je mikrokontroler bez napájení ztrácí všechna data uložená v paměti a je proto nutno je uchovávat. Na mikrokontrolerech jsou dvě možnosti, jakými lze data uchovávat: EEPROM a SPIFFS. Nevýhodou EEPROM je její omezená velikost a ta je pouze 512 bytů a její životnost je zhruba sto tisíc až milion zápisů. SPIFFS má delší životnost a lze s jeho pomocí ukládat soubory. Vzhledem k jednoduchosti uložených dat jsou všechna data uložena v souboru *config.txt* ve stejném formátu jako jsou zpracovávána data při komunikaci (viz. kapitola 4.2.3).

O ukládání a načítání se starají čtyři funkce z nichž jedna inicializuje SPIFFS systém a druhá zjišťuje, jestli je v zařízení *config.txt* soubor.

Funkce *writeToConfigFile* umožňuje zápis nové konfigurace do systému. Nejdříve se načte celý config soubor do bufferu, následně se zapíše nová konfigurace a na konec se zapíše původní stav konfigurace. Tento systém není moc efektivní, ale jelikož zápis konfigurace neprobíhá tak často není třeba tento systém zápisu upravovat.

Funkce *readConfig* pročítá celý konfigurační soubor a získává jednotlivá data. Následně porovná hlavičku paketu s flagem a pokud jsou identické vrátí data z těla. Tato metoda vrací vždy první získaná data, to znamená že pokud je nějaká položka duplicitní použije se ta nejnovější.

# 5. Závěr

Cílem mé maturitní práce bylo vytvoření zařízení a mobilní aplikace pro ovládání adresovatelných LED diod. Výsledkem je mobilní aplikace a zařízení určené pro ovládání LED diod a analýzu zvuku. Mobilní aplikace umožňuje uživateli nastavit různé druhy efektů zobrazovaných, jejich barvu a jas. Grafika aplikace byla vyvíjena pro jednoduché použití a co nejmenší komplikovanost z uživatelské strany. Pomocí aplikace lze také nastavovat budíky. Zařízení je schopné zpracovávat data přijatá z mobilní aplikace a reagovat na ně. Lze na něm také spustit vizualizaci hudby která funguje za pomocí rychlé Fourierovy transformace.

Ze zadání nebylo splněno ovládání hlasem pomocí Google asistenta a změna jasu podle okolního osvětlení. Ovládání hlasovým asistentem jsem se rozhodl nezabývat po prozkoumání možností jakými lze asistenta využívat. Přímo do mobilní aplikace lze integrovat propojení s asistentem, avšak jen pomocí omezeného množství příkazů, které nebyly uživatelsky přívětivé, a tak by jejich využití bylo pro uživatele komplikované. Změna jasu světla podle okolního osvětlení by nebyla při reálném použití efektivní a pro uživatele je mnohem snazší změnit jas manuálně. Zapojení obsahuje integrované čipy MSGEQ7, které byly použity při vývoji, ale nebyly použity ve finálním zařízení. Vzhledem k nedokonalosti analyzování zvuku pomocí Fourierovy transformace by tyto čipy mohly sloužit jako ověřovací a pro nízké frekvence které Fourierova transformace nedokáže efektivně analyzovat.

Celý systém by se dal vylepšit například TCP komunikací která na rozdíl od UDP zaručuje doručení dat do zařízení nebo vyvinutím serveru na kterém by byla uložena všechna data o zařízeních a bylo by tak uživateli umožněno světla ovládat mimo lokální síť.

# 6.Zdroje


- https://www.electroschematics.com/avc-automatic-volume-control/ [cit. 26. února 2020]
- https://www.alldatasheet.com/view.jsp?Searchword=Tl082&gclid=Cj0KCQjwy6T1BRDXARIsAIqCTXpQrXXcG4w9ZB3oyc_e5Ez2hsDl45eVCmp-mbqP-Em__344u04liNEaAiRaEALw_wcB [cit. 10.ledna 2020]
- https://www.sparkfun.com/datasheets/Components/General/MSGEQ7.pdf [cit. 26. února 2020]
- https://material.io/develop/android/components/menu/ [cit. 17. dubna 2020]
- https://en.wikipedia.org/wiki/ASCII [cit. 26.října 2019]
- https://cs.wikipedia.org/wiki/Fourierova_transformace [cit. 28. února 2020]
- https://cs.wikipedia.org/wiki/Rychl%C3%A1_Fourierova_transformace [cit. 28. února 2020]
- https://cs.wikipedia.org/wiki/Diskr%C3%A9tn%C3%AD_kosinov%C3%A1_transformace [cit. 29. února 2020]
- https://randomnerdtutorials.com/esp32-dual-core-arduino-ide/ [cit. 8. března 2020]
- https://randomnerdtutorials.com/esp32-ntp-client-date-time-arduino-ide/ [cit. 29. března 2020]
- https://github.com/espressif/arduino-esp32/tree/master/libraries/AsyncUDP [cit. 28. prosince 2019]
- https://developer.android.com/reference/java/net/DatagramSocket [cit. 29. prosince 2019]
- http://www.robinscheibler.org/2017/12/12/esp32-fft.html [cit. 2. března 2020]
- https://www.norwegiancreations.com/2017/08/what-is-fft-and-how-can-you-implement-it-on-an-arduino/ [cit. 2. března 2020]
- https://github.com/FastLED/FastLED/wiki/Pixel-reference [cit. 4. února 2020]

https://techtutorialsx.com/2018/08/05/esp32-arduino-spiffs-writing-a-file/ [cit. 14. dubna 2020]

https://www.raywenderlich.com/155-android-listview-tutorial-with-kotlin [cit. 15. ledna 2020]

https://developer.android.com/reference/android/content/BroadcastReceiver [cit. 4. ledna 2020]

https://material.io/ [cit. 26. února 2020]

# 7. Seznam obrázků