Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Fejes Péter

AUDIOVIZUALIZÁCIÓS ESZKÖZ TERVEZÉSE

Szakdolgozat

KONZULENS

Dr. Györke Péter

BUDAPEST, 2017

Tartalomjegyzék

Összefoglaló **5**
Abstract **6**
1 Bevezetés **7**
2 Irodalomkutatás **9**
2.1 Létező megoldások 9
2.1.1 Szoftveres megoldások 9
2.1.2 JBL Pulse (Bluetooth hangszóró) 10
2.1.3 RGB LED vezérlők 11
2.2 Használt technológiák 12
2.2.1 Digitális jelfeldolgozás, DSP (Digital Signal Processing) 12
2.2.2 Fourier transzformáció 14
3 Tervezés **16**
3.1.1 Rendszerterv 16
3.1.2 Konfiguráció 20
4 Megvalósítás **22**
4.1 Eszközválasztás 22
4.1.1 Mikrokontroller 22
4.1.2 Megjelenítő 24
4.1.3 Bluetooth audio receiver 27
4.1.4 Jelkondícionálás 28
4.1.5 Módosult blokkdiagram 29
4.2 A vezérlő programja 30
4.2.1 Használt programok, könyvtárak 30
4.2.2 A beágyazott szoftver 32
4.2.3 Időzítés 41
4.3 Konfiguráló program 42
5 Tesztelés **44**
5.1 TrueSTUDIO fejlesztőkörnyezet hibakeresője 44
5.2 Soros porton keresztüli logolás 45
5.3 Oszcilloszkóp használata 45
6 Továbbfejlesztési lehetőségek, összegzés **46**

Irodalomjegyzék **.. 47**

HALLGATÓI NYILATKOZAT

Alulírott **Fejes Péter**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2017. 12. 08.

...…….
Fejes Péter

Összefoglaló

A zene ütemére mozgó kijelzők és fények közkedvelt látványelemek napjainkban. Szakdolgozatom célja egy ilyen audió vizualizációs eszköz tervezésének és megvalósításának bemutatása mely valós idejű vizualizációt képes előállítani lejátszott audió tartalom alapján.

A megtervezett eszköz egy különálló, az audió rendszerektől független működésre képes megjelenítő egység, mely lejátszókhoz csatlakoztatható.

A megvalósítás alapjául egy mikrokontroller szolgál, mely egy címezhető LED panelt vezérel. A kontroller valós idejű jelanalízist végez egy bemenő audió jelet és ez alapján végzi a megjelenítést.

A dolgozatomban bemutatom ezen eszköz megtervezésének lépéseit, majd a tényleges fizikai megvalósítás folyamatát és a folyamat közbeni nehézségeket.

A projekt során kitűzött célokat elértem. Sikerült megterveznem és megvalósítanom is egy a saját elvárásaim, igényeim alapján működő eszközt. A kezdeti elvárásokhoz tartani tudtam magam. A munka elvégzése során sok új technológiával találkoztam, szélesedett a látóköröm és új technológiákat ismertem meg.

Kulcs szavak: mikrokontroller, audió, digitális jelfeldolgozás, vizualizáció, FFT

Abstract

Illustrational displays and lights, pulsing to the rhythm of music are quite popular visual elements today. My dissertation focuses on creating a design for such an audio-visual tool, and – based on this design – creating the device itself. The goal was to create real-time visual effects based on the played audio.

The device had to be a separate, independent visualizer tool, which can be connected to various audio players.

The project was based on a microcontroller, which operates a programmable LED panel. The controller processes the input audio signal with real time signal-analysis, and based on this creates a visualisation.

In my dissertation I go through the various steps of the design creation, then I present a detailed description of the assembly of the device itself, highlighting the various complications, I had to overcome.

During the project, I managed to reach every goal which I determined at the beginning. I successfully designed and created a device to meet my requirements, which were also described at the beginning of the project. I also managed to learn much about various new technologies, and I could widen my knowledge this way.

Keywords: microcontroller, audio, digital signal processing, visualisation, FFT

1 Bevezetés

Az analóg zenevilág fénykorában gyakran találkozhattunk a lejátszó eszközökön úgynevezett kivezérlésjelzőkkel, melyek vizuálisan jelenítették meg a lejátszott tartalom erősségét/teljesítményét, tipikusan a jobb és bal hangsáv alapján, ritkább esetben frekvencia alapján több sávra felbontva a jelet. Ezen kijelzők akkoriban inkább a drágább, jobb minőségű termékeken voltak megtalálhatóak és nem a vizuális élményt voltak hivatottak növelni, hanem arra szolgáltak, hogy megakadályozzák a csatornák tartós túlvezérlését, vagy az eszköz hangzásának beállításában segítsenek.

A zene ütemére mozgó kijelzők, mutatók és fények hamar közkedveltté váltak, mint látványelemek. A technika fejlődésével és a digitális eszközök elterjedésével olyan védelmi eszközök kerültek bele a lejátszókba melyek használatával már nem kell attól tartani a felhasználónak, hogy tönkreteheti kedvenc hi-fi berendezését, ha túl hangosan hallgat rajta zenét. Megjelent az automatikus hangszínszabályozás is, mint funkció. Ezeknek köszönhetően a kivezérlésjelzők elvesztették eredeti funkciójukat és szép lassan eltűntek az eszközökről.

Szakdolgozatom célja egy olyan audió vizualizációs eszköz tervezésének és megvalósításának bemutatása mely modern technológiákon alapulva valósít meg egy ilyen kijelzőt, immár nem más céllal, minthogy vizuális élményt szolgáltasson.

Ilyen eszközökre napjainkban komoly igény mutatkozik, például egyre több olyan hangfallal találkozni a boltokban, melyek a lejátszott zene ütemére villognak. Az ilyen hangfalak azonban egyáltalán nem konfigurálhatóak, nincsenek különböző hangsávok és a vizualizáció csak mellékes, a termékek fő profilja még midig a zene lejátszása. Ezzel szemben az én célom egy olyan eszköz készítése mely saját megjelenítővel rendelkezik, csak köztes megfigyelőként csatlakozik a már meglévő audió rendszerekhez és kiegészítésképpen nyújt vizuális szórakoztatást melléjük, anélkül hogy azok működését megváltoztatná, gátolná. Ennek megfelelően a kiegészített rendszer lehet egy telefon, erősítő, televízió, házimozi rendszer vagy bármilyen audió tartalom lejátszására szolgáló eszköz.

Dolgozatom további fejezeteiben részletesen bemutatom az általam megvalósított eszköz működését, megtervezésének lépéseit, nehézségeit, hogy milyen

technológiákat használ, és hogy mivel nyújt többet, mint a már jelenleg kapható hasonló audió vizualizációs eszközök.

2 Irodalomkutatás

2.1 Létező megoldások

Ezen fejezetben bemutatom, hogy milyen, a témához kapcsolódó termékek, eszközök léteznek. A felsorolt termékek közül mindegyik rendelkezik valamilyen olyan tulajdonsággal, amely inspirációként szolgált és később elvárásként jelent meg az én projektemben. Azonban önmagában egyik sem képes ezeket az elvárásokat hiánytalanul teljesíteni.

2.1.1 Szoftveres megoldások

Sok letölthető program létezik, mely látványvilágban nagyon közelít az általam megvalósítani kívánt eszközhöz, ezek azonban nem különálló fizikai termékek, hanem csak számítógépen használható programok, melyek monitoron jelenítik meg a kívánt látványvilágot. Az én célom egy önmagában használható eszköz elkészítése volt mely mozgatható és csatlakoztatható bármilyen audió eszközhöz. Az elképzelt látványvilág viszont nagyban hasonlít e szoftverekéhez így ezek megfelelő példát mutattak, a cél ezeknek fizikai közegbe való kiemelése volt.

A leginkább ide passzoló ilyen szoftver a Winamp [1] nevű multimédiás lejátszó, mely a 2000-es évek elején elterjedt program volt számítógépes médiatartalom lejátszására. Az alap verzió nem tartalmazott audió vizualizációt, viszont az 1.9 verzió után a program bővíthetővé vált, és ezután megjelentek hozzá vizualizációs pluginek is.

1. ábra: Winamp média lejátszó

Az ehhez hasonló lejátszókon kívül léteznek még nem valós időben működő vizualizáló programok is, tehát nem a lejátszás közbeni szórakoztatásra szolgálnak, hanem főleg zenei videók hátttereként használt animációkat szoktak készíteni velük. Funkcióikban e programok szinte egyáltalán nem teljesítik az én elvárásaimat, pusztán látványviláguk egyezik meg vele.

2.1.2 JBL Pulse (Bluetooth[1] hangszóró)

A bevezetőben említettem, hogy egyre gyakoribbak az olyan hangszórók, melyekbe a gyártó épít valamilyen vizualizációs eszközt. Az én projektemhez legjobban hasonlító ilyen eszköz a JBL Pluse nevezetű Bluetooth hangszóró. Ennél a hangfalnál egy telefonos program segítségével beállítható milyen stílusú legyen a vizualizáció. A termék legnagyobb hátránya az, hogy az ilyen hangfalak fő profilja az audió tartalom lejátszása, és a látványos megjelenítés csak mellékesen társul e mellé. Ha a felhasználó már rendelkezik valamilyen lejátszóval otthon és mellé szeretne látványelemként ilyen hangszórót vásárolni, akkor a már meglévő hangszóróját nem tudja használni, fölöslegesen kell, egyébként nagy összeget kifizetni egy ilyen eszközért. Emellett egy ilyen hangszórónak nem lehet olyan jó hangminősége, mint egy házimozi rendszernek, vagy hifi berendezésnek, ezek mellé viszont másodlagosan nem lehet csatlakoztatni önmagában a látvány elérésének érdekében. Másik nagy hátránya, hogy sok ilyen eszköz nincs ellátva vezetékes csatlakozóval, így nem Bluetooth kompatibilis eszközökkel nem használható.

2. ábra JBL Pulse

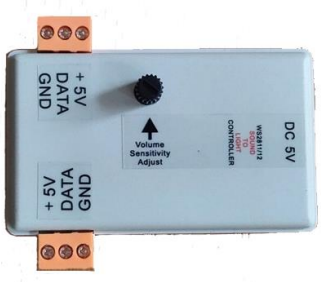
[1] A Bluetooth egy rövid hatótávolságú, adatcseréhez és kommunikációhoz használt, nyílt, vezetéknélküli szabvány.

A modularitása viszont jó alapként szolgál az általam megvalósítani kívánt projekthez. Egy hasonló eszközt szerettem volna elkészíteni, csak kihagyva belőle a hangszóró funkciót, ezzel jóval olcsóbbá, és több eszközzel használhatóvá téve.

2.1.3 RGB LED vezérlők

Léteznek készen vásárolható LED vezérlők is. Ezek között vannak olyanok, melyek RGB kimenettel rendelkeznek és képesek a zene ütemére egy LED szalagot vezérelni, de ezek nem képesek a LED-ek színét egyesével változtatni, tehát egy összetettebb vizualizációra alkalmatlanok. Sokkal kevésbé elterjedtek, de lehet találni olyan kész vezérlőket is, melyek címezhető LED-eket tudnak meghajtani. Ezek elméletben már képesek bonyolultabb vizualizációra is. Viszont ezek a vezérlők nem tartalmazzák magát a LED panelt, azt külön kell megvásárolni hozzájuk, és összeszerelni velük.

Mégsem ez a legnagyobb probléma ezekkel. Az eszközök általában távol keleti gyártmányúak, alacsony minőséggel. Dokumentáltságuk minimális. Legtöbbször nem tartozik hozzájuk használati utasítás, és működésükről szóló leírást, specifikációt szinte sosem mellékelnek. A vásárló megrendeléskor nem tudja, hogy ténylegesen milyen működésű eszközt fog kapni. Ezen kívül ezek a kész vezérlők nem konfigurálhatók vagy átalakíthatók, ha esetleg erre lenne szükség. Általában nehezen bővíthetők vagy fejleszthetők. Ez azért probléma, mert a csatlakozási lehetőségek általában nagyon limitáltak. Ha képesek vezeték nélküli csatlakozásra, akkor szinte soha nem implementálják e mellé a vezetékes csatlakozást. Sokszor még önmagában a vezetékes csatlakozás is hiányzik és az eszköz csak mikrofonnal képes a jelet fogadni. Ezekhez a hátrányokhoz viszonyítva azonban az áruk túl magas.



3. ábra LED vezérlő

Hosszas keresés után találtam egy olyan modellt mely vezetékes csatlakozásra képes, és rendelkezik olyan kimenettel melyhez programozható LED kapcsolható. Ez az eszköz a 3. ábraán látható. Azonban ehhez a termékhez sem tartozik semmilyen dokumentáció, vagy adat arról, hogy milyen vizualizációt fog megjeleníteni, és hogy egyáltalán milyen elrendezésben csatlakoztassuk hozzá a LED-eket. Ára ennek ellenére körülbelül 30 ezer forint.

Projektem során ezek miatt a bizonytalanságok miatt én magam fogok egy ilyen LED vezérlőt elkészíteni. A saját megvalósítás miatt működése teljes mértékben az én elvárásaimhoz igazítható és kiszámítható lesz.

2.2 Használt technológiák

Ebben a fejezetben azon technológiákat és folyamatokat fogom ismertetni, melyek már a projekt kezdetekor felmerültek, mint megoldandó feladat. Ezek ismerete a projekt elkezdéséhez és megértéséhez elengedhetetlen.

2.2.1 Digitális jelfeldolgozás, DSP (Digital Signal Processing)

A leghamarabb felmerülő kérdés az, hogy hogyan is lehet a zene ütemére vezérelni valamit, hogyan tudjuk a zenei tartalom karakterisztikáját megfigyelni.

Az analóg audió jelre általánosságban úgy tekinthetünk, mint egy időben változó feszültségérték, mely a hangot reprezentálja.[2] A jel értéktartománya ugyan általában korlátos, azonban mivel analóg jelről van szó, így értékkészlete végtelen. A számítógépekkel való feldolgozáshoz ezt a jelet véges bitszámon ábrázolható számsorozattá kell alakítanunk. Ezt az átalakítást hívjuk analóg-digitális jelátalakításnak (Analog-Digital Conversation, ADC).

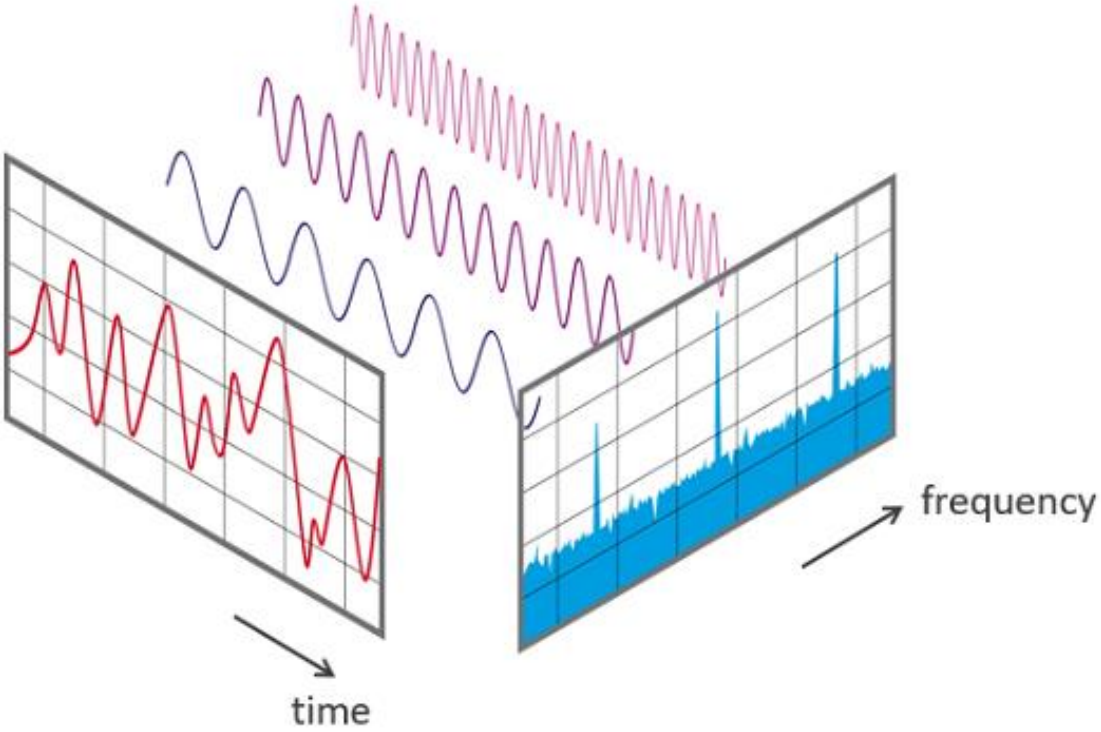
Ez az átalakítás általában két lépésből áll. A mintavételezésből és a kvantálásból. A mintavételezés során az analóg jelet megfigyeljük, és adott, állandó időközönként megmérjük annak értékét. Ezt az állandó időközt nevezzük a mintavételi időnek, ennek reciproka a mintavételi frekvencia. Ezek a mért értékek még továbbra is analóg értékek, viszont időben nem folytonos értéksort alkotnak. A kvantálás során a mintavételezett jelek korlátlan értékkészletét a kimeneti értékkészletre szűkítjük. Ez úgy történik, hogy a mintavételezett jel értékkészlet tartományát sávokra bontjuk, annyi sávra ahány elem van a kimeneti értékkészletben. Ezután ezekhez a sávokhoz, és az azokban lévő értékekhez hozzárendeljük a kimeneti készlet egy elemét.

Fontos megemlíteni a Shannon mintavételi törvényt, mely kimondja, hogy egy folytonos idejű jel tökéletesen visszaállítható a mintavételezési eljárás során keletkezett mintáiból, ha a mintavételezési frekvencia az analóg jel sávszélességének legalább kétszerese. Ez azért fontos számunkra, mert a hallható hang frekvenciatartománya körülbelül 20 Hz és 22 KHz között van, tehát ha legalább 44 kHz-el mintavételezünk, abban az esetben ez az eljárás nem okoz veszteséget.

Ez után az eljárás után egy a digitális adatsort kapunk, amelyet megfelelő programmal feldolgozhatunk. Ez a jel az úgynevezett időtartománybeli jel. Ez az jelenti, hogy a mintavételi időpontokban megadja a jel amplitúdóját. Ez önmagában kevés információt hordoz a látványos megjelenítéshez, habár a zene üteme már ebből is megfigyelhető.

A Fourier tétel alapján tudjuk, hogy az időtartománybeli hullámalakot le lehet írni szinusz- és koszinusz függvények súlyozott összegeként [3]. Ugyanazt a hullámalakot megjeleníthetjük frekvenciatartományban is, mint az egyes frekvencia-összetevők amplitúdóját és fázisát. Előállíthatunk bármely hullámalakot szinusz hullámok összegeként, ahol az egyes szinuszos tagok önálló amplitúdó- és fázisértékkel rendelkeznek. Audió jel esetén a frekvenciatartomány e módon való felbontásával kaphatjuk meg a hangsávokat[2].

2 A tipikus frekvenciafelosztás a következő: 20-60Hz a szub-basszus, 60-150Hz a basszus, 150Hz-500Hz: alsó-közép tartomány, 500Hz-2kHz középtartomány, 2-4kHz a felső közép tartomány,4-6kHz: Precence, 6-20kHz: Brilliance



4. ábra Idő- és frekvenciatartomány közötti összefüggés

Egy jel frekvenciatartományba való transzformálásához használható a Fourier transzformáció.

2.2.2 Fourier transzformáció

A Fourier transzformáció elméletének bemutatása annak komplexitása és mélyebb matematikai tartalma miatt nem célja a szakdolgozatnak. A transzformáció elvégzéséhez rendelkezésre állnak kész algoritmusok, továbbiakban ezen algoritmusok tulajdonságait ismertetem.

Az algoritmust, amelyet arra használunk, hogy időtartománybeli mintavételezett jel értékeket frekvencia tartományba transzformáljunk, Diszkrét Fourier transzformációnak (DFT) nevezzük. Ezen algoritmus futtatása során, ha rendelkezésünkre áll egy N mintából álló adatcsomag egy mérésadatgyűjtő berendezésből (esetünkben az analóg digitális konverter kimenete), akkor a diszkrét Fourier transzformációt erre az időtartománybeli N mintára futtatva, az eredmény szintén N értéket fog tartalmazni és a jel frekvencia tartománybeli tulajdonságait jeleníti meg. [4] A diszkrét Fourier transzformáció bonyolultsága $O(N^2)$, azaz N hosszúságú bemenet esetén N^2 lépést hajt végre. Ez az érték a

számítástechnikában nagyon nagy költségnek számít, ezért kifejlesztették a Gyors Fourier transzformációt (FFT).

Az FFT algoritmus az „oszd meg és uralkodj” elvet követi, azaz rekurzív módon feldarabolja a bemeneti értékeket és annak részhalmazain alkalmazza a DFT algoritmust, majd e részértékekből számítódik ki az eredeti egész halmaz Fourier transzformált értéke. Ezen algoritmus N mintaszám mellett $N log_2 N$ számítást végez, mely sokkal kedvezőbb, mint a DFT transzformáció. Digitális jelanalízis során ezt az algoritmust használják elterjedten.

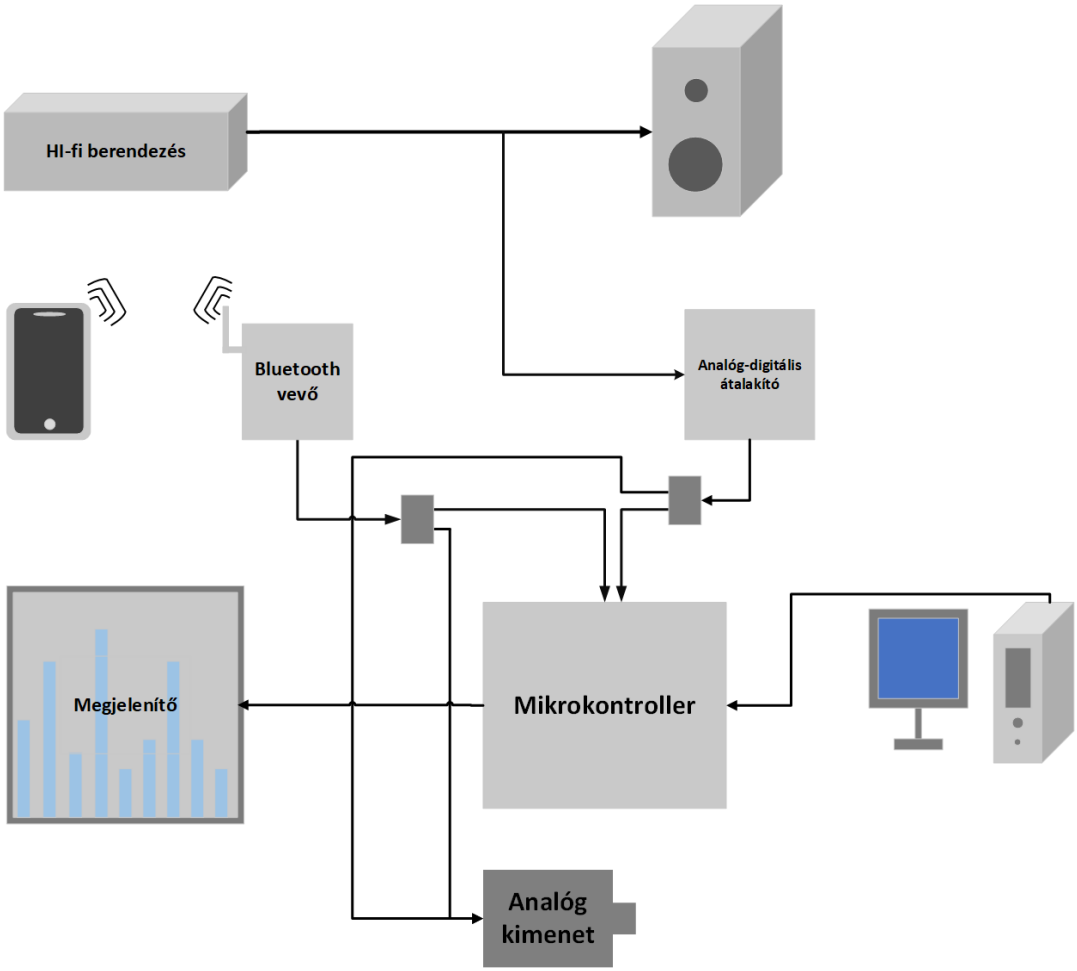
3 Tervezés

Ezen fejezetben a már felvázolt eszköz részletes tervezésének folyamatát fogom bemutatni. Az eszköz megtervezése közben fontos szempont volt, hogy egy olyan rendszer készüljön el, mely korábban ismertetett már létező megoldások és eszközök hiányosságait pótolni tudja.

A legalapvetőbb elvárás, hogy képes legyen a lejátszott zene ütemére valamilyen vizualizációt végrehajtani, melyhez rendelkeznie kell egy megjelenítővel. Fontos szempont volt ezen kívül, hogy önálló, hordozható eszköz legyen, melynek működéséhez nem szükséges semmi, csak a vizualizálni kívánt audió tartalom, és tápellátás. Cél volt továbbá, hogy minél több eszközzel legyen kompatibilis, és hogy konfigurálható legyen a működése, előállítási költsége ne legyen magas.

3.1.1 Rendszerterv

A fentebb említett követelményeket figyelembe véve összeállt egy alap architektúra. Az eszköznek legyen egy vezérlő, és egy megjelenítő egysége. A vezérlő egység bemenetként fogadja az audió tartalmat, valós idejű jelanalízist végez azon, és ez alapján vezérli a megjelenítő egységet.



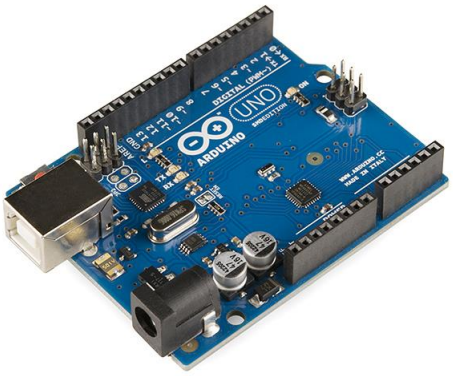
5. ábra a rendszer blokkdiagramja

A vezérlő egységnek kis méretűnek kell lennie, de mégis komplex funkciókat kell ellátnia, relatív nagy számítási teljesítménnyel. Ezen indokok miatt úgy döntöttem, hogy egy felső kategóriás mikrokontroller fogja e feladatokat ellátni.

A mikrokontrollerek vagy mikrokontroller kártyák olyan kisméretű beágyazott eszközök, melyeknek minden egysége egyetlen áramköri lapkába van integrálva[5]. Ezen eszközök tartalmaznak egy vagy több processzort, a szükséges rendszermemóriát, és programozható be- és kimeneti periféria csatlakozókat, és a programot tartalmazó tárhelyet, továbbá számos egyéb perifériát.

Ezen kontrollerek, felprogramozás után képesek a rájuk töltött program önálló futtatására, csak áramellátás szükséges hozzájuk. Számítási kapacitásban ugyan elmaradnak a mai számítógépektől, de számos komplex probléma megoldására kitűnően alkalmasak. A projektemhez szükséges számításokat el tudják valós időben végezni, a sok periféria illesztőnek köszönhetően pedig, mind a többféle bemenet kezelése, mind pedig a megjelenítő panel vezérlése megoldható. Ezen kívül manapság már az áruk sem

túl magas, elég sok gyártó fejleszt ilyen termékeket, sok kiegészítő vásárolható hozzájuk és működésük megfelelően dokumentált. Az alábbi képen egy mikrokontrolleres fejlesztőkártyája látható.



6. ábra Arduino mikrokontrolleres fejlesztőkártya

A rendszer másik fontos pontja a megjelenítő. A projekt során a hagyományos kijelző használatát elvetettem, mivel ezek közül a színes változatok meglehetősen drágák, főleg az akkora méretűek, amelyeken látványosan megjelenítést lehetne elérni. Ezen kívül a képalkotás is bonyolult rajtuk a relatív magas felbontásuk miatt. Audió vizualizációhoz nincs is szükség nagy felbontású kijelzőre. Ha a hagyományos elrendezést nézzük, melyben frekvenciasávokat rendelünk megjelenítendő oszlophoz, könnyen látható hogy nincs feltétlenül szükség 15-20 oszlopnál többre a megjelenítőn, és az oszlopok magasságának sem kell túl nagynak lenniük a látványos megjelenéshez. Ezek miatt úgy döntöttem, hogy egy RGB LED-ekből álló panelt fogok használni, körülbelül 20x20 LED-es méretben. Mivel nem valódi képalkotásra használom, nem szükséges az egyes képpontoknak közel lenniük egymáshoz, így kis felbontás mellett is nagyméretű lehet a kijelző, mely nagyobb vizuális élményt nyújthat.

A kompatibilitás biztosításához szükség van az elterjedt csatlakozási módok biztosítására az általam tervezett rendszerben is. A leggyakoribb mód még napjainkban is a vezetékes csatlakozás, szabványos 3.5mm-es Jack-csatlakozók[3] használatával. A

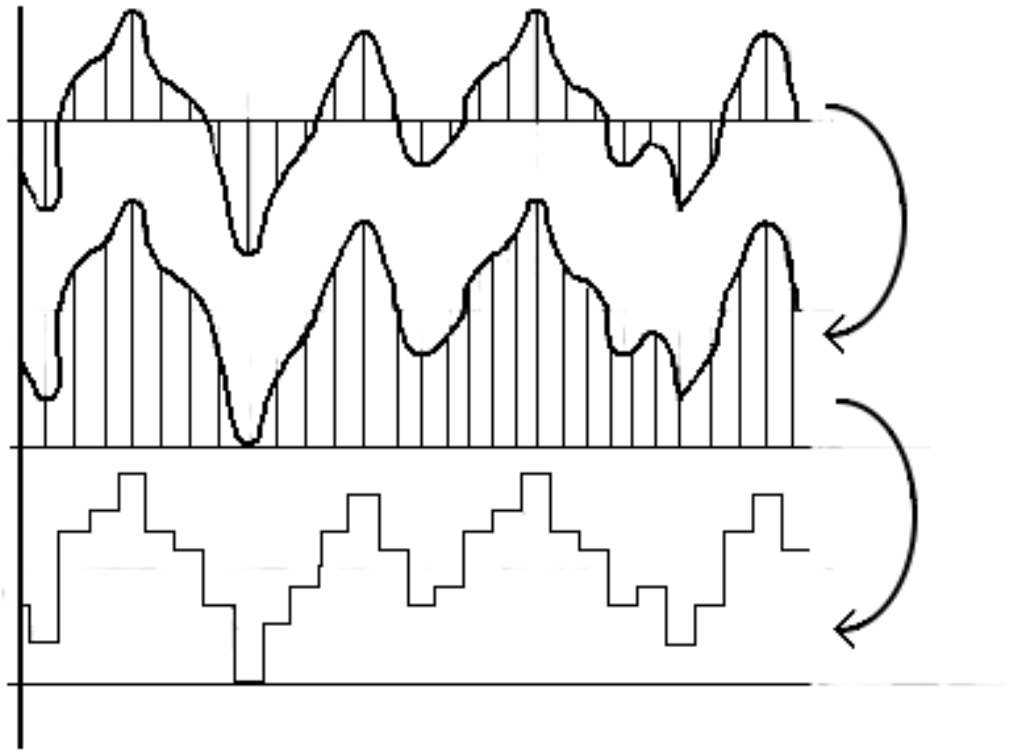
[3] A „Jack” csatlakozó kifejezés, idegen néven phone connector vagy phone jack, egy elekronikai csatlakozó családot jelöl, melyet tipikusan analóg audió jel továbbításakor használnak.

második legelterjedtebb mód a Bluetooth alapú vezeték nélküli csatlakozás. A projektem során ezt a két csatlakozási módot fogom megvalósítani, ez elég széles körű kompatibilitási lehetőséget biztosít.

Első probléma, hogy az eszközökön található vezetékes Jack kimenetek által szolgáltatott jel analóg. Az informatikában viszont nem tudunk folytonos jelekkel dolgozni azok végtelen értéktartománya miatt. Ahhoz tehát hogy jelanalízist tudjon végezni a mikrokontroller, digitális jelre van szükségünk. Erről részletesen a 2.2.1: Irodalomkutatás című fejezetben írtam. Ennek következménye, hogy egy köztes analóg-digitális jelátalakítót kell beiktatni a mikrokontroller elé.

Szintén problémát jelent, hogy az audió jel nem tisztán pozitív értékű jel, hanem a nulla, vagyis a föld értéktől való pozitív és negatív irányba való feszültség eltérést is tartalmaz. Ezt a jelet még az A/D átalakítás előtt pozitív tartományba kell transzformálni.

A folyamatot a 7. ábra szemlélteti, melyen a legfelső jel az eredeti bejövő analóg jel, a középső jel pedig ennek pozitív tartományba transzformált változata. Ezután megtörténik az analóg-digitális jelátalakítás, melynek végeredménye a legalsó sorban látható digitális jel.



7. ábra Analóg jel átalakítása

Bluetooth kapcsolat esetén ez analóg-digitális átalakítás folyamata elkerülhető, mivel kapható olyan Bluetooth modul, mely egyből digitális PCM (Pulse Code Modulation) jelet biztosít.

Elvárás, hogy a vizualizáló eszköz csatlakoztatása ne befolyásolja az eredeti rendszer működését. Ez vezetékes csatlakozás esetén egy hi-fi vagy egy házimozi berendezésnél nem jelent problémát, a vizualizáló eszközt lehetséges a berendezések fejhallgató kimenetéhez csatlakoztatni, ezek az eszközök általában képesek párhuzamosan jelet biztosítani, minőségromlás nélkül, a saját hangszórójukon és fülhallgató kimenetükön egyaránt. Vannak eszközök melyekhez, ha fülhallgató kimenetükre csatlakoztatunk más eszközt, akkor eredeti hangszórójukon nem folytatják az audió tartalom lejátszását. Tipikusan ilyenek a televíziók. Azonban ennek legtöbbször nem fizikai akadályai vannak, csak a gyártók feltételezik, hogy egyszerre csak az egyik kimenetet szeretné a felhasználó használni és legtöbbször ez a funkció kikapcsolható. Amennyiben ezek közül egyik megoldásra sem alkalmas az eszköz, arra az esetre egy belső audió jel elosztót építek a vizualizáló rendszerbe, melynek egyik fele a vizualizációhoz szolgál bemenetként, a másik fele pedig kimenetként jelenik meg az eszközön. Ezzel elérhető olyan működés, hogy a lejátszót vezetékkel csatlakoztatjuk a vizualizálóhoz, majd ennek hangkimenetéhez csatlakoztatjuk a hangfalat. Ez a megoldás már hangminőség vesztéssel jár a jel kettéosztása miatt, viszont ha nincs más megoldás, akkor is használhatóvá teszi az eszközt. Vezeték nélküli csatlakozásnál szintén problémás a párhuzamos lejátszás és a vizualizálóhoz való vezeték nélküli kapcsolódás, ugyanis ilyenkor az eszközök szintén általában csak a vezeték nélküli csatornán bocsájtanak ki jelet. Amennyiben ezt beállítással nem tudjuk kiküszöbölni az eszközön, abban az esetben szintén használható a vizualizáló eszköz audió kimenete.

3.1.2 Konfiguráció

Fontos eleme a rendszernek a felhasználó általi konfigurációs lehetőségek biztosítása. Az elvárt beállítások között szerepel, hogy a felhasználó kiválasztja a csatlakozási módot, konfigurálni tudja a megjelenítő stílusát, azaz hogy hogyan történjen a tartalom frekvenciasávjainak hozzárendelése a megjelenítőhöz, és hogy beállíthassa a használt színeket, fényerőt.

Két fő választási lehetőség merült fel a tervezés során. Az egyik a rendszer ellátása megfelelő mennyiségű nyomógombbal ahhoz, hogy minden beállítást el tudjon

végezni a felhasználó közvetlenül az eszközön. Mikrokontroller használatával ez nem okoz technikai problémát a sok bemeneti csatlakozó miatt. A másik lehetőség egy külső konfigurációs szoftver használata, mely egy különálló eszközön, például asztali számítógépen vagy okos telefonon fut. Ez a program segít a felhasználónak kiválasztani a beállításokat, ezekből egy konfigurációs fájt készít és utána feltölti az eszközre, mely a következő konfigurációig ezeket a beállításokat használja.

Önmagában egyik lehetőséget sem találtam megfelelőnek. A rendszer teljes konfigurálásához elég mély hierarchiájú menürendszerre van szükség főleg, ha a későbbiekben esetleg új beállítási lehetőséget szeretnék felvenni. A korábban kiválasztott technológia, miszerint egy alacsony felbontású LED panelt szeretnék használni, a menüben való navigálás vizuális megjelenítését nagymértékben lecsökkenti. Az ilyen menürendszerek kezelése pusztán nyomógombok alapján nagyon nehézkes, a felhasználó könnyen elveszti, hogy melyik menüben is tart éppen. A legtöbb beállítást a felhasználó ritkán használja, ilyen például a színek beállítása. Így ezekre ideális megoldás lehet a külső konfigurációs szoftver. Ellenben a gyakran használt beállítások vezérlését, mint például a csatlakozási módváltás, fölöslegesen bonyolítaná, ha csak ilyen módon lenne konfigurálható a rendszer.

Ezen okok miatt úgy döntöttem, hogy a beállításokat szétbontom. Az egyszerű, de gyakran használt kapcsolódási módválasztást nyomógombbal vezérelhetővé teszem, ezáltal ez gyorsan módosítható lesz, míg a bonyolultabb stílus beállításokhoz egy konfigurációs programot készítek. Ebben a programban a felhasználó grafikusan is látni fogja a kiválasztott stílust. A program platformjául asztali számítógépet választottam, ugyanis ehhez könnyebb a mikrokontrollerek csatlakoztatása, míg telefonos applikáció esetén egy újabb vezeték nélküli csatlakozó modulra lenne szükség[4].

[4] Az USB OTG egy olyan technológia, mely lehetővé teszi usb képes eszközöknek, mint például egy telefon hogy hosztként is működjenek, így más usb eszközök csatlakozhatnak hozzájuk.

Ezzel a technológiával a telefonkészülék is képes lehet soros porttal csatlakozni a mikrokontrollerhez.

4 Megvalósítás

4.1 Eszközválasztás

A megvalósítás legfontosabb része a hardver alkotóelemek kiválasztása melyből a rendszer fel fog épülni. Ezek a döntések meghatározzák a felépítés komplexitását, működését. Ezen fejezetben az alkotóelemek kiválasztásának folyamatát, az egyes alkotóelemek melletti döntés okát fogom részletezni.

4.1.1 Mikrokontroller

A központi építőelem a rendszer vezérléséért felelős mikrokontroller. Ennek megfelelő kiválasztása nagyon fontos lépés volt. Az iránta támasztott követelmények többek között, hogy:

- megfelelő sebességgel, számítási kapacitással,
- elegendő ki- és bemeneti csatlakozóval,
- megfelelő periféria modulokkal,
- beépített A/D konverterrel,
- beépített programozó áramkörrel rendelkezzen és
- ne legyen túl magas ára.

Ezek közül az egyik legfontosabb tulajdonság a számítási kapacitás. Mint ahogy azt a jelanalízisről szóló fejezetben említettem, az audió jel mintavételezéséhez legalább 40kHz-es frekvenciával kell mintavételezni, majd átalakítani az analóg jelet. Tipikusan egy ilyen átalakítás 15-20 órajel ideig tart. Ez alapján a mikrokontrollernek legalább 60-70 MHz-es órajelen kell tudnia működni.

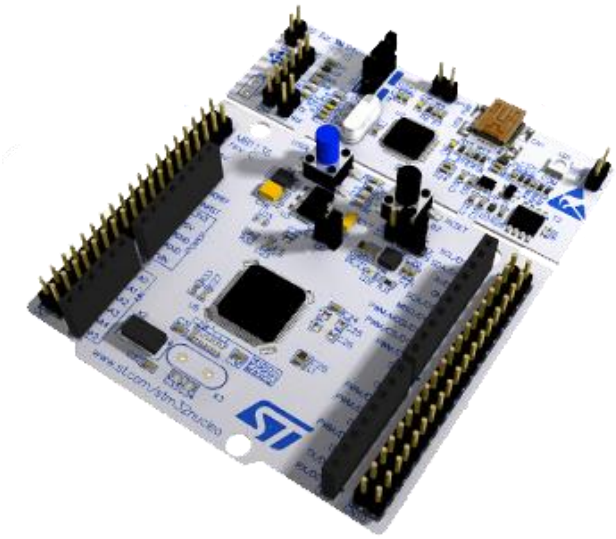
Fontos a ki és bemeneti csatlakozók típusa és mennyisége is. A mikrokontrollerek alapfunkcióinak bővítésére sokféle kiegészítő modult lehet kapni. Létezik például analóg-digitális átalakító modul, melyet a projektem során használhattam volna, vagy Bluetooth vevő modul is. Minél több ilyen típusú modult támogat a kiválasztott mikrokontroller, annál könnyebben bővíthető a projekt során.

Sok mikrokontroller gyárilag tartalmaz az áramkörbe integrált analóg-digitális átalakítót, mely könnyebb használatot biztosít, mint egy külső modul. Fontos hogy a mikrokontroller képes legyen USB porton keresztül kommunikálni asztali PC-vel, nem csak a felprogramozás idejében, hanem működése közben is. Erre azért van szükség, hogy a konfiguráló program adatokat tudjon küldeni.

A mikrokontrollerek felprogramozására alapvetően egy programozói áramkörre van szükség, mely segítségével feltölthető rájuk a futtatott kód. Az ilyen áramkörök elkészítése nem egyszerű, ezért a legtöbb kártyagyártó beleépíti ezt a termékébe.

Ezen szempontokat figyelembe véve a választásom az ST Microelectronics által gyártott STM Nucleo64 F446RE [6] nevű mikrokontrolleres kártyára esett.

Ez a kártya egy 32 bites ARM Cortex M4 [7] processzorral szerelt mikrokontrollert tartalmaz, mely akár 180 MHz-es órajellel is működni képes, ez elegendő a projektemhez. Ezen kívül ez a kártya rendelkezik Arduino V3 típusú csatolófelülettel, így használhatóak vele az egyik legnagyobb választékot kínáló gyártó, az Arduino bővítőmoduljai. Képes virtuális soros portot (VCP) biztosítani, mellyel USB csatlakozással tud asztali számítógéppel kommunikálni.



8. ábra STM Nucleo F446RE [FORRÁS]

Ezek mellett az elvárt tulajdonságok mellett sok egyéb pozitív tulajdonsággal rendelkezik a kártya, mely ideálissá teszi a projekt megvalósításához. A kártyán lévő mikrokontrollerben megtalálható egy beépített analóg-digitális átalakító modul, így ezt nem kell külön beszerezni, ezen kívül mivel közvetlenül a kontrollerbe épített,

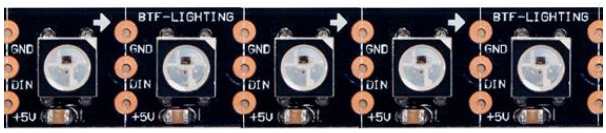
használata is jóval egyszerűbb. Az analóg jelet elég közvetlenül a kártya egy bemeneti csatlakozójára kapcsolni, ezt fogja tudni olvasni az átalakító és a program futtatása során már a digitális jellel tudok dolgozni. Ezek részletes működéséről a mikrokontroller által futtatott program leírásakor fogok beszélni. Ezen kívül nagy pozitívum, hogy a gyártó készített és a termék mellé elérhetővé tett egy grafikus paraméterező programot, melyben megadhatjuk a kontroller kezdeti beállításait, mint például órajel, használt modulok, lábkiosztás, stb. Ez a program képes generálni egy fordítható programvázat, melyet fejlesztőkörnyezetbe importálva a fejlesztés során a tényleges működés megvalósításával kell csak foglalkozni, a programban kiválasztott be- és kimeneti portok, egyéb modulok már inicializálva és felparaméterezve rendelkezésre állnak. A program használatát szintén a következő fejezetben fogom bemutatni.

Ez a kártya teljesítményben és kínált funkciókban is felülmúlja a projekthez szükséges minimális elvárásokat, de mivel ára nem túl magas, az első fejezetben említett megoldásoknál jóval kevesebb, így ez nem okoz problémát, sőt plusz biztonságérzetet ad, hogy az eszköz nem használja ki a teljes teljesítményt, így későbbiekben fejlesztési lehetőségeket biztosít.

4.1.2 Megjelenítő

A megjelenítő egységről már tervezési fázisban eldöntöttem, hogy egy RGB LED-ekből álló panelt fogok használni. Ezen panellel szemben állított elvárások, hogy színes és a mikrokontroller által vezérelhető (egyenként címezhető LED-ek) legyenek.

Választásom a WS2812b típusú LED-ekre esett, melyek a mikroelektronikában gyakran használt LED chipek. Kaphatók eleve panelben gyártva, vagy szalagban is. Én a projektem során szalagban vásároltam őket és abból építettem meg a panelt, de ennek csak anyagi okai voltak, működése nem tér el az eleve panelben vásárolt típustól.



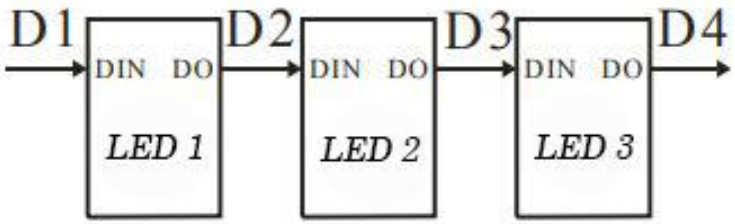
9. ábra WS2812 LED-ekből álló LED szalag

A kiválasztott LED-ek képesek az RGB színskálán színenként 8 bites felbontású megjelenítésére, és tartalmaznak egy kiegészítő fehér LED diódát is, nem csak a három alapszínt. Működésükhöz 5 Volt feszültség szükséges. A LED-eknek csak egy része üzemel egyszerre, és azok sem maximális fényerőn. Így az alapműködés esetén a mikrokontrolleres panel is tudja biztosítani a szükséges tápellátást.Maximális fényerő mellett ezek a LED-ek körülbelül 50mA-t használnak, emellett viszont már nyilvánvaló, hogy ha minden LED egyszerre üzemelne teljes fényerőn, azt már nem tudná a mikrokontroller meghajtani, ugyanis az USB csatlakozón keresztül csak 500mA-t tud felvenni maximálisan. Ebben az esetben külső áramforrás csatlakoztatására van szükség.

Ezek úgynevezett címezhető LED-ek, tehát egyesével megszabható, hogy melyik milyen színű legyen a szalagban. Minden chip rendelkezik 5V és föld csatlakozókkal, ezen kívül mindegyiken található egy adat bemenet (DIN) és egy adat kimenet (DOUT) csatlakozó pont.

A LED-ek egymáshoz csatlakoztatása úgy lehetséges, hogy az első DOUT csatlakozóját a következő DIN csatlakozójához kapcsoljuk, és így sorba kötünk minden további chipet. Ezután a mikrokontrollert az első LED-hez csatlakoztatjuk, a vezérlő jelet annak DIN csatlakozójára kell küldjük.

Ezen típusú LED-ek használata során a legnagyobb nehézséget a megfelelő vezérlőjel előállítása okozza. Egy LED chip beállításához egy 24 bites adatcsomagra van szükség, melynek első 8 bitje a zöld szín, középső 8 bitje a piros szín és utolsó 8 bitje a kék színkomponens kívánt értékét tartalmazza. A teljes panel összes LED-jének beállításához annyiszor 24 bitet kell egymás után fűzni ahány LED-ből a panel áll, majd ezt az elkészült csomagot egyben az első panel DIN portjára küldeni. Ezután az első LED beolvassa az első 24 bitet, azt levágja az üzenetről, és a maradékot továbbküldi DOUT portján. Ezt a következő LED olvassa, levágja az első 24 bitet, majd továbbküldi a következő lednek. Ez a folyamat ismétlődik, amíg az üzenet hossza nagyobb, mint 24 bit.

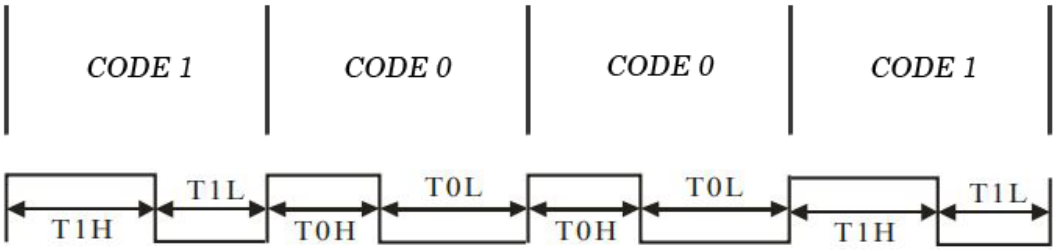


10. ábra LED-ek sorba kötése

Ez a működési elv pozitív és negatív tulajdonságokat is magában hordoz. Pozitív, hogy a LED sor hossza rugalmasan változtatható, megfelelően megírt vezérlőprogram esetén az üzenet meghosszabbítása nem nagy feladat. Másik előnye, hogy mivel egy aszinkron pulzusszélesség modulált jelet használnak vezérlőjelként, így nem szükséges külső órajelet biztosítani számukra.

Nehézséget okoz viszont az, hogy az üzenet előállítása és küldése során pontos időzítési feltételeket kell tartani, valamint a teljes csomag kiküldése során nem lehet szünet a jelben, mert ezt reset feltételnek érzékelik a LED-ek. A LED gyári dokumentációjában szereplő adat szerint 800 Kbit/s-os sebességgel kell az adatokat küldeni. Egy bitet reprezentáló jelrésznek 1.2 µs hosszúságúnak kell lennie és ettől maximálisan 300 ns mértékben térhet el pozitív vagy negatív irányba. Ezt az értéket az egész üzenet hosszán keresztül tartani kell.

Az adott bit értékét az azt reprezentáló jelrész kitöltési tényezője, azaz a jel magas és alacsony állapota közötti arány határozza meg. Minden bit egy felfutó éllel kezdődik, ezután 0-ás bit esetén 0.35 µs-ig marad a jel magas értékben, 1-es bit esetén 0.7 µs-ig, ezután lefutó él következik és 0-s bit esetén 0.8 µs-ig, 1-es bit esetén 0.6 µs-ig marad alacsony állapotban a jel. Az ezután következő felfutó él már a következő bit kezdetét jelzi. Az alábbi ábrán az 1001 bitsorozat jele látható.



11. ábra 1001 bitek jelkódolása

A LED panel két frissítési ciklusa között ki kell adni egy legalább 50 µs hosszú alacsony állapotú jelet, ez jelzi az első LED számára, hogy az előző csomag véget ért, tehát a következőkben bejövő első 24 db bitet ismét sajátként dolgozza fel és csak a maradékot küldi tovább.

Ezen időzítési nehézségek ellenére is egy jól használható programozható LED típus. Elterjedt használata miatt viszonylag sok segédlet és dokumentáció található használatához.

4.1.3 Bluetooth audio receiver

A vezeték nélküli csatlakozáshoz szükség volt egy Bluetooth audióvevő modulra, melyet a mikrokontrollerhez tudok csatlakoztatni. A tervezés kezdeti fázisában úgy gondoltam, hogy olyan modult fogok használni, mely nem tartalmaz digitális-analóg átalakítót és kimenetén digitális PCM jelet biztosít, ezzel elkerülve a szükséges átalakítást. A megvalósítás során viszont rájöttem, hogy ez nem egyszerűsíti sem a működést, sem a megvalósítást. Mivel a mikrokontroller saját átalakítóját használom a vezetékes csatlakozás esetén, ezért a két forrás során különböző típusú bemenetekkel kellene dolgozni, amely a futtatott kódot bonyolítaná.

Léteznek olyan Bluetooth vevők is, melyek analóg jelet szolgáltatnak. Ilyen vevőegység használata esetén ezt az eszközt szintén rá tudom kötni a mikrokontroller átalakítójára, így a program belső működése semmiben sem fog különbözni a két csatlakozási mód esetén. A mikrokontroller két teljesen különálló átalakítóval rendelkezik, melyek egyenként 12 bemenetet tudnak kezelni, tehát nem okoz problémát több analóg forrás bekötése.

A Bluetooth kommunikáció során a jel digitális formában érkezik, tehát egy ilyen vevő ezután visszaalakítja analóggá a jelet, majd én ezt újra digitalizálom, és úgy kezdem el feldolgozni. Ez az eljárás rontja a hangminőséget ugyan, de nagyban leegyszerűsíti a működést. Mivel a vizualizációhoz nem szükséges a hangminőség tökéletes megtartása, ezért ezt a megoldást választottam.

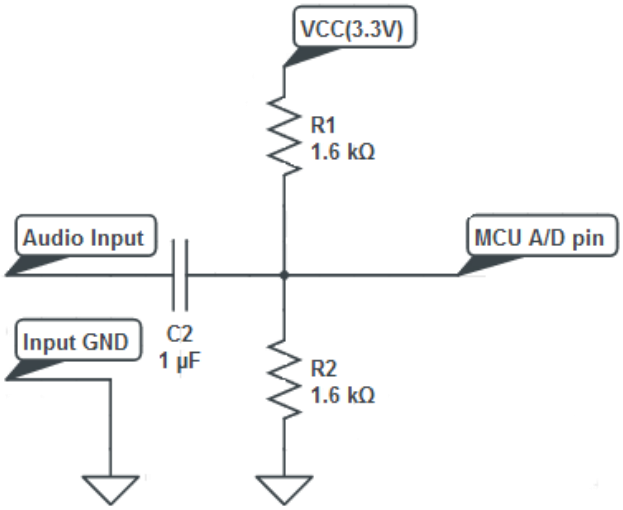
A konkrét kiválasztott eszközt, melyet a projekt során használtam, egy webáruházból rendeltem. Egy bekapcsoló gombbal és egy jack csatlakozó kimenettel és egy mikroUSB csatlakozóval rendelkezik, semmilyen beállítást vagy konfigurációt nem igényel a használata. Egy töltéssel körülbelül 4-6 órán keresztül képes üzemelni. Az

átviteli hangminősége elég gyenge, viszont a projekt során erre nincs is szükség, ezért inkább egy ilyen olcsóbb modellt választottam. A termék az alábbi képen látható.

12. ábra Bluetooth vevő

4.1.4 Jelkondícionálás

A 3.1.1 –es fejezetben említettem, hogy feldolgozás előtt szükség van a jel pozitív tartományba való transzformálására. Ezt egy jelkondícionáló áramkör közbeiktatásával tettem meg.

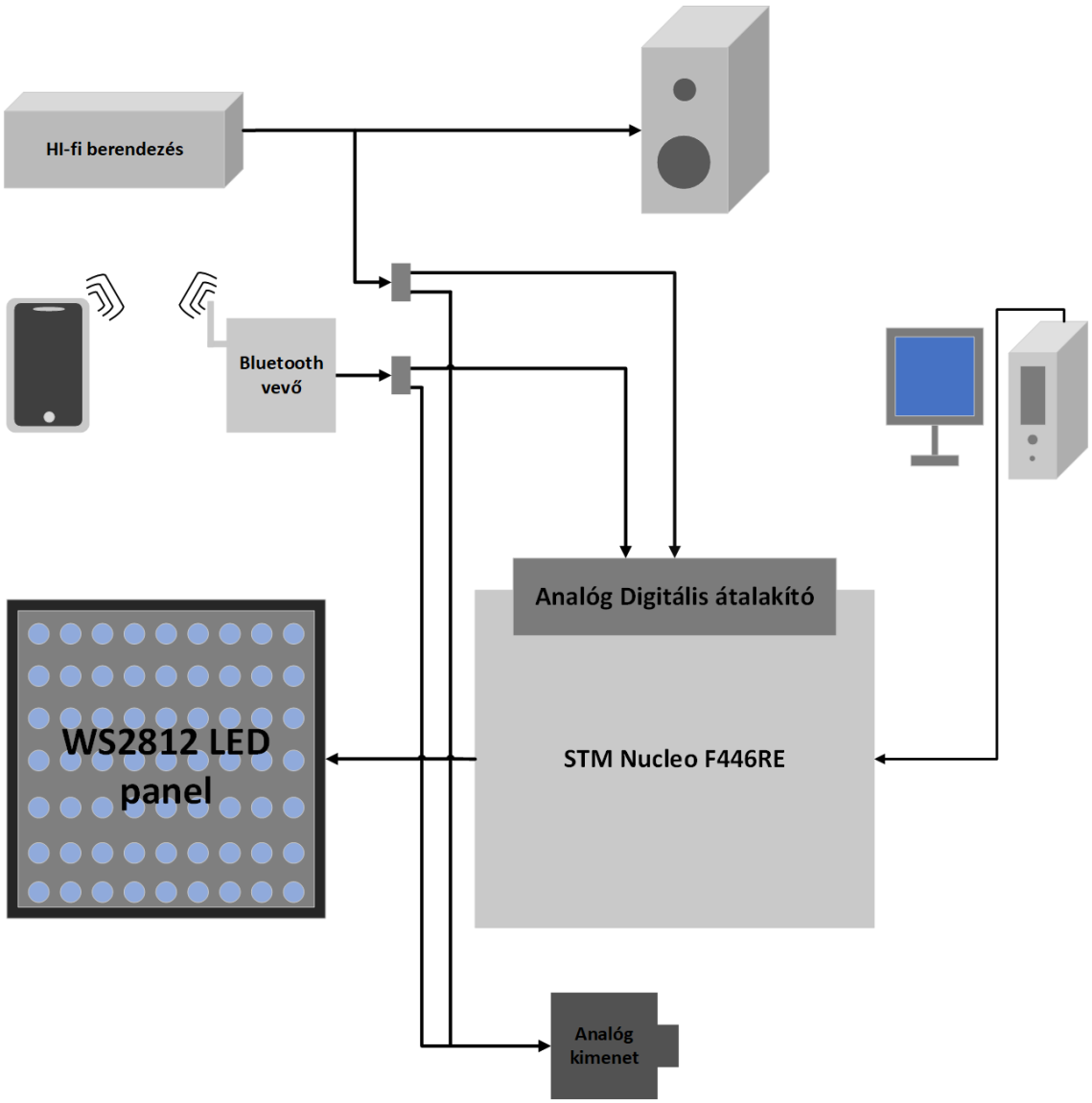


13. ábra Jelkondícionáló áramkör kapcsolási rajza

Ezzel az áramkörrel beállítható a jel középszintje. Az egyenlő ellenállások miatt ez ebben az esetben a tápfeszültség fele, azaz 1.6 V-nál lesz. Ettől az értéktől fog negatív és pozitív irányba kitérni a jel. Mérésekkel megállapítottam, hogy körülbelül 2 V tartományt fed le egy átlagos eszközből érkező audió jel, így ezzel a beállítással biztosan pozitív értékben tartható a jel.

4.1.5 Módosult blokkdiagram

A hardverek kiválasztása során pontosodott, és kis mértékben módosult a rendszer architektúrája. Az ez alapján elkészült új blokkvázlat az alábbi képen látható.



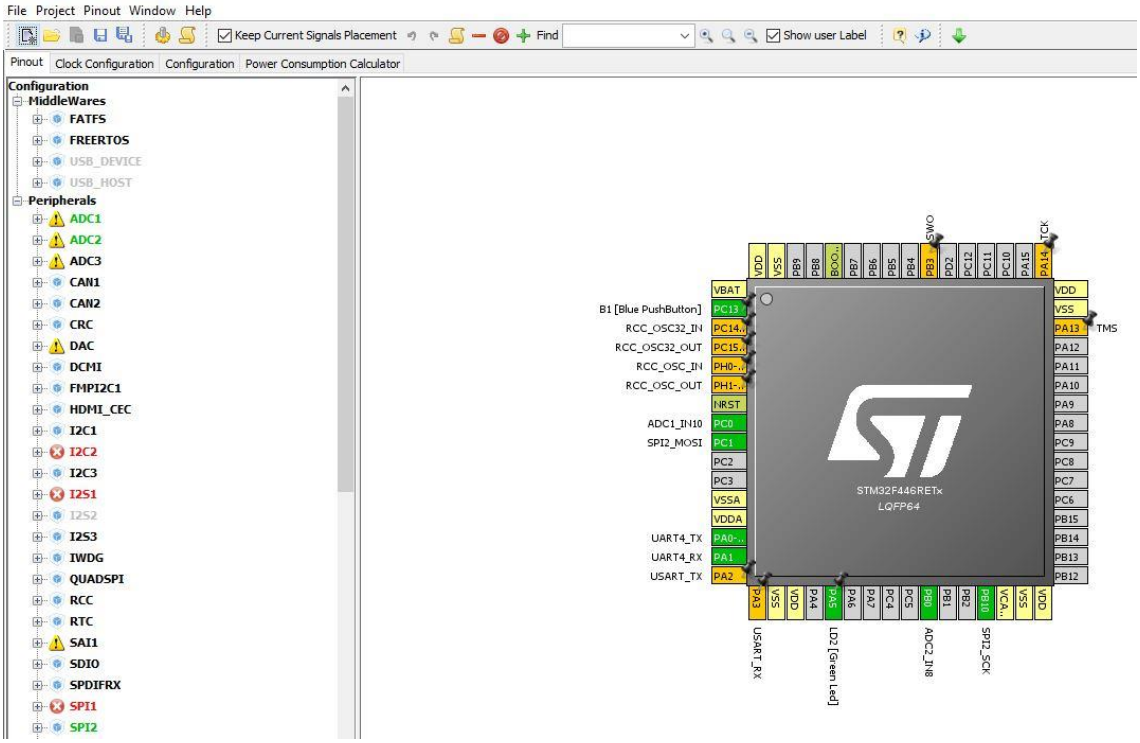
14. ábra Blokkdiagramm

4.2 A vezérlő programja

4.2.1 Használt programok, könyvtárak

A mikrokontroller által futtatott program fejlesztésében nagy segítséget nyújtottak azok a hardver specifikus segédprogramok és programkönyvtárak melyek a mikrokontroller gyártójától, vagy annak típusából adódóan rendelkezésre álltak. Ebben a fejezetben ezeknek a működését, használatát fogom bemutatni.

Az első ilyen használt program neve STM32 CubeMX. Ezt a programot az ST Microelectronics készítette fejlesztőkártyái mellé. A programban grafikus felületen lehet a mikrokontroller kezdeti beállításait megadni.



15. ábra CubeMX, Lábkiosztás képernyő

Új projekt létrehozásakor ki kell választani a fejlesztő kártyát (vagy csak a mikrokontroller típusát, amennyiben saját kártyánk van), ezután minden lehetséges beállítási lehetőség megjelenik. A program négy fő képernyőből áll. Ezek közül az elsőn, mely a 15. ábraán látható, a kártya által biztosított perifériák vannak felsorolva. Melyek közül kiválaszthatjuk, hogy miket szeretnénk használni. A program jelzi, hogy mely perifériák használhatóak egyszerre, és hogy melyik modult melyik ki-, vagy bemeneti lábra csatlakoztatja.

A második képernyő az órajel konfigurációért felelős. Ezen a képernyőn megtalálható az összes belső órajel, és hogy azok a fő órajel milyen átalakításával keletkeznek. A harmadik képernyőn, az első oldalon kiválasztott modulok konfigurációját lehet elvégezni. Az utolsó képernyőn a kártya energiafogyasztását lehet megtekinteni és konfigurálni. Ez abban az esetben fontos, ha akkumulátorról szeretnénk használni a kártyát.

A program legnagyobb előnye, hogy miután mindent beállítottunk, lehetőséget nyújt rá, hogy ezekből a beállításokból egy fordítható kezdeti projektet generáljunk. Többféle fejlesztőkörnyezetet is támogat, és az adott környezetbe importálható projektet készít el. Ebben az elkészült programvázban jelölve vannak a megfelelő helyek, ahová a felhasználó saját kódját írhatja. Ha csak ezeket a helyeket használjuk, akkor később tudjuk módosítani a CubeMX-ben a projektet, új modult adhatunk hozzá, beállításokat módosíthatunk, és azután újragenerálhatjuk a programvázat anélkül, hogy az addigi munka elveszne.

A program által generált váz egy main.c fájlból és egy programkönyvtárból áll melynek neve HAL library (Hardware Abstraction Layer). Ez a könyvtár, mint ahogy a neve is jelzi, a kártya hardverkomponenseinek használatát teszi lehetővé magasabb absztrakciós szinten. Használatával nem szükséges a perifériák regiszter szerinti címzése, beállítása, mert ez a könyvtár mindenre kínál kódból hívható metódust. A main fájlban található a main függvény, melyben a program elvégzi az egyes komponensek inicializálását, és létrehoz a modulokra globális változókat, melyek segítségével hívhatóak a HAL könyvtárban lévő metódusok. Ez nagyban megkönnyíti a kártya használatát. Projektem során nagy hasznát vettem ennek a programnak, az általam a programban használt konkrét modulok és beállítások ismertetését a következő fejezetben fogom megtenni.

Egy másik fontos programkönyvtár használatát a mikrokontroller processzorának típusa tette lehetővé. A Cortex típusú processzorok mellé létezik egy úgynevezett Cortex Microcontroller Software Interface Standard (CMSIS), mely egy olyan alkalmazásprogramozási interfész réteg (Application Programming Interface, API), ami lehetőséget nyújt a gyártók számára processzor közeli metódusok definiálására.[8] A processzor szolgáltat egy interfészt, melyet a gyártók saját termékeiknek megfelelően megvalósíthatnak, ezzel lehetővé téve olyan műveleteket melyek teljes mértékben kihasználják a processzor képességeit. Ha a gyártók nem

készítik el ezt a megvalósítást saját termékükhöz, akkor egy köztes, úgynevezett „third party vendor” fél is megteheti ezt.

Ez az interfész réteg több programkönyvtárból épül fel, nekem ezek közül egyre volt szükségem, a matematikai műveleteket támogató könyvtárra, ugyanis ennek része egy digitális jelfeldolgozó könyvtár (DSP library), mely teljes mértékben megvalósítja az FFT algoritmust, így nekem nem kellett saját implementációt készíteni hozzá. Ráadásul ez a megvalósítás kihasználja a processzor adottságait. A processzor FPU-t (Floating Point Unit) tartalmaz, így a törtszámokkal történő számítások gyorsabbak.

A fejlesztés során az Atollic TrueSTUDIO nevű fejlesztőkörnyezetet használtam. Ez egy Eclpise alapú fejlesztőkörnyezet ARM architektúrájú eszközökön futó programok fejlesztéséhez. A program sok lehetőséget nyújt a hibakeresés támogatására. Ezen lehetőségekről a Tesztelés című fejezetben fogok beszélni.

4.2.2 A beágyazott szoftver

Az egész rendszer működéséért a mikrokontroller által futtatott program a felelős. A projekt lényegi része e program elkészítése volt. Ebben a fejezetben az elkészítés folyamatát, a program működését fogom bemutatni.

A fejlesztés során az inkrementális fejlesztés elvét használtam, azaz lépésről-lépésre haladtam a funkciók megvalósításával. Elsőként a lehető legkevesebb funkciót tartalmazó programot készítettem el, majd ezt bővítettem ki egyesével egyre több funkcióval.

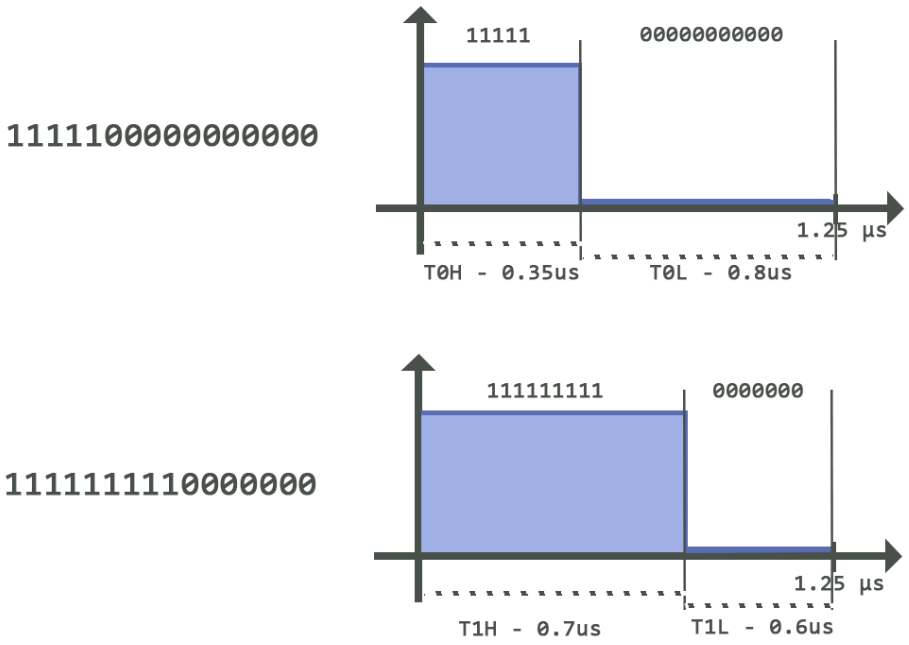
A folyamat első lépéseként azt valósítottam meg, hogy a mikrokontroller képes legyen a LED-ek vezérlésére. A 4.1.2-es fejezetben leírtak alapján a legnagyobb problémát a megfelelő jel előállítása és időzítése okozta. Ennek a problémának a megoldására a Soros Periféria Interfész buszt (Serial Peripheral Interface, SPI) választottam. Ez a technológia eredetileg perifériák szinkronizált, kis távolságú kommunikációját teszi lehetővé valamilyen mester egységgel.[9] Beágyazott rendszerekben gyakran használják. Ez egy külső órajel által szinkronizált kétirányú kommunikációs csatorna.

Azért fontos ennek használata, mert az SPI egység hardveresen implementált, így az üzenetküldés mentes a szoftver által okozott esetleges jitterektől. Ha egy

szoftveres megoldást választottam volna, azok működése sokkal ingadozóbb lenne, mely tönkretenné az időzítést.

Mivel nekem csak egyirányú kommunikációra volt szükségem, így csak a MOSI (Master Output Slave Input) irányt használtam. Ezen kívül a modul által biztosított órajelet sem kellett használnom. Viszont a jel előállítására és küldésére így is tökéletesen megfelel. Ezen kívül a kommunikáció sebessége beállítható és ezáltal az üzenet időzítése is pontosan megadható.

Először létrehoztam a 4.2.1 –es fejezetben ismertetett CubeMx programban egy új projektet. Ezután kiválasztottam használatra az SPI2 modult, mivel az SPI1 által használt lábakat foglalta a rendszer által automatikus bekapcsolt USART kommunikációs modul (Univerzális Szinkron és Aszinkron Küldő és Fogadó). Ez hibakeresési funkciókhoz fontos soros üzenetküldést tesz lehetővé, melynek használatáról egy későbbi fejezetben fogok beszélni. Az SPI2 modul bekapcsolása után annak konfigurálása következett. A két legfontosabb paraméter a küldési sebesség és az adatcsomagok mérete. Az adatcsomagok mérete lehet 8 vagy 16 bit. Ahhoz, hogy a LED számára szükséges jel bitjeit elő tudjam állítani egy pontos kitöltési tényezőjű jelre volt szükség. Ez úgy értem el, hogy 16 bites csomagméretet választottam, és egy darab ilyen 16 bites csomag reprezentál, egy darab, a LED felé küldött bitet. Egyes értékű bit esetén a 16 bites csomag első kilenc bitje 1-es, a maradék hét bit pedig 0-ás, nullás bit esetén pedig az első öt bit 1-es, a maradék tizenkettő 0-ás értékű. Az így előállított csomagok láthatóak a 16. ábraán. Felül a nulla értékű bitsorozat, alul az egyes értékű bitsorozat.



16. ábra SPI csomagok

Mivel a LED-ek 800Kbit/másodperces sebességű adatot várnak, így az ily módon előállított 16 bites csomagoknak kell ezzel a sebességgel érkezniük. Tehát a tényleges adatküldés sebességének 800Kbit/másodperc 16 szorosának, azaz 12.8 Mbit/másodpercnek kell lennie. Ezek alapján tehát megkaptuk a két fontos konfigurációs adatot: 16 bites adatcsomag méret, és 12.8 Mbit/másodperces sebesség. A CubeMX-ben az adatcsomag méretének beállítása egyszerű, csak ki kell választani egy legördülő menüből, hogy 16 bit, a sebesség beállítása viszont nem triviális. Ehhez először a kártya specifikációjából meg kellett keresnem, hogy melyik belső órajelet használja az SPI2, majd ezután ezt az órajelet olyan értékre állítani, amely a modul saját beállított órajelszorzója mellett megfelelő sebességet biztosít. Az SPI konfigurációs felületén jelzi, hogy a beállított órajel mellett milyen sebességű lesz a kommunikáció. Az SPI2 az APB2 jelzésű órajelet használja, ennek 51 MHz-re való állítása és 2-szeres órajelszorzó beállítása mellett 12.75 Mbit/másodperces sebesség érhető el, mely tolerancia határon belül van.

A kívánt beállítások megadása után a TrueSTUDIO fejlesztőkörnyezetet kiválasztva generáltam egy programvázat, melyben az így beállított SPI csatornát már használni tudtam.

Minden LED 3 x 8 bitre vár, mely az RGB értékeket jelenti számára. Az elkészült programvázban létrehoztam tehát egy globális tömb tárolót, mely 8 bites

számokat tárol. Ennek a tárolónak a mérete a LED-ek számának háromszorosa, mert minden LED-hez három ilyen színérték tartozik. Az első LED színe a tömb első, második és harmadik eleméből áll össze, a második LED színe a negyedik, ötödik és hatodik elemből, és így tovább. Ezután létrehoztam egy metódust, mely ezen a tárolón végighaladva ezeket a 8 bites értékeket alakítja át az SPI által kiküldendő csomagokká. Egy 8 bites színből egy 8 x 16 bites üzenet lesz.

```
for (i = 0 ; i < COLOR_COUNT; i++) {
     for (j = 0; j < 8; j++) {
          if ((colors[i] & (128 >> j)) != 0) {
               output[i * 8 + j+20] = 0b1111111110000000;
          } else
               output[i * 8 + j+20] = 0b1111100000000000;
     }
}
```

A metódusban az `i` ciklusváltozó jelöli, hogy éppen hányadik színt dolgozzuk fel a globális tárolóból. Végighalad a színeken, és megvizsgálja az adott szín minden bitjét, hogy az 1-es vagy 0-s és annak megfelelő 16 bites értéket teszi a kimeneti tömbbe. Fontos megjegyezni, hogy a LED-ek a színértékek beolvasása során a legmagasabb helyiértékű bitet várják elsőként (MSB first). A 8 bites érték egyes bitjeinek vizsgálatát úgy végzem, hogy egy másik 8 bites számmal hasonlítom össze a bitenkénti ÉS logikai műveletet használva. Ezen másik szám kezdeti értéke 128, mely binárisan ábrázolva az 10000000 szám. Tehát a logikai művelet kimenetének az első 7 bitje mindenképpen 0 lesz. Ha a vizsgált szín legmagasabb helyiértékű bitje 0-ás, akkor biztosan a végeredmény is 0 és az ennek megfelelő 16 bites érték kerül a kimenetbe, ha 1-es, akkor a végeredmény nagyobb, mint nulla, és az egyesnek megfelelő 16 bites érték kerül a kimeneti tárolóba. Ezután a következő iterációban jobbra shiftelem a 128-as értéket és a vizsgálatot ugyanígy végzem. Ezzel az eljárással egyszerre tudom vizsgálni a bitek értékét, és mivel az 10000000 szám volt a kezdeti érték melyet jobbra shiftelek, ezért ez az eljárás meg is fordítja a helyiérték sorrendet.

Így a kimeneti tömbben végül a színek száma * 8 * 16 bites érték lesz. Az így elkészült kimeneti tömböt az SPI-al egyben kiküldöm.

Ez a küldés a HAL könyvtárnak köszönhetően csak egyetlen metódushívás, amelynek paraméterei a kiküldendő adatra mutató pointer és az adathalmaz mérete.

Ezután már csak csatlakoztatni kellett az SPI kimeneti portját (MOSI) az első LED adatbemeneti (DIN) portjára[5], és a kontroller képes lett a megjelenítő vezérlésére. A beállítani kívánt színeket az említett globális tömbbe kell megadni.

Ezután létrehoztam a szükséges belső segédmetódusokat, melyek a színeket tároló tömbnek az egyszerűbb feltöltését teszik lehetővé. Létrehoztam egy metódust mely paraméterként a három szín értékét és a beállítani kívánt LED indexét várja. Létrehoztam egy olyat is, mely egész oszlopokat képes a megadott színre beállítani, paraméterként a színeken kívül az oszlop indexét és a kitöltés magasságát várja. Ezen függvények a kezdeti használathoz már elegendőek voltak.

A következő lépés a fejlesztés során az audió jel beolvasása és digitalizálása volt. Ehhez a mikrokontroller analóg/digitális átalakítóját kellett használnom, így újra megnyitottam a projektet a CubeMx programmal, és bekapcsoltam az ADC1 modul 10-es csatornáját. Bekapcsolása után ezt is be kellett konfigurálni. Ez a periféria elég sok beállítási lehetőséget kínál. Meg kell adnunk a belső órajel szorzóját és a minták felbontását, pontosságát. Ez a két adat együtt fogja meghatározni a mintavételi frekvenciát. A kiválasztott felbontás mellett jelzi a program, hogy mennyi órajel ciklus időbe fog telni egy mintavétel. Én 12-bites felbontást használtam, e beállítás mellett 15 órajelciklusba kerül egy mintavétel. Ezután az órajel beállítással érhetjük el a 44 kHz-es mintavételi frekvenciát. Ezen kívül engedélyeztem a folytonos konvertálás módot, ami azt teszi lehetővé, hogy indítás után folyamatosan futni fog, és ciklikusan írja a számára adott memóriaterületet. A többi beállítási lehetőséget alapbeállításon hagytam.

Ezek után a program CubeMx általi újragenerálása után az ADC modul használhatóvá vált. Használatához létre kellett hozni egy tárolóterületet, ahová a mintavételezett értékeket írni tudja. Ez egy 16 bites értékeket tároló lista. Ezután elég a HAL könyvtár által biztosított Start metódus meghívása, melynek paraméterül kell adni a tárolóra mutató pointert, és egy konvertálási ciklus alatt vett minták számát. A lefoglalt tároló méretének legalább akkorának kell lennie, mint a minták száma.

[5] A LED-ek 5V-os TTL (Transistor-Transistor Logic) jelet várnak, míg a mikrokontroller 3.3V-os CMOS rendszerű. Ez kifelé irányban nem jelent problémát, de ha a LED is kommunikálna visszafelé irányban a kontrollerrel, az túlfeszültséget okozhatna.

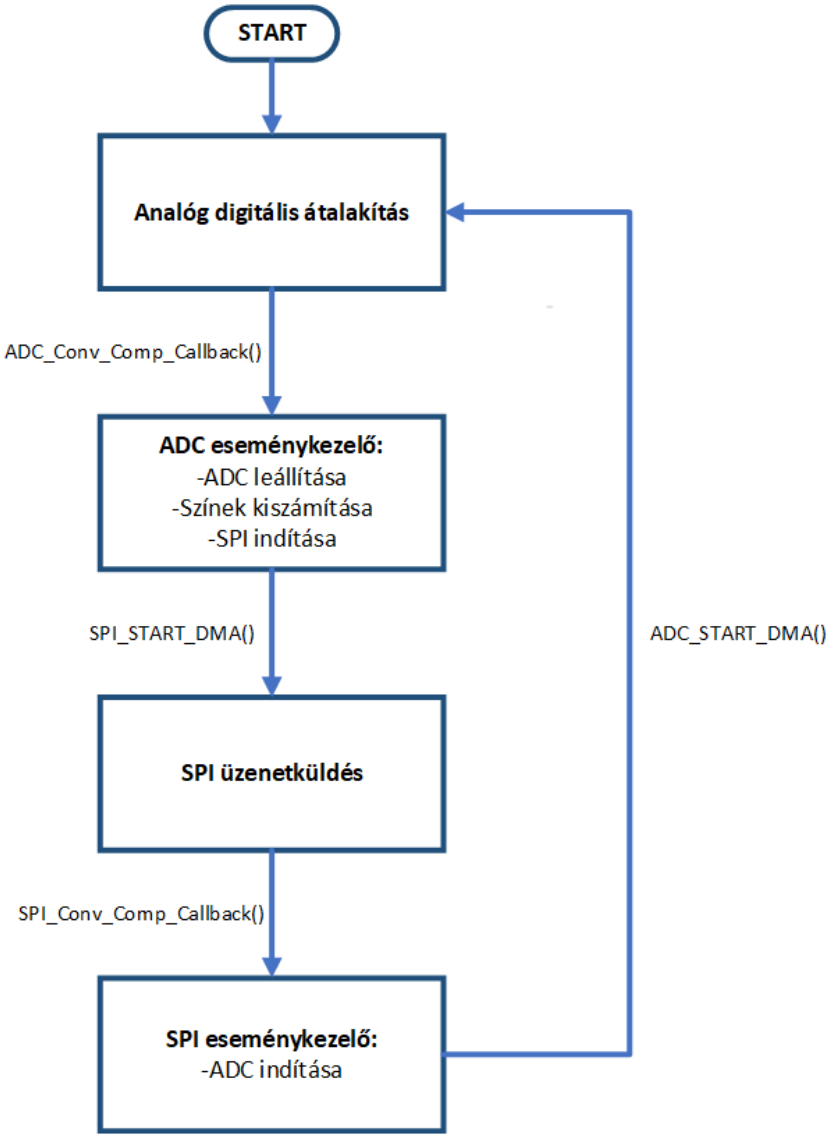
A vett minták száma meghatározó paramétere az egész rendszernek. Ez az egyetlen rugalmasan változtatható rendszerszintű paraméter, mellyel a rendszerszintű időzítések módosíthatóak. Az egész program globális időzítéséről egy önálló fejezetben fogok beszélni miután minden modul bemutatásra került.

A modul rendelkezik esemény megszakító metódusokkal. Ezek között szerepel az az eseménykezelő, mely akkor hívódik meg, mikor egy átalakítási ciklus befejeződött. Ez a metódus úgy van definiálva a HAL könyvtárban, hogy felül kellett írnom azt egy saját változattal, melyben egyedi működést adhattam meg. Ez a kezelőfüggvény megfelelő hely lesz a program későbbi FFT analízis funkciójának indítására, ugyanis ez az esemény akkor hívódik meg, mikor az ADC modul elkészül a minták egész tömbjével. Ekkor elvégezhetjük rajta a Fourier transzformációt, majd megjeleníthetjük a kívánt értéket.

A projekt jelenleg elkészített funkcióinak teszteléseként első lépésként a vett mintákat átlagoltam, majd ezt az átlagértéket a LED panel összes oszlopnak magasságához rendeltem. Ezen teszt során szembesültem azzal a problémával, hogy az ADC modul és az SPI kommunikáció párhuzamos használata elrontja az SPI üzenetküldés időzítését. A processzor ütemezője félbeszakítja az üzenetek folyamatos küldését az ADC kérések kiszolgálására, így a komplett üzenet folytonossága megszakad, emiatt hamis színértékek jelentek meg a LED-eken.

Ennek a problémának a megoldására egy teljesen új programarchitektúrát kellett kitalálnom, melyben semmilyen más modul nem aktív az SPI kommunikáció alatt. Ehhez megszakítás vezérlésűvé tettem a programot, és ezen kívül a már meglévő ADC és SPI moduloknál bekapcsoltam a közvetlen memória hozzáférés (Direct Memory Access, DMA) funkciót [10]. Ez a technológia lehetővé teszi az egyes moduloknak, hogy a CPU-tól függetlenül írjanak a rendszermemóriába. Ennek bekapcsolása szintén a CubeMX programban lehetséges, az egyes modulok konfigurációjánál. Ezután a modulok metódusaiból létrejön egy „`_DMA`”-ra végződő nevű változat is, mely már használja ezt a technológiát, például: `HAL_ADC_START_DMA()`. A másik feladat a megszakításokon alapuló működés megvalósítása volt. A kitalált megoldás során a program indulásakor elindítja az ADC modult. Majd ennek a már korábban említett azon eseménykezelő függvényében, mely akkor hívódik meg, mikor az átalakítás egy ciklusa befejeződött, leállítom az ADC modul működését, kiszámítom a megjeleníteni kívánt színeket a globális színtárolóba (később ilyenkor fog történni az FFT átalakítás

is). Miután ez elkészült, elindítom az SPI adatküldést. Az SPI modul is rendelkezik az A/D átalakítóéhoz hasonló eseménykezelő függvényekkel. Ezek közül is azt használtam, mely akkor hívódik meg, ha az egész kommunikáció befejeződött. Ebben a kezelőfüggvényben újraindítom az ADC modult, és innentől ismétlődik a folyamat. Ennek folyamatábrája látható az alábbi ábrán.



17. ábra Megszakítás alapú vezérlés

Ezzel a módszerrel már zavartalanul tud működni az SPI kommunikáció.

A működés szempontjából a következő nagy funkció az FFT függvénykönyvtár integrációja volt. A kezdeti próbálkozások során egy saját implementációt próbáltam meg elkészíteni. Az algoritmus működése utáni kutatás közben találkoztam a 4.2.1 fejezetben már említett CMSIS könyvtárral. Ennek megismerése után úgy döntöttem, hogy saját implementáció helyett ez fogom használni. Azonban ennek a

programkönyvtárnak a használatba vétele nem volt olyan egyszerű, mint az átlagos programkönyvtáraké. Mint ahogy azt a könyvtár leírásában is említettem, ez csak egy interfész réteg, melyet a gyártók megvalósíthatnak. Habár az általam használt kártyánál ennek vannak kezdetleges nyomai, működő gyári megoldást nem sikerült találnom. A CubeMX által generált programvázban szerepel a CMSIS library összes definíciós fájlja (header file), de az implementációs fájlok nem kerülnek bele a generált projektbe, annak ellenére, hogy a program által használt csomagban szerepel ezen a fájlok nagy része.

Ez valószínűleg nem azt jelenti, hogy a gyártó által szolgáltatott könyvtár nem működne. Valószínűleg a CubeMX programgeneráló részében van a hiba.

A gyári megoldás hibája miatt a már szintén bemutatott third party beszállítók általi megvalósítás után néztem. Szerencsére az interneten elég sok ilyen megvalósítás található. Kiválasztottam egy megbízhatónak tűnő projektet és ennek számomra szükséges implementációs forrásfájljait bemásoltam a projektembe. Ezek után a projekt szintű szimbólumokhoz fel kellett vennem az „ARM_MATH_CM4” szimbólumot, mely a processzor típusát jelöli a könyvtár számára.

Ezután már használhatóvá váltak a könyvtár által kínált metódusok. Ezek közül nekem kettőre volt szükségem. Az egyik az FFT algoritmust végzi el egy bemeneti, komplex számokból álló adathalmazon. Ennek futtatásához létre kellett hoznom egy olyan adattömböt, melyben minden páratlanodik elem a mintavételből származó egy érték, és minden páros elem nulla, a páratlan elemek a komplex számok valós részei, a páros elemek a képzetes részek. Tehát mérete a minták számának kétszerese. Ezt az adathalmazt és a minták számát paraméterként adva a CMSIS könyvtár FFT metódusának és lefuttatva azt, ugyanebbe a tárolóba visszaírja a transzformáció utáni értékeket. Ezután a könyvtárból használt másik függvény segítségével megkaphatjuk a komplex számokat tartalmazó FFT kimenet valós számokká alakított értékeit. Ennek egyik bemenete a komplex számokat tároló tömb, másik bemenete egy új tároló, melynek számossága feleakkora. Ebben az új tárolóban lesznek az átalakított minták.

Habár a minták számáról csak a következő fejezetben fogok beszélni, az nyilvánvaló, hogy több lesz, mint ahány frekvenciasávra szeretném bontani a jelet. Ehhez egyenlő intervallumokra bontom a mintákat, és az intervallumokon belül átlagolással egy értéket képzek, mely az adott sáv értéke lesz. Ezeket már könnyebben lehet meg jeleníteni.

A futtatott program alapja ezzel elkészült. Ezután következett a Bluetooth vevő beépítése a rendszerbe, ezzel biztosítva a második csatlakozási módot. Ehhez, mint ahogy a 4.1.3 fejezetben írtam, egy olyan vevőegységet használok, mely analóg jelet szolgáltat. Annak érdekében, hogy a két forrás ne tudja zavarni egymást egy új ADC csatornára van szükség. A két forrás kezelése megoldható lett volna az ADC1 modul egy új csatornájának bekapcsolásával is, de mivel a mikrokontroller két teljes értékű A/D konverterrel rendelkezik és a második modul beüzemelése egyszerűbb, mint egy modul több csatornájának kezelése, ezért az ADC2 modul bekapcsolása mellett döntöttem, melyhez a Bluetooth vevőt csatlakoztattam. Ehhez újra a CubeMx programban kellett módosítanom a projektet és bekapcsolnom az új modult. A modul beállításai és használati módja teljes mértékben megegyeznek a már használt ADC1 moduléval.

Fontos, hogy a perifériák ugyanazon a megszakítás kezelő rutinon (IRQ) osztoznak, tehát mindkettő ugyanazt a metódust fogja hívni. Az eseménykezelő metódus paraméterül kap egy azonosítót, mely alapján lehetséges beazonosítani, hogy melyik modul hívta.

Ezek után a mikrokontrolleren található felhasználói gomb megnyomására kellett olyan működést megvalósítanom, mely a két ADC modul között vált. A nyomógomb már a kezdeti projektvázba is automatikusan belekerül. Paraméterezhető hogy tárolja-e az állapotát, vagy pedig állapotmentesen működjön. Mivel a nyomógomb fizikai kialakítása nem túl precíz és a kiadott jelen gyakran pergés jelentkezik, ezért egy lekérdező (polling) jellegű működést valósítottam meg. Ennek során az ADC modulok már használt eseménykezelő függvényét kiegészítettem azzal, hogy minden egyes megszakításnál megvizsgálja a nyomógomb állapotát, melynek változása során egy saját globális flag-et állítok. Mivel csak minden iterációban egyszer vizsgálom meg a gomb állapotát, ez a késleltetés elég arra, hogy az állapot ingadozást kiszűrje.

Ez a flag szabályozza, hogy az SPI eseménykezelő metódusa melyik ADC modult indítsa.

Az utolsó fontos lépés a paraméterező program megvalósításának előkészítése volt. A program soros porton keresztül tud kommunikálni a mikrokontrollerrel. Mint említettem korábban, debug céllal már alapból be van kapcsolva ez a periféria a projektben. A konfigurációhoz csak az üzenet fogadását kellett megvalósítani és az az alapján a paraméterek beállítását.

Az üzenet fogadásának kezelésére szintén a megszakítás kezelő rutint használtam. Az UART periféria rendelkezik olyan eseménnyel, mely akkor hívódik meg mikor egy üzenetfogadás lezajlott. Ebben a metódusban először egy minimális szintű csatlakozási eljárást valósítottam meg. Egy adott kezdeti üzenetet vár a mikrokontroller. Ennek megérkezése jelzi számára, hogy a konfiguráló program csatlakozott. A csatlakozó jelzés megérkezése után leállít minden egyéb működést és egy válaszüzenetet küld. Ezután egy következő üzenetben fognak a beállítások érkezni, ezt az üzenetet fel kell dolgozni egy buffer segítségével. Majd miután az adatokat beolvasta és elmentette egy szétcsatlakozás üzenetet küld a programnak.

Meg kellett alkotnom egy saját üzenetcsomagot, mellyel a két programom kommunikálni tud. Ehhez vesszővel elválasztott számokból álló „üzenetet” használok.

Az első szám meghatározza a stílust. Ezekből kettő előre elkészített van a mikrokontroller programjában, de ezek később szabadon bővíthetőek. Egy ilyen stílus azt határozza, meg hogy hogyan történjen a ledmátrixhoz a frekvenciasávok hozzárendelése. Ez a szám egy globális változóba kerül, hogy minden metódus elérje.

Ezután következik kötelezően 9 darab szám, melyek RGB értékek. Ez három darab színt határoz meg. Minden megjelenítési stílusnak legalább ennyi színt használnia kell, az a stílustól függ, hogy ezek a színek pontosan mit változtatnak. Ezek az értékek is globális változókba kerülnek kiírásra a kezelőfüggvényben.

Ezt követheti még tetszőleges számú, de maximum 11*9 darab szín érték, ha az adott téma további színbeállítást is igényel. Ezek az a további értékek egy listába kerülnek, az adott séma onnan tudja felhasználni őket.

4.2.3 Időzítés

A megszakításokon alapuló rendszervezérlés miatt fontos volt egy globális rendszeridőzítés megvalósítása melyben körülbelül 25Hz-es képfrissítés elérése a cél. Tehát az SPI kommunikációt 40 ms-enként kell elvégezni. A globális időzítést az egyes megszakítások időzítésének módosításával lehet változtatni, azaz hogy azok mennyi ideig futnak. A legtöbb rendszermodulnak a paraméterezése nem változtatható, így a lefutási ideje is fix.

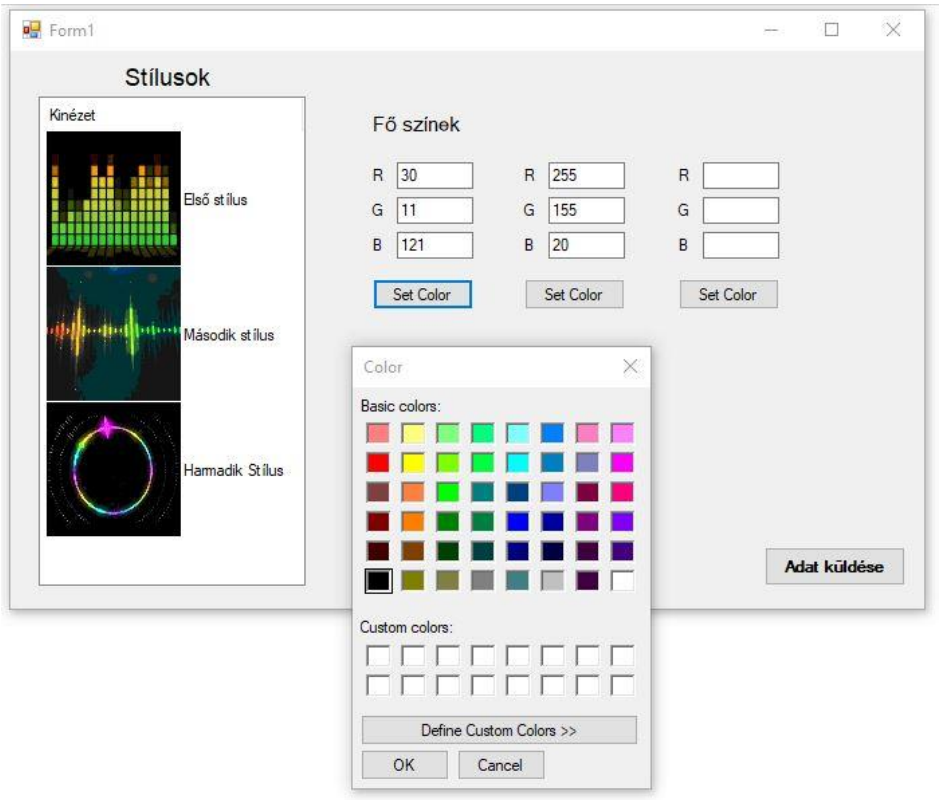
A rendszerben az egyetlen szabadon módosítható paraméter a vett minták száma. Ez a paraméter az analóg/digitális átalakítás idejét befolyásolja.

Az egy periódusban lévő összes minta begyűjtésének periódusideje 44Khz osztva a minták számával. Ahhoz hogy ez a modul 40ms- ideig fusson tehát 44KHz / 0.04s = 1100 mintát kellene venni. A minták száma viszont mindenképp kettő egész számú hatványa kell hogy legyen. Ehhez a legközelebbi érték az 1024, így a minták számát ekkorára választottam.

4.3 Konfiguráló program

A konfiguráló egy teljesen különálló grafikus program, melyet C# nyelven írtam, ugyanis ez a nyelv tartalmaz beépített metódusokat soros portok kezelésére, és a grafikus interfész elkészítése is könnyen megoldható a hozzá tartozó VisualStudio fejlesztőkörnyezet beépített GUI (Graphical User Interface) szerkesztőjével.

A program felülete viszonylag egyszerű, a kezdőképernyőn az üdvözlő szöveg mellett egy csatlakozás gomb található. Ennek megnyomása során a program csatlakozási üzenetet küld a kontroller felé, és annak válaszára vár. Ha a válasz nem érkezik meg akkor a csatlakozás sikertelen volt, ha megérkezik, akkor megnyílik a konfigurációs képernyő, mely az alábbi képen látható.



18. ábra Konfiguráló program

Ezen a képernyőn a bal oldali menüben kiválaszthatjuk a vizualizáció témáját. Ezután a jobb oldali panelben megjelenik a színbeállítások lehetősége. A felül látható három alapszín minden stílusnál kötelező elem. A színválasztás gombra kattintva egy új ablak jelenik meg, a hagyományos Windows-os színválasztóval. Miután a felhasználó megadta a szükséges paramétereket, elküldheti azokat a jobb alsó sarokban található küldés gombra kattintva.

Ezután a program a kapcsolat bontása üzenetre vár, miután ezt megkapta, tudja, hogy a küldés sikeres volt-e és jelzi ezt a felhasználónak, majd visszalép a kezdőképernyőre.

5 Tesztelés

A tesztelés az egész projekt megvalósítása alatt fontos szerepet kapott. Mivel az inkrementális fejlesztési technológiát követtem, ezért minden egyes lépés végén le kellett tesztelnem a rendszer helyes működését. Ráadásul nem volt elég az újonnan hozzáadott modul vagy periféria tesztelése, ugyanis az kihatással lehetett a korábban elkészített munkára. Ennek köszönhetően a fejlesztés előrehaladtával egyre nagyobb kódbázist kellett tesztelnem. A fejlesztési folyamatom, habár nem tudatosan, de hasonlóságot mutatott az elterjedt Teszt Alapú Fejlesztés (TDD, Test Driven Development) módszertannal. Egy fejlesztési lépcső első megvalósítása ritkán működött elsőre, ezután az elvégzett tesztek alapján próbáltam a funkcionalitását javítani. A hasonlóság azért nem volt tudatos, mert beágyazott rendszerek fejlesztésénél ez a módszertan nehezen követhető, mert nincsen jól működő teszt keretrendszer a C nyelvhez. Ezen kívül a technológiák sokszínűsége miatt a tesztelési folyamatba is több eszközt kellett bevonnom.

Ebben a fejezetben ezen eszközöket és szükségességüket, valamint a projektben való használatukat fogom ismertetni.

5.1 TrueSTUDIO fejlesztőkörnyezet hibakeresője

A legtöbbet használt eszköz a fejlesztőkörnyezet beépített debug eszköze volt. Az általam használt kártya lehetőséget biztosít arra, hogy a fejlesztőeszköz debug módban csatlakozhasson rá.

Ennek során a programban töréspontok (breakpoint) helyezhetőek el, melyekkel megállíthatjuk, léptethetjük annak futását. Eközben lehetőség van az aktuális memóriatartalom kiolvasásra is. Tehát minden a hagyományos fejlesztőkörnyezetek hibakereső módjával azonos, azzal a különbséggel, hogy itt ténylegesen a mikrokontrolleren fut a kód.

Ez a hibakeresési mód sok esetben nagy segítséget nyújtott. A megszakítás kezelő rutinokban elhelyezett breakpointtal jól tudtam tesztelni, hogy megfelelő sorrendben hívódnak e meg azok, vagy nincsenek-e felesleges megszakítások. Másik jó példa a használatára az ADC modul tesztelése volt. Ennek során meg tudtam minden

mintavétel után állítani a programot és ellenőrizni, hogy változtak-e a minták értékei abban az esetben, ha szól az audió tartalom vagy mikor le van állítva.

5.2 Soros porton keresztüli logolás

Már többször említettem a dolgozat során a kártyán található sorosportot, mely hibakeresési céllal is használható. Ezt a leghasznosabban különböző állapotok és értékek kiírására használható. Össze kell állítani egy üzenetet, majd egy egyszerű függvényhívással elküldeni azt. Majd egy terminál alkalmazással a soros portra csatlakozva minden küldött üzenet megjeleníthető.

```
sprintf(printbuffer, "AD=%d\r\n", adc.getValue());
HAL_UART_Transmit(&huart4, printbuffer, strlen(printbuffer), 5000);
```

Ezt a fentebb látható kódrészletet a mintavételezés tesztelésekor használtam. Minden egyes vett mintát kiküldtem a soros porton, így a terminál ablakban valós időben tudtam követni a vett minták alakulását annak függvényében, hogy hogyan változtatom a lejátszott tartalmat.

A projekt kezdeti fázisaiban nagy hasznát vettem, azonban később egyre kevesebbszer tudtam használni, ugyanis az üzenetküldés zavarja az SPI modul működését és elrontja annak időzítését, így azzal párhuzamosan nem használhattam.

5.3 Oszcilloszkóp használata

A projekt során többször is szükségem volt a rendszer bemenő vagy kimenő jeleinek megfigyelésére, mérésére. Ehhez oszcilloszkópot használtam.

A LED-ek számára előállított vezérlőjel ellenőrzéséhez tudtam hasznát venni. Ennek segítségével le tudtam mérni a jel időparamétereit, meg tudtam vizsgálni az előállított jelet grafikusan is. Nagy segítséget nyújtott az SPI modul elkészítése közben.

Ezen kívül az audió jel jelkondicionálásakor is ezt használtam a megfelelő jelszint megtalálásához.

6 Továbbfejlesztési lehetőségek, összegzés

A tervezési fázisban külön figyelmet fordítottam arra, hogy a projekt bővíthető maradjon. Ez mind az eszközválasztásra, mind a rendszer logikai felépítésére igaz. A mikrokontroller erőforrásai nincsenek teljes mértékben kihasználva, ezen kívül a megjelenítő típusa és a jel előállításának módja is olyan, hogy bármikor bővíthető egy nagyobb panellel, több LED-el.

A lehetőség tehát adott, és habár próbáltam a legfontosabb funkcionalitásokat lefedni a projekt során, mégis sok fejlesztési lehetőség van még.

Egy ilyen bővítési lehetőség, mely későbbi terveim közt szerepel, egy moduláris kijelző elkészítése, melynek a képernyő-aránya változtatható. Ez megvalósítható úgy, hogy kisebb LED panelek (3-6 LED) egymáshoz csatlakoztatásával lehetne egy kijelzőt „építeni” tetszőleges szélességgel és magassággal. A kijelző méretarányát a konfiguráló programban lehetne megadni.

A projekt során kitűzött célokat elértem. Sikerült megterveznem és megvalósítanom is egy a saját elvárásaim, igényeim alapján működő eszközt úgy, hogy eközben rengeteg új technológiával ismerkedtem meg.

A dolgozat témájának kitalálásakor még kevéssé voltam jártas a beágyazott rendszerek készítésében. Ez volt az első komolyabb ilyen rendszer, amit megvalósítottam. Viszont már korábban is vonzott a mikroelektronika, a beágyazott rendszerek világa. Tanulmányaim során ez kissebb hangsúlyt kapott, így ez egy ideális alkalom volt arra, hogy szélesítsem a technológiai ismeretimet és tapasztalatot szerezzek a mikrokontrollerek programozásában.

Irodalomjegyzék

[1] Winamp. (2017. 11 20). Letöltés dátuma: 2017. 11 20, forrás: Wikipedia: https://hu.wikipedia.org/wiki/Winamp

[2] Hangfeldolgozás. (2017). Letöltés dátuma: 2017. 11 28, forrás: http://itl7.elte.hu/hlabdb/hangfeld/hangf.html

[3] (2012), D. L. (2017. 11 28). Jelfeldolgozás és számítógépes irányítás. Letöltés dátuma: 2017. 11 28, forrás: Tankönyvtár: http://www.tankonyvtar.hu/en/tartalom/tamop412A/2010-0017_36_jelfeldolgozas_es_szamitogepes_iranyitas/ch04.html#id508513

[4] Diszkrét frekvencia analízis. (dátum nélk.). Letöltés dátuma: 2017. 11 25, forrás: Mechatronika, Optika és Gépészeti Informatikai Tanszék honlapja: http://www.mogi.bme.hu/TAMOP/mereselmelet/ch11.html#ch-XI.3

[5] Mikrokontroller. (dátum nélk.). Letöltés dátuma: 2017. 11 20, forrás: Wikipedia: https://en.wikipedia.org/wiki/Microcontroller

[6] STM32 Nucleo-64 board User manual. Letöltés dátuma : 2017. 11. 24., forrás: http://www.st.com/content/ccc/resource/technical/document/user_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/translations/en.DM00105823.pdf

[7] STM32F46RE Mikrokontroller Adatlap. Letöltés dátuma: 2017. 11. 25., forrás: http://www.st.com/content/ccc/resource/technical/document/datasheet/65/cb/75/50/53/d6/48/24/DM00141306.pdf/files/DM00141306.pdf/jcr:content/translations/en.DM00141306.pdf

[8] Cortex Microcontroller Software Interface Standard. Letöltés dátuma: 2017. 11 30, forrás: ARMDeveloper - https://developer.arm.com/embedded/cmsis

[9] Serial Peripheral Interface Bus. Letöltés dátuma: 2017. 12. 02, forrás: Wikipedia: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

[10] Direct memory access. Letöltés dátuma: 2017. 11 29, forrás: Wikipedia: https://en.wikipedia.org/wiki/Direct_memory_access