**Szegedi Tudományegyetem Informatikai Intézet**

Beltéri hangvezérelt világítás megvalósítása ESP32 mikrovezérlővel

Indoor voice-controlled lighting

using ESP32 microncontroller

Szakdolgozat

|  |  |
| --- | --- |
| Készítette: | Témavezető: |
| **Fodor Teodor** | **Dr. Mingesz Róbert** |
| mérnökinformatikus szakos hallgató | egyetemi adjunktus |

Szeged 2024

# Feladatkiírás:

ESP32 mikrovezérlővel számos IOT-projektet elkészíthetünk. A mikrovezérlő sokoldalúságának és könnyen proramozhatóságának köszönhetően a lehetőségek tárháza végtelen. Gyakori jelenség, hogy az épületek beltéri megvilágítását különböző elven működő érzékelővel vezérlik. A diplomamunkám célja, hogy a megszokott vezérléseket figyelmen kívül hagyva, egy sokkal kényelmesebb és képzeletet felülmúló, hangvezérelt megoldással álljak elő, amely lehetőleg gyors, könnyen felhasználható és alacsony hibaárányú. A dolgozat írása során természetesen törekszem megtalálni az arany középutat a felhasznált komponensek árában és minőségében, hiszen a legjobb ötlet sem valósítható meg a megfelelő céleszközök nélkül.

A feladat megvalósításának lépései:

1. Pontos specifikáció meghatározása
2. ESP32 fejlesztőkit és mikrofonok kiválasztása
3. Hardverek tesztelése (osszcilloszkóp, egyéb mérések)
4. Program implementálása
5. Program tesztelése
6. Áramkör elkészítése próbapanelen
7. Program javítása és bővítése

# Tartalmi összefoglaló:

## A téma megnevezése:

Beltéri hangvezérelt megvilágítás megvalósítása ESP32 mikrovezérlővel

## A megadott feladat megfogalmazása:

A feladat lényege, hogy egy két zónás led-világítás tudjunk hanggal vezérelni.

## A megoldási mód:

A kiválasztott ESP32 fejlesztőkit és mikrofonok közötti kommunikáció szoftveres megvalósítása.

## Alkalmazott eszközök, módszerek:

A programot a hivatalos, Espressif cég által biztosított ESP-IDF keretrendszert felhasználva készítem el, FreeRTOS-s környezetben. A kapcsolást először breadboard-on rakom össze, majd egy próbapanelen összeforrasztom (a két műveletet megelőzi egy áramkörtervezés).

## Elért eredmények:

Mind a program, mind az áramkör is elkészült, így a szobámban már megtalálható a működő prototípus.

## Kulcsszavak:

ESP-IDF, ESP32, I2S, INMP441, FreeRTOS

# Bevezetés:

A hangvezérlés egy olyan technológia, amely lehetővé teszi a különböző eszközök és alkalmazások vezérlését beszédhanggal. A hangvezérlést biztosító rendszerek használata egyre elterjedtebb a mindennapi életben, például okostelefonok, autók és egyéb elektronikus készülékek esetében, illetve ma már egy korszerű okosotthon szinte teljesen elképzelhetetlen hangvezérlés nélkül. A hangvezérlés egy olyan alternatíva, amely egy érintésmentes és kényelmes szolgáltatást biztosít a hagyományos vezérlési módszerekhez képest.

Felmérések alapján a felhasználók nagy része többnyire a következő 4 platform mellett teszi le a voksát:  Apple Siri, Samsung Bixby, Amazon Alexa vagy Google asszisztens. A felsorolt szolgáltatások mindegyikére jellemző, hogy online, azaz felhő-alapú hangfelismerő algoritmusokat futtatnak nagy teljesítményű szervereken, a platformokat támogató termékek (Amazon echo dot, Amazon echo studio, Apple Homepod, Samsung televízió, stb.) valójában egy mikrofonként szolgálnak, amelyek továbbítják a beszédhangot az előbb említett távoli számítógépeknek. Ezen rendszerek általában drágábbak illetve a használatukhoz egyszerre több terméket is meg kell vásárolnunk. A diplomamunkámmal azt próbálom bemutatni, hogy lehetőség van jutányos áron beszerezni, kiépíteni egy olyan rendszert, amellyel egyszerű eszközök is hangvezéreltté tehetők. Például egy Amazon Alexát támogató villanykörte körülbelül 30 euróba, a hozzá tartozó legegyszerűbb Amazon Echo Dot 50 euróba kerül - az ellátott feladatot természetesen teljesítik az eszközok, azonban az ár egy kissé magas (az Alexás termékek árai messze elmaradnak az Apple termékeihez képest) egyetlen funkcionalitás megvalósításához. Az elkészített projekttel be szeretném mutatni, hogy sokkal kevesebb anyagi ráfordítással is megteremthető a fenti kényelmi funkció.

Fontos megjegyezni, hogy ha az ember ESP32 mikrovezérlővel megvalósított hangfelismerés offline, azaz az algoritmus magán az eszközön fut, ezért egy erősebb chip megvásárlása szükséges, amelyben több memória található, illetve az órajele is sokkal magasabb. Az ilyen chipek ára még mindig messzemenőleg elmarad egy felhő-alapú rendszerhez képest. A legegyszerűbb megoldás az, ha az ember vásárol egy olyan fejlesztőkitet, amiben már megtalálható a mikrofon is, így nem kell foglalkozni a hang eljuttatásával a mikrovezérlőbe. Találtam több kitet is, amelybe már integrálva voltak a mikrofonok, azonban a beszerzésük hosszadalmas és drágább folyamat lett volna. Ezért úgy döntöttem, hogy külön-külön beszerzem a fejlesztőkitet és a két mikrofont(lehetőleg olyan típusút, amelyek a hivatalos kitekbe is integrálva vannak) és elkészítem a közöttük lévő kommunikációhoz szükséges illesztőprogramot. A hangfelismerő algoritmust nem én készítetettem, már része az ESP-IDF keretrendszernek, és sajnos nem nyílt forráskodú a framework ezen könyvtára.

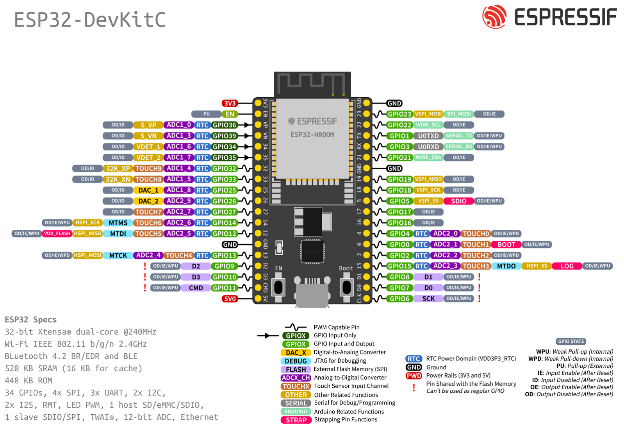
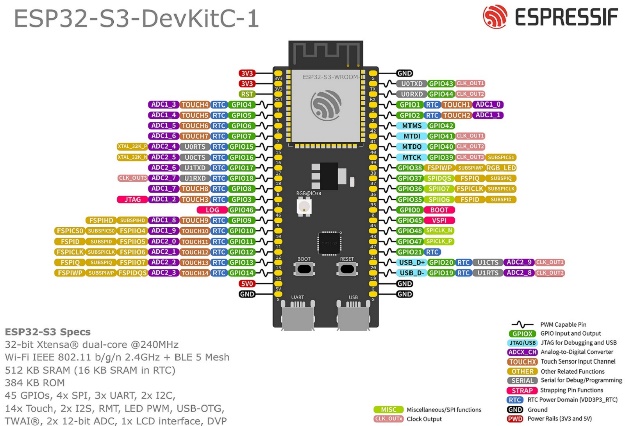
Nem titkolt célja a projektnek, hogy egy pénztárcabarát és könnyen használható, illetve konfigurálható eszközt tervezzek, amely egyszerűbb vezérlési feladatok elvégzésére képes.

# Hardverkörnyezet bemutatása:

A felhasznált hardverek kiválasztásakor alábbi négy szempont szerint jártam el:

Legyen költséghatékony, könnyen elérhető, lehetőleg gyors legyen a kiszállítás és ne legyen gyárilag hibás.

## A felhasznált mikrovezérlő és fejlesztőkit:

Az Espressif ESP termékcsaládja eddig is népszerű volt a hobbi építők és az olcsó IoT eszköz gyártók körében, révén hogy nagy tudása mellett, olcsó és könnyen programozható. Az elterjedt ESP8266 (illetve ESP8265 és változatai) leváltására érkezett az erősebb ESP32. Az ESP32 az elődjéhez képest sokkal több perifériával, nagyobb memóriával és magasabb órajellel rendelkezik. A programozható pinek száma megnőtt, illetve egyre több hálózati kommunikációs protokollt(Thread, Zigbee, Matter) is képesek támogatni a megszokott WiFi-n és Bluetooth-on kívül az újszériás ESP32 chipek. Az elmúlt évek során rengeteg változat jelent meg, mindegyik széria rendelkezik valamilyen specifikus képességgel, így az adott feladathoz megfelelően tudjuk kiválasztani a számunkra megfelelő chipet.

Ábra 1

Ábra 2

A diplomamunkámhoz az ESP32-S3 AI SoC-t választottam, amely a már jól bevált két processzormagos, akár 240 MHz-es órajel elérésére is képes Xtensa LX7 vezérlőegységet tartalmazza. A mikrovezérlőt kifejezetten AIoT alkalmazásokhoz gyártott le az Espessif Systems cég 2020-ban. Az ilyen chipet tartalmazó modulok különlegessége, hogy tartalmaznak egy külső PSRAM (Pseudostatic RAM) memóriaegységet, 2,4,8 és 16 MB-os kapacitással, verziótól függően. A PSRAM nagy méretének köszönhetően, lehetővé teszi a magas számításigényű mesterséges intelligencia algoritmusok (esetünkben hangfelismerő algoritmusok) futását, a nagy mennyiségű digitalizált hangadat pufferelését – kizárólag ezért használható csak az S3-as szériához a keretrendszer hangfelismerő könyvtára. A jelenlegi projekthez én ESP32-S3-DevKitC-1U-N8R8 chip-el ellátott ESP32-S3-DevKitC-1 v1.1 fejlesztőkitet választottam 8MB flash-memóriával és 8 MB PSRAM-al.

Ábra 3 – Espressif cég által gyártott külső PSRAM csip

## A felhasznált mikrofonok:

Annak érdekében, hogy a beszédhang digitalizásálával ne kelljen foglalkoznom, mindenképpen egy MEMS-technológiát használó mikrofont választottam, illetve egy olyan kommunikációs protokollt, amellyel könnyen le tudom implementálni a mikrofon és a fejlesztőkit közötti adatátvitelt. Ha körül nézünk a boltok polcain, akkor legtöbb ilyen típusú mikrofon az I²S kommunikációs szabványt támogatja. Az ESP-IDF pedig rendelkezik olyan könyvtárral, amely a fenti kommunikáció lebonyolításáért felelős.

### A MEMS-technológia:

A MEMS mikrofonokat vagy szilíciummikrofonokat már régóta előszeretettel használják okostelefonokban, okostévékben, távirányítókban. Ez nagy részben olyan hangvezérelt személyititkár-alkalmazásoknak köszönhető, amilyen a korábban is más sokszor emlegetett Amazon Alexa, a Google Assistant és az Apple termékek Siri-je. Ezek az alkalmazások olyan hangparancsokra várnak, amelyek ébresztik magát a rendszert, majd a parancs alapján hajtanak végre különböző utasításokat. A tervezők szűrőfunckiókat is integrálnak a mikrofonokba, hogy azok pontosan, jelentős környezeti zajban is messziről érzékeljék a beszédhangot.

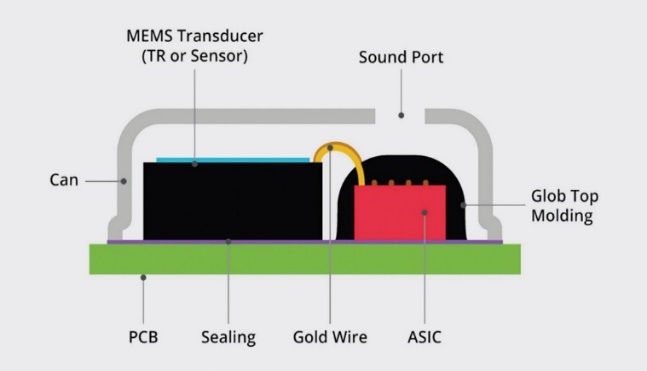
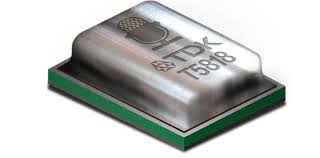
A MEMS mikrofonok jellemzően két, egyazon tokban elhelyezett alkatrészből állnak: egy MEMS membránból, amely átalakítja a hanghullámokat elektromos jellé, valamint egy erősítőből, amely impedanciaátalakító-ként működve használható analóg kimenőjelet ad a hangfrekvenciás jelfeldolgozó láncra. A harmadik alkotóelem – ha digitális kimenőjelre van szükség – egy analóg-digitális átalakító, amely szintén elhelyezhető ugyanabban a tokban, számunkra ez a fontos a projekt esetében.

A MEMS-mikrofonok fontosabb paraméterei a következők:

Jel-zaj viszony(signal-noise ratio, azaz SNR): a hasznos és a zavaró jel aránya dB-ben kifejezve. Minnél alacsonyabb, annál több zaj jelenik meg a digitalizált adatokban. Törekedni kell a nagyobb jel-zaj viszonyú mikrofonok kiválasztására, így az eszköz kevesbé „zajosabb”.

Frekvenciaátvitel: megadja azt a hangfrekvencia tartományt, ahol a mikrofon képes érzékelni a beszédhangot.

Dinamikatartomány: azon leghangosabb és leghalkabb hangnak megfelelő hangnyomásszint közti különbség, amelyek között a mikrofon jelleggörbéje lineáris.

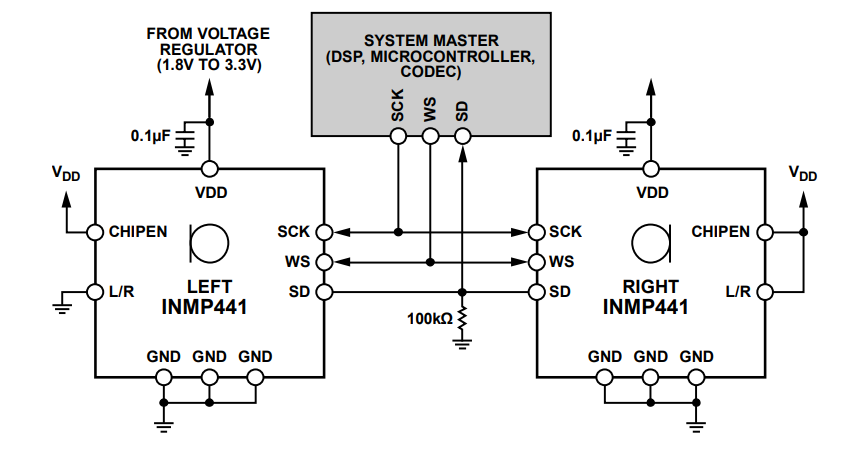
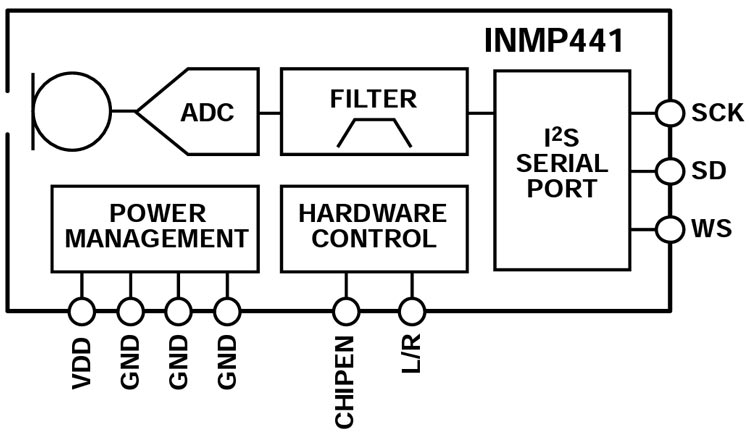
Érzékenység: megadja, hogy a mikrofon egy 1 kHz-es 94 dB hangnyomású hanghullámra adott analóg/digitális kimeneti értéket

Ábra 4

Digitális MEMS mikrofon használata esetén a mikrofon kimenőjele közvetlenül rávezethető digitális áramkörökre, általában egy mikrovezérlőre Az erős villamos zajú környezetekre tervezett hangvezérléses felhasználói felületek jellemzően inkább a digitális mikrofonokat részesítik előnyben, mert a digitális kimenőjeleknek nagyobb a zajtűrésük, mint az analógoknak.

Ábra 5

### INMP441:

Az INMP441 egy olcsó és könnyen beszerezhető, alacsony fogyasztású MEMS mikrofon modul. Fontos tulajdonsága, hogy I²S kimeneti interfésszel rendelkezik és omnidirekcionális, amely azt jelenti, hogy minden irányból egyformán érzékeli a hangot. A kimenete 24-bites és könnyen alkalmazható sztereó módban, azaz egyszerre 2 azonos típusú mikrofont is könnyen használhatunk, az architektúrájából kifolyólag szinkronizálva van az adatátvitel.

Ábra 6 – INMP441-el szerelt mikrofon modul

Ábra 7 az INMP441 típusú MEMS-mikrofon blokkadiagrammja

Ábra 8 – 2 INMP441 típusú mikrofon kapcsolása sztereó módban

A fenti ábrán jól látható a „L/R” pin a mikrofon blokkdiagramján – ennek segítségével konfigurálható, hogy az eszköz a „jobb” vagy a „bal” oldali csatornán szolgáltassa a

kimeneti adatot. Az alábbi ábra szemlélteti a sztereó-kimenet kapcsolását.

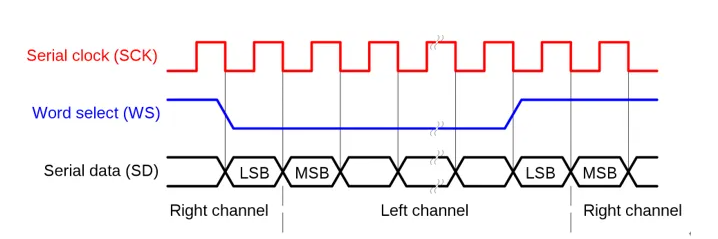
Jól látható, hogy az egyik mikrofon csip L/R pinje tápfeszültségre, míg a másik eszköz L/R pinje GND-re van csatlakoztatva. Az ábrán feltüntetett szűrőkondenzátorok és ellenállások már alapból megtalálhatóak a mikrofonos lapocskán, ezért használatuk elhanyagolható. Az ábra utolsó fontos eleme, a „SYSTEM MASTER” blokk esetünkben nem más, mint maga az ESP32 S3-as fejlesztőkit.

### I²S szabvány:

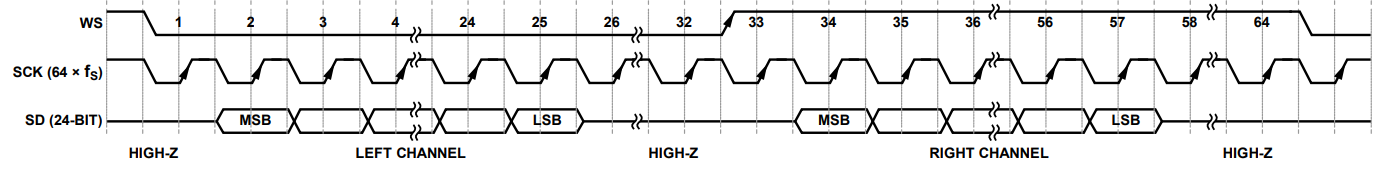
Az I²S (Inter-IC Sound) egy elektronikus soros busz interfész szabvány, amelyet digitális audioeszközök összekapcsolására használnak. Ezt a szabványt először a Philips Semiconductor vezette be 1986-ban. Digitális audió adatok átvitelére használják az elektronikus eszközök integrált áramkörei között. A kimenet PCM-formátumú.

Összesen 3 jelalak jellemzi:

* Bit órajel – az általunk használt mikrofon esetében ez az SCK-pinen (serial clock) jelenik meg. Feladata az átviteli adat bitjeinek az ütemezése
* Szó óravonal – formálisan ez a mikrofon WS-pinjének (word select) a jele. Feladata, hogy sztereó kimenet esetén kiválasztja, hogy a jobb vagy bal oldali csatornáról szeretnénk fogadni az adatot. Logikai alacsony szint esetén a bal-, míg logikai magas jelszint esetén a jobb csatornáról történik az adat olvasása. Leegyszerűsítve ez a mintavételi frekvencia egy szóra vonatkozóan, ami esetünkben 24-bit (32 bit).
* Soros adatvonal – ezen az adatvonalon jelennek meg a mikrofon által digitalizált adatok bájtjai. Az INMP441 mikrofon esetén ez az SD-pin(serial data).



Ábra 9 – Az I2s idődiagramja



Ábra 10 – Az INMP441 mikrofon I2S idődiagramja sztereó-kimenet esetén

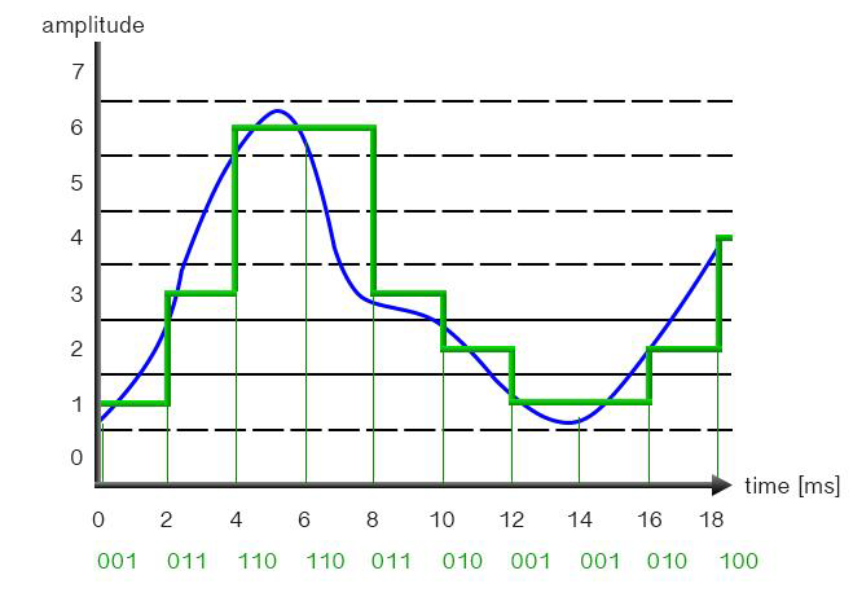
Adatlap alapján a mikrofon kimenetén 24-bites adatok jelennek meg, amelyek a legnagyobb helyiértékű bittel kezdődnek. A dolog érdekessége, hogy mégis 32 impulzus van egy fél-WS-ciklus(1 darab csatorna olvasása ) folyamán. Ez azzal magyarázható, hogy a mikrofon ilyenkor egy magas impedanciás állapotba kerül – ezek a bitek számunkra közömbösek és a szoftveres lekezelésénél is figyelmen kívül hagyjuk őket. Kicsit nehezebben észrevehető, de még mielőtt kiolvasnánk a digitalizált adat legnagyobhelyiértékű bitjét a WS-ciklus elején szükségünk van egy bit-órajel impulzusra, ez a formátum a „Philips formátum”. Az ESP-IDF I2S könyvtárában ez szerencsére gond nélkül konfigurálható.

Fontos még kiemelni, hogy egy teljes WS-ciklus alatt 64 SCK impulzusnak kell megjelennie annak ellenére, hogy csak 24 bit a számunkra értékes információ.

### PCM – Impulzus-kódmoduláció:

Az impulzus-kódmoduláció a távközlésben régóta széleskörben használt módszer analóg jelek digitálissá alakítására és átvitelére egy digitális csatornán. A módszer két fő komponense az időbeli diszkretizálás, a mintavételezés, és a minták értékeinek diszkretizálása, azaz kvantálása.

* Mintavételezés: Az analóg jelből a mintavételezési frekvenciának megfelelő gyakorisággal mintát vesznek. A mintavételezés szigorúan a Shannon-mintavételezési törvénynek megfelelően kell, hogy történjen: az eredeti jel rekonstruálásának érdekében a mintavételezési frekvencia legalább kétszerese kell legyen a mintavételezett jel frekvenciájának.
* Kvantálás: A mintavételezett értékeket diszkrét szintekre (kvantumszintekre) kerekítik. Ez a folyamat során minden mintát a legközelebbi kvantumszinthez rendelnek hozzá, ami bizonyos fokú pontatlanságot, az úgynevezett kvantálási zajt eredményez.
* Kódolás: Utolsó mozzanatként a kvantált értékeket binárissá alakítjuk impulzusok reprezentálásával.

Az ábrán kék színnel van jelölve az analóg, mintavételezett jelünk. Zölddel vannak jelölve az egyenlő időközönként kvantált értékek, a vízszintes tengely alatt található zöld értékek pedig a bináris kódok. Például a 4-es és 8-as időpillanat között a kvantált érték 6, ezért ezen az időtartományon belül „110” bináris értékkel kódoljuk a jelet. A 16-os és 18-as időpillanatban a kvantált érték 2, amit a bináris „010” értékkel reprezentálunk.

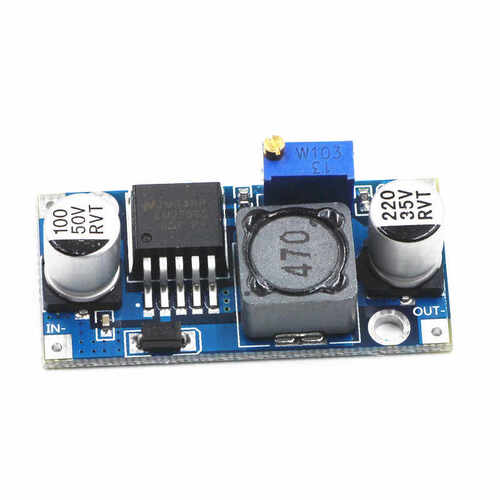
Ábra 11-PCM-moduláció

## A felhasznált LED szalag:

Végül, de nem utolsó sorban a megvilágításért felelős LED szalagról teszek említést. A tervek szerint a szalagot a szobámban található két szekrény hátoldalán szeretném elhelyezni. Miután lemértem a szekrények hosszait és felragasztottam őket, ehhez a szalagon található ragasztón kívül még univerzális ragasztót is felhasználtam a biztos tapadás érdekében. Összesen 3 méter szalagra volt szükségsem, amelyekről a gyári csatlakozót le kellett cserélnem csupasz vezetékekre.

A műszaki paramétereket áttekintve egy 12 V-on működő, méterenként 120 LED diódával szerelt szalagról van szó, 3000K színmélységgel („ún. meleg fehér” a szalag által emittált fénysugár). Számomra fontos paraméter volt a dimmelhetőség, mivel egyszere 5-6 különböző fényerősséget szerettem volna felkínálni, amelyeket a hangparancsokkal lehet változtatni. A különböző fényerősségek eléréséhez természetesen PWM-jelet alkalmaztam.

Ábra 12

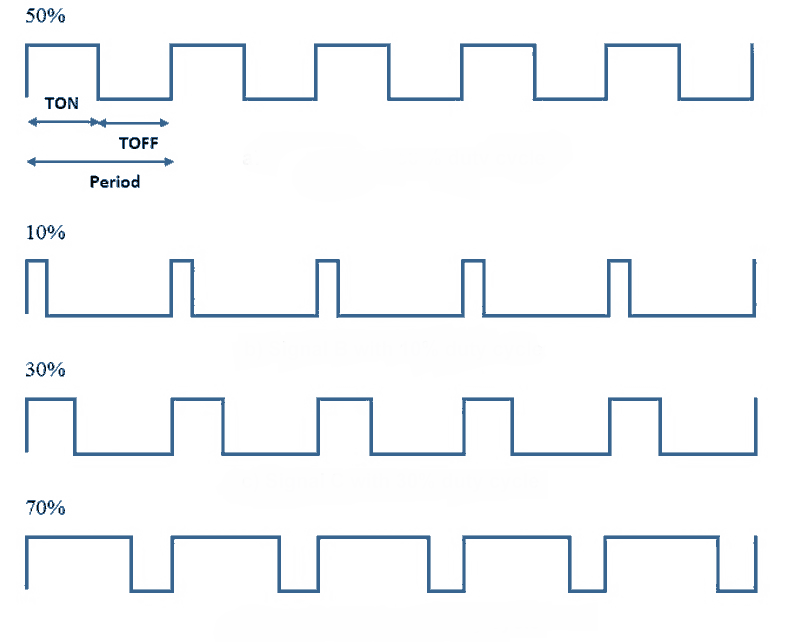
Kihagyhatatlan volt a felhasznált teljesítmény és áramerősség meghatározása is, hiszen a rendszerhez egy megfelelő méretű tápegységet is be kellett szereznem. . Mérések és kísérletezés alapján a két LED szalag egyenként 750 mA-t fogyasztott(maximális fényerősség esetén), illetve a mikrovezérlő elenyésző fogyasztását is belaszámoltam, aminek eredménye egy 3A áramerősségű 12 voltos tápegység lett. (230V hálózati feszültségről alakít). A fejlesztőkit tápfeszültséggel való ellátásáért egy LM2595 DC-DC konvertert használtam, melynek kimeneti feszültségét 3.3V-ra állítottam.

Ábra 13

Ábra 14

### PWM, a digitális analóg jel:

Azok az idők, ahol a fényt csak arra használták, hogy egy teret megvilágítsanak, rég elmúltak. A modern otthonokban már elengedhetetlenek a dimmelhető, azaz fényerejüket szabályozható LED égők és szalagok. Elsősorban a költséghatékonyság érdekében hozták létre a technológiát, ezzel is csökkentve az áramfogyasztást.

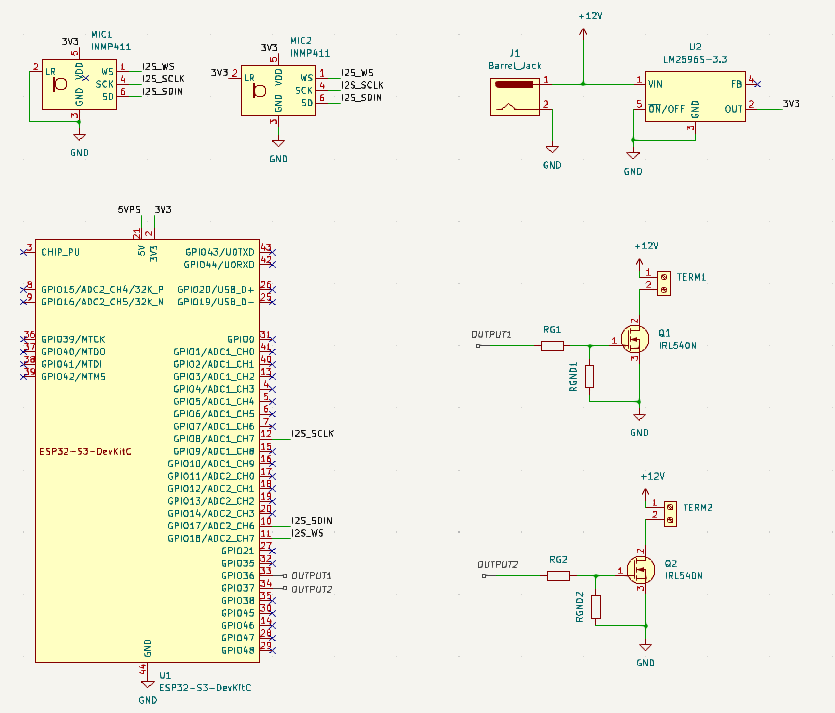
A dimmelhetőség mögött rejlő technológia a PWM moduláció. Ennek elve az, hogy az analóg kimenő feszültségjelek helyettesíthetők digitális impulzussorozat-jelekkel, amelyek hosszabb időtartamra vonatkoztatott átlagfeszültsége egyenértékű az analóg feszültségjellel: fényerő intenzitását azzal szabályozzák, hogy a PWM-jel egy periódusa alatt milyen hosszú ideig van bekapcsolt és kikapcsolt állapotban a világítótest, ezt nevezzünk kitöltési tényezőnek. Ezen egyszerű eljárás az emberi szem „lassú” reakciódejejét használja ki: szemünk a 100 Hz-es villogást már képtelen érzékelni, ezért látjuk folytonosnak ha világít a LED szalag.

Ábra 15 – Különböző kitoltési tényezőjű PWM-jelek

## Egyéb felhasznált eszközök:

* Műanyag doboz – 95 x 60 mm méretű, az áramkör tárolására
* Próbapanel – a dobozhoz mérten méretre vágtam, erre forrasztottam az áramkört
* IRL540N logikai MOSFET – a két LED szalag meghajtására használtam. Önmagában az ESP32 fejlesztőkit GPIO lábai nem képesek 12 voltos eszközöket meghajtani. A tranzisztor threshhold feszültszége 1.8-1.9 V, ami könnyen kapcsolható kit 3.3 V-os pinjeivel.
* Ellenállások – 100 Ω és 10 kΩ értékkel
* 2.54mm lábtávolságú aljzat: a fejlesztőkit és a mikrofonok könnyebb rögzítéséhez és esetleges cseréjéhez
* DC-csatlakozó – egy tápegység, egy csatlakozó. Az egyszerűség kedveért barrel-jack-t használtam
* Kétutas csavaros sorkapocs – ezzel kötöttem a LED szalagokat az áramkör kimeneteire

## A felépített áramkör:

A fenti ábra alaposan bemutatja az általam tervezett áramkört. A program, amelyet használtam a kapcsolási rajz elkészítéséhez a nyílt-forráskódú és ingyenesen letölthető KiCad. A könnyebb átláthatóság érdekében a vezetékek helyett címkéket használtam (pl. OUTPUT1, I2S\_SDIN, stb.). Szeretném kiemelni, hogy az áramkör 3 dimenziós modelljét, amely a nyomtatott áramköri lapot tartalmazz, nem készítettem el, ezért is érdemesebb a rajzra egy funkcionális diagramra tekinteni, amely a forrasztás során segített létrehozni a megfelelő kapcsolásokat(erre jó példa a LM2596 DC-DC átalakítónak csak a integrált áramköri komponense van feltüntetve az ábrán). A sokszor felhasznált részeket (+12v, GND) egységesítettem.

Ábra 16

Jobb felső sarokban található az áramkör tápellátásért felelős részegység. A J1-es jelzésű BarrelJack-en keresztül tápláljuk meg 12 volttal a rendszert a 13-as ábrán található AC-DC tápegységgel. Ezt a 12 volt feszültséget használjuk fel a LED szalagok meghajtására illetve közvetett módon a fejlesztőkit áramellátására is: az LM2596 DC-DC átalakító a 12 voltot 3.3 voltra alakítja át, amely a fejlesztőkit 3V3 kivezetésére csatlakozik.

Az ábra bal alsó sarkában látható a ESP32 S3 fejlesztőkit. Az „I2S” előtaggal rendelkező kivezetéseket szabadon választottam, ügyelve arra, hogy ne az eszköz működéséhez szükséges pineket (BOOT, GND, RST, stb.) használjam. Hasonló módon jártam el az OUTPUT1 és OUTPUT2 pinek esetén is, általános célú lábakat választottam, amelyek a későbbiekben a váltakozó kitöltési tényezőjű PWM-jeleket juttatják el a logikai MOSFET tranzisztorokra.

Az ábra bal felső sarkában található a két darab INMP441-es omnidirekcionális mikrofonmodul. Ezeket is LM2596-os feszültségkonverter látja el tápfeszültséggel, adatlapi értékek alapján maximálisan 3.6V-al táplálhatóak. A korábban említett sztereó konfiguráció érdekében az egyik L/R kivezetését a rendszer földpontjára, míg a másikat 3.3 V feszültségre kötjük. A további három vezeték az I2S által szüséges jelek továbbítására szolgál, amelyek az I2S\_WS (word select), I2S\_SD(serial data) és I2S\_SCK (serial clock).

Az áramkör utolsó részegysége a LED szalagok meghajtásáért felelős aláramkör a kapcsolási rajz jobb alsó sarkában. Az RG1 és RG2 ellenállások 100 Ω-sak, míg az RGND1 és RGND2 10kΩ-sak. Az RGND1 és RGND2 ellenállásoknak lehúzó ellenállásként funkcionálnak, így biztosítva azt, hogy a térvezérlésű tranzisztor gate-jén vagy logikai magas(3.3V) vagy logikai alacsony (0V) feszültség jelenik meg – a PWM jel bináris értékeinek és a kiválasztott tranisztor karakterisztikájának alapján kapcsolóüzemű a meghajtó áramkör Továbbá a mikrovezérlő pinjeinek a védelmére is szolgál, hiszen sohasem lehetünk biztosak benne, hogy a program inicializációs szakaszát megelőzően milyen állapotban lesznek. Az RG1 és RG2 ellenállás megléte nem kötelező, hiszen a logikai térvezérlésű tranzisztorokat elméleti szinten kizárólag feszültséggel vezéreljük, azonban tudjuk, hogy a gyakorlatban némi áram is megjelenik a mosfet gatején – a két ellenállások ezeknek a töltéseknek az elszivárogtatásáért felelős. Noha a egy feszültségosztó áramkört kapunk, az RG1 és RG2 ellenállásértékei elenyészőek a 10kΩ-hoz képest, ezért a jelenlétük nem fogja zavarni a működést:

A kapott „V” feszültség érték messze meghaladja az IRL540 fet threshold feszültéségét(~ 1.8-1.9 V). A tranzisztor drain kivezetése a csavaros sorkapocs egyik lábára csatlakozik, természetesen arra, amelyre a LED szalag negatív polarítású vezetékét kötöttem, míg másikra, amelyen a 12V-os tápfeszültsg van, a szalag pozitív vezetéke került bekötésre. A tranzisztor N-csatornás, így a PWM-jel a szalag földpontját „szaggatja” nagyfrekvencián (dimmeli), a váltakozó kitöltési tényező pedig a jel effektív értékét, azaz a fényerőt szabályozza.

# Szoftverkörnyezet bemutatása:

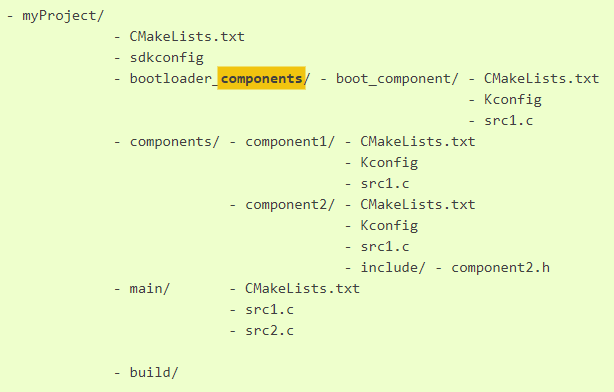
## Espressif IoT Development Framework:

Az ESP-IDF a kínai Espressif cég által fejlesztett keretrendszer, amelyet az ESP8266- és ESP32 sorozatú mikrovezérlők programozására használunk. Az szoftvercsomag rengeteg API-t kínál, amellyel kényelmesen programozható a mikrovezérlő minden egyes perifériája és felkínált szolgáltatása C és C++ programozási nyelveken egyaránt. Fontos kiemelni, hogy maga a keretrendszer és annak szoftverkomponensei a FreeRTOS valós-idejű alapjaira vannak lefektetve, kivétel nélkül mindegy egyes Espressif-es csipet támogat.

A keretrendszer, hogy nagy előnye, hogy teljese nyílt-forráskodú, teljesen szabadon megtekinthetőek a forrásfájlok a hivatalos GitHub repository-ban, emellett jól dokumentált, minden apró részletre kitér és, ami a legfontosabb, hogy az egyes könyvtárakhoz példaprogramokat is bemutat, amelyek a dokumentáció olvasásával kombinálva megkönnyítik és felgyorsítják a fejlesztést.

## Build System:

Az IDF projekt struktúrája a már korábban említett komponensekre (components) épül: például a Wifi-vagy Bluetooth kommunikációért felelős API és ezzel együtt minden felkínált vagy a szoftverfejlesztő által készített component az alábbi struktúrát kötelezően követi:

A „components” egy kitüntetett direktórium, ide került minden egyes építőkocka, amelyek szükséges a program futásához. Az IDF a CMake build rendszert használja, amely a C/C++ programokódot tartalmazó forrásfájlok buildelési folyamat konfigurálásáért és az automatizálásáért felelős.

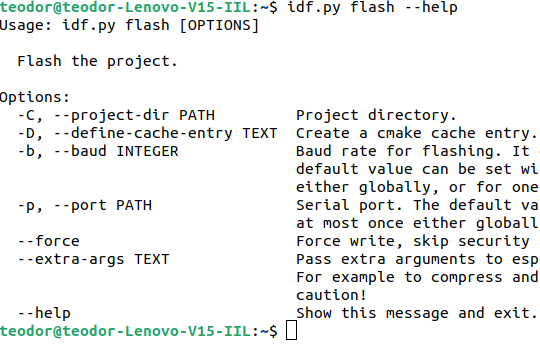
Ábra 17 – egy esp-idf példaprojekt struktúrája

Megjegyzem, hogy nem csak „hagyományos” CMake parancsokat használhatóak, az idf ún. „wrapper” parancsokat támogat. A „CMakelists.txt” fájlok alapján a Ninja elnevezésű build tool ezután buildeli a projektet, nagy előnye, hogy sokkal gyorsabb, mint egy hasonló, hagyományos tool, mivel kihasználja a többmagos processzorok erejét

Ábra 18 – a projekt egyik komponensének CMake fájlja

és párhuzamosítási eljárásokat alkalmaz. A ninja által készített fájlok a „build” mappában tárolódnak: itt találhatóak meg az object fájlok (a fordított program bináris alakban), a linkelt, azaz végrehajtható és a csipre feltölthető program és egyéb CMake fájlok is.

## idf.py:

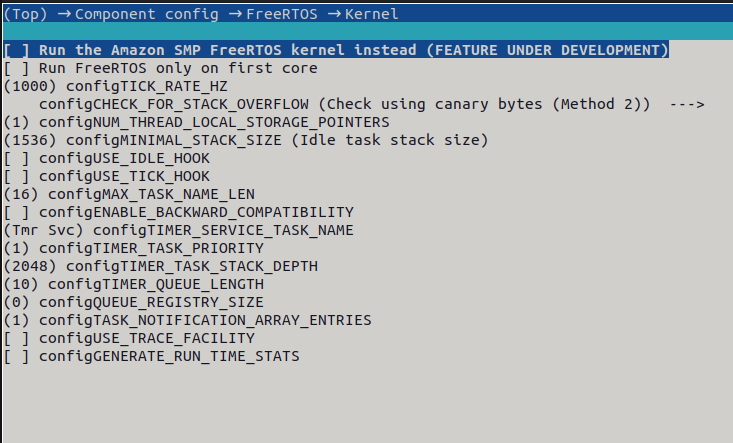
Ez egy parancssori-eszköz (command-line tool), amely egy egyszerű kis frontend felületet biztosít a projekt buildeléséhez, konfigurálásához, monitorozásához illetve az elkészített firmware a mikrovezérlőre való feltöltéséhez is.

Ábra 19

## A projekt konfigurálása:

## Az IDF-projekteket az idf.py eszközzel konfigurálhatjuk, melynek eredményeket áll elő az „sdkconfig” fájl. Ezen fájl segítségével az általános beállításoktól a komponensekre vonatkozó specifikus beállításokig szinte minden szabadon változtatható.

## Ha például az ESP32-modul flash memóriájának a partíciónálását szeretnénk testreszabni, akkor készítenünk kell egy .csv kiterjesztésű fájl a projektben belül majd az elérési útvonalát megadni a konfigurátor interfészében. Ezen kívül a FreeRTOS órajelét, wifi-pufferek számát, méretét a mikrovezérlő frekvenciáját változtathatjuk és. egyedi konfigurációs beállításokat is létrehozhatunk (ezt egy Kconfig fájlban kell elkészíteni). A konfigurátor segítségével a programozott ESP32 csip szériája is, így az adott sorozatra vonatkozó(és nem vonatkozó) specifikus beállítások automatikus kerülnek felkínálásra.



Ábra 20 – az idf.py menuconfig parancs futtatása után megjelenik a konfigurátor interfész, jelen esetben a FreeRTOS-ra vonatkozó opciókat láthatjuk