

TAR DÁNIEL  
SZAKDOLGOZAT

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM  
GÉPÉSZMÉRNÖKI KAR  
MECHATRONIKA, OPTIKA ÉS GÉPÉSZETI INFORMATIKA TANSZÉK



SZAKDOLGOZATOK



BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM  
GÉPÉSZMÉRNÖKI KAR  
MECHATRONIKA, OPTIKA ÉS GÉPÉSZETI INFORMATIKA TANSZÉK

TAR DÁNIEL  
SZAKDOLGOZAT  
**Android vezérlésű okos LED-rendszer  
fejlesztése**

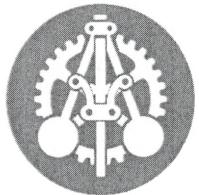
*Development of android controlled wireless LED-lighting system*

Témavezető:

*Szakály Norbert*  
tanszéki mérnök

Budapest, 2018

Szerzői jog © Tar Dániel, 2018.



# SZAKDOLGOZAT FELADATKIÍRÁS (BSc)

<b>AZONOSÍTÁS</b>	Név: Tar Dániel			Azonosító: 76190778292
	Képzéskód:	2N-AM0	Specializáció kódja:	Feladatkiírás azonosító:
	Szak:	Mechatronikai mérnöki alapképzési szak		2N-AM0-MB-2010 SZD-GEMI-2018/19/1-gutoy7
	Szakdolgozatot kiadó tanszék:	Záróvizsgát szervező tanszék:		
	Mechatronika, Optika és Gépészeti Informatika			Mechatronika, Optika és Gépészeti Informatika
Témavezető: Szakály Norbert tanszáki mérnök, 71528961780, szakaly@mogi.bme.hu, 463-2945				

<b>FELADAT</b>	Cím	Android vezérlésű okos LED-rendszer fejlesztése Development of Android controlled wireless LED-lighting system
	Részletes feladatak	<ol style="list-style-type: none"> <li>Okos LED-rendszerek felkutatása, forgalomban lévő eszközök áttekintése.</li> <li>Tervezési feladat követelményeinek felállítása.</li> <li>Eszköz (elektronikai és beágyazott szoftverének) tervezése és összeállítása.</li> <li>Androidos alkalmazás elkészítése.</li> <li>Elkészült eszköz értékelése, költségterv számolása.</li> </ol>
Hely	A szakdolgozat készítés helye: BME MOGI Tanszék 1111 Budapest, Bertalan Lajos u. 4-6. „D” ép. IV. em. Konzulens:	

<b>ZÁRÓVIZSGA</b>	1. záróvizsga tárgy(csoport)	2. záróvizsga tárgy(csoport)	3. záróvizsga tárgy(csoport)
	<b>Mechatronika</b> BMEGEFOAMM1 (3 kr) BMEGEFOAMM2 (3 kr)	<b>Analóg és digitális technika</b> BMEVIAUA009 (3 kr) BMEVIAUA010 (3 kr)	<b>Automatika</b> BMEGEFOAMA2 (5 kr) BMEGEMIAMG2 (3 kr)

<b>HITELESÍTÉS</b>	Feladat kiadása: 2018. szeptember 3.	Beadási határidő: 2018. december 7.
	Összeállította:   témavezető	Ellenőrizte:  Mechatronika, Optika és Gépészeti Informatika Tanszék tanszékvezető/tanszékvezető-h.
	Alulírott, a feladatkiírás átvételével egyúttal kijelentem, hogy a Szakdolgozat készítés c. tantárgy előkötélményeit maradéktalanul teljesítettem. Ellenkező esetben tudomásul veszem, hogy a jelen feladatkiírás és a tárgy felvétele érvényét veszti.	
	Budapest, 2018. szeptember 3.	 ..... hallgató

# NYILATKOZATOK

## *Elfogadási nyilatkozat*

Ezen szakdolgozat a Budapesti Műszaki és Gazdaság tudományi Egyetem Gépész mérnöki Kara által a Diplomatervezési és Szakdolgozat feladatokra előírt valamennyi tartalmi és formai követelménynek, továbbá a feladatkiírásban előírtaknak maradéktalanul eleget tesz. E szakdolgozatot a nyilvános bírálatra és nyilvános előadásra alkalmasnak tartom.

A beadás időpontja: 2018. december 13.



témavezető

## *Nyilatkozat önálló munkáról*

Alulírott, Tar Dániel (GUTOY7), a Budapesti Műszaki és Gazdaság tudományi Egyetem hallgatója, büntetőjogi és fegyelmi felelősségem tudatában kijelentem és sajátkezű aláírásommal igazolom, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, és dolgozatomban csak a megadott forrásokat használtam fel. minden olyan részt, melyet szó szerint vagy azonos értelemben, de átfogalmazva más forrásból átvettettem, egyértelműen, a hatályos előírásoknak megfelelően, a forrás megadásával megjelöltem.

Budapest, 2018. december 13.

  
Tar Dániel  
szigorló hallgató

# TARTALOMJEGYZÉK

<b>1. Bevezetés</b>	<b>1</b>
1.1. Célkitűzések . . . . .	1
1.2. Áttekintés . . . . .	1
<b>2. Okos LED-rendszer felkutatása, forgalomban lévő eszközök áttekintése</b>	<b>2</b>
2.1. LED világítás és előnyei . . . . .	2
2.1.1. Energiatakarékosság . . . . .	2
2.1.2. Jó színvisszaadás . . . . .	2
2.1.3. Környezetbarát . . . . .	3
2.1.4. Széles működési tartomány . . . . .	3
2.2. Forgalomban lévő eszközök áttekintése . . . . .	3
2.2.1. Philips HUE . . . . .	4
2.2.2. LIFX . . . . .	4
2.2.3. TP-Link okos villanykörte . . . . .	4
2.2.4. Általános vezérelhető LED szalag . . . . .	5
<b>3. Tervezési feladat követelményeinek felállítása</b>	<b>7</b>
3.1. Rendelkezésre álló hardverek . . . . .	7
3.1.1. Wifi modul: ESP8266 - ESP01 . . . . .	7
3.1.2. WS2811-es IC-vel ellátott címezhető LED szalag . . . . .	8
3.1.3. Mikrovezérlő: STM32F103C8T6 . . . . .	9
<b>4. Eszköz elektronikai tervezése és összeállítása</b>	<b>11</b>
4.1. Elektronikai tervezés . . . . .	11
4.1.1. DC/DC tervezés . . . . .	13
4.1.2. Logikai jelszint átalakító tervezése . . . . .	16
4.1.3. STMF103C8T6 mikrokontroller és perifériák . . . . .	17
4.1.4. Alkatrészek megrendelése és a NYÁK legyártatása . . . . .	20
4.2. Az elkészült eszköz beforrasztása, tesztelése . . . . .	21
4.3. Védődoboz tervezése, 3D-nyomtatással prototípus gyártása . . . . .	22
4.3.1. Védődoboz 3D modellezése . . . . .	22
4.3.2. 3D nyomtatás és az elkészült hardver . . . . .	23
<b>5. Beágyazott szoftver elkészítése</b>	<b>25</b>
5.1. STM32F103C8T6 mikrokontroller inicializálása . . . . .	26
5.2. LED sorral való kommunikáció . . . . .	27

5.2.1. PWM jel létrehozása a kimeneteken . . . . .	28
5.2.2. PWM vezérlése DMA segítségével . . . . .	29
5.3. ESP8266-os Wifi Modul . . . . .	30
5.3.1. A Wifi modul bekonfigurálása . . . . .	30
5.3.2. Mikrokontroller felkészítése az UART-on való kommunikálásra	32
5.3.3. UART vezérlése DMA segítségével . . . . .	32
5.4. Az ESP8266-os modulról érkező adatok és a LED sor vezérlés összekötése . . . . .	33
<b>6. Androidos alkalmazás elkészítése</b>	<b>35</b>
6.1. Androidról általában . . . . .	35
6.2. Android-os szoftverfejlesztés megismerése . . . . .	35
6.3. Android alkalmazás rövid felépítése . . . . .	36
6.3.1. Services . . . . .	36
6.3.2. Activities . . . . .	36
6.3.3. Broadcast Receivers . . . . .	36
6.4. Az elkészült alkalmazás funkciói és felhasználói kézikönyv . . . . .	37
6.4.1. LED sor IP-címének a beállítása . . . . .	38
6.4.2. Simple Color Picker mód . . . . .	40
6.5. Color Palette mód . . . . .	41
6.6. Party mód . . . . .	41
6.7. Audio Visualizer . . . . .	41
6.8. Beállítások menü . . . . .	43
6.8.1. Helyi hálózati IP cím beállítása (65. és 66. ábra) . . . . .	43
6.8.2. Vizuális értesítők bekapcsolása . . . . .	43
6.8.3. Automata színbeállítás a Simple Color Picker módhoz . . . . .	45
<b>7. Összefoglalás: Elkészült eszköz értékelése és költségterv számolása</b>	<b>46</b>
7.1. Az elkészült eszköz értékelése . . . . .	46
7.2. Költségterv számolása . . . . .	47
7.3. Fejlesztési lehetőségek . . . . .	47
<b>Hivatkozások</b>	<b>49</b>
<b>Summary</b>	<b>54</b>
<b>A. Függelék</b>	<b>55</b>
A.1. Az elkészült LED sor vezérlő kapcsolási rajzai . . . . .	55

# ELŐSZÓ

Szakdolgozatomban egy kollégiumi szobában kigondolt gólyakori álmot váltottam valóra. Megismertem ez által egy összetett, szerteágazó rendszer egyes építőelemeit, és megszereztem a létrehozásához szükséges tudást.

Elsősorban köszönettel tartozom, témavezetőmnek, Szakály Norbert Tanár Úrnak, sok segítségért és támogatásáért. Köszönöm szüleimnek, akik mindig a precíz és kitartó munkára neveltek.

Budapest, 2018. december 13.

*Tar Dániel*

# 1. BEVEZETÉS

## 1.1. Célkitűzések

A céлом egy olyan feladat megvalósítása volt, amely során képes leszek komplex rendszerek tervezésére, értelmezésére, illetve széleskörű tudásra tehetek szert. Több közül végül még egy kollégiumi szobában megfogalmazott feladat mellett döntöttem. Az elképzelésünk az volt, hogy az Ebay-en fellelhető olcsó LED-szalagokhoz készítsek egy olyan hardvert, amit az én és a szobatársam Android-os mobiltelefonjával tudunk vezélni, illetve állapotokat megjeleníteni (például hívás, SMS érkezését) Wifi-n keresztül. Továbbá szerettük volna, hogy zenére és a mobiltelefon mozgatására is tudjon villogni.

A szakdolgozat írásakor, kivitelezésekor törekedtem az átlátható logikus gondolkodásra, szerkezeti tagolásra. Programozásnál igyekeztem a clean code elvek betartására, és az adott nyelvnek megfelelő elnevezési konvenciók alkalmazására [1][2][3].

## 1.2. Áttekintés

Az első fejezetben ismertetem az interneten fellelhető, forgalomban lévő LED-es eszközöket, rendszereket. A második fejezetben kitérek az elektronikai tervezésre, megvalósításra és a védődoboz elkészítésére. A hardver után beszámolok a beágyazott szoftver implementációjáról és a külső eszközökkel való kommunikációról. A következő részben magáról az Android-ról, illetve az alkalmazás szerkezetéről fogok írni. Végül egy felhasználási útmutatóban keresztül ismertetem az alkalmazás működését.

## 2. OKOS LED-RENDSZEREK FELKUTATÁSA, FORGALOMBAN LÉVŐ ESZKÖZÖK ÁTTEKINTÉSE

Az evolúció során az emberi szem úgy fejlődött ki, hogy nappali fényhez - világoshoz - gyorsan alkalmazkodik, és éles képet alkot. Sötéthez, szürkülethez csak lassan képes alkalmazkodni és akkor sem éles az a kép, amit látunk. A fejlődő világunkban nélkülözhetetlenné vált a világítás, hogy napnyugta után sem álljon meg az élet és tovább tudjuk folytatni tevékenységeinket.

Az általam készített termékkel főként háztartásokban lévő okos világítás megvalósítása a cél, minél energia-gazdaságosabb módon. Erre a legmegfelelőbb technológia a LED világítás[4].

### 2.1. LED világítás és előnyei

A fényt kibocsájtó diódák (angolul: light-emitting diode - LED) a mai legenergiatakarékosabb és leggyorsabban fejlődő világítástechnológiák közé tartoznak. Általánosságban igaz az, hogy a LED-es izzók tovább bírják, ellenállóbbak és hasonló vagy még jobb minőségű megvilágítást biztosítanak, mint más fényforrások. [5]

A következőkben a LED-es világítások előnyeit foglaltam össze.

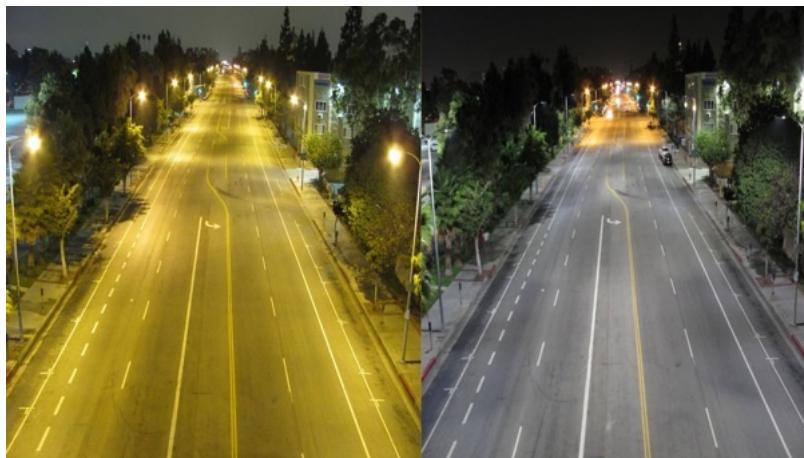
#### 2.1.1. Energiatakarékkosság

A LED-es izzók legalább 75%-kal kevesebb energiát használnak és 25x hosszabb ideig bírják, mint a hagyományos izzólámpák. A LED-es izzók elterjedése lenne az egyik legnagyobb hatással az energiatakarékkosságra. Az USA-ban például ezzel 2027-ig 348 TWh elektromos munkát (egy éves teljesítménye 44, egyenként 100 Megawattos erőműnek) spórolhatnánk meg, ahhoz képest, mintha nem használnánk egyáltalán LED-es izzókat, ami több mint 30 milliárd dollár lenne.

A hagyományos izzók teljes teljesítményének 90%-ából hő termelődik, ami azt jelenti, hogy csak 10%-a fordítódik fénykibocsátásra. A LED-ek esetében alig termelődik hő. [5]

#### 2.1.2. Jó színvisszaadás

A színvisszaadási index, röviden 'CRI' (Color Rendering Index), a fényforrás azon képességét méri, hogy különféle tárgyakat megvilágítva vele, mennyire képes azok színét visszaadni. [6]



**1. ábra:** Utcai megvilágítás kisnyomású nátriumlámpával és LED-del [7]

#### 2.1.3. Környezetbarát

A higanylámpával és a fénycsővel ellentétben, a LED-es megoldások nem tartalmaznak higanyt.

#### 2.1.4. Széles működési tartomány

A LED-ek hidegben és melegben is egyaránt jól funkcionálnak, működésükhez esetekben nagyon kis feszültségek is elegendők, ezáltal alkalmasak kültéri megvilágításokra is.

### 2.2. Forgalomban lévő eszközök áttekintése

Manapság rengeteg gyártó kínál okos világítás rendszereket. Az okos égők nem csak átlagos LED lámpák, amiket a foglalatba lehet tekerni. Okkal hívják őket okos izóknak. Vezeték nélkül csatlakoztathatók mobiltelefonjainkhoz vagy Smart Home rendszerünkhez (Amazon Echo, Google Home), ezzel korlátlan lehetőséget létrehozva.

Szinte minden ilyen fényforrásnak lehet állítani a fényerősségét anélkül, hogy bármiféle fényerősségszabályzó kapcsolót kéne a falba szerelni. Bárhonnan vezérelhetők, illetve be- és kikapcsolási rutinok állíthatók be rajtuk. Ez például egy nagyszerű biztonsági funkció, mert amikor nyaralni vagyunk, akkor annak a látszatát kelti, mintha otthon lennének. A ház elhagyása után a véletlen felkapcsolva maradt izzókat le lehet kapcsolni, nem pazarolva ezzel az elektromos áramot. A legtöbb ilyen okos lámpának a színét is lehet állítani, ezáltal a meleg- és hidegérzetünket befolyásolhatjuk. Hangulatunknak megfelelően is behangolhatjuk, például pirosas színre állítva romantikázás esetén. A piacon zene lejátszására képes égők is elérhetők, bár ezek nem fogják a házimozi rendszerünket helyettesíteni.[8][9]

### 2.2.1. Philips HUE

Az egyik legnagyobb háztartás- és szórakoztató elektronikai eszközgyártó cég is kínál okos világítás rendszereket. Talán az övé a legtöbb funkcióval rendelkező rendszer. Kinti, benti megoldásokat is kínálnak, zenére változó világítást és rengeteg hanggal vezérelhető opciót valósítottak meg. A telepített eszközöket, égőket egy központi egység, úgynevezett hub vezérli, enélkül működőképtelen a rendszer. A Philips-hez hűen megbízhatók az eszközök, viszont cserébe elég borsos árat kell fizetni.[10]



2. ábra: Philips HUE eszközök[11][12]

### 2.2.2. LIFX

A LIFX egy okos világításra specializálódott cég. A Philips HUE termékektől annyiban különbözik, hogy ezek számára nem kell egy központi vezérlőegység, hanem képesek külön-külön működni. Egyedül a LED szalag és csempe termékükhez kell külön vezérlőegység. Ez a termékcsalád is vezérelhető hanggal, támogatja továbbá a Google Asszisztentet, az Apple HomeKit-et, az Amazon Alexa-t és a Logitech Harmony rendszereket is. Árát tekintve hasonló kategóriába esik a Philips termékekkel.

### 2.2.3. TP-Link okos villanykörté

A kedvező áru, vezeték nélküli hálózati eszközökről ismert TP-Link is forgalmaz okos villanykörtéket, illetve egyéb okos otthon megoldásokat. Az ár ez esetben is kedvező, habár a vevői visszajelzések nem olyan jók, mint az előző két termék esetében. A hangvezérlés Amazon Alexa-val és Google Asszisztenssel itt is támogatott.[15]



3. ábra: LIFX termékek[13][14]



4. ábra: TP-Link okosizzó[15]

#### 2.2.4. Általános vezérelhető LED szalag

Talán a legolcsóbb LED-es világítások az Ebay-en, Amazon-on, Aliexpress-en, és hasonlókon rendelhető LED sorok. Ezeket többnyire áramforrással és egy egyszerűbb infrás távirányítóval működtethető vezérlőegységgel adják. Általában csak egy szín

állítható be az egész szalagon, ellentétben a címezhető LED szalagokkal. [16]



5. ábra: Általános LED szalag[16]

### 3. TERVEZÉSI FELADAT KÖVETELMÉNYEINEK FELÁLLÍTÁSA

A feladatot az Ebay-en vásárolt olcsó modulok segítségével szerettem volna megvalósítani, amik már a rendelkezésemre álltak. A cél egy olyan áramköri elem megépítése volt, ami viszonylag könnyedén beforrasztható, külső egyenáramú tápellátásról vezérelni tudja a rákapcsolt LED sorokat, és egy cserélhető Wifi modulon keresztül képes az Androidos alkalmazásunkból érkező, általam definiált protokollú, adatok fogadására. Továbbá egy olyan védődoboz tervezése, amivel különböző helyekre lehet majd felszerelni. A következőkben pontokba szedtem az összes tervezési követelményt:

- Legyen telefonnal működtethető vezeték nélküli hálózaton keresztül
- Rendelkezésre álló modulokból épüljön fel
- Legyen minél kisebb, de még kézzel beforrasztható
- Védve legyen a külső behatások ellen (védődoboz)
- Legyen hordozhatóvá tehető, akkumulátorról működtethető
- Tápfeszültségek szempontjából
  - 12V DC tápfeszültség a LED sor számára
  - 3,3V DC tápfeszültség a mikrovezérlő és a Wifi-modul számára
  - 5V DC tápfeszültség a mikrovezérlőből kijövő 3,3V jelének 5V-sra alakításához

#### 3.1. Rendelkezésre álló hardverek

##### 3.1.1. Wifi modul: ESP8266 - ESP01

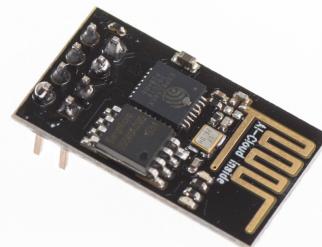
Az ESP8266 egy olyan IC, ami IoT világának[17] megfelelően lett tervezve. Képes egy 802.11 típusú hálózaton TCP/IP protokoll segítségével kommunikálni. Teljesen címezhető SPI vagy UART segítségével, és hozzáférést ad a GPIO-hoz.[18]

Néhány tulajdonsága:

- 802.11 b/g/n szabványokat támogat
- Wi-Fi Direct (P2P), soft-AP módú működés

- integrált TCP/IP protokoll
- integrált WEP, TKIP, AES motorok
- 3.3V DC tápfelszültség

Ennek az IC-nek az ESP01 típusú modulját (6. Ábra) használtam. Körülbelül 500 Ft-ért lehet hozzájutni az Ebay-en, és egy 2x4-es csatlakozóval lehet az áramkörünk-höz csatlakoztatni.



**6. ábra:** ESP01-es Wifi Modul[18]

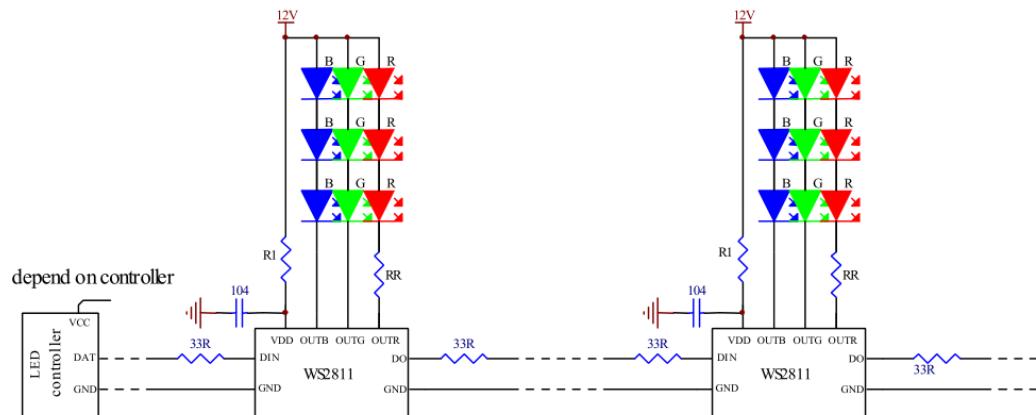
### 3.1.2. WS2811-es IC-vel ellátott címezhető LED szalag



**7. ábra:** Ebay-ról vásárolt LED szalag

A WS2811 egy 3 kimeneti csatornás speciális LED meghajtó IC. Egy jelvezetéken küldött 24 bit szélességű bitsorozattal lehet vezérelni 400 vagy 800 Kbps sebességgel. Az IC-ben van belső jelerősítő, hogy egymás után lehessen kötni őket. Az általam vásárolt LED szalagon IC-nként 3 [db] RGB LED található, és ezek az IC-k az

imént említett módon vannak összekötve. Ezt a felépítést szemlélteti a 8. ábra.



**8. ábra:** Ebay-ről vásárolt LED szalag felépítése[19]

LED szalag tulajdonságai:

- 12V DC tápfeszültség
- 1 A maximális áramfelvétel
- 5V-os jelvezeték
- 5 m hosszú, 150 RGB LED-et tartalmaz
- 3 [db] RGB LED/IC

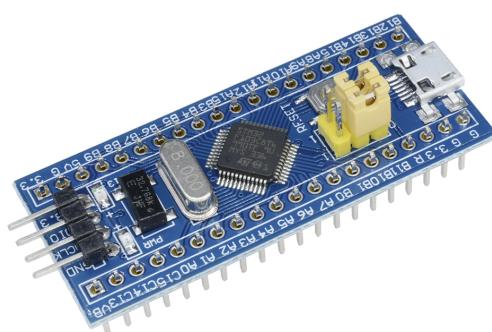
### 3.1.3. Mikrovezérlő: STM32F103C8T6

Ez a mikrovezérlő az F103-as család tagja, azon belül pedig az a változat, aminek 48 lába van (C), 64 Kbyte Flash memóriája van (8), LQFP csomagú (T) és -40-től +85°C-ig működik (6). [20]



**9. ábra:** STM32F103C8T6 típusú mikrovezérlő[21]

Egy maximum 72 MHz-es, 32-bit-es ARM Cortex-M3-as processzor a magja, ami a fent említett Flash és SRAM memória és megannyi más egység (2x A/D konverter, DMA vezérlő 37 I/O port, 7x timer, és 9 kommunikációs interfész) is kiegészít. Számomra az időzítők (PWM generálás, és időzítési feladatok), az UART interfész és a DMA vezérlő játszik fontos szerepet, illetve az A/D konverterek bővítési lehetőségeként.



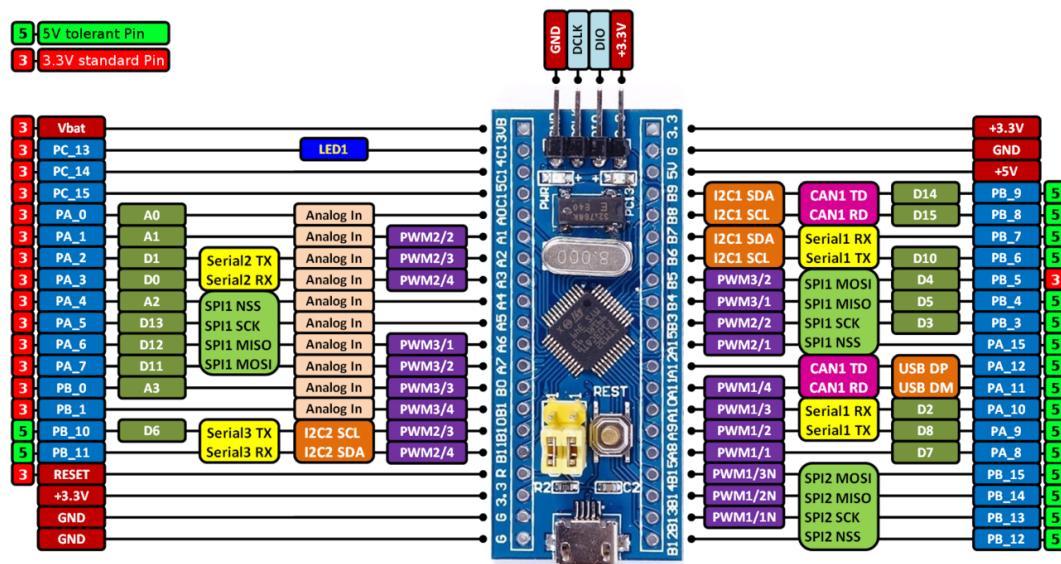
**10. ábra:** Ebay-ről vásárolt STM32F103C8T6-es minimum fejlesztői lap[22]

## 4. ESZKÖZ ELEKTRONIKAI TERVEZÉSE ÉS ÖSSZEÁLLÍTÁSA

### 4.1. Elektronikai tervezés

Az elektronikai tervezést 3 fázisra és az utolsó fázis hibáinak kijavítására bontottam le. Az elsőnél a meglévő modulokat duggostam össze próbapanelen, a másodiknál vasalásos technikával készítettem NYÁK-ot a stabil, kevésbé zajos csatlakozások érdekében. A harmadik fázisnál került sor a modulok egy NYÁK-lapra való integrálására.

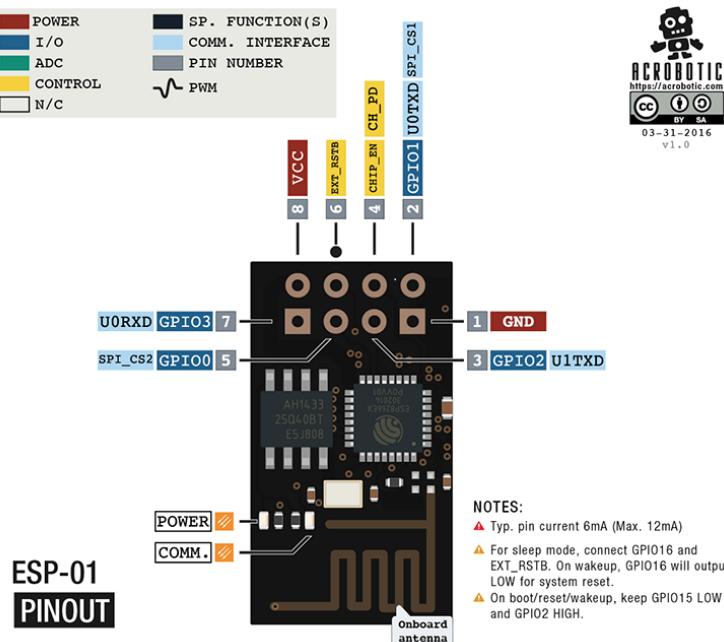
Az első fázis első lépései között szerepelt az Ebay-ról vásárolt eszközök tesztelése: működik-e mindegyik, programozható-e a fejlesztői lap, konfigurálható-e a Wifi modul, stb. Második lépésként a mikrovezérlő lábkiosztása (11. ábra) alapján kiválasztottam a lábak általam használt funkcióit.



11. ábra: STM32F103C8T6 fejlesztői lap pinout-ja [23]

A PB\_0-s lábat választottam a LED sor vezérlő lábának, és a PA\_9, PA\_10 (TX,RX) lábakat pedig Wifi modullal kommunikáló UART lábaknak. minden modulra az előírt tápfeszültséget kötöttem, illetve az összes földpotenciált összehuzaloztam, ezzel egy közös földpontot létrehozva. A ESP-01-s modul pinout-ja (12. ábra) alapján az UART lábakat is összekötöttem a megfelelő módon: a mikrovezérlő RX lábat (PA\_10) a Wifi modul U0TXD lábával és mikrovezérlő TX lábat (PA\_9) a Wifi modul U0RXD lábával.

A WS2811-es IC adatlapján a logikai jelszintes részt áttanulmányozva megállapítottam, hogy a mikrovezérlő 3,3V-os vezérlő jelét még éppen tolerálja a LED sor



12. ábra: ESP-01 pinout-ja[24]

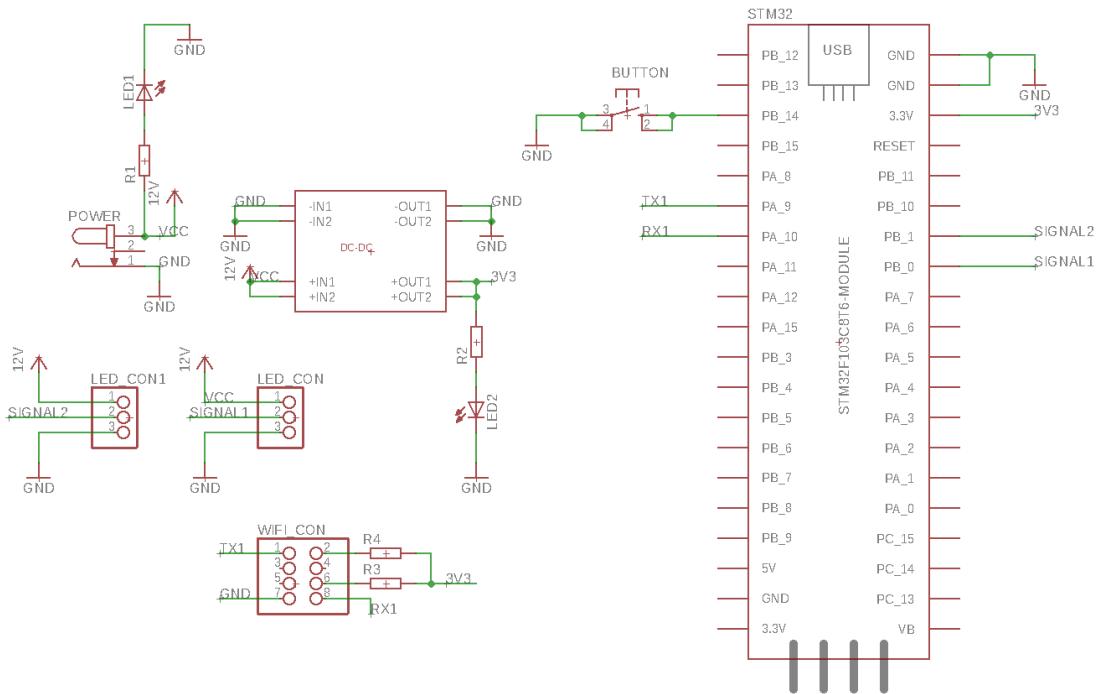
vezérlő IC 5V-os bemeneti lába (3,2V-os jel igaznak felel meg).

Ezek alapján állítottam össze a kapcsolást breadboard-on, és az internetről letöltött mintaprogramokat kicsit módosítva elértem, hogy külön-külön működjön a Wifi modullal történő kommunikáció, illetve a LED soron meg tudtam jeleníteni egy fehér színt. Az összeállítással több probléma is volt; a tesztelés során a breadboard-ból gyakran kicsúsztak a kábelek és a csatlakozások kontaktosak voltak, hamis színeket okozva ezzel a LED soron.

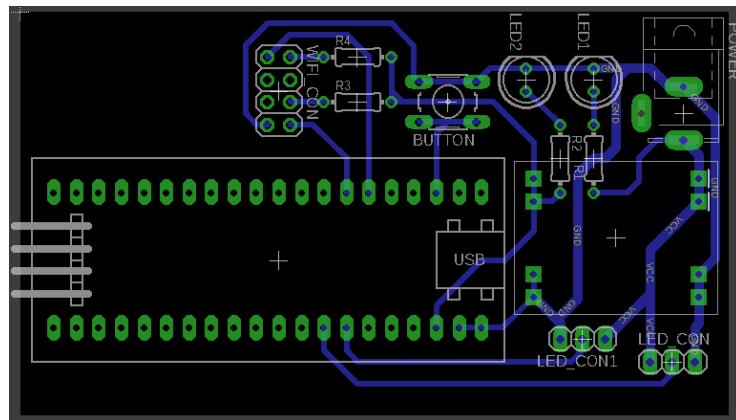
A problémák orvoslásaként került sor a második fázis kivitelezésére, amit az Autodesk Eagle nyáktervező programmal valósítottam meg. Az internetről letöltöttem és beimportáltam az STM32F103C8T6 mikrovezérlő fejlesztői laptának, az ESP8266-os modul ESP-01-es típusának a könyvtárait [23][25]. Létrehoztam egy saját könyvtárt is az Ebay-ról rendelt DC/DC modulnak. Ez a feszültségátalakító állítja elő a 12V-os tápfeszültségből a 3,3V-osat. Egy pár csatlakozó, állapotjelző LED és ellenállás hozzáadása után összehuzaloztam a megfelelő elemeket.

Az elkészült kapcsolási rajz (13. ábra) alapján elhelyeztem és összekötöttem az alkatrészeket a NYÁK-on. Ezen a verziót bekötésre került egy extra gomb (BUT-TON) és egy extra csatlakozó (LED\_CON1), hogy 2 LED sort legyen képes vezérelni az eszköz párhuzamosan. A 14. ábra szemlélteti az így összehuzalozott NYÁK-ot.

A NYÁK-ot vasalásos technikával készítettem el. A tervet lézernyomtatával fényes papírra kinyomtattam, egy egy-oldalas NYÁK-lapra helyeztem, a tonerport vasalóval átvittem a rézfelületre, a tonerrel nem fedett réz bevonatú részeket ma-



13. ábra: 2. fázisú kapcsolási rajz



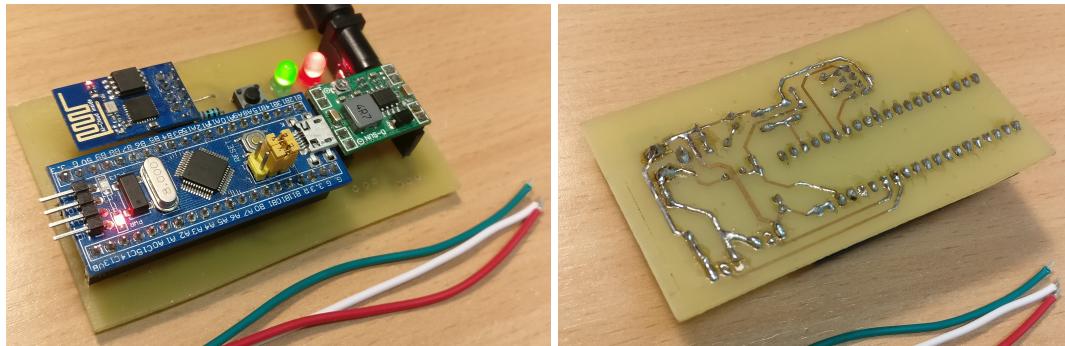
14. ábra: 2. fázisú NYÁK-terv

ratószerrel lemarattam, és acetonnal lemostam, hogy a megmaradt rézfelületre forrasztani lehessen. Az így elkészült NYÁK-lapra (15. ábra) beforrasztottam az alkatrészeket. Ezzel a NYÁK-kal már el tudtam kezdeni a beágyazott szoftver fejlesztését, mivel az elektronikai rész stabilan működött.

A 3. fázis tartalmazza a DC/DC áramkörök tervezését és a mikrovezérlő perifériáival történő integrálását egy nyomtatott áramköri lapra.

#### 4.1.1. DC/DC tervezés

Mivel nagy a különbség a bemeneti és kimeneti feszültségek között (12V-ról 5V-ra), ezért egy DC/DC konverter sokkal jobban megfelel, mint egy LDO (Low-DropOut

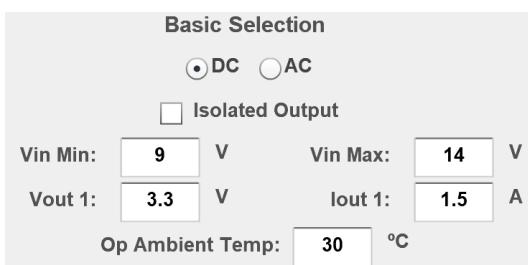


15. ábra: 2. fázisú beforrasztott áramkör

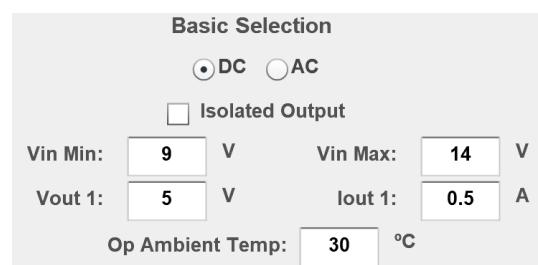
regulator), hiszen egy LDO-nak a hatékonysága nagyon alacsony (12V-ról 5V-ra kb. 41%) és a maradék energiát elfűti. Ez a megoldás semmiképpen sem előnyös, illetve még hűtőborda alkalmazását is megköveteli. [26][27]

A 3,3V és 5V-os tápfeszültségek ellőállítására 2 DC/DC áramkörre is szükségem volt, és ezért olyan megoldást kerestem, ahol ugyanazzal az IC-vel, külső komponensek minimális változtatásával ez megoldható. Célom volt továbbá, hogy egy 3 cellás LiPo akkumulátorról is működjön az áramkör, ezáltal hordozható legyen. A bemeneti feszültség, 3V-os minimum és 4,2V-os maximum cella feszültségnél, egy 3 cellás LiPo-nál 9V és 12,6V közzé fog esni. A felső értéket egészre kerekítve és eggyel nagyobbra választva (14V) egy kisebb biztonsági sávot is kaptam.

A DC/DC áramkörök tervezésénél nagy segítségemre volt a TI (Texas Instruments) Webench Power Designer [28]. A tervezési paramétereket megadva nem csak egy megfelelő IC-t, hanem több komplett kapcsolást is felkínál. A kimeneti névleges áramot, a 3,3V-os kimeneti feszültség esetében 1,5A-ra választottam (16. ábra), hogy akár még más áramkori lapkát is képes legyen meghajtani az eszköz. Az 5V-os kimenet esetén (17. ábra) is túlméreteztem a névleges áramot (0,5A).

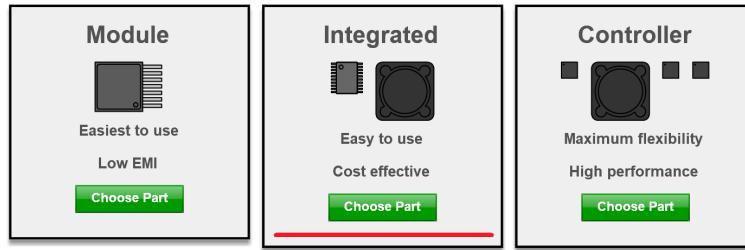


16. ábra: Tervezési paraméterek megadása 3,3V-os kimentre[28]



17. ábra: Tervezési paraméterek megadása 5V-os kimentre[28]

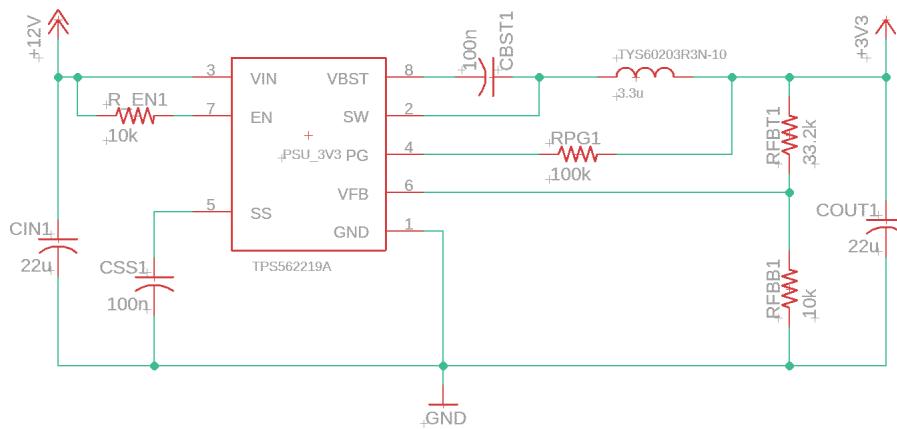
Ezek után az integrált, könnyen használható és költséghatékony tervezési séma-ra szűrve (18. ábra), elkezdtem minél olcsóbb és hatékonyabb megoldás után keresgálni a legenerált esetek között.



18. ábra: Tervezési séma kiválasztása[28]

A TPS562219A típusú IC minden konfigurációban (3,3V - 5V) előfordult. A két verzió között csak egy tekercs és egy ellenállás volt a különbség.

A TI Webench Designer által kínált kapcsolási rajzok komponensei nem minden megfelelők, illetve beszerezhetők. Tovább bonyolította a helyzetet, hogy SMD alkatrészeket sohasem forrasztottam, tehát az alkatrészek mérete sem volt mindegy számomra. A korábbi tapasztalataim alapján úgy döntöttem hogy 1206-os (metrikus: 3216) SMD méretet fogok használni. Az egyik legnagyobb elektronikai kereskedőnél, a Mouser Electronics-nál, a megfelelő paraméterek figyelembevételével újraválasztottam az alkatrészeket. Miután néhány komponensnek megcsináltam az Eagle könyvtárát, elkészítettem a kapcsolási rajzokat a Webench Designer és a TPS562219A IC adatlapja alapján [29]. Mivel a két konfiguráció (3,3V - 5V) minimálisan különbözik egymástól ezért csak az egyiket ábrázoltam (19. ábra).



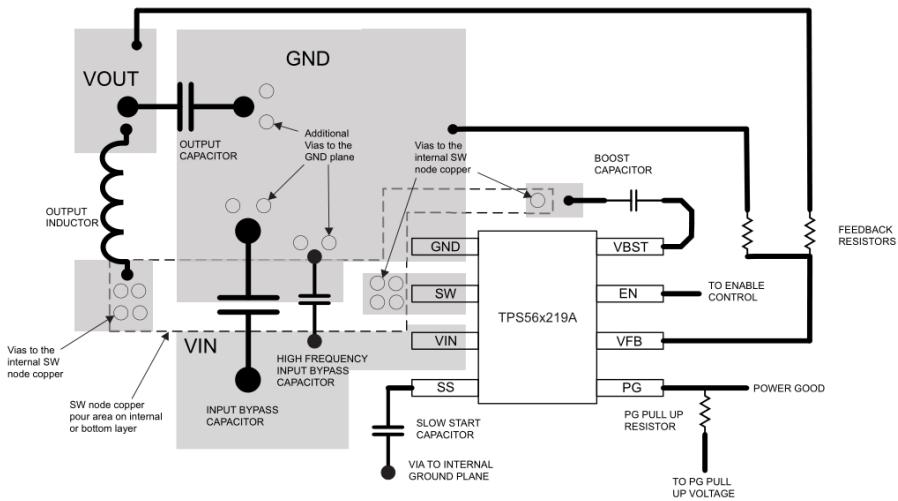
19. ábra: 3,3V-os DC/DC kapcsolási rajza

Általában nem mindegy, hogy az alkatrészeinket az adott IC körül milyen kiosztásban helyezzük el. A TPS562219A típusú IC adatlapjában pontokba van szedve mire kell odafigyelni, illetve egy elhelyezési minta (20. ábra) is megtalálható mellette [29].

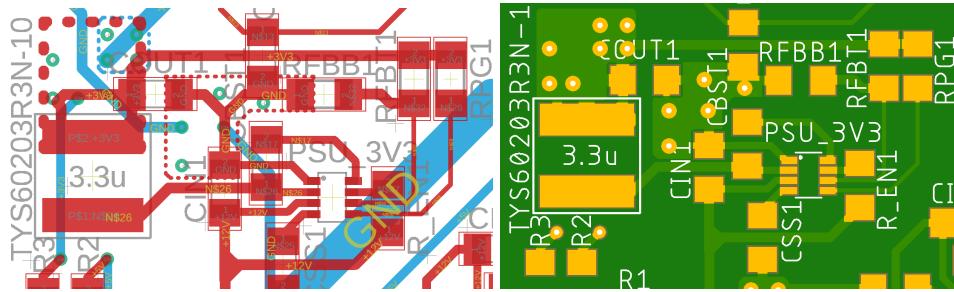
Az irányelveket követve az elkészült NYÁK-tervet a 21. ábra szemlélteti.

Hasonló módon készítettem el az 5V-os átalakítót is, csak az a rész NYÁK-on az óra mutató járásával megegyezően 90°-al el lett forgatva.

## 10.2 Layout Example



20. ábra: DC/DC elhelyezési mintája [29]



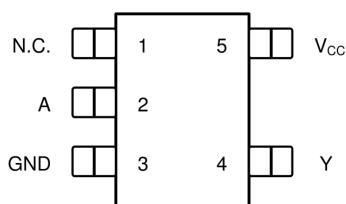
21. ábra: 3,3V-os DC/DC nyákterve

## 4.1.2. Logikai jelszint átalakító tervezése

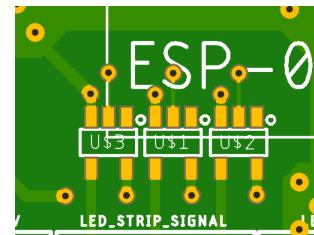
A WS2811-es IC adatlapjában [19] le van írva, hogy a tápfeszültség ( $V_{DD} = 4,5..5,5V$ ) 70%-nak kell lennie a vezérlő jelnek, hogy az IC logikai 1-es szintnek értelmezze. Tehát minimum 3,15V-os ( $0,7 \cdot 4,5V = 3,15V$ ) jel kell az IC vezérléséhez. Így a 3,3V-os mikrovezérlővel a határon mozgok és feszültségingadozás esetén zavarok léphetnek fel. A megbízható, stabil működéshez egy logikai jelszint átalakítót kell beépítenem az áramkörbe.

Törekedtem minél több alkatrészt a TI-től (Texas Instruments) kiválasztani, mert a diákok (érvényes egyetemi e-mail-címmel rendelkezők) számára egy pár ingyenes mintadarabot kínál, amikkel jobb esetben elvégezhető a prototípus fejlesztése. Így esett a választásom a SN74LV1T34 típusú logikai jelszint átalakítóra.

A kis SOT-23-as csomagolású IC (22. ábra) rendkívül egyszerűvé tette a feladatot. A 'GND' lábára a földet, az 'A' lábára a bemeneti jelet, az 'Y' lábára a kimeneti jelet, a 'V<sub>cc</sub>' lábára pedig a kívánt kimeneti logikai jelszintnek megfelelő 5V tápfeszültséget kötöttem. Az adatlapban[30] leírt forrasztási maszk (23. ábra) alapján



**22. ábra:** SN74LV1T34 logikai jelszint átalakító IC lábkiosztása[30]



**23. ábra:** Három logikai jelszintátalakító elhelyezve a NYÁK-on

elkészítettem az alkatrész Eagle könyvtárát. A 3. fázisú NYÁK-kal már három LED sort szerettem volna vezérelni, ezért a kapcsolási rajzon és a nyákon is három-három darab alkatrész került elhelyezésre.

#### 4.1.3. STMF103C8T6 mikrokontroller és perifériák

A mikrokontrollert külső komponensekkel is el kell látni ahhoz, hogy megfelelően működjön. Az elsők és legfontosabbak a bufferkondenzátorok a mikrokontroller megfelelő lábain. Az alkalmazási útmutató (AN2586[31]) leírja, hogyan kell a kondenzátorokat megfelelően bekötni (24. ábra): minél közelebb a megfelelő lábakhoz, és ha lehetséges, akkor alkalmazzunk galvanizált furatokat a föld ( $V_{SS}$ ) és a tápfeszültség ( $V_{DD}$ ) rétegekre. Az adatlapban (24. ábra) megtalálható, hogy melyik lábra mekkora és milyen kondenzátort kell bekötni.

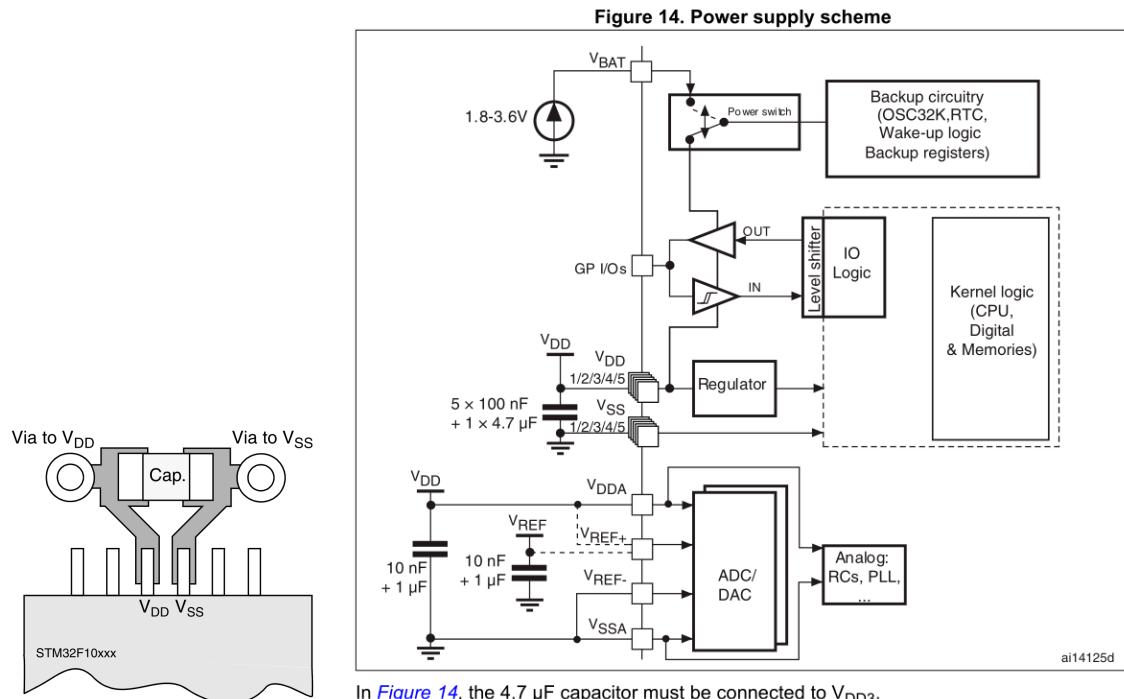
Többféleképpen is felprogramozható (UART-on keresztül bootloader segítségével, JTAG-en vagy pedig SWD-n) a mikrokontroller, viszont DEBUG-olni csak JTAG-en és SWD-n (Serial Wire Debug) lehet. A kettő közül az utóbbit választottam, mert rendelkezésemre állt egy USB-s SWD programozó. Ahhoz, hogy a mikrokontrollert SWD-n keresztül programozhassuk, ki kell vezetni a SWCLK (Serial Wire Clock), illetve SWDIO lábakat egy csatlakozóra. A kétirányú kommunikáció miatt (programozás, debuggolás), az SWDIO vezetéket egy  $100[\text{k}\Omega]$ -os ellenállással ajánlott felhúzni a tápfeszültségre [32].

Három különböző helyről képes a mikrokontroller bootolni. Az alapértelmezett fő flash memóriából való bootolást a boot lábak földre húzásával valósítottam meg.

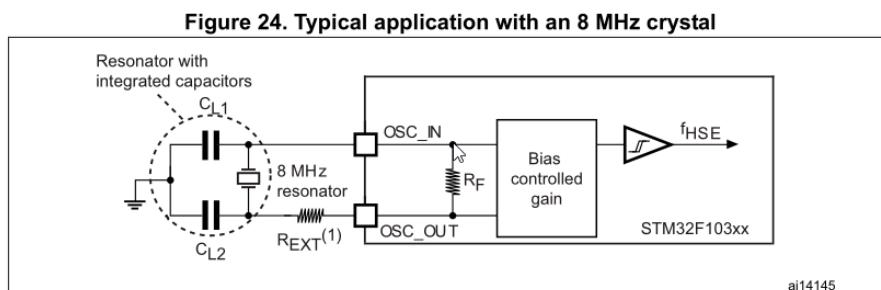
A maximális, 72MHz-es, órajellel való működéshez külső oszcillátor áramkörre is szükség van, anélkül csak 64MHz lenne elérhető. A referencia terv (25. ábra) alapján egy 8MHz-es kristály oszcillátort és két 20pF-os kerámiakondenzátort használtam a megvalósításhoz.

A reset gombot is bekötöttem, szintén az adatlap által javasolt referencia terv (26. ábra) alapján.

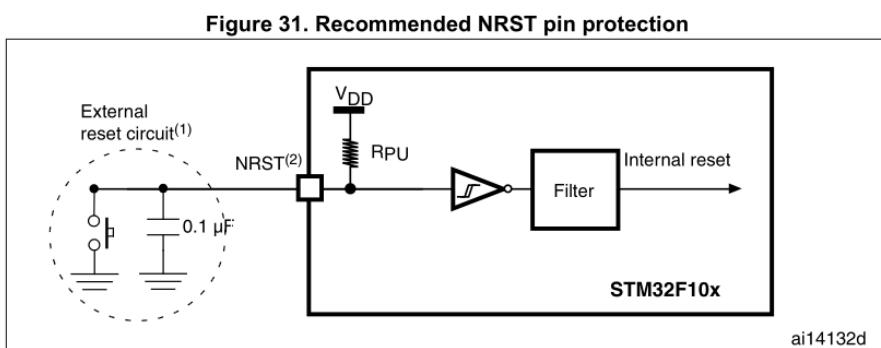
A mikrokontrollert összekötöttem a Wifi modullal, illetve további 3 LED-es álla-



**24. ábra:** Bufferkondenzátorok ajánlott áramköri elhelyezése balra[31], méreteik, bekötési helyei jobbra[20]



**25. ábra:** Nagy sebességű oszcillátor referencia terv[20]

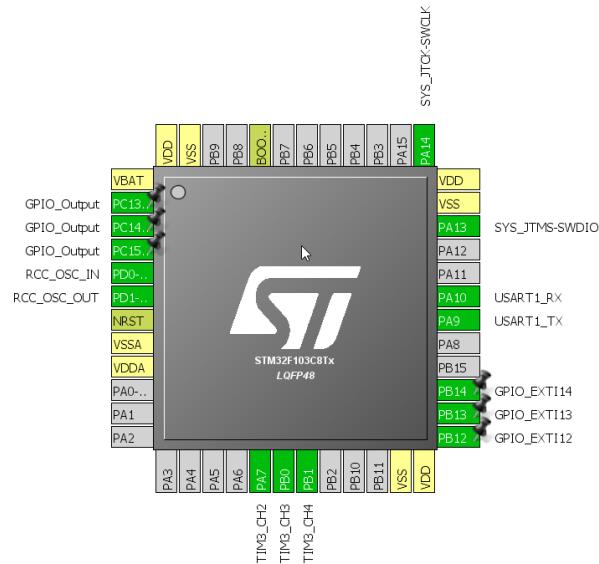


**26. ábra:** Reset gomb bekötésének referencia terv[20]

potjelző kimenettel és 3 szoftveresen földre húzott gombbal is elláttam a kapcsolást. Csak az egyik LED-nek szántam szerepet, a többi I/O továbbfejlesztési lehetőségeinek

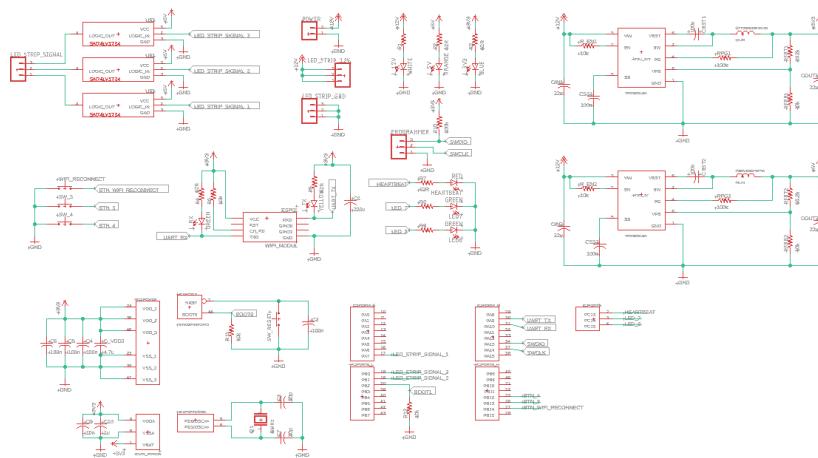
lett megtervezve.

Az ST Microelectronics gyártó STM32CubeMX nevű programjával ellenőriztem a megfelelő lábkiosztást, hogy a megfelelő lábakra kivezethetők-e az adott belső perifériák (TIMER-ek, UART) jelei.



**27. ábra:** Lábkiosztás a STM32CubeMX programban

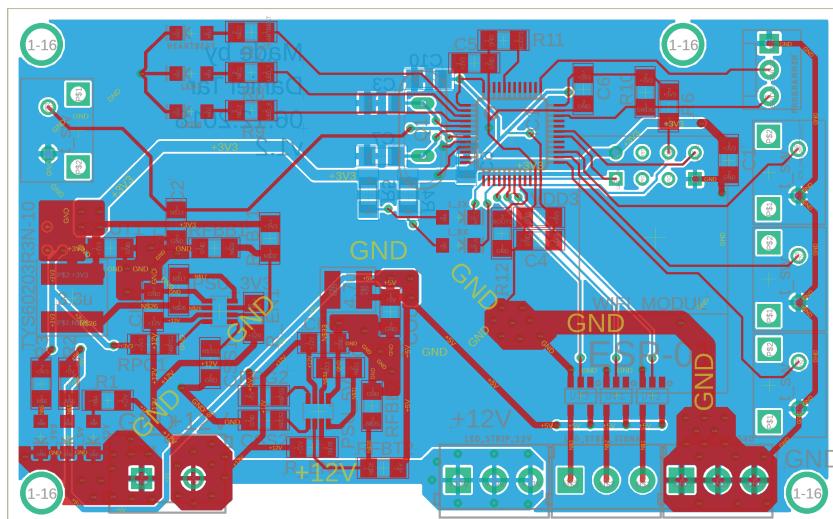
A mikrokontrollerem EAGLE könyvtárát az internetről letöltöttem és beimportáltam a programba. A tervezek alapján elkészítettem a kapcsolási rajzot (28. ábra) pár extra visszajelző komponenssel ellátva: UART RX, TX jelvezetékeire, a 12V-os és a DC/DC áramkörök kimeneti tápfeszültségeire kötöttem visszajelző LED-eket.



**28. ábra:** Lekicsinyített kapcsolási rajz (felnagyított verzió a függelékben található)

A kapcsolási rajz után az alkatrészek elhelyezése és huzalozása következett. Az EAGLE fejlesztői itt is nagy segítségemre voltak, és a 10 leghasznosabb tippet [33]

összegyűjtötték a témaban nem annyira jártasok számára. A NYÁK-on logikai szempontok alapján elhelyeztem az alkatrészeket. A tápfeszültség és egyéb visszajelző LED-eket, gombokat - csoportonként - egymás mellé és a reset gombot a többi gombtól elkülönített helyre raktam. A LED sor és a Wifi modul kommunikációs jelvezetékeit minél távolabba helyeztem a zajforrásoktól, mint például a DC/DC áramköröktől és az oszcillátor áramkörtől. Az elrendezett komponenseket összehuzaloztam a tervezési irányelvek és a fejlesztői tippek alapján (29. ábra).

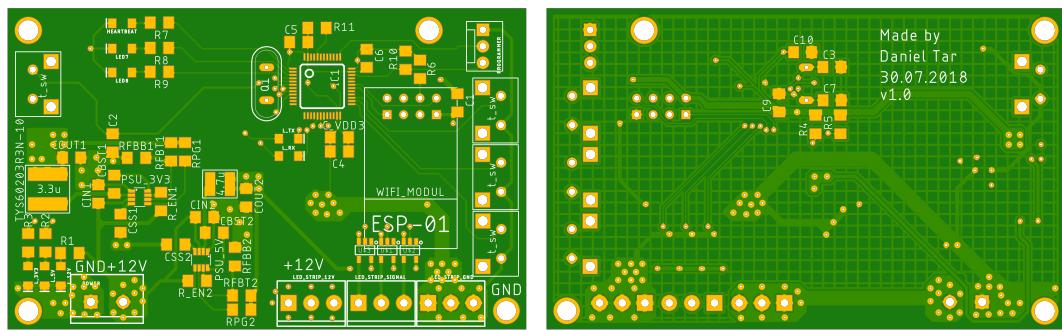


29. ábra: Összehuzalozott NYÁK terv

#### 4.1.4. Alkatrészek megrendelése és a NYÁK legyártatása

Az alkatrészeimet az egyik legnagyobb internetes elektronikai viszonteladónál, a Mouser Electronics-nál, választottam ki. A terveket úgy hoztam létre, hogy 1206-os méretű alkatrészeket használtam, mert ezt a méretet nagy biztonsággal képes leszek beforrasztani. A projekt nagy tanulsága viszont, hogy ebben a méretben egyáltalán nem biztos, hogy minden értékben megtalálhatók az alkatrészek, és ha igen, akkor is jellemzően jóval drágábban, mint más méretekben. További probléma volt, hogy a TI Workbench által javasolt alkatrészek a DC/DC átalakítóhoz, kifutott termék révén, már elfogytak vagy nem is forgalmazták őket. A megfelelő szempontokra figyelve kiválasztottam az új alkatrészeket, frissítettem a kapcsolási rajzot és a NYÁK tervet (a frissített verzió képei találhatók az előző alfejezetben).

A sok NYÁK gyártó közül én a JLCPCB-t, egy prototípusgyártásra specializálódott céget, választottam. Az EAGLE-ben kigenerált gerber fájlokat (30. ábra) kellett feltöltenem valamilyen tömörített állományként vagy zip vagy rar formátumban. Az alapbeállításokat használva minden összes 2\$-ért és szállítási költségért két héten belül megkaptam.



30. ábra: Gerber fájlok előnézetei

## 4.2. Az elkészült eszköz beforrasztása, tesztelése

Az eszközt fokozatosan készítettem el, részegységenként haladva. Először beforrasztottam, majd ellenőriztem a csatlakozásokat, rövidre zárást multiméterrel (diódateszter funkció). Ha nem sikerült valamit rendesen beforrasztani, akkor megpróbáltam még egyszer, és utána újra ellenőriztem a dolgok helyességét. Amennyiben minden rendben találtam, akkor rákötöttem az eszközöm a labortápra (megfelelő feszültséget, és áramkorlátozást beállítva) és oszcilloszkóppal mértem a megfelelő működést. Szépen haladtam sorban a 3,3V-os, az 5V-os DC/DC, a mikrokontroller és a többi részen. Többnyire első próbálkozásra sikerültek a forrasztások, a STM32F103-as IC-nél másodikra.

Az elkészült első prototípus után jöhett a szoftveres tesztelés. Az első és legfontosabb az volt, hogy programozható legyen a mikrokontroller. 10 alkalomból 7-szer sikerült is felprogramozni az IC-t, ami nem éppen az elvárt működés volt. További tesztek során a Wifi modullal való kommunikációt is sikeresen leteszteltem. Az egyik bekapcsolásnál viszont váratlanul kisült az egyik tág IC. Huzamosabb kísérletezés után, megismétlődött az előbbi eset és elkezdtem keresni a probléma okát. Időbe telt, mire kiderült, hogy a tág IC EAGLE könyvtárának készítése során felcseréltem két lábat. Viszont a hiba kijavítása után sem szűnt meg az eredeti probléma. A végső megoldást, az úgynevezett *Soft Start* kondenzátor ( $C_{SS}$ ) kicserélése jelentette. Ez a kondenzátor felelős azért, hogy milyen gyorsan kapcsoljon be, és állítsa be a kimeneti feszültséget a DC/DC áramkör[29]. A TI Workbench által javasolt 8,2nF értéket 100nF-ra cseréltem, amivel a készülék azóta is jól működik. Ezzel az értékkel 20ms felfutási ideje lett a tágfeszültségeknek.

Az eszköznek jelenleg hálós föld rétege van, mert az első verzió tervezésénél úgy emlékeztem, hogy ennek előnyösebb tulajdonságai vannak mint az egybefüggő földrétegeknek. Azóta utána jártam, és kiderült, hogy az egybefüggő sokszor jobb [34]. A NYÁK terven ez kijavításra került, de azóta még nem került legyártásra.

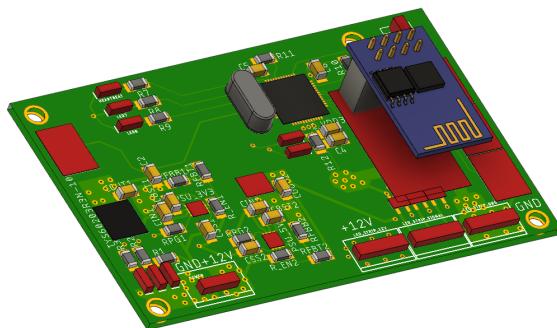


**31. ábra:** A tág IC felcserélt lábainak kijavítása az elkészült NYÁK-on

### 4.3. Védődoboz tervezése, 3D-nyomtatással prototípus gyártása

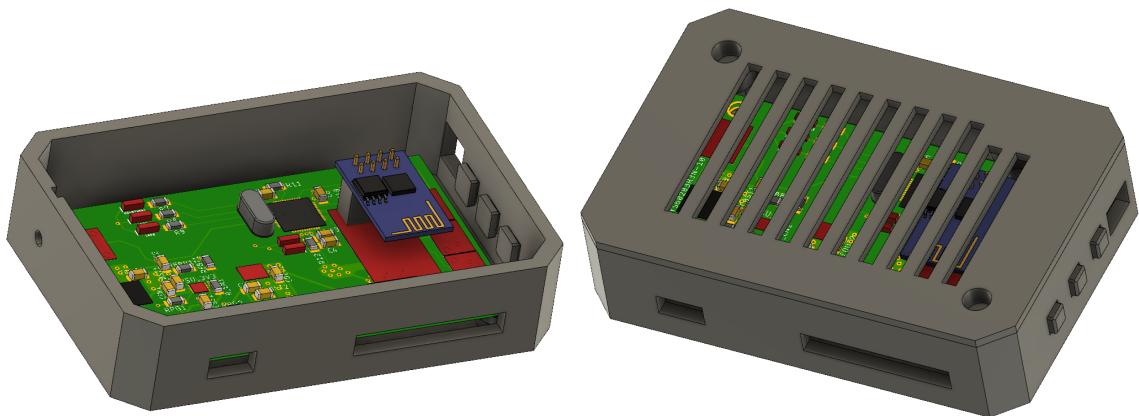
#### 4.3.1. Védődoboz 3D modellezése

Az utóbbi időben egyre gyorsabban fejlődik az EAGLE. Megjelentek a 3D-s alkatrészkönyvtárak, és az Autodesk Fusion 360 (továbbiakban Fusion) nevű programba való exportálás, feltöltés lehetősége is. A Fusion egy felhő alapú program CAD, CAM és CAE eszközökkel. A felhő alapú szolgáltatás lényege, hogyha a NYÁK tervet az EAGLE-ben frissítem, akkor az Autodesk szerverein is frissül, ezáltal a Fusion-be érzékelve a változtatásokat. A NYÁK-ot és a grabcad.com oldalról letöltött Wifi modult beimportáltam és összeillesztettem (32. ábra).

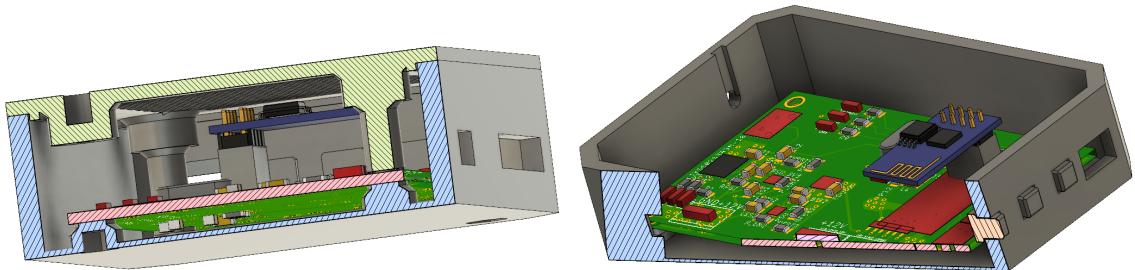


**32. ábra:** Fusion-be importált NYÁK és ESP8266-os Wifi modul

Az elektronikai részek 3D-s környezetbe illesztése után elkezdtem a védődoboz modellezését. Parametrikus modellezést és a NYÁK referenciaméreteit használtam a gyors testreszabhatóság és az automatikusan változó méretek miatt. A dobozon nyílást hagytam a tágvezetékeknek, a LED sorok bekötésére és a programozó csatlakozónak (33. ábra). Furatot készítettem a reset gomb számára, hogy szükség esetén benyomható legyen, illetve a 3 a házon elhelyezett nyomógombot kapott a 3 extra gomb is.



**33. ábra:** Védődoboz 3D-s tervei



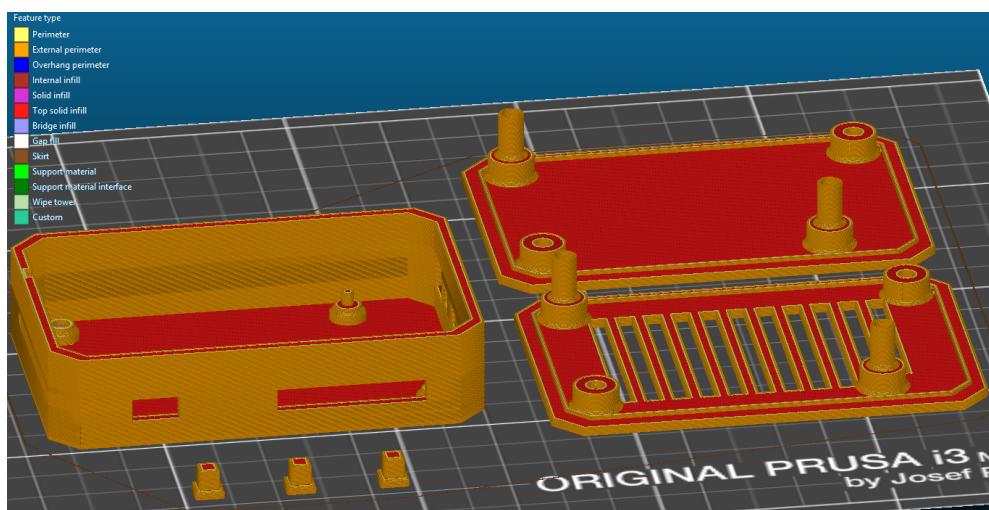
**34. ábra:** 3D-s tervek metszetben

A NYÁK-on 4 furat található, ebből kettőt a dobozban való pozicionálásra használtam, kettőt pedig a doboz házának és fedelének a rögzítésére (34. ábra). A fedélből több verziót is készítettem. Az egyiken hálós hűtő felület került elhelyezésre és főleg benti használatra lett tervezve, míg a másik teljesen zárt testtel rendelkezik. Az otthoni terasz megvilágítására például a zárt verziójút szereltem fel az eresz alá.

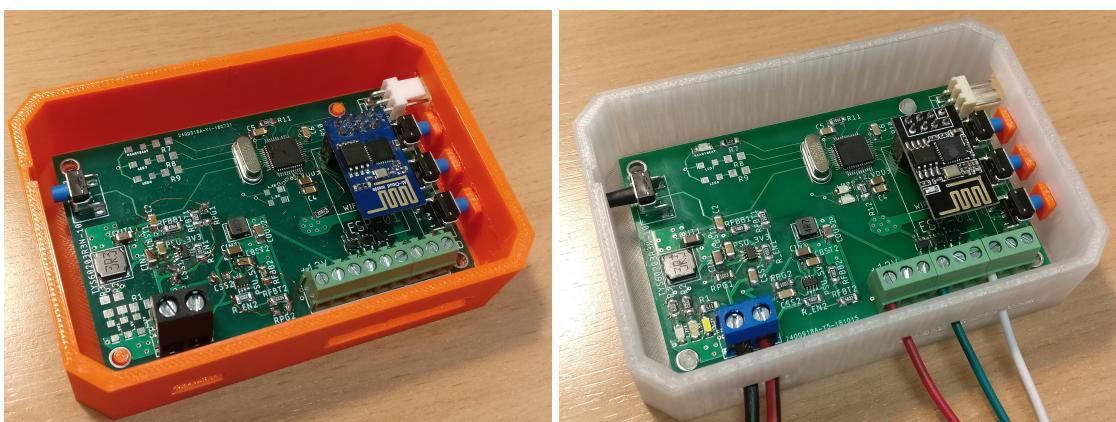
#### 4.3.2. 3D nyomtatás és az elkészült hardver

Manapság a 3D nyomtatás a prototípusgyártás megkerülhetetlen része lett. Olcsón, gyorsan, valós méretű, mérethelyes alkatrészeket készíthetünk. A meglévő technológiák közül a számomra elérhető FDM-et (Fused Deposition Modelling) használtam. A módszer lényege, hogy egy megfelelő polimertekercsből, rétegenként állítjuk elő a kívánt modellünket. Slic3r PE egy olyan program, ami elvégzi a modellünk slice-olását (35. ábra) (rétegekre bontását), és a 3D nyomtató számára legenerálja a megfelelő G-Code-ot. Ez a kód tartalmazza azokat az információkat, hogy milyen hőmérsékletre kell fútni az adott anyagot, mikor, milyen sebességgel és hova kell mozogjon az extruder (nyomtató fej).

A nyomtatás után összeszerelt és beforrasztott ledsorvezérlőket a 36. és a 37. ábrák szemléltetik.



35. ábra: Védődoboz slice-olás után



36. ábra: 1. és 2. verziójú NYÁK-ok a 3D nyomtatott védő dobozukban



37. ábra: A 3D nyomtatott védő dobozok tetővel

## 5. BEÁGYAZOTT SZOFTVER ELKÉSZÍTÉSE

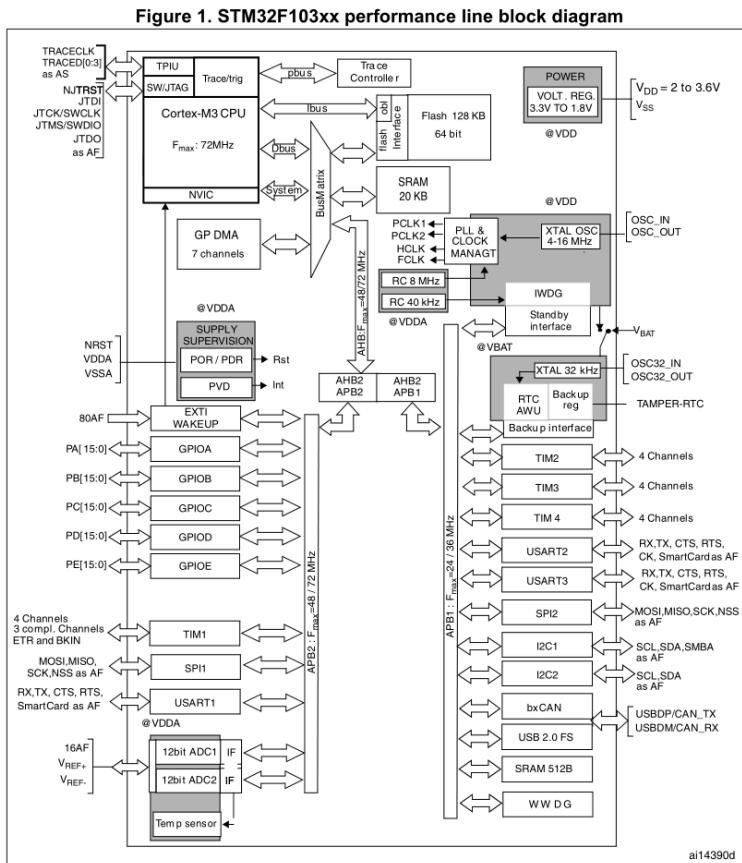
A beágyazott szoftver feladata, hogy a Wifi modultól érkező csomagokat fogadja, értelmezze, majd ez alapján meghatározza és kiküldje a LED sor által értelmezhető formátumban a jeleket. A feladatot erre a mikrokontrollerre három féle könyvtár-csomaggal is meg lehet oldani.

- Arduino környezettel
- HAL (Hardware Abstraction Layer) könyvtárral
- SPL (Standard Peripherals Library) könyvtárral

Az Arduino környezetben már rengeteg megoldás létezik az interneten az ilyen típusú LED sorok vezérlésére. Az egyiket kiválasztva, letöltve és kisebb módosításokat végezve a programkódon a Wifi modullal való kommunikáció is megoldható. Ez a módszer már elsőre is jól hangzik, de projekt célja nem a gyors működésre bírás, hanem a háttérben futó folyamatok megértése, és az Arduino-s megoldás pont ezt rejti el. Hasonló megoldást kínál az ST gyártó a HAL könyvtárakkal. A már használt STM32CubeMX programban a megfelelő IDE-be (Integrated Development Environment - integrált fejlesztői környezet) lehet generálni majdnem kész projekteket. Egy pár kattintással be lehetne állítani az órajel konfigurációt, az I/O portok, időzítők (továbbiakban Timer), megszakítók (továbbiakban Interrupt), stb. működését elektünket jelentősen megkönnyítve. Viszont, mint ahogy a neve is sugallja, hardver absztrakciós réteg, ez a megoldás is elfedi a belső működését a mikrokontrollernek. Az igényeimet végeredményül az SPL könyvtár fogja kielégíteni, ami hozzáférést ad a regiszterekhez, de emellett alapszintű függvényeket is biztosít a folyamatok kezeléséhez. [35][36]

A perifériakezelést mindenkorban DMA-val (Direct Memory Acces - direkt memória hozzáférés) valósítottam meg, hogy a processzor ez idő alatt másossal tudjon foglalkozni. Az egyes részelemeket az átláthatóság kedvéért igyekeztem külön könyvtárakba csoportosítani, illetve az extra moduloknak további alkönyvtárokat létrehozni.

Az adott lábak, periféria, órajelek engedélyezésénél nagy segítségemre volt az adatlapban megtalálható mikrokontroller blokkvázlata (38. ábra). A diagramm mutatja, hogy melyik buszt, illetve mikor mirek az órajeleit kell engedélyezni. Például, ha a SPI1-t szeretném használni, akkor nem csak SPI1 vezérlő számára kell biztosítani a működéshez szükséges órajelet, hanem az APB2 (Advanced Peripheral Bus 2) részére is engedélyezni kell.



**38. ábra:** STMF103XX blokk diagramma[20]

A beágyazott szoftver megírását négy részre bontottam le:

1. Mikrokontoller inicializálása
2. LED sorral való kommunikáció (PWM)
3. Wifi modullal való kommunikáció (UART)
4. Az előző kettő összekötése

A szakdolgozatomban nem fogok kitérni arra, hogy konkrétan melyik regisztek, illetve milyen értékek lettek beállítva, hanem a lényegi összefüggéseket írom le.

### 5.1. STM32F103C8T6 mikrokontroller inicializálása

A mikrokontroller programozására és debuggolására az ST tulajdonában lévő, ingyenesen elérhető, cross platform Atollic TrueStudio-t használtam. Az IDE Eclipse

alapú és rengeteg beágyazott szoftver fejlesztésében segítő eszközzel van kiegészítve. Az STM32CubeMX programban létrehozott projektünket egy az egybe lehet exportálni az IDE-be, persze ilyenkor HAL könyvtárakat használ a projekt. Új projekt készítésekor a régi mikrokontrollerekhez, viszont az SPL-t is lehet használni. A mikrokontroller és a debugger fajtája (JTAG/SWD), és még egy pár opció beállítása után egy azonnal fordítható kódot generál. A projektet lebuildelve és feltöltve a mikrokontrollerre, az a *main.c* állományt fogja futtatni, illetve a *main()* függvény a program belépési pontja.

```

70
71 int main(void){
72     SystemInit();
73
74
75     #ifdef DEBUG
76         InitDBG();
77     #endif
78
79     /* configure SysTick for delay functions */
80     InitSysTick();
81
82
83

```

**39. ábra:** main.c állomány eleje

A *main()* függvény első soraiban (39. ábra) kell beállítani a kívánt működési órajelet. Ezt a SPL egyik könyvtárában lévő *SystemInit()* nevű függvény végzi el. A függvény alapértelmezett beállítása a 72MHz-es órajel, pont, amit én is be akarok állítani. A következő sorokban a DEBUG szó definiálására lefutó *InitDBG()* függvény [37]. Ebben a függvényben beállításra került, hogyha debuggolásnál megállítjuk a CPU futását, akkor a Timer-ek is megálljanak. Az inicializálást az *InitSysTick()* függvény zárja, amiben be lett állítva, hogy a *SysTick\_Handler()* interrupt kezelő függvény mikor hívódjon meg. Erre a függvényre épülnek a *util* könyvtárban megtalálható primitív *Delay..()* függvények, és a mikrokontroller helyes működését visszajelző LED villogtatások.

## 5.2. LED sorral való kommunikáció

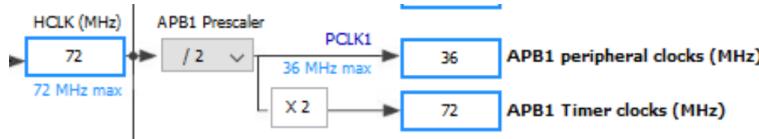
Egy LED szalagot egy darab, egy irányú jelcsatornán lehet vezérelni. Ezen a vezetéken kivezérelt megfelelő frekvenciájú PWM (Pulse Width Modulation - impulzus szélesség moduláció) jel kitöltési tényezőjének változtatásával lehet a WS2811-es IC számára a megfelelő bitkombinációt elküldeni. 400kHz-es üzemmódban a '0'-s bithez egy 0,5µs ideig magas, és 2,0µs ideig alacsony, azaz 20%-os kitöltési tényezőjű PWM ciklus felel meg. Az '1'-s bithez pedig egy 1,2µs ideig magas, és 1,3µs ideig alacsony, azaz 48%-os kitöltési tényezőjű PWM ciklus felel meg. Egy IC számára egy RGB kódnak megfelelő 24bit szélességű adatot (24 darab PWM ciklus) kell elküldeni, 8-8-8 bites felbontással. Ezen felül érkező biteket továbbítja a következő -

vele sorba kapcsolt - IC-nek. Az IC-k a kimenetére kötött LED-eken akkor fog megjelenni az elküldött szín, mikor egy reset jel érkezik. A reset jel egy  $50\mu s$ -nál hosszabb alacsony feszültség szint. A 800kHz-es üzemmódban az idők a reset jel kivételével a felére csökkennek. Az általam vásárolt LED sorokon 50 darab ilyen WS2811-es IC található, IC-ként 3 darab RGB LED-del. Ezeket a LED-hármasokat neveztem el ledgroup-oknak (LED csoportoknak), mert ezeknek a színeit lehet külön-külön változtatni. [19]

### 5.2.1. PWM jel létrehozása a kimeneteken

A LED sor vezérlésére a hármas Timer (továbbiakban TIM3) 2,3,4-es csatornáit (továbbiakban CHx, ahol az 'x' jelenti az adott csatorna számát) használtam, mert ezek felelnek meg a kapcsolásban bekötött lábaknak (PA7, PB0, PB1). Első körben engedélyezni kell az I/O portokra (esetben A és B portokra) és az AFIO-ra (alternate function I/O) az órajelet. Az utóbbira azért van szükség, mert a TIM3 ezen keresztül fogja vezérelni az mikrovezérlő lábait. A I/O porton beállítottam a megfelelő lábakat kimenetre és hogy milyen sebességgel fognak maximálisan operálni.

Második lépés a TIM3 működési frekvenciájának beállítása. A TIM3 az APB1-en található (38. ábra), de az összes többi perifériával ellentétben a timerek megkapják a 72MHz-es órajelet (40. ábra). Alapvetően a 800kHz-es módra konfiguráltam fel a TIM3-at, mert az előosztó (prescaler) megváltoztatásával 400kHz-es működési mód érhető el. A legnagyobb felbontás elérében egyre választottam az órajelosztást, ami azt jelenti, hogy 72MHz-en fog üzemelni a timer. Ezen a frekvencián 90 órajelciklus fel meg  $1,25\mu s$ -nak, a 800kHz-es jel periódusidejének. A TIM3 tehát 0-tól fog számolni 89-ig, ahol nullázódik és kezdi újra a számlálást. Azt, hogy ez idő alatt mit csinál, az Output Compare mód beállításával lehet megadni. Az előosztót (prescaler) a LED sor könyvtárában való sebesség módok definiálásával változtathatóvá tettem (41. ábra). Eredményül a lassabb 400kHz-es mód esetén felére csökkenti a TIM3 működési frekvenciáját, ezzel a kívánt működést létrehozva.



40. ábra: APB1 órajelei az STM32CubeMX programból

Harmadik lépés a működési mód és tulajdonságainak bekonfigurálása. Kimenetbe, azon belül a PWM1 üzemmódba állítottam a timert. Az alapértelmezett polaritás beállításoknál az annyit tesz, hogy mikor elkezd számlálni, akkor magas jelszintet ad ki, és mikor eléri a Capture Compare (továbbiakban CC) értéket akkor vált ala-

csony (föld) jelszintre. A CC értékkel lehet majd beállítani, hogy milyen kitöltési tényezőnél történjen meg ez a váltás. 90 számlálásnál a logikai igaznak értelmezett 48%-os kitöltési tényező 43-nak, míg a logikai hamis 20%-os kitöltési tényező 18-nak fog megfelelni (41. ábra). A DMA vezérlés célja, hogy a CC értékét periódusonként változtassuk, ezzel a 24bit-es szín kombinációkat létrehozva a LED szalagon található WS2811 IC-k számára.

```

35  /*=====
36  *                               WS2811 SETTINGS
37  =====*/
38
39  /* Select operating mode according to your led strip */
40 // #define MODE_800kHz
41 #define MODE_800kHz
42
43  /* high period times in counter ticks @72MHz */
44 /* 90 tick is equal with 1.25 us - prescaler = 0 */
45 /* 90 tick is equal with 2.5 us - prescaler = 1 */
46 #ifdef MODE_400kHz
47     #define TIM_PRESCALER 1
48 #endif /*MODE_400kHz*/
49 #ifdef MODE_800kHz
50     #define TIM_PRESCALER 0
51 #endif /*MODE_800kHz*/
52
53 /* according to ws2811 datasheet */
54 #define TIM_PERIOD 90    /* 90 */
55 #define T1H      43    /* 43 */
56 #define T0H      18    /* 18 */

```

**41. ábra:** Képernyőkép a WS2811 nevű könyvtár header állományából

### 5.2.2. PWM vezérlése DMA segítségével

A beágyazott szoftverben egy tömbben tárolom, hogy melyik LED sornak melyik ledgroupján milyen színt kell megjelenítsek. Ez a tömböt fogja majd később frissíteni a Wifi modulról érkező adat. Viszont a LED sor számára az egyes biteket, illetve a biteknek megfelelő kitöltési tényezőjű (CC értékű) PWM-et kell küldeni. Egy ledgroupnak 3 bájton tárolom a színét, viszont, hogy ennek a három bájtnak, azaz 24 bitnek, a kitöltési tényezőjét eltároljam, 24 bájtra lenne szükségem (43, és 18 értékeket a legkisebb egységen, egy bájton lehet eltárolni). Szóval egy ledgroup kitöltési tényezőjének eltárolásához nyolcszor annyi hely szükséges, mint csak a színadataihoz. Ha 6 LED sort csatlakoztatnák az eszközönre, akkor ( $6[\text{ledstrip}] \cdot 50[\frac{\text{ledgroup}}{\text{ledstrip}}] \cdot 3[\frac{\text{byte}}{\text{ledgroup}}] = 900[\text{byte}]$ ) 900 bájtot és a kitöltési tényezőket, vagyis még 7200 bájtot kellene eltároljak. Összesen tehát körülbelül 7,9 kilobájtot ( $(900+7200)/1024 \approx 7,9[\text{kilobyte}]$ ), ami a 64 kilobájtos tárhelynél már egészen drasztikus. Ezért ezt valamilyen bufferezési technikával kell megoldanom.

A reference manual [32] timer regiszterei szekciót olvasva találtam rá az úgynevezett DMA Burst funkcióra. Ez a funkció lehetővé teszi, hogy a memóriában lévő tömbből az adatokat, ne csak egy, hanem több timer regiszterbe írja. A működés így

a következőképpen készítettem el: az adatok írását a CH2 CC regiszterével kezdi és a CH4 CC regiszterével fejezi be. 6 elemű adattömb esetén az 0-s indexű elem kerülne a CH2 CC regiszterébe, 1-es a CH3, 2-es a CH4, 4-es újra a CH2 CC regiszterébe, és így tovább. Tehát a buffer szerkezetemnek úgy kell kinéznie, hogy a párhuzamosan kapcsolt LED sorok azonos indexű bitjei legyenek egymás után sorba rendezve.

A DMA ciklikus üzemmódban is beállítható, így a memóriában található adatokon iterál végig, újból és újból. A DMA buffer méretét és a ledgroupok számának ismeretében meghatároztam, hogy hány ciklus után kell lekapcsolni a DMA-t. Így már csak a buffer megfelelő frissítését kellett megoldanom a működéshez. A DMA vezérlőn be lehet kapcsolni a HT (half transfer - fél transzfer) és FT (full transzfer - teljes transzfer) interruptokat, amik jelzik, hogy a vezérlő végzett a fél, illetve egész buffer kiküldésével. Amikor megjelenik a HT interrupt akkor fogom a buffer első felét frissíteni a következő adatokkal, a FT interruptnál pedig a buffer második felét.

Utolsó lépésként engedélyeztem a DMA megszakításkezelő függvényeit, és beállítottam a prioritásukat.

A LED sor frissítése tehát a következő pontokra bomlik le, amit a *refreshLedStrip()* függvényhívással (42. ábra) indíthatunk el:

1. A ledgroup színei alapján a kitöltési tényezők meghatározása, és a bufferbe való betöltése
2. A DMA vezérlő állapotának resetelése, buffer méretének a beállítása
3. TIM3 elindítása
4. A buffer frissítése a DMA megszakításkezelőjében
5. Az adatok kiküldése után a TIM3 leállítása és várakozás legalább a  $50\mu\text{s}$  reset-jel teljesüléséig

## 5.3. ESP8266-os Wifi Modul

### 5.3.1. A Wifi modul bekonfigurálása

A Wifi modullal való beszélgetés előtt, fel kell konfigurálnom. Be kell állítani az üzemmódját, hogy milyen hálózatra csatlakozzon automatikusan, avagy sem, illetve az UART beállításait. Az eszközön egy AT firmware fut és UART-on keresztül küldhetünk neki AT parancsokat. Az AT parancsok leírása a gyártó által kiadott ESP8266 AT Instruction Set című dokumentumban található [38].

A modul számára egy USB-Serial átalakítóval küldtem a megfelelő parancsokat. Az eszköz alapértelmezett UART beállításait használtam: 115200 Baudrate, 8

```

321  /**
322   * @brief Sends out the color data to the led strip:
323   * - disables the TIM, sets its CCRx register values to 0
324   * - set the DMA data counter
325   * - initialize the buffer and pixel_id variables
326   * - set the transmission flag to ON
327   * - clear the DMA Flags
328   * - enable the DMA and TIM controllers
329   * @param None
330   * @retval None
331 */
332 void refreshLedStrip(void){
333
334     TIM_Cmd(TIM3, DISABLE);
335     DMA_Cmd(DMA1_Channel3, DISABLE);
336     DMA_SetCurrDataCounter(DMA1_Channel3, DMA_BUFFER_SIZE);
337
338     /* buffer initialization*/
339     Init_DMA_Buffer();
340     txOn = 1;
341
342     DMA_ClearFlag(DMA1_FLAG_HT3);
343     DMA_ClearFlag(DMA1_FLAG_TC3);
344     DMA_Cmd(DMA1_Channel3, ENABLE);
345     TIM_Cmd(TIM3, ENABLE);
346 }

```

**42. ábra:** Képernyőkép *refreshLedStrip()* függvényről

bit szóhosszúság és 1 stop bit. A kommunikáció sebességét nem lehet és felesleges növelnem, mert a LED sor frissítéséhez jóval több idő kell, mint amit az UART-on keresztül érkező csomag fogadása igénybe vesz.

Az áramkörömet úgy terveztem meg, hogy cserélgetni lehessen a Wifi modulokat, a környezet megváltozásának megfelelően. Hárrom modult konfiguráltam fel különbözőképpen: az egyiket a kollégiumi Wifi hálózatomra, a másikat az otthoni Wifi hálózatra és a harmadikat pedig a laptopom által létrehozott Hotspot hálózatra. Mindegyik AP (Access Point - hozzáférési pont) módba, és az adott hálózatra való automatikus csatlakozóra lett beállítva.

A LED sorok színét a telefonos applikációban lévő színválasztással fogom vezélni. Azért, hogy telefonon lévő változtatásokat minél gyorsabban lekövesse a rendszerem, az online számítógépes játékoknál is alkalmazott, UDP (User Datagram Protocol) protokollt választottam. Az UDP protokoll nem biztosítja a hálózaton közlekedő csomagok megérkezését, de ha a sok színváltoztatás közben kimarad egy érték, az sem probléma. A rendszer a legtöbb esetben egy eszközön (routeren) alapuló lokális hálózaton keresztül fog üzemelni, és ebben az esetben a csomagok kiesése elég valószínűtlen.

A Wifi modul bekapcsolása és halózatra csatlakozása körülbelül 8 másodpercet vesz igénybe. Ezután a *AT+CIPSTART="UDP","0",0,1302,2* parancs elküldésével hozható létre az UDP kapcsolat az 1302-es porton keresztül. A parancs elküldése után az eszközünk fogadja a hálózaton érkező csomagokat a következő formában: *+IPD,<len>:<data>*, ahol *len* az csomag hossza, és *data* a csomag tartalma. A mikrokontrollert az UDP kapcsolat felállításáért felelős karaktertömb elküldésére, és az előző formátumban érkező adatok fogadására kell felkészíteni.

### 5.3.2. Mikrokontroller felkészítése az UART-on való kommunikálásra

A folyamat hasonlóan néz ki, mint a PWM létrehozása esetében. Engedélyeztem a lábakat tartalmazó portra és az AFIO-ra az órajelet. A TX lábat kimenetre, az RX lábat pedig bemenetre állítottam.

A lábak felkonfigurálása után az UART periféria beállítása következett. Ugyanazokat a beállításokat alkalmaztam, mint ami a Wifi modul esetében: 115200 Baudrate, 8 bit szóhosszúság és 1 stop bit. Megint engedélyeztem a perifériának az órajelét és az RX és TX módokat (43. ábra).

```

137  /**
138   * @brief This function initialize the gpio pins for the UART1 (PA9 - TX, PA10 - RX)
139   * @param GPIO_InitTypeDef variable
140   * @retval None
141   */
142 void InitGPIO_UART1(GPIO_InitTypeDef* GPIO_InitStruct){
143   /* TX on PA9 */
144   GPIO_InitStructInit(GPIO_InitStruct);
145   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
146
147   GPIO_InitStruct->GPIO_Mode = GPIO_Mode_AF_PP;
148   GPIO_InitStruct->GPIO_Pin = GPIO_Pin_9;
149   GPIO_InitStruct->GPIO_Speed = GPIO_Speed_50MHz;
150   GPIO_Init(GPIOA, GPIO_InitStruct);
151
152   /* RX on PA10 */
153   GPIO_InitStructInit(GPIO_InitStruct);
154   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
155
156   GPIO_InitStruct->GPIO_Mode = GPIO_Mode_IPU;
157   GPIO_InitStruct->GPIO_Pin = GPIO_Pin_10;
158   GPIO_InitStruct->GPIO_Speed = GPIO_Speed_50MHz;
159   GPIO_Init(GPIOA, GPIO_InitStruct);
160 }
161
162 /**
163  * @brief This function initialize the UART1 settings
164  * @param USART_InitTypeDef variable
165  * @retval None
166  */
167 void InitUART1(USART_InitTypeDef* USART_InitStruct){
168   /* USARTx configured as follow:
169    * - BaudRate = 115200 baud
170    * - Word Length = 8 Bits
171    * - One Stop Bit
172    * - No parity
173    * - Hardware flow control disabled (RTS and CTS signals)
174    * - Receive and transmit enabled
175   */
176   /* deinitialize before use */
177   USART_DeInit(USART1);
178   RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
179
180   USART_InitStructInit(USART_InitStruct);
181   USART_InitStruct->USART_BaudRate = 115200;
182   USART_InitStruct->USART_WordLength = USART_WordLength_8b;
183   USART_InitStruct->USART_StopBits = USART_StopBits_1;
184   USART_InitStruct->USART_Parity = USART_Parity_No;
185   USART_InitStruct->USART_HardwareFlowControl = USART_HardwareFlowControl_None;
186   USART_InitStruct->USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
187   USART_Init(USART1, USART_InitStruct);
188
189   USART_Cmd(USART1, ENABLE);
190 }

```

**43. ábra:** Képernyőképek az UART inicializálásról

### 5.3.3. UART vezérlése DMA segítségével

A küldésre (TX) és fogadásra (RX) külön-külön kell definiálni a DMA tulajdonságait. A TX-hez a 4-es csatornát (CH4), RX-hez pedig az 5-ös csatornát (CH5) kell beállítani (44. ábra).

Table 78. Summary of DMA1 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1	-	-	-	-	-	-
SPI/I <sup>2</sup> S	-	SPI1_RX	SPI1_TX	SPI2/I2S2_RX	SPI2/I2S2_TX	-	-
USART	-	USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I <sup>2</sup> C	-	-	-	I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1	-	TIM1_CH1	-	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
TIM2	TIM2_CH3	TIM2_UP	-	-	TIM2_CH1	-	TIM2_CH2 TIM2_CH4
TIM3	-	TIM3_CH3	TIM3_CH4 TIM3_UP	-	-	TIM3_CH1 TIM3_TRIG	-
TIM4	TIM4_CH1	-	-	TIM4_CH2	TIM4_CH3	-	TIM4_UP

44. ábra: DMA hozzáférés csatornái [32]

Habár már a PWM jel generálása esetén engedélyeztem az órajelet DMA számára, itt is meg fogom tenni. Ennek az az oka, hogy a kódot igyekeztem minél modulárisabban megírni, hogy az UART és TIMER perifériák külön-külön is működőképesek legyenek. A TX-el csak egyetlen egyszer kell, egy ismert hosszúságú karaktertömböt kiküldeni, amivel létrehozzuk az UDP kapcsolatot. A DMA-nak ilyenkor a karaktertömb elemein kell végigmennie, és az UART periféria adatregiszterébe (USART1->DR) másolhatnia az adatokat. A RX-nél viszont folyamatosan kell fogadni az üzeneteket, és az UART adatregiszteréből másolhatni a fogadó adat területre. Az Android-os applikációból ismert adatszerkezetű üzeneteket küldök, 5 bajtot. Ez az adat a következőképpen épül fel: az első bajt biztonsági funkciót tölt be, hogyha ez nem egyezik meg a mikrokontrollerbe beállított értékkel, akkor az eszközünk nem fog semmit csinálni. A második bajt a módválasztó, ezzel választom ki, hogy sima színt, színpalettát, vagy animációt jelenítsek meg. A további három bajt mindenkor különböző paramétereknek felel meg, például a sima szín esetében az R,G,B komponenseknek. A Wifi modul az érkező csomagot az ismert karakterekkel kiegészíti és így küldi tovább a mikrokontroller számára. Ezen információk birtokában, a DMA vezérlőn beállítottam a fogadandó karakterek számát, és a ciklikus üzemmódot, mert minden ugyanezt a hosszúságú adatot fogja a mikrokontroller megkapni. A DMA csatornáinak bekonfigurálása után, már csak a megszakításkezelő függvényeket kell engedélyezni és prioritásukat beállítani.

#### 5.4. Az ESP8266-os modulról érkező adatok és a LED sor vezérlés összekötése

Az ESP8266-os inicializálása során átadásra kerül egy általam definiált *callback* típusú függvénypointer (45. ábra). A DMA RX megszakítás kezelő függvényben (46. ábra), azaz, ha egy csomag megérkezett, akkor a függvénypointer által mutatott függ-

vény fog meghívásra kerülni. A meghívott függvény az *OnUART\_DataReceived()* (47. ábra). Ebben kerül lekezelésre a csomag értelmezése és a LED sor frissítésének elindítása. A LED sor frissítésére jelenleg akkor kerül sor, ha új csomag érkezik. Ez a módszer viszont nem alkalmas animációk megjelenítése, mert ott folyamatosan változtatni kéne az értékeket.

```
21 typedef void (*callback)(void); /* callback function definition */
```

**45. ábra:** *callback* nevű függvénypointer definíciója

```
379 /*
380     * @brief This function handles the UART1_RX DMA
381     * @param None
382     * @retval None
383     */
384 void DMA1_Channel5_IRQHandler(void){
385     /* all data received */
386     if(isCallbackSet){
387         (*ptr_CallbackOnUART_DataReceived)();
388     }
389     DMA_ClearFlag(DMA1_FLAG_TC5);
390     //isNewDataArrived = 1;
391 }
392
```

**46. ábra:** A DMA megszakításkezelőjében a callback függvény hívása

```
208 void OnUART_DataReceived(void){
209     if(uart_receive_array[SECURITY_BYT] == security_code){
210         switch(uart_receive_array[MODE_BYT]){
211             // simple color
212             case 0 :{
213                 ColorRGB solidColor;
214                 solidColor.r = uart_receive_array[PARAM_START];
215                 solidColor.g = uart_receive_array[PARAM_START+1];
216                 solidColor.b = uart_receive_array[PARAM_START+2];
217
218                 if(!txOn){
219                     fillSolid(solidColor);
220                     refreshLedStrip();
221                 }
222             }break;
223             // color palette
224             case 1 :{
225                 if(!txOn){
226                     fillPattern(uart_receive_array[PARAM_START]);
227                     refreshLedStrip();
228                 }
229             }break;
230
231             default:break;
232         }
233     }
234 }
```

**47. ábra:** *OnUART\_DataReceived()* callback függvény

## 6. ANDROIDOS ALKALMAZÁS ELKÉSZÍTÉSE

### 6.1. Androidról általában

Az Android a világ legnápszerűbb mobil operációs rendszere. Több milliárd eszközön fut, mint például a telefonokon, órákon, táblagépeken, TV-ken, és még sok máson. Különböző alakú és méretű eszközökön egyaránt elfut, ezzel óriási flexibilitást biztosítva az alkalmazás fejlesztők számára. [39]

A nemrégen megjelent *Android Things* lehetővé teszi az okos, internetre csatlakoztatott eszközök építését, nem csak általános, hanem kereskedelmi és ipari felhasználásra is. Egy ilyen fejlesztői készletet mutat be az 48. ábra. A meglévő Androidos fejlesztői eszközökön kívül elérhetővé válik az alacsony szintű I/O könyvtárak kezelése is.

Azért döntöttem az Android alapú vezérlés mellett, mert megbízható, biztonságos és olcsó eszközökön is tökéletesen működik.



48. ábra: PICO-PI-IMX7 Startkit[40]

### 6.2. Android-os szoftverfejlesztés megismerése

Az Android-os alkalmazások fejlesztéséhez elengedhetetlen a Java programozási nyelv, hiszen Java API keretrendszerre épít a platform. A platformmal először még 3. félévben egy villanykaros szabadon választható tárgy keretében találkoztam. Akkoriban még nem ismertem a Java-t, ezért ennek az elsajátításával kezdtem meg

a tanulást[41]. Az Android specifikus részek megismerésében nagyon nagy segítségemre volt a tantárgy maga, az Android Developer weboldal[42] és egy ismerősöm által ajánlott *Android Programming: The Big Nerd Ranch Guide* című könyv[43]. Az alkalmazás fejlesztése során igyekeztem a tanultakon kívül a clean code elvek használatára[3].

### 6.3. Android alkalmazás rövid felépítése

Az android alkalmazások alapvetően négy fő komponensből épülnek fel, ahol a Content Provider-ek kivételével minden alkalmazásra került[44]:

1. Activities - felületi (UI) elemek
2. Services - háttérben futó folyamatok
3. Content Providers - adat szolgáltatók
4. Broadcast Receivers - rendszer szintű eseményekre reagálás

#### 6.3.1. Services

Androidos környezetben a szolgáltatások egy hosszabb ideig, háttérben futó folyamatot jelképeznek. Nem rendelkeznek felhasználói felülettel és más komponensek elindíthatják vagy vezérlés céljából rácsatlakozhatnak. Az alkalmazásomban két darab ilyen Service is megtalálható: az egyik az UDP csomagok küldésért felelős, a másik az alkalmazásba belekódolt zeneszám lejátszásáért.

#### 6.3.2. Activities

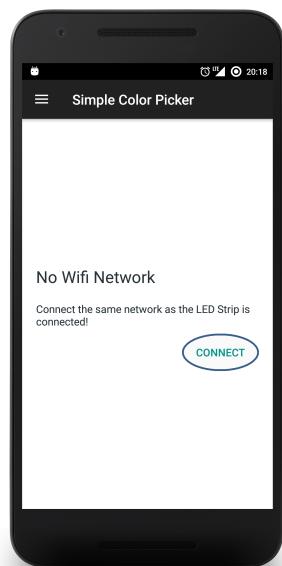
Az Activity-k, pontosabban egy Activity-re csatolt Fragment-ek, alkotják a felhasználói felületet. A felhasználói felületet a manapság gyakori 5.5-inch-es, 16:9-es képarányú, FullHD (1920x1080) felbontású mobiltelefonokra terveztem és valósítottam meg, de ennél nagyobb kijelzőjű tabletéken is elfut az alkalmazás. Az Activity-k feladata Service-ek vezérlése a felhasználói felület (csúszka mozgatása, gomb megnyomása), illetve a gyorsulásérzékelő és magnetométer szenzoradatok változására.

#### 6.3.3. Broadcast Receivers

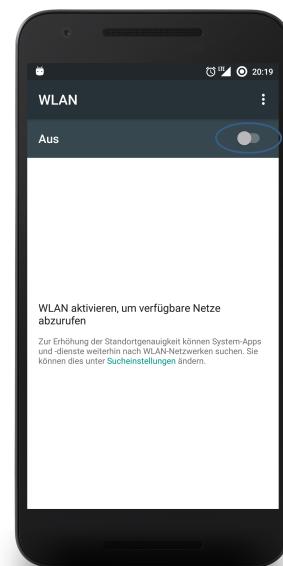
Broadcast Receiver-ek segítségével iratkozhatunk fel rendszer szintű eseményekre. Ilyen események például az SMS és a bejövő telefonhívás, amiknek a bekövetkezésekor beállítható, hogy mit csináljon az alkalmazás.

## 6.4. Az elkészült alkalmazás funkciói és felhasználói kézikönyv

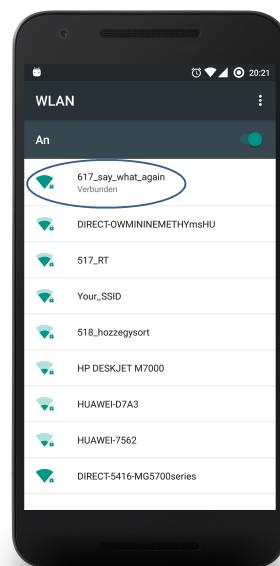
Az alkalmazás indításakor szükséges, hogy valamilyen WiFi hálózathoz legyen csatlakoztatva az eszköz. Ekkor az alapértelmezett Simple Color Picker képernyő töltődik be, amit később a beállításokban módosíthatunk kedvünk szerint. Ha nincs WiFi hálózatra csatlakoztatva az eszköz, akkor egy hibát jelző oldal jelenik meg, amin a csatlakozás gombra kattintva eljuthatunk a készülék WiFi beállítás menüjébe, és ott csatlakozhatunk a hálózatra (49., 50. és 51. ábrák).



**49. ábra:** Koppintson a csatlakozás gombra!



**50. ábra:** Majd a bekapcsolásra!

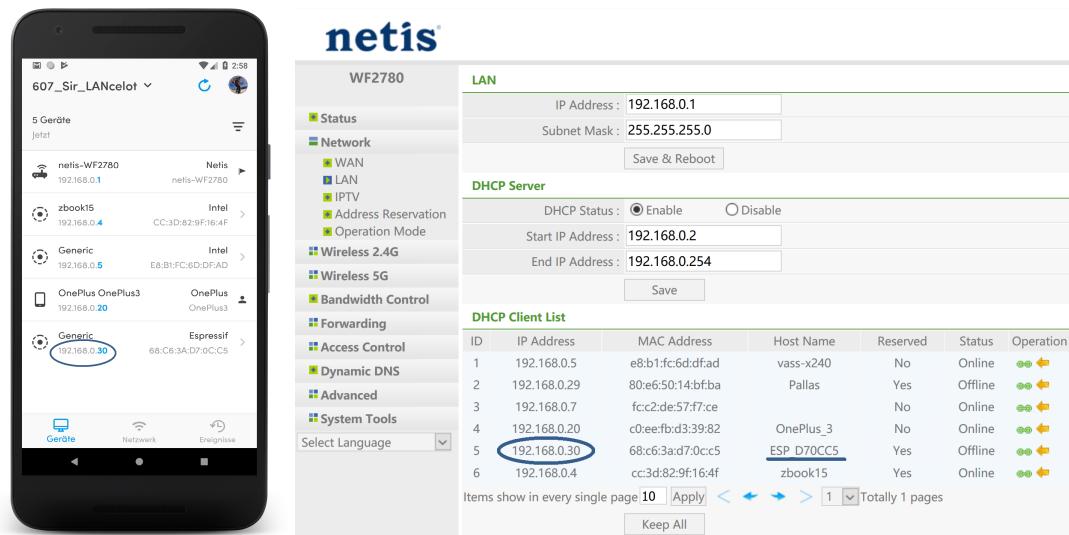


**51. ábra:** Végül válasza ki azt a hálózatot amire csatlakoztatva van a LED sor!

#### 6.4.1. LED sor IP-címének a beállítása

A LED sor IP címe két módon szerezhető meg (52. ábra):

1. Mobiltelefonunkon a Fing nevű alkalmazással könnyedén felderíthetjük a lokális Wifi hálózaton található eszközöket IP címükkel együtt, ami innen egyszerűen kimásolható.
2. A helyi hálózati routerre csatlakozott eszközök listájából.



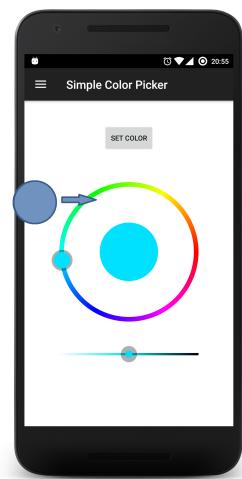
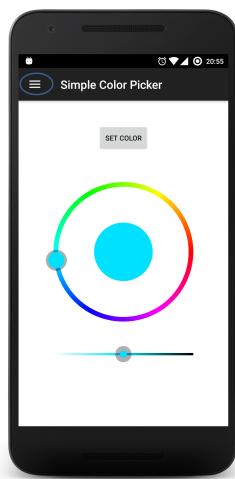
**52. ábra:** LED sor IP címének felderítése Fing nevű alkalmazással és routerünk međijéből

Ezek után visszalépve az alkalmazásba, átnavigálva a beállítások menüre beállítható a LED sor IP címe:

1. Navigációs menü előhozása

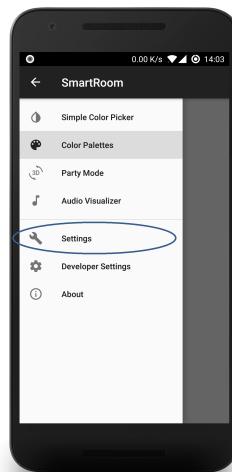
Az alkalmazáson belül bármelyik képernyőről ezzel (53. és 54. ábrák) a két módszerrel lehet előhozni a navigációs menüt.

2. Beállítások fül kiválasztása (55. ábra)
3. Helyi hálózati IP cím beállítása (56. és 57. ábrák)

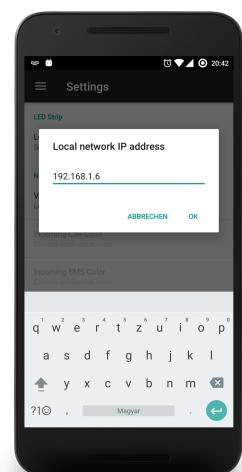
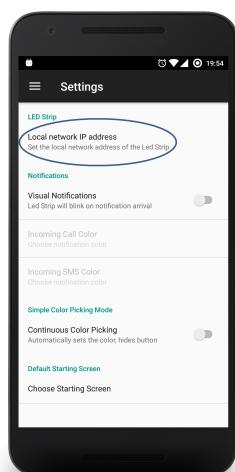


**53. ábra:** A menü gombra való kattintás-

**54. ábra:** "Swipe gesture segítségével"



**55. ábra:** Koppintson a Beállítások fülre!

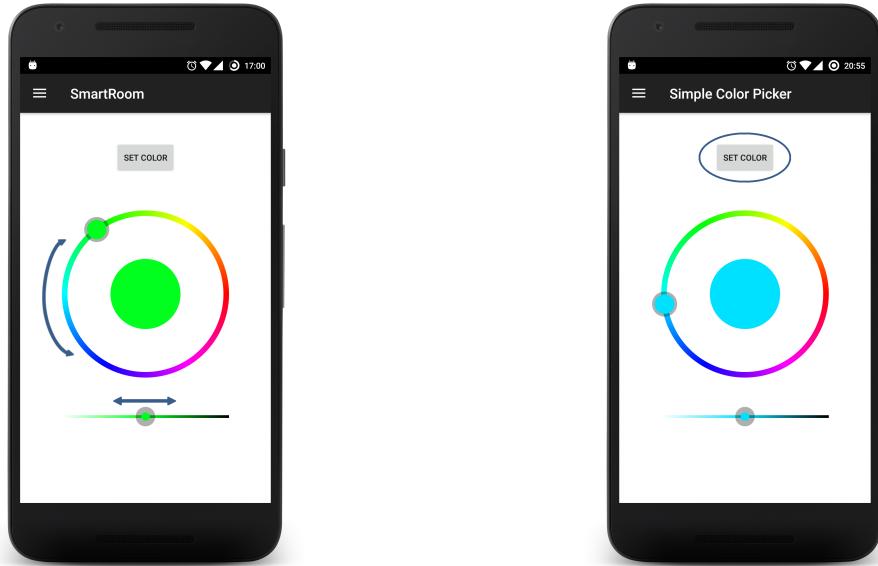


**56. ábra:** Koppintson a helyi hálózati IP cím beállításra!

**57. ábra:** Állítsa be az IP címet, majd koppintson az Ok gombra!

#### 6.4.2. Simple Color Picker mód

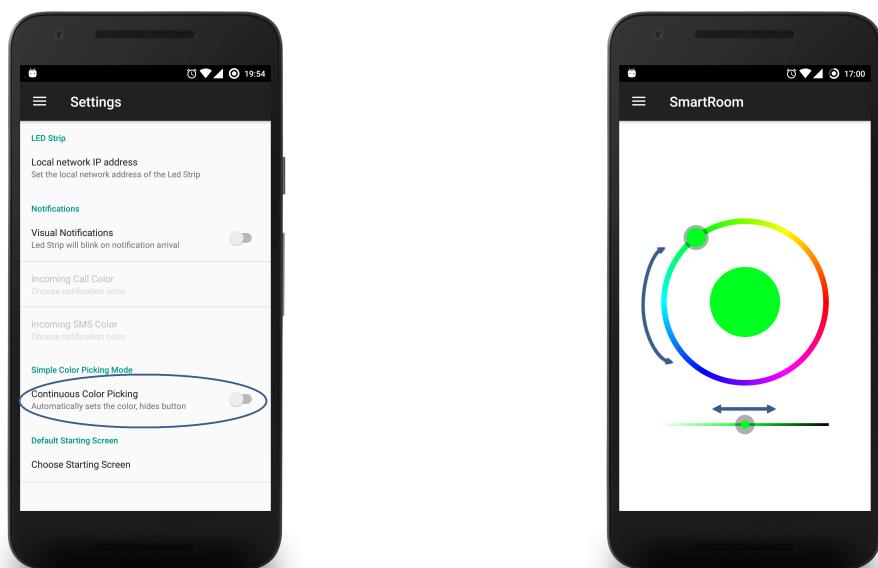
Ezen a képernyőn egy külső forrásból származó Color Picker található[45]. A csúszkák mozgatásával tudjuk kiválasztani az adott színt, majd a *Szín beállítása* gomb megnyomásával állíthatjuk be a LED sor színét (58. és 59. ábrák).



**58. ábra:** A csúszkák mozgatásával változtathatjuk a megjelenítendő színt

**59. ábra:** Szín beállítását a gombra koppintással végezhetjük el

A képernyőt személyre szabhatjuk a beállítások fülön lévő automata színválasztó funkció bekapcsolásával (60. és 61. ábrák).

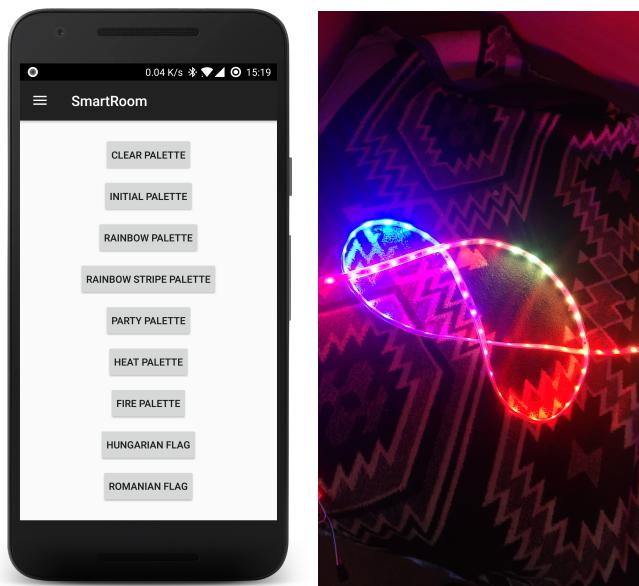


**60. ábra:** Automatikus színválasztás funkció bekapcsolása

**61. ábra:** Ezek után a csúszkákat mozgatva automatikusan változtatja a színt

## 6.5. Color Palette mód

A gombokra való koppintással a LED sor vezérlőbe beleégetett színpaletták jeleníthetők meg (62. ábra).



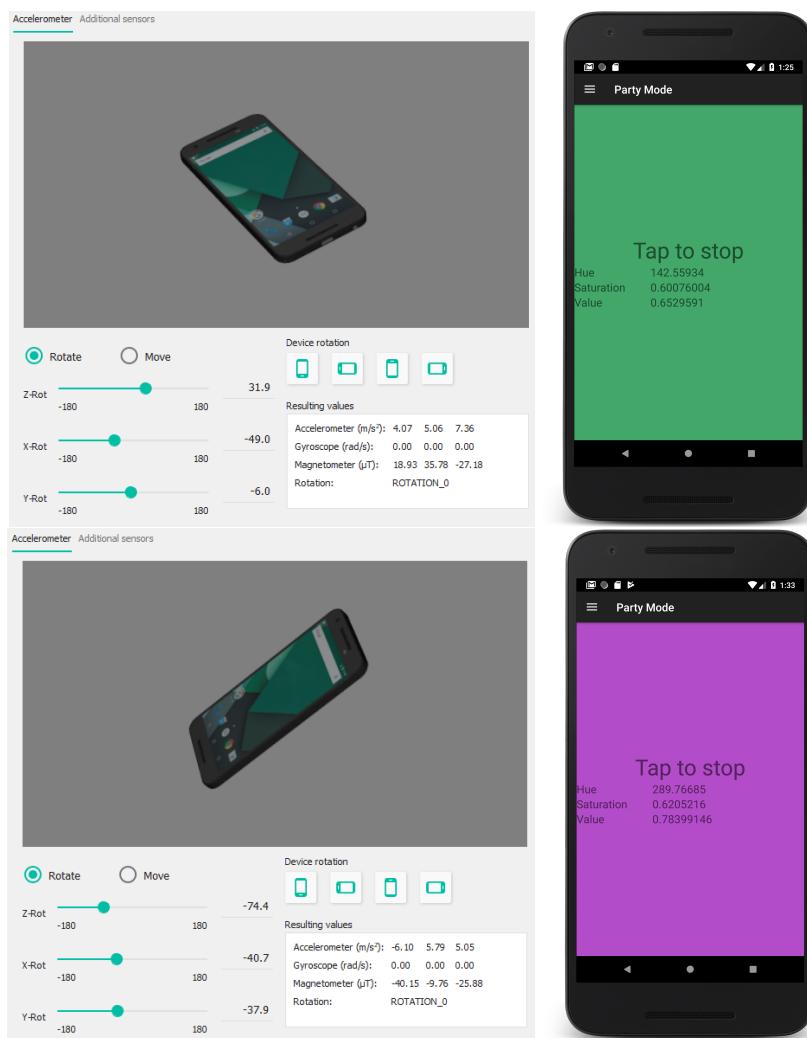
**62. ábra:** A *Party Palette* gomb megnyomására megjelenő minta a LED soron

## 6.6. Party mód

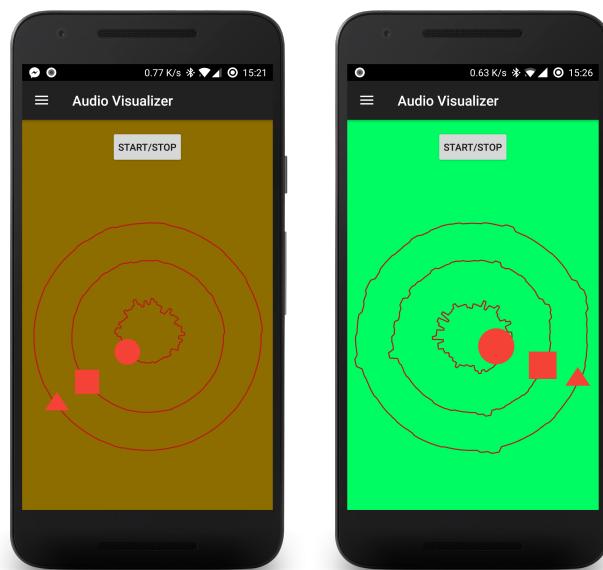
Ebben a módban a telefon orientációjával állítható be a LED sor színe (63. ábra), vagyis ha táncolunk vagy ugrálunk akkor aszerint változik a szín. A fejlesztői beállításokban ki lehet választani, hogy melyik tengelyek körül orientáció, melyik komponensnek feleljen meg a HSV (Hue Saturation Value) színtérből.

## 6.7. Audio Visualizer

Az alkalmazásba hardcodeolt zeneszámot a Start/Stop gomb megnyomásával indítható el. Az zene elindításával körkörösen mozgó mély, közép, és magas értékek találhatóak, rendre a kör középpontjától kifelé haladva (64. ábra). Ezen adatokat az Androidba beépített Visualizer API FFT (Fast Fourier Transform) mintakódjával nyerem ki. A mély értékekből számítom a HSV színtér Value értékeit, ezzel a LED sor elsötétedését és felvillanását állítva. A megjelenítendő szín (Hue) a magas és mély adatok összekombinálásával határozzom meg.



**63. ábra:** A telefon orientációja és a hatására megjelenő színek

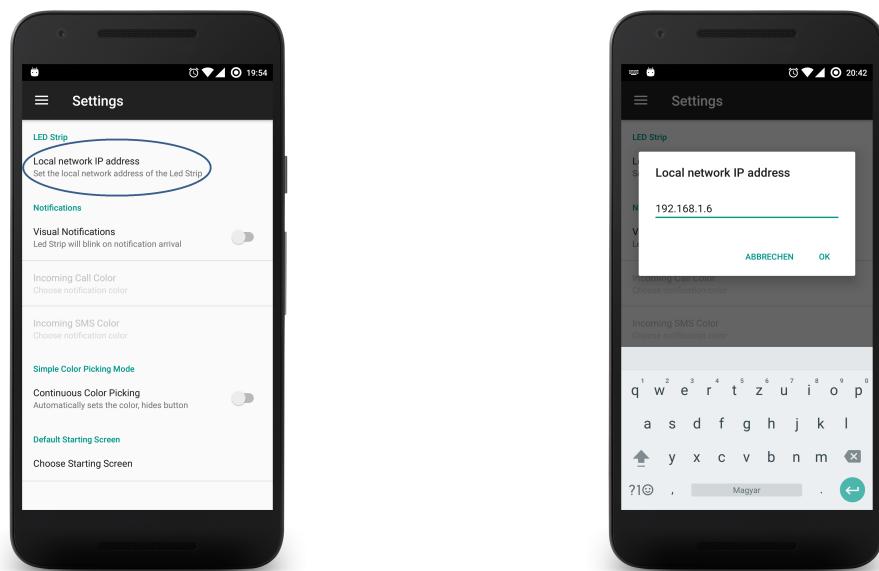


**64. ábra:** Szín változtatás a zene spektruma alapján

## 6.8. Beállítások menü

Ezen a nézeten lehet testre szabni az alkalmazást.

### 6.8.1. Helyi hálózati IP cím beállítása (65. és 66. ábra)



**65. ábra:** Koppintson a helyi hálózati IP **66. ábra:** Állítsa be az IP címet, majd koppintson az Ok gombra!

### 6.8.2. Vizuális értesítők bekapcsolása

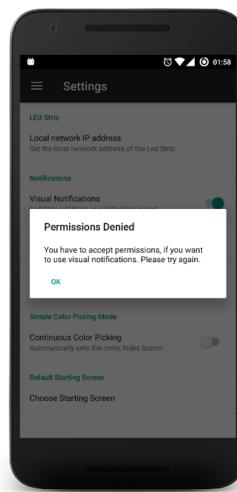
Ennek funkciójának a használatához el kell fogadni a megfelelő engedélyeket, különben nem fog működni.

Amennyiben az engedélyeket nem adjuk meg, egy hiba üzenet ugrik fel (67. ábra).

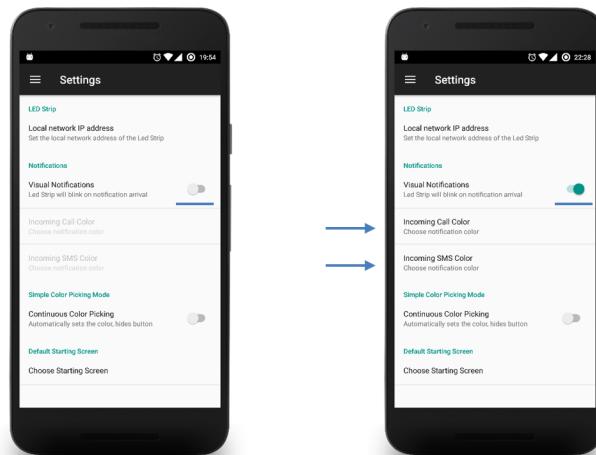
Ha mégis használni szeretnénk ezt a funkciót, akkor újra be kell állítani, majd ezt követően megadni meg az engedélyeket.

A vizuális értesítések bekapcsolása után további két beállítás válik elérhetővé, amikkel testre lehet szabni, hogy hívás, vagy sms érkezése esetén milyen színnel villogjon a LED sor ( 68. ábra).

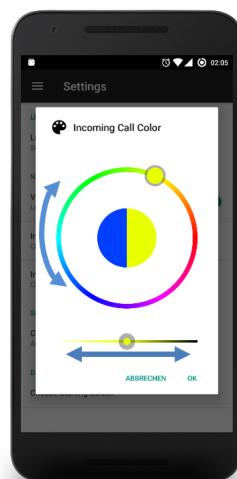
A bejövő hívás vagy SMS szín opcióra koppintva előhozhatunk egy ablakot (69. ábra), ahol beállíthatjuk a kívánt értesítési színt.



67. ábra: Felugró hibaüzenet



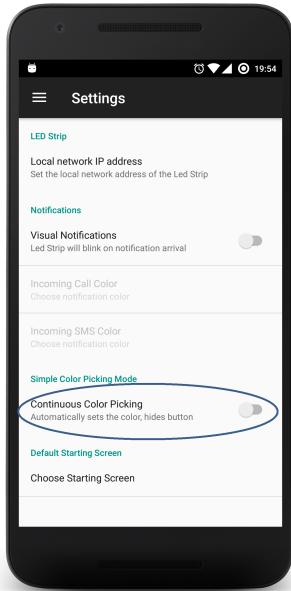
68. ábra: Az engedélyek elfogadása után elérhető két új beállítási lehetőség



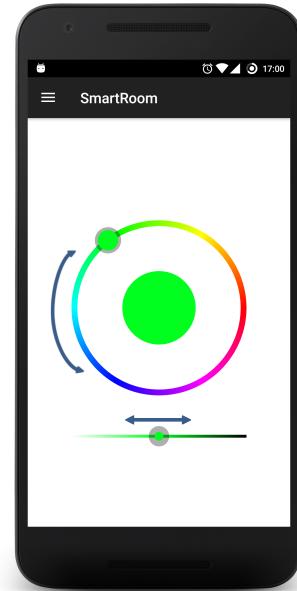
69. ábra: Bal oldalt a jelenlegi, jobb oldalt a beállítani kívánt szín látható, amit az Ok gomb lenyomásával ment el az alkalmazás.

### 6.8.3. Automata színbeállítás a Simple Color Picker módhoz

Az automata színválasztás bekapcsolása és a hatására változó Simple Color Picker mód ablak ( 70. és 71. ábrák).



**70. ábra:** Automatikus színválasztás funkció bekapcsolása



**71. ábra:** Ezek után a csúszkákat mozgatva automatikusan változtatja a színt

## 7. ÖSSZEOGLALÁS: ELKÉSZÜLT ESZKÖZ ÉRTÉKELÉSE ÉS KÖLTSÉGTERV SZÁMOLÁSA

### 7.1. Az elkészült eszköz értékelése

Összesen 5 eszközhöz elegendő alkatrészt rendeltem meg, amelyekből 3 kész LED sor vezérlő került sikeresen elkészítésre. Kettő javított verziójú és egy pedig az eredetileg elkészített NYÁK-kal működik. Mind a három egység stabil, üzembiztos viselkedést mutat. Az egyik egység az otthoni teraszunkra (72. ábra), a másik a kollegiumi szobába került felszerelésre. A harmadik egység tartalékként működik. Az eszköz tesztelve lett akkumulátorról is, egy Halloween-i jelmez kellékeként. Három cellás (12V-os), 2200mAh-s LiPo (Lítium Polimer) akkumulátorról üzemeltetve 5 órán keresztül működött. Az elkészült Android-os alkalmazással és LED sor vezérlővel a feladat célkitűzéseit teljesítettem.



**72. ábra:** Az elkészült eszköz különböző színeiben pompázva

A feladat során megtanultam a Git verzió kezelő rendszer használatát, amivel többek között nyomon követhettem a haladási folyamatomban. Elmélyítettem az objektumorientált Java, Android, illetve a hardverközeli C programozói tudásom is. Megtanultam a *datasheet*-ek, *reference manual*-ok, és *application note*-ok segítségével megvalósítani beágyazott szoftvert. Megterveztem és legyártattam a LED sor vezérlő áramkörömet. Átismételtem, és kibővítettem paraméteres modellezési technikával a 3D-modellezési ismereteim. És végső soron új tapasztalatokat, ismereteket szereztem a forrasztás és a 3D nyomtatás területén.

## 7.2. Költségterv számolása

A költségtervbe csak az elkészült eszközök (3 db) alkatrész igényeit számoltam bele, illetve a működéshez szükséges perifériákat. Az feltüntetett árak csak körülbelüli értékek, illetve árfolyam függők.

Darab	Eszköz megnevezése	Egységár	Összesített ár
6	5m-es LED szalag	4500 Ft	27000 Ft
3	12V-os 5A-es egyenáramú áramforrás	3300 Ft	9900 Ft
3	ESP8266 Wifi modul	500 Ft	1500 Ft
1	ST-Link V2 programozó	600 Ft	600 Ft
1	2x5 db NYÁK legyártása és szállítási díj	8500 Ft	8500 Ft
1	Mouser Electronics-tól rendelt alkatrészek	22000 Ft	22000 Ft
3	Védődoboz nyomtatása	1000 Ft	3000 Ft
1	Extra kábelek, csatlakozók	1500 Ft	1500 Ft
Összesen:			74000 Ft

Három LED sor vezérlő legyártásához, vezérlőként két LED szalaggal, összesen 74000 Ft-ra van szükség. Tehát egy darab vezérlő egység (két LED sorral) körülbelül 25000 Ft-ba kerül. Ez az összeg már így is vetekszik a Philips és LIFX által forgalmazott eszközök áraival, de sorozatgyártással az ár tovább csökkenthető. A Mouser Electronics adataiból kiindulva, az árak 4000 darabos széria számánál a harmadára csökkennek, feltételezhetjük az összköltségünk is legalább a felére fog csökkenni.

## 7.3. Fejlesztési lehetőségek

Mind a beágyazott rendszer, mind az elkészült Android-os alkalmazáson nagyon sok minden tovább lehetne fejleszteni, illetve rengetek plusz funkcióval elláttni. Első lépésben a kommunikációs protokoll továbbfejlesztésével kezdeném, hogy komplexebb szerkezetű adatokat, mint például új színpalettákat, lehessen küldeni a vezérlőegység számára.

A beágyazott rendszer esetében fontosnak tartom egy erősebb mikrokontroller és egy mikrofon alkalmazását. Így nem kellene az Androidos alkalmazáson keresztül a zeneszámot Fourier transzformálni és 20-30 milliszekundumonként színeket küldözgetni a vezérlő egységnek, hanem saját magának számolná ki a szükséges adatokat. Ezzel jelentős mennyiségben lehetne a hálózati kommunikáció forgalmát csökkenteni és a telefon akkumulátoridejét növelni. A szoftvert fel lehetne készíteni komolyabb animációk megjelenítésére, folyamatok beüzemelésére. Fejleszteni lehetne a zenére való színváltoztatást, kivilágosodást.

Az Android-os applikációban biztosítani lehetne a különböző zeneszámok lejátszását. Ki lehetne bővíteni hangvezérléssel, és egy olyan nézettel, ahol a ledgroup

csoportoknak, de akár egyesével is, lehetne változtatni a színét. Több nyelvű felhasználói felület megalkotása sem lenne utolsó szempont, az angolul nem tudó felhasználók számára.

## HIVATKOZÁSOK

- [1] Michael Barr. *Embedded C Coding Standard*. 2009.
- [2] Michael Barr. *Programming embedded systems: with C and GNU development tools*. 2006.
- [3] clean code - android-guidelines.  
[https://github.com/ribot/android-guidelines/blob/master/project\\_and\\_code\\_guidelines.md](https://github.com/ribot/android-guidelines/blob/master/project_and_code_guidelines.md)  
hozzáférés dátuma: 2018-12-13.
- [4] Arun Kumar, Pushpendu Kar, Rakesh Warrier, Aditi Kajale, and Sanjib Kumar Panda. Implementation of Smart LED Lighting and Efficient Data Management System for Buildings. *Energy Procedia*, 143:173–178, dec 2017.
- [5] LED Lighting.  
<https://www.energy.gov/energysaver/save-electricity-and-fuel/lighting-choices-save-you-money/led-lighting>  
hozzáférés dátuma: 2018-12-12.
- [6] Színvisszaadás – Wikipédia.  
<https://hu.wikipedia.org/wiki/Színvisszaadás>  
utolsó megtekintés: 2018-12-12.
- [7] Color Temperature & Color Rendering Index.  
<https://www.eledlights.com/blog/post/color-temperature-color-rendering-index-what-the-numbers-tell-you/>  
hozzáférés dátuma: 2018-12-12.
- [8] 5 reasons why your next light bulb should be a smart bulb - CNET.  
<https://www.cnet.com/how-to/why-your-next-light-bulb-should-be-a-smart-bulb/>  
hozzáférés dátuma: 2018-12-12.
- [9] Best Smart Light Bulbs 2018 - Smart Home Devices Lab Tested Reviews by PCMag.com.  
<https://www.pcmag.com/article2/0,2817,2483488,00.asp>  
hozzáférés dátuma: 2018-12-12.
- [10] Smart Home Lighting | Philips Hue.  
<https://www2.meethue.com/en-us>  
hozzáférés dátuma: 2018-12-12.

- [11] Philips Hue Smart lighting | Philips Hue.  
<https://www2.meethue.com/en-us/philips-hue-benefits>  
hozzáférés dátuma: 2018-12-12.
- [12] Hue White and color ambiance Starter kit E26 046677530228 | Philips.  
<https://www2.meethue.com/en-us/p/hue-white-and-color-ambiance-starter-kit-e26/046677530228>  
hozzáférés dátuma: 2018-12-12.
- [13] LIFX Color BR30 LED Smart Bulb.  
<https://www.lifx.com/products/lifx-br30-e26?variant=15387557396554>  
hozzáférés dátuma: 2018-12-12.
- [14] LIFX Tile Kit.  
<https://www.lifx.com/products/lifx-tile>  
hozzáférés dátuma: 2018-12-12.
- [15] TP-Link smarte WLAN Glühbirne.  
[https://www.amazon.de/TP-Link-Glühbirne-funktioniert-warmwei\T1\ss-erforderlich/dp/B01N3634DR/ref=sr\\_1\\_9?ie=UTF8&qid=1542973740&sr=8-9&keywords=SMART%2BLIGHT&th=1](https://www.amazon.de/TP-Link-Glühbirne-funktioniert-warmwei\T1\ss-erforderlich/dp/B01N3634DR/ref=sr_1_9?ie=UTF8&qid=1542973740&sr=8-9&keywords=SMART%2BLIGHT&th=1)  
hozzáférés dátuma: 2018-12-12.
- [16] EBay-en található LED szalag .  
<https://www.ebay.com/itm/0-5M-5M-5050-RGB-LED-Strip-Waterproof-USB-LED-Light-Strips-Flexible-Tape-DC-5V/263038757504?hash=item3d3e54ea80:m:m8cgVDTTWdemEaRY1oMGGDg:rk:1:pf:0>  
hozzáférés dátuma: 2018-12-12.
- [17] Eleonora Borgia. The Internet of Things vision: Key features, applications and open issues. *Computer Communications*, 54:1–31, dec 2014.
- [18] Jayconsystems - ESP8266.  
<https://www.jayconsystems.com/esp8266-wifi-tranceiver.html>  
hozzáférés dátuma: 2018-12-12.
- [19] WS2811 datasheet.  
<https://cdn-shop.adafruit.com/datasheets/WS2811.pdf>  
hozzáférés dátuma: 2018-12-12.

- [20] STMicroelectronics. STM32F103 datasheet, 2015.  
<https://www.st.com/resource/en/datasheet/cd00161566.pdf>  
hozzáférés dátuma: 2018-12-12.
- [21] Hacktronics WebShop.  
<https://hacktronics.co.in/microcontrollers/stm32f103c8t6-cortex-m3-32-bit-risc-core-72-mhz-microcontroller-lqfp48-package>  
hozzáférés dátuma: 2018-12-12.
- [22] STM32F103C8T6 minimum devBoard.  
<https://s3-ap-southeast-1.amazonaws.com/a2.datacaciques.com/00/NDAy/17/12/16/9e54ifgdt2zm4pk9/6fec5ff2017d7c73.jpg>  
hozzáférés dátuma: 2018-12-12.
- [23] Zoltan Hudak. STM32F103C8T6 board, alias Blue Pill.  
[https://os.mbed.com/users/hudakz/code/STM32F103C8T6\\_Hello/wiki/Homepage](https://os.mbed.com/users/hudakz/code/STM32F103C8T6_Hello/wiki/Homepage),  
hozzáférés dátuma: 2018-12-12.
- [24] ACROBOTIC Industries. ACROBOTIC ESP8266 ESP-01 Serial to Wi-Fi Module -.  
<https://acrobotic.com/products/acr-00020>  
hozzáférés dátuma: 2018-12-13.
- [25] Eagle library for the ESP8266.  
[https://github.com/wvanvlaenderen/ESP8266-Eagle\\_Library](https://github.com/wvanvlaenderen/ESP8266-Eagle_Library)  
hozzáférés dátuma: 2018-12-12.
- [26] LDO and DC-DC - Quora.  
<https://www.quora.com/Where-can-I-use-LDO-and-DC-DC-converters>  
hozzáférés dátuma: 2018-12-12.
- [27] DC/DC vs LDO.  
<https://electronics.stackexchange.com/questions/160218/how-does-dcdc-save-power-over-an-ldo>  
hozzáférés dátuma: 2018-12-12.

- [28] WEBENCH® Designer.  
<https://webench.ti.com/webench5/power/>  
hozzáférés dátuma: 2018-12-12.
- [29] TI. TPS56x219A, 2016.  
<http://www.ti.com/lit/ds/symlink/tps562219a.pdf>  
hozzáférés dátuma: 2018-12-13.
- [30] TI. SN74LV1T34 Single Power Supply Single Buffer GATE CMOS Logic Level Shifter, 2017.  
<http://www.ti.com/lit/ds/symlink/sn74lv1t34.pdf>  
hozzáférés dátuma: 2018-12-13.
- [31] STMicroelectronics. AN2586 - Application note Getting started with STM32F10xxx hardware development, 2011.  
[https://www.st.com/content/ccc/resource/technical/document/application\\_note/6c/a3/24/49/a5/d4/4a/db/CD00164185.pdf/files/CD00164185.pdf/jcr:content/translations/en.CD00164185.pdf](https://www.st.com/content/ccc/resource/technical/document/application_note/6c/a3/24/49/a5/d4/4a/db/CD00164185.pdf/files/CD00164185.pdf/jcr:content/translations/en.CD00164185.pdf)  
hozzáférés dátuma: 2018-12-13.
- [32] STMicroelectronics. RM0008 - Reference Manual, 2011.  
[https://www.st.com/content/ccc/resource/technical/document/reference\\_manual/59/b9/ba/7f/11/af/43/d5/CD00171190.pdf/files/CD00171190.pdf/jcr:content/translations/en.CD00171190.pdf](https://www.st.com/content/ccc/resource/technical/document/reference_manual/59/b9/ba/7f/11/af/43/d5/CD00171190.pdf/files/CD00171190.pdf/jcr:content/translations/en.CD00171190.pdf)  
hozzáférés dátuma: 2018-12-13.
- [33] Top 10 PCB Routing Tips for Beginners | EAGLE.  
<https://www.autodesk.com/products/eagle/blog/top-10-pcb-routing-tips-beginners/>  
hozzáférés dátuma: 2018-12-13.
- [34] Solid ground-plane vs Hatched ground-plane.  
<https://electronics.stackexchange.com/questions/5139/solid-ground-plane-vs-hatched-ground-plane>  
hozzáférés dátuma: 2018-12-13.
- [35] Standard Peripherals Library vs CMSIS vs HAL vs Low Level Library | PurpleAlienPlanet.  
<https://www.purplealienplanet.com/node/61>  
hozzáférés dátuma: 2018-12-13.

- [36] arm - STM32F4 and HAL - Electrical Engineering Stack Exchange.  
<https://electronics.stackexchange.com/questions/225568/stm32f4-and-hal>  
hozzáférés dátuma: 2018-12-13.
- [37] Peter Vass. program for the stm32f103.  
[https://github.com/petervass/led\\_strip](https://github.com/petervass/led_strip)  
hozzáférés dátuma: 2018-12-13.
- [38] Espressif Systems IOT Team. ESP8266 AT Instruction Set, 2016.  
[https://www.espressif.com/sites/default/files/documentation/4a-esp8266\\_at\\_instruction\\_set\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf)  
hozzáférés dátuma: 2018-12-13.
- [39] Android.  
<https://developer.android.com/about/>  
hozzáférés dátuma: 2018-12-13.
- [40] Android Things Starter Kit.  
<https://shop.technexion.com/pico-pi-imx7-startkit-rainbow-hat.html>  
hozzáférés dátuma: 2018-12-13.
- [41] Kathy. Sierra and Bert. Bates. *Head first Java*. O'Reilly, 2005.
- [42] Developer Guides | Android Developers.  
<https://developer.android.com/guide/>  
hozzáférés dátuma: 2018-12-13.
- [43] Bill (Software engineer) Phillips, Chris (Computer programmer) Stewart, Kristin Marsicano, and Big Nerd Ranch (Firm). *Android programming : the Big Nerd Ranch guide*.
- [44] Android Application Fundamentals.  
<https://developer.android.com/guide/components/fundamentals>  
hozzáférés dátuma: 2018-12-13.
- [45] Holo Color Picker.  
<https://github.com/LarsWerkman/HoloColorPicker>  
hozzáférés dátuma: 2018-12-13.

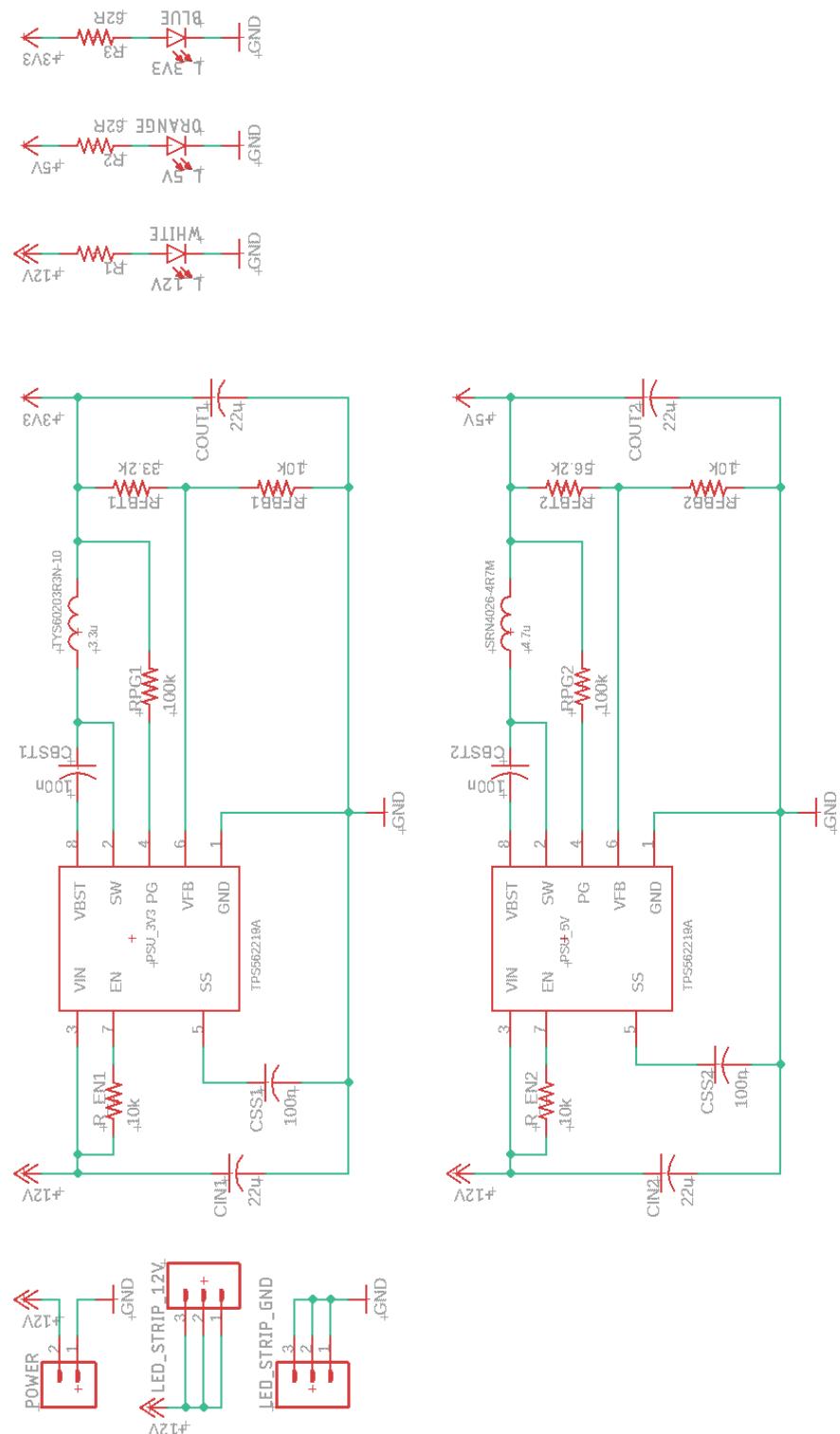
## SUMMARY

The goal of my thesis was to develop an android controlled wireless LED-lighting system. First, I researched the similar products already available on the market and set up my requirements. I accomplished the hardware development in three main phases (from the breadboard circuit until the manufactured PCB). I created a 3D modell of the enclosure housing of the board with the Autodesk Fusion 360 program and 3D printed it on my own Prusa i3 MK3 3D printer. I soldered and tested the PCBs and corrected the inaccuracies. In the meantime, I implemented the embedded software and the Android application. I configured the ESP8266 Wifi module with AT commands and connected it with the LED strip controller board. Finally, I installed the finished product on our teracce and in my dorm room.

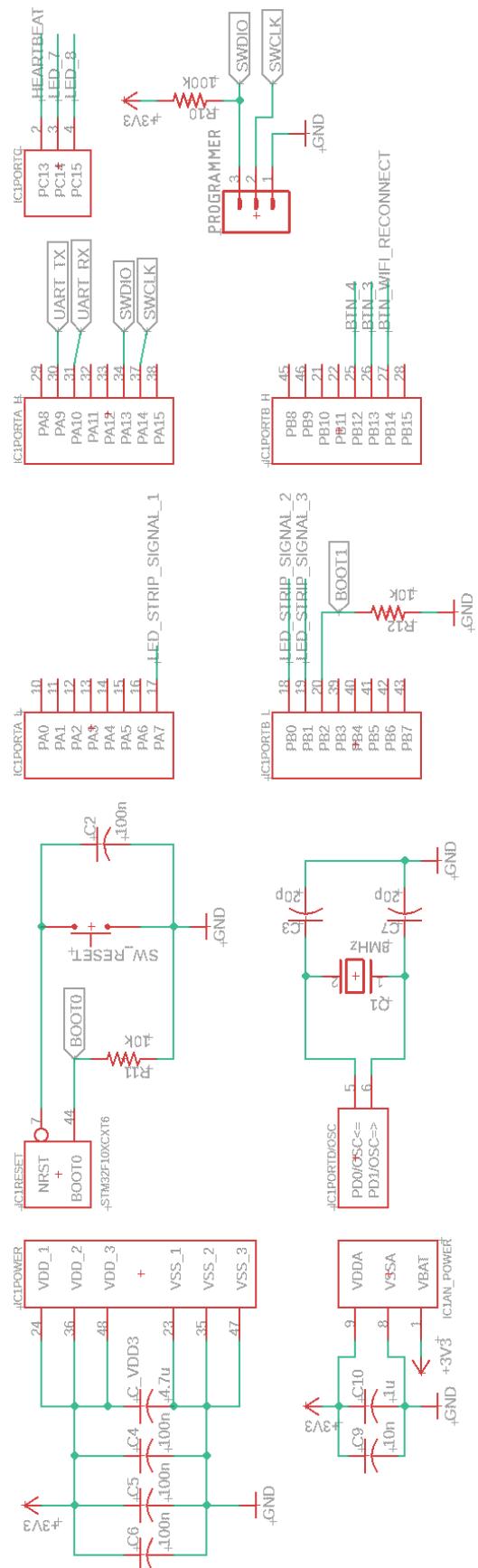
During my thesis, I learned how to use the Git version control system and GitHub, the hosting solution for Git repositories. I expanded my knowledge in the object oriented Java and Android programming. While writing the program, I attempted to use the language specific naming conventions, the developer guidelines and the clean code principles. I became familiar with the working mechanisms of microcontrollers, also on the level of registers. From the datasheets, reference manuals and application notes, I was able to implement my own embedded software. I learned how to plan and develop electrical circuits. According to the datasheets, I created the schematic layout. Corresponding to the layout design guidelines and the top-10-tips-by-electrical-engineers, I placed the components and routed the PCB, which was manufactured by JLCPCB. I refreshed my 3D modelling knowledge and completed it with parametric modelling skills. Lastly, I got more experienced with soldering and using an oscilloscope during the measurements and error correction.

## A. FÜGGELÉK

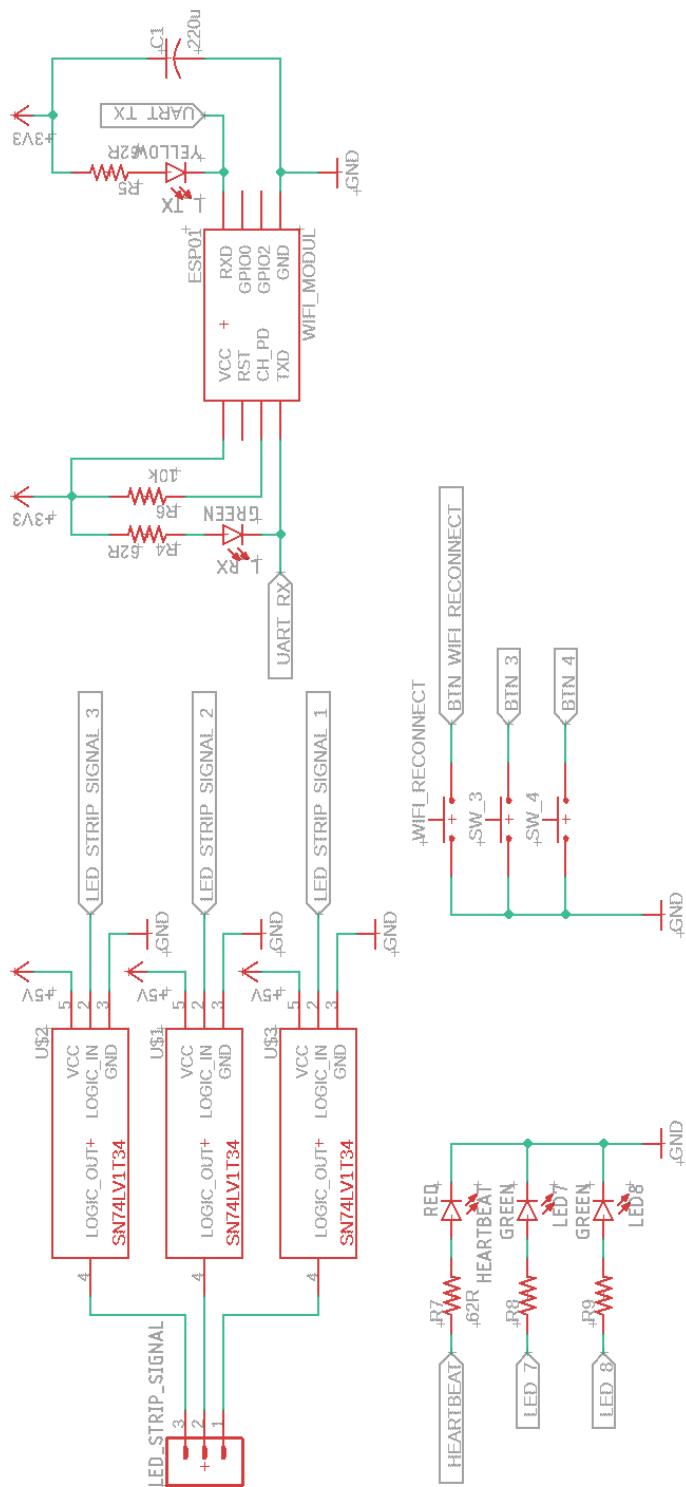
### A.1. Az elkészült LED sor vezérlő kapcsolási rajzai



73. ábra



74. ábra



75. ábra