Dinamikus szegmensdefiniálás ESP32 vezérlőn, ARGB LED-szalag alkalmazásához

Dynamic segment definition on ESP32 controller for ARGB LED strip

SIMON Róbert[1], Dr. TROHÁK Attila[2]

[1, 2]: Miskolci Egyetem, 3515 Miskolc-Egyetemváros, +36 46 565 111, www.uni-miskolc.hu

Abstract

The subject of my research is an automation solution that can be applied to fixed mood lighting, light painting, and aims to make the electronics that control it easier to program. In a previous research, I have already created a method to control a relatively large number of pixels in the form of short messages, and by this control, lighting segments are given unique colour and behaviour. In my current research, I now aim at a dynamic assignment of individual lighting segments, which handles the fine-tuning of the system in a convenient way, while also allow more flexible scenario definition.

Keywords: WAGO, ESP32, Smart Home, ARGB, Lighting control

Absztrakt

A kutatásom tárgya egy olyan automatizálási megoldás, amely fixen beépített hangulatvilágításokhoz, fényfestésekhez alkalmazható, fókuszálva az ezt vezérlő elektronika könnyebb programozhatóságára. Egy korábbi kutatásomban már létrehoztam azt a metódust, amellyel rövid üzenetek formájában, relatíve nagy számú pixel vezérelhető, és e vezérlés által a világítási szegmensek egyedi színt és viselkedést kapnak. A jelenlegi kutatásomban most az egyes világítási szegmensek dinamikus kijelölését tűztem ki célul, amely a rendszer finomhangolását kezeli kényelmes formában, egyben pedig rugalmasabb szcenáriódefiniálást is lehetővé tesz.

Kulcsszavak: WAGO, ESP32, Okos Otthon, ARGB, Világítás vezérlés

1. BEVEZETÉS

Napjaink világában egyre nagyobb hangsúlyt kapnak az olyan megoldások, amelyek a humán kényelem mellett, a mesterséges intelligencia egyszerűbb vagy összetettebb implementációját is alkalmazzák. Egyszerűbb alkalmazások közé tartozik az, amikor hangutasításokra, gesztusokra, vagy valamilyen eseményhez kötötten – például hőmérsékletváltozás, napszakok változása, időzítés – avatkozik be a logika a rendszer működésébe. Hőmérsékletváltozás esetén például bekapcsol a fűtés, vagy éppen ellenkezőleg, bekapcsol a klíma; a napszakok változásával felkapcsol a lámpa, leereszkedik a redőny, vagy éppen ellenkezőleg, a reggeli fény beköszöntével a redőnyök automatikusan felemelkednek. Időzítést pedig bármilyen esemény elindításához, vagy működésének limitálásához be tudunk állítani. Ezek mindegyike tehát az egyszerűbb implementációk közé tartozik, viszont már ezek alkalmazásai is jelentősen tudják növelni az egyén komfortérzetét. Olyannyira, hogy ilyen és ehhez hasonló alkalmazásokat, megoldásokat találunk a modern okos-otthon rendszerekben is. Természetesen a tényleges felhasználás, a kivitelezési helyszín lehetőségeit figyelembe véve, a tervező fantáziáján, a vevő igényein, és legvégül, az anyagi lehetőségeken múlik. [5]

Összetettebb implementációra még nagyon kevés példa van jelenleg, bár ez minden bizonnyal változni fog a jövőben, hiszen az elmúlt időszakban az egyszerű felhasználó számára is elérhetővé váltak olyan Mesterséges Intelligencia (MI) rendszerek, amelyekkel egyénileg paraméterezhető képek és szövegek, esszék generálhatóak perceken belül. Persze ezen rendszereknek megvannak a saját korlátjaik, hiszen alapvetően az előállított eredmények (képek, szövegek stb.) emberek alkotta műveken alapulnak, azok felhasználásával készülnek. Ennek etikai, morális kérdéseit nem tisztem e cikkben vitatni, ahogyan az emberi tanulás analógiáját sem kívánom vizsgálni az MI szempontjából. Ám az jól látszik, hogy népszerűsége exponenciálisan növekszik, újabb és újabb nyelvi modellek kerülnek kifejlesztésre, amelyek hónapról hónapra újabb lehetőségeket mutatnak be felhasználásukra. Ekképpen a modern okos-otthonok is előbb-utóbb adaptálni fogják őket, bár az még nem világos, hogy milyen módon. [6]

2. A VILÁGÍTÁS SZEREPE AZ OTTHONBAN

Az okos-otthonok kényelme igen tág fogalom, nagyon sok olyan összetevője van az emberi individuumnak, ami jelentőséggel bír, és ez ráadásul egyénenként, sőt hangulattól függően is változik. A hőérzet és a fizikai tevékenységek közötti összefüggést az ember az élete során, már gyerekkorban felismeri, ám számos más összefüggés is van az ember és környezete között, amikre sokszor nem is tudatosan, de reagál – például hangulattal, motiváltsággal, türelemmel, fókuszáltsággal, vagy éppen ezek ellentétjeivel. A reakció mértékét számos tényező befolyásolja, így például a személy neme, személyisége, hajlamossága, neveltetése, de a pillanatnyi lelkiállapota, fizikai kondíciója is ugyanígy fontos. Ezek tudatos felismerése és alkalmazása a környezetünkben, jelentősen hozzájárulhat a kívánt tevékenység hatékonyságához. [7]

Jelen kutatásomban most a világítással foglalkozok, és azon belül is egy olyan megoldással, amely képes a vezérlő logika utasításainak megfelelően, a rábízott világítótestek, LED-ek fényeit tetszőlegesen szegmentálni, és az egyes szcenáriók, vagy egyéni döntés, beavatkozás alapján azokat dinamikusan változtatni.

2.1. A világítási rendszer felépítése

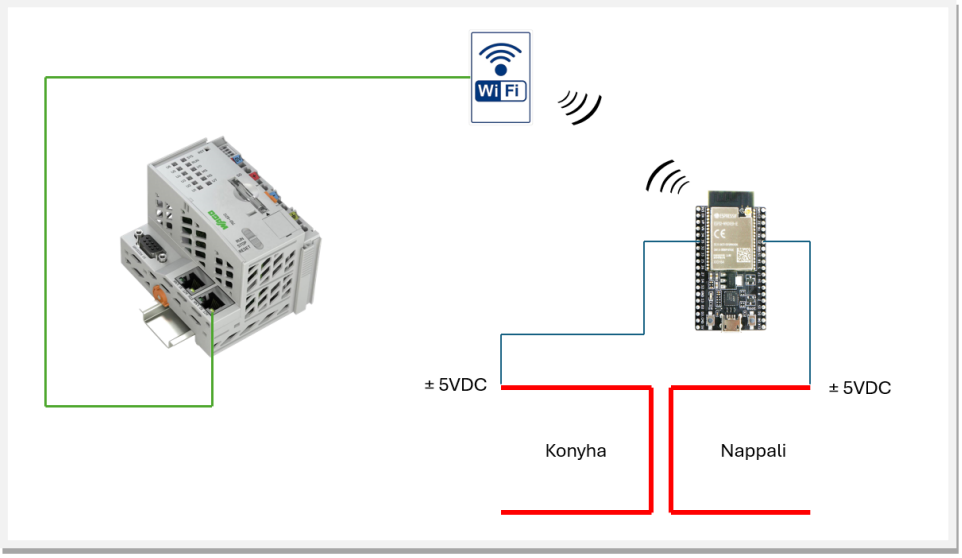
A rendszer egy családi házban található, amelyben a nappali, a konyha és az étkező egyetlen helyiségben van, így egyetlen légtér több funkciónak is egyidejűleg helyt tud adni. Az 1. ábra e helyiség mennyezetét mutatja, amelyen jól láthatók azok a téridomok, amelyek az egyes helyiség-szegmenst elválasztják egymástól.

1. ábra. A multifunkciós nappali tagolt mennyezete

A kép a konyharészben állva készült, így a kép felső része a konyha mennyezetét mutatja, míg lejjebb haladva, a szemben lévő falon végződve található a nappali szegmense, ahol a TV és pihenő terület is található. A kettőt pedig egy keresztirányú téridom, az összekötő határolja egymástól, ami alatt, középen található az étkezőasztal. Az egyes helyiségek megvilágításáról lámpák gondoskodnak, ám a cél, hogy a téridomok belsejében is elhelyezzünk világítást, így festve a mennyezetet és adva fényt a szobának. E megoldással a következő világítási módok érhetőek el:

- hangulatvilágítás
- folytonos fényáram
- fókuszált terület
- indirekt megvilágítás
- vészvilágítás
- parti fények

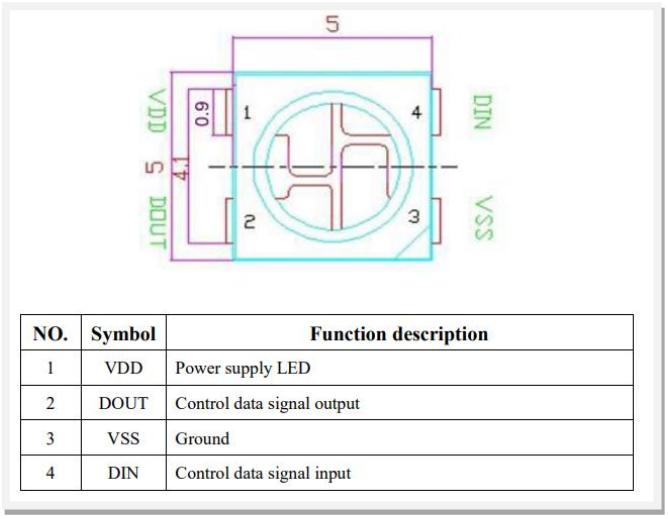
Ezen elvárások sokfélesége hagyományos LED-szalagokkal nem lenne kivitelezhető, ezért ARGB (Addressable RGB – Címezhető RGB) LED-szalag kerül majd alkalmazásra. Ezek a szalagok képesek pixelenként változtatni a színüket, ám ehhez természetesen külön vezérlő is szükséges, nem elegendő csupán a tápfeszültséget szabályozni színcsatornánként, vagy szegmensenként. A rendszer felépítését a 2. ábra mutatja, nem méretarányosan.



2. ábra. A világítási rendszer felépítése

Van tehát egy WAGO PFC200-as PLC (Programmable Logic Controller – Programozható Logikai Kontroller), amely közös Ethernet hálózaton van egy WiFi Access pointtal. Az ESP32-es vezérlőt a PLC így a WiFi-n keresztül éri el, és ezen keresztül tud kommunikálni egymással a két eszköz. Az ESP32 vezérlő feladata a LED-szalagok, illetve az ARGB pixelek vezérlése, amit egy-egy PWM képes digitális pinen keresztül végez el. A LED-szalagok összesített mérete 20 m, így külön 5V-os egyenáramú betáplálás szükséges, amely egyben az ESP32-es vezérlő tápfeszültsége is.

Az ARGB LED-szalag pixeleinek, illetve az azokat alkotó chip-jeinek felépítése a következő - 3. ábra:



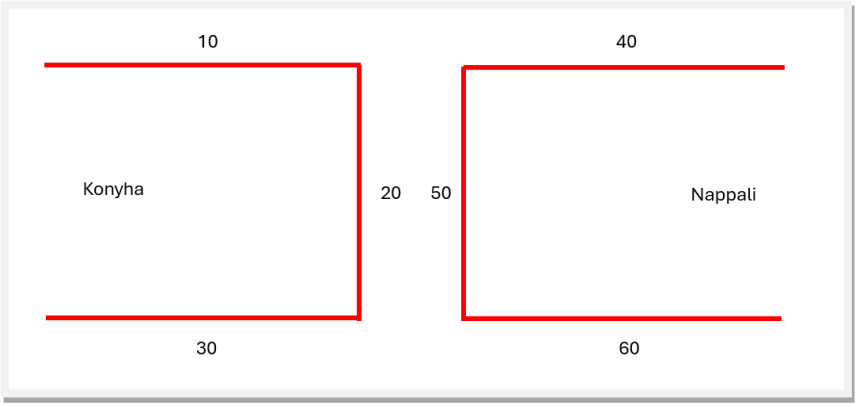
NO.	Symbol	Function description
1	VDD	Power supply LED
2	DOUT	Control data signal output
3	VSS	Ground
4	DIN	Control data signal input

3. ábra. WS2812B LED chip

Az ARGB chip típusa WS2812B [1], és egy 5050-es méretű tokozásban kapott helyet, és négy ponton csatlakozik a külvilághoz. A WS2812B chip továbbfejlesztése az SK6812 [2] azonosítójú chip, ami számos ponton fejlődött az eredetihez képest, ám külső felépítése és kontaktusai ugyanazok maradtak. Így két adat érintkezője (adat-be és adat-ki) mellett a tápfeszültség két érintkezője van, amely 5VDC tápfeszültséget igényel. Fontos különbség azonban, hogy az SK6812 már képes egy negyedik csatornát is alkalmazni, így megjelenik az RGB-csatornák mellett egy fehér színű, W-jelű csatorna is. A LED-es fehér fények sajátja, hogy különböző színhőmérsékletek állnak rendelkezésre, amely sajátosság ezekbe a chip-ekbe is örökül. Így a LED-szalag kiválasztásánál szempont a W színhőmérséklete is. Egyébiránt az ilyen chip-ekkel szerelt LED-szalagokat ARGBW LED-szalagoknak is szokták hívni.

3. A DINAMIKUS SZEGMENSDEFINIÁLÁS

A dinamikus szegmensdefiniálás célja, hogy olyan rendszer álljon össze, amely során a vezérlő, a kezdeti felprogramozása után ne kényszerüljön újabb programozásra. Ugyanis, ha a LED-szalagok a helyükre kerülnek és a tápfeszültség is bekötésre kerül, akkor alapvetően a rendszer automatikusan üzemkésszé kell, hogy váljon. Előfordulhat azonban az az eset, hogy az alapfelépítés - lásd 4. ábra során, a kézenfekvő sorszámozás nem megfelelő eredményt hoz.



4. ábra. A szegmensek alapértelmezett számozása

Ilyen esetben célszerűen meg kell változtatni ezeket a szegmenseket, de ehhez – ha hard-code-olt szegmensek vannak – újra kell programozni a vezérlőt, és nem megfelelő PLC program esetén a PLC felhasználói programján is módosítani kell. Mi okozhat ilyen igényeket? Például az, ha a 20-as szegmens esetén kettős átmenetet szeretnénk megvalósítani. Például a 10-es és 30-as szegmensek legyenek sötétek, viszont a 20-as sáv közepe legyen világos. Ekkor két szegmensre van szükség, egy sötét-világos átmenettel, és egy másik világos-sötét átmenettel. Erre egyszínű világítási igénynél nincsen szükség. Ugyanígy olyan igény is megjelenhet, ami az alapszegmenseken átível. Például a 10-es közepétől a 20-as harmadáig tart egy szakasz, a 20-as szakasz középső harmada egy újabb szakasz, majd a 20-as harmadik harmadától a teljes 30-as szakasszal bezárólag alakítunk ki egy újabb szakaszt. Ekkor az átmenetek automatikusan képződnek, és szép grádienseket kapunk, tetszőleges kezdő és befejező színeket is választhatunk.

A másik megoldás az lenne – ha nem lenne lehetőség dinamikus szegmentálásra –, ha az alapszegmenseket mikroszegmensekre bontanánk, például 10-10 szegmensre. Így a 10-es szakasz 10-től 19-ig terjedne. Ám ebben az esetben számolni kell az egyes pixeleket, hogy hanyadiknál kezdődik és hanyadiknál ér véget a mikroszakasz. Ez azonban még egyszerű feladatnak is tűnik, ahhoz képest, hogy egy teljes, mind a tíz szakaszt érintő grádiens esetén mi magunk kell meghatározzuk a kezdő és végpontok színeit is. RGB-s színátmenet esetén, ez igen összetett számítást igényel, hiszen mind a három csatorna átmeneteit számolnunk kell. Ez, az informatikai erőforrások ignorálását jelentené, humán oldalról pedig extra erőfeszítést igényelne – holott ennek épp fordítva kellene működnie.

3.1. A PLC és az ESP32-es vezérlő kommunikációja

Szükségszerű volt tehát egy olyan megoldás, amivel a szegmensek megvalósítása dinamikusan, a PLC által kiadott paranccsal megvalósítható. Mielőtt azonban ebben elmerülnénk, nézzük meg, hogy jelenleg hogyan kommunikál egymással a PLC és az ESP32-es vezérlő.

Az alapkritérium az volt, hogy olyan kommunikációt hozzak létre, amivel nagy számú pixeleket tudok, relatív rövid üzenetek formájában vezérelni. Ehhez egy egyszerűen programozható, UDP (User Datagram Protocol – Felhasználói Adatcsomag Protokoll) alapú kommunikációt választottam, amit mind a PLC, mind az ESP32-es vezérlő kezelni tudott. Az üzeneteket pedig a következő séma alapján hoztam létre:

1. Fejléc (4 byte)
2. Fényerősség (1 byte)
3. Adatcsomag (8 byte * n)
4. CRC (2 byte)
5. Farok (4 byte)

A *Fejléc* és a *Farok* részek 4-4 byte-ja fix karaktersorozat, amivel elkerülhetők a véletlen üzenetértelmezések a hálózat egyéb UDP műsorszóróitól. A *CRC* (Cyclic Redundancy Check – Ciklikus Renduncia Ellenőrzés) az üzenet helyességét hivatott biztosítani, alkalmazásával csekély kockázata van a téves világítási paraméterek realizálásának. A tényleges adatsor kettő lényeges elemből áll. Az első a *Fényerősség*, amivel a teljes szalag fényereje szabályozható, vagy akár le is kapcsolható. Így elkerülhető, hogy a kurrens szcenárió újra elküldésre kerüljön, és egyúttal a le-fel kapcsolás eseménye is kényelmesen programozhatóvá válik. A másik lényeges elem az *Adatcsomag*, amelyben *n* db szakasz világítása paraméterezhető. Az *n* egy olyan pozitív egész szám, amely 0 is lehet, így tulajdonképpen pusztán a fényerősség is szabályozható a LED-szalag szegmenseinek módosítása nélkül.

A LED-szalag szegmensei azok a szakaszok, amelyekre a teljes LED-szalag felosztásra kerül. A pixelek sorban, egymás után helyezkednek el, így a szakasz definíciója kettő számot tartalmaz – a kezdő LED sorszámát, illetve a befejezőét. Mindegyik szakasz egyedi sorszámmal rendelkezik, így az *Adatcsomag*-ban ez feltüntetésre is kerül. Eképpen az *Adatcsomag* felépítése a következő:

1. AreaID (1 byte)
2. Motívum (1 byte)
3. Color1 RGB (3 byte)
4. Color2 RGB (3 byte)

Vagyis az *AreaID* lesz a szegmens azonosítója, sorszáma. A *Motívum* olyan jelző, ami a szegmens világítási típusát meghatározza, ezek jelenleg a *statikus*, a *grádiens* és a *direkt pixel* motívumok. Később ez bővíthető villogással, stroboszkóppal, esetleg mozgó (szegmensek közti transzfer) motívumok létrehozásával. *Statikus* világítás esetében csupán az egyik szín 3 byte-ja van használva, a másik 3 byte kihasználatlan marad. *Grádiens* esetén a két végpont színét kell megadni, a mikrovezérlő kiszámolja a köztes pixelek színeit és fényerejét, így a *grádiens* színátmenet sötétítésre és világosításra is használható (pl. feketéből fehérbe, és fordítva). A *direkt pixel* motívumban szintén csak a *Color1* használt a szín meghatározására, a *Color2* byte-jaiból kettő az abszolút pixel pozíció meghatározására alkalmas. Így a szegmenseket részben felülírva direkt világító pixeleket tudunk létrehozni, amelyek így egy-egy különleges szerepű LED vezérlését teszik lehetővé a teljes soron.

3.2. A dinamikus szegmentálás elméleti megoldása

A világítás vezérlési metódus megismerése után nézzük, hogy miként van lehetőség a szegmenseket dinamikusan definiálni. Az üzenetek azonosítására továbbra is megtartottam az üzenet fejlécét és farok részét, ahogyan a CRC szakaszt is, az adatok konzisztenciájának védelme miatt. A kérdés az, hogy az adatcsomag milyen adatokat tartalmazzon. Az előző, 3.1 pontban bemutatott szakaszonkénti vezérlés jól működő, és hatékony világításvezérlést eredményezett, tehát az adatcsomagnak e szegmenseket kell definiálnia. Mivel a direkt címzések esetén 2 byte-ot használtam a pixel címének meghatározásához, így összesen 65536 pixel címezhető. Ez 60 pixel/m LED-szalag esetén 1km-nyi hosszt képes lefedni, ami olyan nagy szám, hogy még sokkal nagyobb pixelsűrűség esetén is elméleti maximumot ad, vagyis a gyakorlatban nem fog kompromisszumot jelenteni a maximális LED-szám a világítási vezérléseknél. A másik lényeges eleme az előzőleg bemutatott vezérlési adatcsomagnak a szakasz azonosítására használt 1 byte, ami 256 különböző szegmenst képes meghatározni. Ezen ismeretek alapján tulajdonképpen kézenfekvő a következő üzenetfelépítés meghatározása:

1. Fejléc (4 byte)
2. Adatcsomag (5 byte * n)
3. CRC (2 byte)
4. Farok (4 byte)

Ebben a *Fejléc* és *Farok* ugyanúgy az üzenet meghatározására szolgál, ám célszerűen eltérő karaktersorozatokkal, hogy az üzenetfeldolgozó el tudja dönteni, hogy 5 vagy 8 byte-os *Adatcsomagokkal* fog dolgozni, illetve, hogy a kapott adatokkal mit is kezdjen. A *CRC* szerepe ugyanaz, mint az előző adatsor esetén, vagyis az adat konzisztencia biztosítása. Az *Adatcsomag* felépítése a következő szerint történik:

1. AreaID (1 byte)
2. Kezdő pixel – abszolút cím (2 byte)
3. Végpont pixel – abszolút cím (2 byte)

3.3. A dinamikus szegmensdefiniálás gyakorlati megvalósítása

Az előző módszerrel tehát dinamikusan definiálhatók lesznek a szakaszok, ám kérdés, hogy hogyan ültethető át a gyakorlatba a használata. Mivel a LED-szalag elemei pixelek, így a reprezentációjuk a vezérlő memóriájában a tömb adattípussal realizálható. Monokróm szalagok esetén elegendő egy 1 dimenziós tömb használata, ám ARGB vagy ARGBW szalagok esetén 3, illetve 4 dimenziós tömböket kell alkalmaznunk. Az ARGB LED-szalagok vezérléséhez az Adafruit Neopixel library-ja használható, amely képes arra, hogy az egyenként beállított pixelekhez rendelt színeket az egyes pixelekhez továbbítsa – adatcsomag formájában, hiszen ne felejtsük el, hogy egyetlen adatpin-ünk van, ami sorosan továbbítja az adatokat az egyes pixelekhez, illetve az egyes pixelek az adatokat egymásnak továbbadják. A vezérlő feladata annyi tehát, hogy az egyes pixelek fényerő paramétereit beállítsa, majd aztán kiadja a parancsot, hogy ez realizálódjon a szalagon.

A vezérlőben kettő darab ilyen tömb van, az egyik a library tömbje, amit a library kezel. A másik az a tömb, amelyet én állítok össze. Ez utóbbit a program bármely pontjáról elérem, és manipulálhatom, így a szegmens definíciók alapján, a színvezérlő adatcsomag érkezésekor a tömb meghatározott értékeit módosítani tudom. Ezek feldolgozása után pedig aktiválom az adatküldés függvényét és a változás láthatóvá is válik a szalagon. A dinamikus szegmens kezeléséhez egy újabb, 3 dimenziós tömbre van szükség, amely mérete

*255 * (1 + 2 + 2) = 1275 byte*

Azért használok 255 értéket csupán, mert egy értéket (a 0-t) fenntartom a szegmensek számának rögzítésére. Ekkor a szegmensdefiníciós adatcsomag *AreaID-ja* 0, a *Kezdő pixel* word-jében tárolom a szegmensek számát (pl. 10), a *Végpont pixelt* pedig figyelmen kívül hagyom. Ez azért lényeges, mert így az egyes szegmensdefiníciók külön üzenetben is elküldhetők, hogyha a vezérlő, vagy a PLC nem képes hosszabb, kB méretű üzeneteket küldeni, fogadni. A szegmensdefiníciós telegrammok ezért nincsenek kihatással a színezésre, illetve a LED-szalag fényeire. Azon majd a következő színvezérlő adatcsomag érkezésekor lesz látható módosítás, hiszen így a szegmens(ek) méreteinek változásával az érintett szakasz is változni fog.

4. ÖSSZEGZÉS

A kutatásom célja egy olyan metódus kidolgozása volt, amely képes egy fixen telepített világítási rendszeren – amennyiben az ARGB LED-szalagokat használ – dinamikusan új szegmenseket definiálni, vagy meglévőket módosítani. Véleményem szerint az itt bemutatott megoldás ezt olyan módon képes kezelni, amely a PLC és vezérlő már jelenleg is alkalmazott kommunikációját alapul véve, ahhoz illeszkedve valósul meg. Ez alapján a kutatást sikeresnek vélem és a gyakorlatban alkalmazhatónak találom. Gyakorlati alkalmazása a az 1. ábrán található lakóhelyiségben fog megtörténni az elkövetkező hónapok során. A további kutatások – a gyakorlati tapasztalatok összegzése után – a megfelelő szcenráriók megalkotására fognak fókuszálni. Mely tevékenységek milyen megvilágítással harmonizálnak, és milyen módon vezérelhető – vagy lehetőség szerint automatizálható – ezen megvilágítások váltása.

5. IRODALOMJEGYZÉK

[1] ***, WS2812B Intelligent control LED integrated light source. Adafruit, https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf (2024.02.02)

[2] ***, SK6812 Technical Data Sheet. Adafruit, https://cdn-shop.adafruit.com/product-files/1138/SK6812+LED+datasheet+.pdf (2024.02.02)

[3] ***, Adafruti_NeoPixel, github, https://github.com/adafruit/Adafruit_NeoPixel (2024.02.03)

[4] ***, Ethernet Shield Sending and Receiving String via UDP, https://docs.arduino.cc/tutorials/ethernet-shield-rev2/udp-send-receive-string/ (2024***, Ethernet Shield Sending and Receiving String via UDP, https://docs.arduino.cc/tutorials/ethernet-shield-rev2/udp-send-receive-string/ (2024.02.24)

[5] Szabó P., Összehasonlítás az okos otthon rendszerek között, https://intelligenshaz-okosotthon.hu/inspiralo-blogcikkek/harmadik-szamu-blog.html (2024.03.26)

[6] Juan C. Augusto & Chris D. Nugent, Smart Homes Can Be Smarter, Designing Smart Homes. Lecture Notes in Computer Science(), vol 4008. Springer, Berlin, Heidelberg

[7] C L B McCloughan, P.A. Aspinall, R.S. Webb, The impact of lighting on mood, International Journal of Lighting Research and Technology, vol 31, issue 3, 1999