



MATEMATIKAI ÉS INFORMATIKAI INTÉZET

# Programozható elektronikák alkalmazásai

Kompakt okos otthon létrehozása mikrokontrollerekkel

**Készítette**

Bagoly Gábor

Programtervező Informatikus

**Témavezető**

Dr. Geda Gábor

Egyetemi Docens

EGER, 2023

# Tartalomjegyzék

<b>Bevezetés</b>	<b>4</b>
<b>1. Kereskedelmi forgalomban elérhető, főbb okos otthon rendszerek</b>	<b>5</b>
1.1. Nagyobb cégek által létrehozott ökoszisztemák . . . . .	5
1.1.1. Mik is a virtuális asszisztensek? . . . . .	5
1.1.2. Amazon - Alexa Smart Home . . . . .	6
1.1.3. Google - Google Home és Google Nest . . . . .	7
1.1.4. Xiaomi - Mi Home . . . . .	8
1.2. Nyílt forráskódú rendszerek . . . . .	8
1.2.1. Mi az open source? . . . . .	8
1.2.2. Home Assistant . . . . .	9
1.2.3. OpenHAB . . . . .	10
<b>2. Alkalmazott eszközök</b>	<b>11</b>
2.1. A rendszer lehetséges hardver-elemei . . . . .	11
2.1.1. Mi az IoT? . . . . .	11
2.1.2. Raspberry Pi 4B . . . . .	11
2.1.3. NodeMCU - ESP-WROOM-32 . . . . .	12
2.1.4. ESP32-CAM - Wi-Fi-s kamera modul . . . . .	13
2.1.5. DHT22 - hőmérésklet és páratartalom szenzor . . . . .	14
2.1.6. Mi is az RFID? . . . . .	15
2.1.7. RFID-RC522 - RFID olvasó . . . . .	15
2.1.8. Szilárdtest relé . . . . .	16
2.2. Alkalmazott szoftverek, keretrendszerek, és programozási nyelvek . . . . .	17
2.2.1. C++ és az Arduino IDE . . . . .	17
2.2.2. HTML és CSS . . . . .	17
2.2.3. PHP . . . . .	18
2.2.4. JavaScript . . . . .	18
2.2.5. jQuery . . . . .	18
2.2.6. Chart.js . . . . .	18
2.2.7. Tailwind CSS . . . . .	19
2.2.8. dbDiagram.io . . . . .	19

2.2.9. Font Awesome . . . . .	20
2.2.10. MySQL . . . . .	20
2.2.11. PlantUML . . . . .	21
2.2.12. Fritzing . . . . .	21
2.2.13. Laravel . . . . .	21
<b>3. A rendszer felépítése és működése</b>	<b>23</b>
3.1. Raspberry Pi 4B alkalmazása . . . . .	23
3.2. Adatbázis felépítése . . . . .	24
3.3. Laravel működése . . . . .	25
3.3.1. Modellek és migrációk . . . . .	25
3.3.2. Route-olás . . . . .	27
3.3.3. Controller . . . . .	28
3.3.4. Blade . . . . .	29
3.4. A kezelői felület bemutatása és működése . . . . .	30
3.4.1. A főoldal . . . . .	30
3.4.2. Beállítások . . . . .	32
3.4.3. RFID beállítások . . . . .	34
3.4.4. RFID használati táblázat . . . . .	36
3.4.5. Hőmérsékleti és páratartalom előzmények - ChartJS . . . . .	37
3.5. Eszközök kommunikációja a webszerverrel . . . . .	38
3.5.1. Hőmérséklet és Páratartalom szenzor . . . . .	39
3.5.2. Eszköz kapcsoló . . . . .	40
3.5.3. Kamera . . . . .	41
3.5.4. RFID kártya olvasó . . . . .	42
<b>4. Tesztelések</b>	<b>44</b>
4.1. Cypress automatizált tesztelések . . . . .	44
4.2. Manuális tesztelések . . . . .	45
<b>5. Rendszer telepítése</b>	<b>48</b>
5.1. Raspberry beállítása mint Wi-Fi hozzáférési pont . . . . .	48
5.2. Webalkalmazás telepítése . . . . .	50
<b>Összegzés</b>	<b>52</b>
<b>Irodalomjegyzék</b>	<b>53</b>

# Bevezetés

Mai világunkban az okos otthonok és az okos eszközök rendkívül népszerűek és elterjedtek. Általában, hogy ha megkérdezünk valakit ezzel a témaval kapcsolatban, akkor nagy valószínűsséggel azt tudják mondani, hogy rendelkeznek legalább egy okos otthonban alkalmazható eszközzel. Ezek az eszközök lehetővé tehetik a kényelmesebb és hatékonyabb életmódot.

De mi is tesz egy eszközt okossá? Feltételezhetjük azt, hogy ha valamelyik eszköz internetre kapcsolódik, esetleg távolról beállíthatjuk, vagy automatizálhatjuk előre meghatározott dolgokra, akkor azt az eszközt „okosnak” tudjuk mondani.

Hogy ha valaki már rendelkezik több ilyen eszközzel, akkor bizonyára találkozott már azzal a problémával, hogy egy bizonyos ökoszisztemában<sup>1</sup> használatos eszköz nem feltétlenül tud működni egy másikban.

Minderre próbáltam egy olyan megoldást kitalálni, ami abból a szempontból közelíti meg mindezt, hogy még egy „nem okos” eszközt (például egy izzót) integrálok úgy a rendszerbe, hogy azt egyszerűen tudjunk kezelni bárholnan az otthonunkból. Mindehhez egy olyan rendszert hoztam létre, amelyben a háttérben lévő folyamatok lebonyolítását egy webes alkalmazás végez el, és az általunk ismert legtöbb eszközön használható, amin internetezni is tudunk: legyen az számítógép, Androidos, vagy iOS-es telefon, tablet, vagy akár okos óra is.

Azért esett erre a témara a választásom, mert számonra felettesebb érdekes az, hogy egy ilyen okos otthonban az eszközök hogyan is kommunikálnak, és szeretnék ebbe egy belátást nyerni, hogy hogyan is épül össze minden.

**Céljaim azok voltak, hogy:**

- Belelássak egy ilyen rendszer működésébe.
- Különböző programozható eszközök alkalmazását jobban megismerjem.
- Egy olyan általános kezelőfelületet tudjak létrehozni, amit könnyen tud a felhasználó alkalmazni.

---

<sup>1</sup> Az ökoszisztemája összekapcsolt termékek, szolgáltatások és technológiák hálózatát jelenti, amelyek együttműködnek annak érdekében, hogy zökkenőmentes felhasználói élményt hozzanak létre.

# 1. fejezet

## Kereskedelmi forgalomban elérhető, főbb okos otthon rendszerek

### 1.1. Nagyobb cégek által létrehozott ökoszisztemák

Amikor arra kerül a sor, hogy okos otthonot szeretnénk összeállítani, akkor ahhoz egy széles palettából tudunk választani eszközöket, legyen az biztonsági kamera, ajtózár vagy akár háztartási eszközök, mint például egy mosógép vagy robot porszívó. Amikor az okos otthon rendszert szeretnénk létrehozni.

Van néhány olyan eszköz, – például az okos izzók – amelyek szerencsére több okos otthon rendszerben is könnyen alkalmazhatóak. Ezek az eszközök általában ipari szabványokat használnak, mint például a Zigbee<sup>1</sup>, ami lehetővé teszi számukra, hogy együttműködjenek a különböző rendszerekkel.

Azonban vannak olyan eszközök is, amelyek csak egyetlen rendszerrel használhatóak. Ezek az eszközök saját szabványokat alkalmaznak, amelyek nem kompatibilisek másokkal. Vegyük például azt, hogy ha egy olyan okos eszközt veszünk, ami csak egy meghatározott okos otthon rendszerrel működik együtt, akkor az eszközt később nem tudjuk használni egy másik rendszerben.

Mindezek alapján, amikor okos eszközt vásárlunk, akkor nagyon oda kell figyelni, hogy kompatibilis-e a választott okos otthon rendszerünkkel, amit általában fel szoktak tüntetni az eszközök leírásában.

#### 1.1.1. Mik is a virtuális assziszensek?

A virtuális assziszensek olyan szoftveres programok, amik különböző feladatokat tudnak elvégezni felhasználó kérésére.

---

<sup>1</sup> Zigbee-t használnak például az Amazon Echo, a Google, az IKEA okos otthon termékei, és a Philips Hue

A virtuális asszisztensek úgy lettek megalkotva, hogy a felhasználó hang vagy esetleg chat alapon tudjon egyszerű kérdéseken keresztül kommunikálni vagy utasításokat adni. A virtuális asszisztensek olyan feladatokban tudnak segíteni, mint emlékeztetők létrehozása, üzenetek küldése, hívások indítása, interneten való keresés, időjárás előrejelzések felolvasása, okos otthoni eszközök irányítása, és egyéb más dolgok.

Számos cég hozott már létre magának virtuális asszisztentst, amik közül a legismertebbek lehetnek a Google által létrehozott Google Asszisztens, az Apple Siri, az Amazon Alexa, a Samsung Bixby, és a Microsoft Cortana.

Hogy ha például egy újabb Samsung telefonja van az embernek, akkor azon két virtuális asszisztens is jelen van (Google Assistant és Bixby), de letölthető akár mellé harmadiknak az Amazon Alexa is.

### **1.1.2. Amazon - Alexa Smart Home**

2014-ben lépett be a piacra az Amazon – *az akkor még újdonságnak számító* – okos hangszórójukkal, az Amazon Echo-val. Ekkor még leginkább csak annyira volt képes, hogy a felhasználó zenét tudja irányítani hang utasításokkal. Ez az eszköz úgy működik, hogy bele van integrálva az Amazon sajátos virtuális asszisztense, amit Alexának hívnak. Ugye mint kezdetleges szoftver, neki sem voltak a képességei túl szerteágazóak. Leginkább arra tudta az ember használni, hogy egyszerű kérdéseket tegyen fel, és zenét tudjon elindítani, leállítani átugrani.

Nem is sokkal később, amikor elkezdett egyre jobban fejlődni Alexa, úgy egyre több mindenre kezdhette el használni az ember: termosztátok beállítása, izzók ki- és bekapcsolására is lehetett használni. Miután az Amazon egyre többet fektetett a rendszerükbe, felettesebb szerteágazó lett annak a használata az otthonokban. Még arra is volt lehetőség már ekkor, hogy akár bevásárló listát készítsen, és azokat meg is tudja rendelni a felhasználó az Amazonról, mindezt Alexa használatával. A cég 2017. május 23-án jelentette be, hogy a *Smart Home Skill API*-jukba[1] innentől kezdve megadható, hogy milyen eszközöket is csatlakoztatunk a rendszerükbe, ami kitárta a lehetőségeket az otthoni okos készülékek automatizációjára. Mindez elvezetett az okos otthon piac és a virtuális asszisztensek szerepének növekedéséhez.

Az eszközök leginkább hangvezérléssel irányíthatóak, miután követtük az általuk biztosított használati útmutatót. Néhány esetben az Amazon Alexa alkalmazás elegendő lehet, de ha olyan eszközt szeretnénk használni, amelyhez saját alkalmazás tartozik, akkor azt is le kell töltenünk.

Az Amazon által szolgáltatott okos otthon rendszer 2023-ra olyan népszerűségi szintre jutott, hogy az Amerikai Egyesült Államokban az ilyen hang vezérelt hangszórók 68.2%-a az Amazon Echo.[2]

Napjainkban több ezer eszköz használható már ezen a rendszeren belül: legyen az

biztonsági rendszer, háztartási gépek, vagy esetleg szórakoztató rendszerek. Érthető is, hogy sokan miért is szeretik ezt a rendszert használni.

Azonban ennek is megvannak azok a hátulütői, mint például az, hogy nem tudunk használni benne olyan eszközöket, amik nem támogatottak az Amazon által. Érdemes arra odafigyelni, amikor egy okos eszközt veszünk, hogy arra fel van-e tüntetve, hogy kompatibilis az Amazon rendszerével. Másik ilyen negatív tényező lehet számunkra az is, hogy az Amazon Alexa még nem használható magyar nyelven.

### 1.1.3. Google - Google Home és Google Nest

Az Amazon Echo sikere után a Google is részesülni akart az okos otthon piacának sikereiből.

Az akkor még Nest Labs által készült termékek olyanok voltak, mint az öntanúló termosztát – amit 2011-ben hoztak létre „Nest Learning Thermostat”<sup>2</sup> néven, ami Wi-Fi-re kapcsolható volt, és szenzorok segítségével alkalmazkodhatott a beltéri hőmérsékleti körülményekre. Ezt követte a következő termékük, a füst és szén-monoxid érzékelő, aminek a neve „Nest Protect” volt. 2014-ben felvásárolták a Dropcam nevezetű céget, ami biztonsági kamerákat készítettek. Ezután a Nest Labs következő terméke 2015-ben egy Nest Cam nevezetű kamera volt.

A Nest Labs egyre jobban látszódó sikerének köszönhetően a Google felvásárolta 2014. januárjában. 2018-ig még önállóan működött a Nest, ami után beolvastották a Google otthoni termékcsaládba, ezzel létre hozva a Google Nest termékcsaládot, ami számos termékkal rendelkezik mostanára: termosztát, ajtócsengő, ajtózár, biztonsági kamera, virtuális asszisztenssel integrált érintőképernyős központi egység.[3]

Az Amazon Echo-hoz hasonló első terméke a Google-nek a Google Home volt, amit 2016. októberében jelentettek be. Ez a cég sajátos virtuális asszisztensével a Google Assistant-tel volt felszerelve, és ugyanúgy lehetett neki utasításokat adni, és kérdéseket felenni. Azóta már több otthonon belül alkalmazható eszköz elérhető egyenesen a Google Store-ból.[3]

Az egyik legnagyobb előnye a Google Home rendszernek, hogy integrálható más Google szolgáltatásokkal, mint például a Google Térkép, a Google Naptár és a Google Fotók. Ez azt jelenti, hogy a felhasználók kéz nélkül is hozzáférhetnek személyes információikhoz és ütemtervükhez, szimplán csak a Google Asszisztensnek feltéve a kérdést.

A Google Home rendszer másik erőssége az, hogy képes felismerni a különböző hangokat, amely lehetővé teszi a személyre szabott válaszokat és információkat minden háztartási tag számára. Ez különösen hasznos lehet több személyes háztartásokban, ahol több ember is használja a rendszert.<sup>3</sup>

---

<sup>2</sup> Magyarul: Nest Tanuló Termosztát

<sup>3</sup> Erre ugyanúgy betanítható az Amazon Alexa is.

Mindezek által a Google Home rendszer erős versenytárs az okos otthon piacon.

Viszont a Google Home-nak is meg vannak azok a hátrányai, mint az Amazon rendszerének. Sajnos inkább csak akkor tudjuk kihasználni ennek a rendszernek az előnyét, ha angolul vagy más támogatott nyelven beszélünk vele, amibe még nem tartozik bele a magyar.

#### 1.1.4. Xiaomi - Mi Home

A Xiaomi 2015. júniusában dobta piacra első okos otthon termékcsomagját, a „Smart Home Kit”-et, amely mozgásérzékelőt, lámpát és kapacitív kapcsolót tartalmazott. Ezeket az eszközöket egy alkalmazás segítségével lehetett vezérelni, ami lehetővé tette a felhasználók számára, hogy különböző utasításokat állítsanak be, például hogy a mozgásérzékelő észlelésekor a lámpa felkapcsoljon, vagy értesítést küldjön a felhasználónak, illetve a csomag támogatta a Xiaomi biztonsági kameráját is. [4]

Ma a Xiaomi okos otthon termékpallettája rendkívül sokrétű, a forrólevegős sütőktől, a robotporszívókon, és a szobamérlegeken át a hőmérséklet- és páratartalom-szenzorokig. Az összes termékük integrálható a Mi Home alkalmazásba, és a felhasználók harmadik féltől származó eszközöket is csatlakoztathatnak hozzájuk.

Nagy előnye a Xiaomi okos otthon termékeknek az, hogy viszonylag olcsóbb a konkurens termékektől, és fel is használható például a Google Home, és az Amazon Alexa Smart Home rendszeren belül is, és használata felettesebb felhasználóbarát.

Kisebb hátránya lehet az, hogy ezeket az eszközöket nem lehet asztali alkalmazáson keresztül irányítani, csakis az Androidos és iOS-es alkalmazáson keresztül, vagy esetleg a Google Home vagy Alexa Smart Home-on belül. Amazon Alexával a párosítás mosztanában sajnos nehézkesebb, mivel valamilyen oknál fogva nagy százalékban nem tud csatlakozni valami hibánál fogva. Még olyan is megesik, hogy nem minden eszköz érhető el például a Google Home felületen. Ilyen lehet példának a biztonsági kamerájuk, ami csak a saját alkalmazásukon érhető el, továbbá mint az előző két esetében, ez sem használható még magyar nyelven.

### 1.2. Nyílt forráskódú rendszerek

#### 1.2.1. Mi az open source?

Open source, azaz nyílt forráskódú szoftver az olyan, aminek a forráskódját szabadon lehet vizsgálni, módosítani, és akár ki is egészíteni. A kód az a része egy szoftvernek, amit a legtöbb felhasználó bizonyára soha nem fog látni. Ez az a kód, amit a programozók változtathatnak, hogy megváltozzon a program vagy alkalmazás működése. Azok a programozók, akik hozzáférhetnek egy szoftver forráskódjához, azokat feljeszthetik,

és javíthatják azzal, hogy például új funkciót adnak hozzá, vagy kijavítanak egy olyan részt, ami nem minden esetben működik helyesen.[5]

Azonban azt, hogy egy szoftver nyílt forráskódú, nem feltétlenül jelenti azt, hogy az adott szoftver ingyenes használatban áll. Nyílt forráskódot fejlesztő programozók kérhetnek pénzt azért a nyílt forrású szoftverért, hogy ha ők alkották meg, vagy hozzájárulásukkal készült el.

Bizonyos esetekben azonban, mivel a nyílt forráskódú licenc megkövetelheti a program forráskódjának kiadását, amikor szoftvert adnak el másoknak, egyes programozók úgy találják, hogy jövedelmezőbb pénzt kérni a felhasználóktól a szoftverszolgáltatásokért és - támogatásért (nem feltétlenül magáért a szoftverért). Így a szoftvereik ingyenesek maradnak, és pénzt keresnek azzal, hogy másoknak segítenek telepíteni, használni és hibaelhárítást végezni.[5]

### 1.2.2. Home Assistant

A Home Assistant 2013. novemberére került publikálásra *GithHub.com*-ra PAULUS SCHOUTSEN által. Ez ekkor még egy Python programozási nyelven megírt alkalmazás volt. Ekkor ez még csak egy egyszerű program volt, ami napnyugtakor felkapcsolta a lámpát. Azóta a szoftver elég érett lett, és körülbelül 20 aktív hozzájáruló dolgozik a projekten. Ennek köszönhetően kéthetente érkezik frissítés a rendszerre.[6]

Ez a szoftver bármely olyan rendszeren működik, ami a Python 3 programozási nyelvet tudja futtatni. Biztosít mobil és számítógép alkalmazást is, mellyel több ezer támogatott eszközt tudunk irányítani. Annyi a különbség ez a rendszer és a nagyobb cégek által biztosított között, hogy ez teljesen lokálisan fut, és hogy ha esetleg internet kimaradás lenne, még akkor is tudjuk ugyanúgy irányítani eszközeinket. A rendszer által számos ismert okos eszköz támogatott, mint például a Philips Hue, IKEA TRÅDFRI, és akár az Amazon Echo, Google Home, és a Xiaomi Mi Home által támogatott eszközök, és még a virtuális asszisztensük is.[7] Mindez annak köszönhető, hogy a Z-Wave és a Zigbee protokoll is támogatott a rendszer által, és mivel nyílt forráskódú a szoftver, ezért bármikor lehet az, hogy valamelyik felhasználó vagy esetleg fejlesztő hozzáadja bizonyos termékekkel kiegészítésként.

A rendszerrel háztartásunk energiafelhasználását is nyilván lehet tartani. Szabadon testre szabható a felület kinézete a felhasználók ízlése szerint. Szintén pozitívum lehet, hogy a szoftver támogatja a magyar nyelvet is.

Ámbár ez a rendszer azoknak ajánlott jobban, akik informatika terén mélyebb ismeretekkel rendelkeznek.

### 1.2.3. OpenHAB

OpenHAB, azaz Open Home Automation Bus<sup>4</sup> fejlesztése 2010-ben kezdődött, és Java programozási nyelven írták. 2013-ban került elérhetővé az első stabil verziója a szoftvernek.

Az OpenHAB legelső körben azon a személyek számára ajánlott, akik jól ismerik informatika és robotika területét. Ez azért szükséges, mert nekünk kell összeállítanunk, hogy mit szeretnénk elérni az okos otthon rendszerünkben. Az alkalmazás rendkívül rugalmas és testre szabható és nagy a támogatottsága. Ez az egyik legelterjedtebb nyílt forráskódú okos otthon rendszer, és folyamatos fejlesztés alatt áll egy nonprofit szervezet által, aminek a fejlesztésébe bárki bekapcsolódhat.

Ez a rendszer képes integrálni a piac különböző eszközeit, mint ahogyan azt a Home Assitant rendszernél is említettem. Szintén fontos tényezője, hogy lokálisan működik, ezért egy internet kimagasánál az okos otthonunk ugyanúgy fog működni.

A szoftvernek hivatalos oldalon megtalálható a teljes dokumentációja, hogy mit hogyan kell összeállítani és használni, ezért lehet kedvelt olyan személyek számára, akik szeretik a dolgokat maguknak összeállítani. Mivel a rendszer nagy támogatottságot élvez és nagy közösséget vonz, így ha felmerülne bármilyen kérdés, biztosan választ kapunk rá.

Az ezzel létrehozott otthonunkat a legtöbb felületen el tudjuk érni, legyen a MacOS, a Windows, a Linux, az Android vagy az iOS.[8]

---

<sup>4</sup> Magyarul: Nyílt Otthon Automatizációs Busz

## 2. fejezet

# Alkalmazott eszközök

Ebben a fejezetben azt fogom taglalni, hogy milyen eszközöket is használtam az okos otthon rendszerem létrehozása során. Legyenek ezek hardveres vagy szoftveres komponensek.

Mindezek közben fogok csatolni kapcsolási rajzokat, amik abba adnak betekintést, hogy hogyan is kell összekötni ezeket az eszközöket, hogy a rendszerben működjenek.

Szoftveres komponensek közé tartoznak azok a szoftverek, amik a projekt létrehozása közben fel voltak használva: legyen az keretrendszer, programozási nyelv, vagy stílus megírásra használt külső komponens.

### 2.1. A rendszer lehetséges hardver-elemei

#### 2.1.1. Mi az IoT?

IoT<sup>1</sup>-k azok olyan eszközök, melyek Wi-Fi hálózaton keresztül párosíthatóak, és irányíthatóak. Ilyenek lehetnek a háztartásunkban levő olyan eszközök, melyeket távolról is tudunk irányítani interneten keresztül. Legyen ez például a Xiaomi forró levegő sütője, vagy egy okos izzó.

Ebbe a kategóriába tartoznak a mikrokontrollerek is, amik mini programozható elektronikák, mint például az ESP32 és a Raspberry Pi 4B.

#### 2.1.2. Raspberry Pi 4B

A projektnek szíve-lelke egy bankkártya méretű mini számítógép, ami szolgáltatja a vezeték nélküli internetet az otthoni eszközök számára, ezzel megvalósítva a később említett programozható elektronikáknak is a kommunikációt.

A Raspberry dolgozza fel az adatokat, és ez szolgáltatja a webalkalmazást is. Felettesebb sokoldalú a használata, és előszeretettel használják okos otthon projektek megval-

---

<sup>1</sup> Internet of Things - magyarul: Internet dolgai

lósításában is, például hogy ha Home Assistant-el, vagy OpenHAB-bal szeretnénk azt megvalósítani.

Eben Upton 2012-ben hozta létre az első ilyen kis számítógépet, a Raspberry Pi 1 Model B-t. Raspberry magyarul azt jelenti, hogy málna mivel, ebben az időben többen is hoztak létre elektronikai termékeket gyümölcs nevekkel felruházva: legyen az Apple (alma), Acorn (makk), Apricot (sárgabarack).

Elsődleges célja az volt Eben Upton-nak hogy olcsó és könnyen elérhető számítógépeket juttassanak el fiatalok számára.

A Raspberry számítógépeknek elsődlegesen két fajtája volt: az A Model, ami olcsóbb, és a B Model, ami gyorsabb volt.[9]

Rendkívül sokoldalú ez a kis számítógép, és ezért is ennyire közkedvelt a használata a programozók körében.

#### **Az általam használt Raspberry Pi 4B specifikációi:**

- **RAM:** 2GB
- **Tárhely:** 16GB-os microSD memória kártya
- **Operációs rendszer:** Raspbian, ami egy Debian alapú Linux operációs rendszer

### **2.1.3. NodeMCU - ESP-WROOM-32**

A NodeMCU az Espressif Systems ESP8266-12E Wi-Fi System-On-Chip<sup>2</sup> rendszeren alapul, amely egy nyílt forráskódú, Lua-alapú firmware-rel<sup>3</sup> van felszerelve. tökéletes az IoT-alkalmazásokhoz és más olyan helyzetekhez, ahol vezeték nélküli kapcsolatra van szükség. Ez a chip nagyon sok hasonlóságot mutat az Arduino-val. Mindkettő mikrokontrollerrel felszerelt prototípus lap, amely az Arduino IDE segítségével programozható.[12]

Az ESP-WROOM-32 mikrovezérlő egy erős és sokoldalú eszköz, amely remek választás lehet egy okos otthon projekt számára. Alacsony energiaigényű processzora és integrált Wi-Fi és Bluetooth képességei alkalmasak szenzorok és eszközök vezérlésére és kommunikációjára.

Az ESP-WROOM-32S mikrovezérlő adatlapja részletes technikai információkat tartalmaz a funkcióiról és képességeiről.[10][11]

Éppen ezért is lehet elsődleges választás robotika oktatás közben egy NodeMCU használata. Én is az egyetemi tanulmányaim során a Robotikai alapjai névű tárgyon találkoztam ezzel az eszközzel először.

Rendszeremben ezeket az eszközöket úgy használom, hogy – kihasználva a Wi-Fi-s felszereltségüket – rákapcsolódnak a Raspberry Pi-ra, és ezen keresztül küldenek

---

<sup>2</sup> Magyarul: Rendszer a Chipen

<sup>3</sup> Magyarul: Fix tárban tárolt információ, rendszer

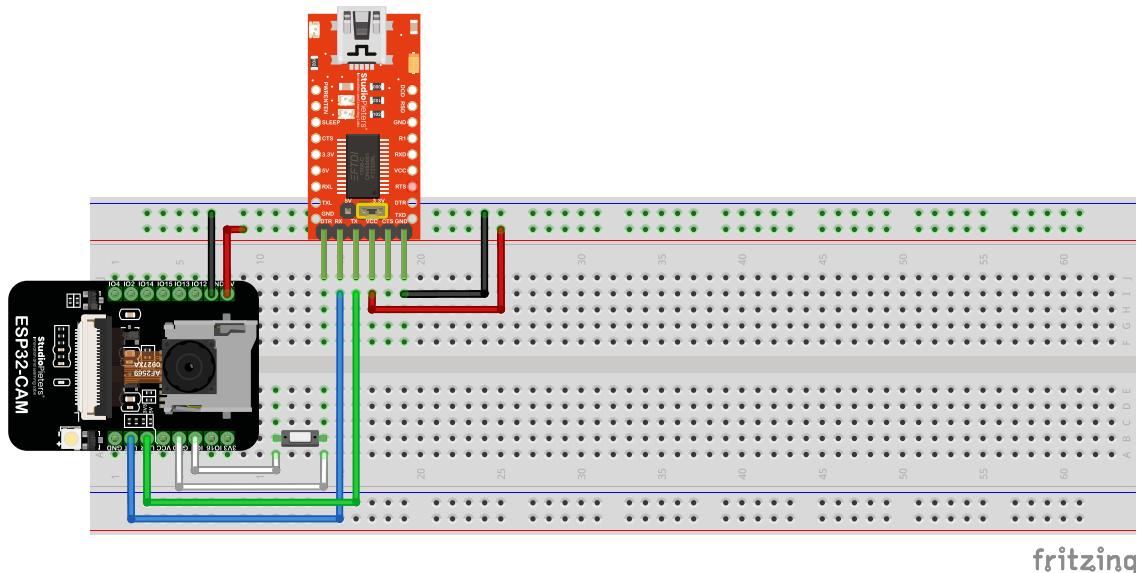
olyan információkat, mint például az adott szobában lévő hőmérséklet és páratartalom szenzor által szolgáltatott információk, és a szobákban lévő eszközök ki- és bekapcsolása.

#### 2.1.4. ESP32-CAM - Wi-Fi-s kamera modul

Mint ahogyan a 2.1.3. szakaszban taglalt NodeMCU, az ESP32-CAM is az IoT és ESP32-es eszközök családjába tartozik, ami egy olyan programozható elektronika ami Wi-Fi-re képes csatlakoztatni.

A rendszerben mint „biztonsági kamera” van alkalmazva és főoldalon szolgáltat elő közvetítést. Szintén oly módon csatlakozik rá a Raspberry Pi-ra, mint a többi mikrokontroller a rendszerben.<sup>4</sup>

Adott a lehetőség ezzel az eszközzel is, hogy akár memóriakártyára mentsünk képeket, amit a kamera felvesz, és van egy saját beépített LED lámpája is. Azonban ezeket a funkciókat a rendszer aktuális verziójában nem alkalmazom, csak és kizárolag elő közvetítésre.



**2.1. ábra.** Az ESP32-CAM bekötése

ESP32-CAM	FTDI Programmer
GND	GND
5V	VCC (5V)
U0R	TX
U0T	RX
GPIO 0	GND

**2.1. táblázat.** ESP32-CAM és az FTDI Programmer bekötése

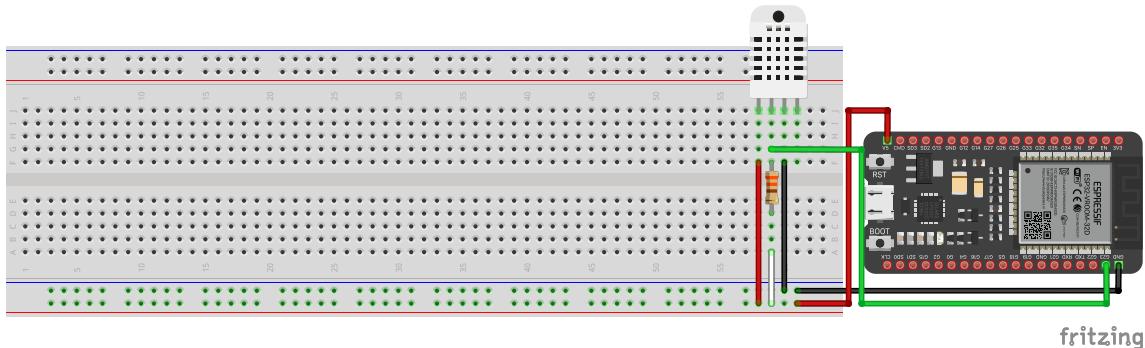
<sup>4</sup> A 3.5 részen ennek a működése részletezve lesz.

**Kapcsoláshoz magyarázat:** Ebben az esetben azért lett így összekapcsolva az ESP32-CAM egy FTDI Programmer-el,<sup>5</sup> hogy a vezérlőkódot fel tudjam rá tölteni. Az adatátvitelt az U0R - TX és az U0T - RX portok összekötése biztosítja. A GPIO 0 - GND kód feltöltéséhez kell, abban az esetben, amikor töltenénk fel a kódot a kamerára, akkor ahhoz meg kell nyomni a gombot, mint ahogyan a 2.1. képen is látható. Az 5V - VCC a tápellátás, a GND - GND a földelés. Ha a programozását elvégeztük, akkor elegendő a tápot biztosítani az ESP32-CAM-nek.

### 2.1.5. DHT22 - hőmérséklet és páratartalom szenzor

Az DHT22 érzékelő egy olyan eszköz, amely méri a levegő hőmérsékletét és a relatív páratartalmát. Az összegyűjtött adatokat feldolgozza és továbbítja. A DHT22 kis mérete, alacsony energiafogyasztása miatt széles körben használható különböző alkalmazásokban.[13]

Ebben a rendszerben úgy van alkalmazva, hogy egy ESP-WROOM-32-höz van kapcsolva, ami 10 másodpercenként elküldi az adatokat a webalkalmazásnak, mely eltárolja az adott pillanatban mért adatokat az alkalmazott szobában, és megjeleníti a főoldalon, mely alapján meg lehet tekinteni az elmúlt 24 órai adatokat egy grafikonon is.<sup>4</sup>



**2.2. ábra.** Az ESP-WROOM-32 bekötése DHT22 hőmérséklet és páratartalom szenzorral

DHT22	ESP-WROOM-32
GND	GND
5V	VCC (5V)
DATA	GPIO 23

**2.2. táblázat.** ESP-WROOM-32 és a DHT22 bekötése

**Kapcsoláshoz magyarázat:** DATA - GPIO 23 azért felelős, hogy az ESP-WROOM-32 elérhesse a DHT22 által mért szenzoros adatokat. Ehhez szükséges egy  $330\Omega$ -os el-

<sup>5</sup> Magyarul: FTDI programozó

lenállást is hozzá kötni, mint ahogyan a 2.2. kapcsolási rajzon is látható. Az 5V - VCC a tápellátás, a GND - GND pedig a földelés.

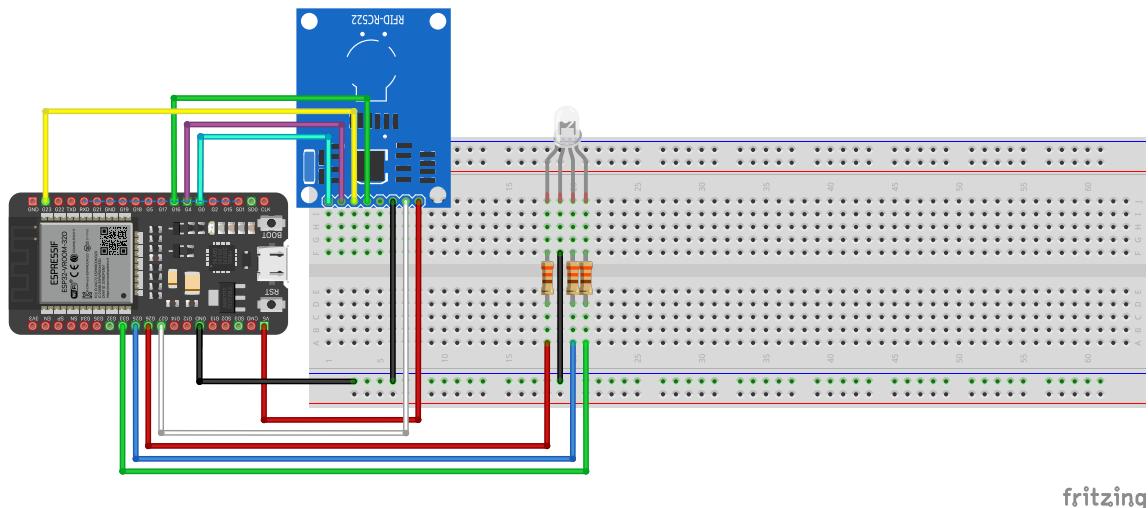
### 2.1.6. Mi is az RFID?

A Radio Frequency Identification<sup>6</sup> (RFID) olyan vezeték nélküli rendszer, amely két összetevőből áll: tag-ekből<sup>7</sup> és olvasókból. Az olvasó egy olyan eszköz, amely olyan antennával rendelkezik, ami rádióhullámokat bocsát ki, és jeleket fogad vissza az RFID-tag-ról. A tag-ek lehetnek passzívak vagy aktívak. A passzív RFID-címeket az olvasó táplálja, és nincs saját áramforrásuk. Az aktív RFID címek akkumulátorral, vagy hálózatról működnek.

Az RFID-címek számos információt tárolhatnak egy azonosítótól több oldalnyi adatig. Az olvasók lehetnek mozgathatóak, így kézben is hordhatók, vagy oszlopra vagy falra is rögzíthetők. Egy olvasó egy szekrény, szoba vagy épület architektúrájába is beépíthető.[14]

### 2.1.7. RFID-RC522 - RFID olvasó

Mint ahogyan a 2.1.6. szakaszban említettem, ez egy RFID olvasó, amely képes RFID tag-eknek az azonosítóját és egyéb adatait beolvasni rádióhullámok alkalmazásával.



**2.3. ábra.** Az ESP-WROOM-32 bekötése, RFID-RC522 olvasóval és egy RGB LED-del

A rendszerben úgy van jelen, mint egy kezdetleges „biztonsági rendszer”, amit például ajtó, vagy szekrények zárására lehet használni. Jelen esetben úgy van alkalmazva, hogy az RFID olvasót működtető ESP-WROOM-32 elküldi a Raspberry Pi számára az

<sup>6</sup> Magyarul: rádiófrekvenciás azonosítás

<sup>7</sup> Magyarul: címkekkel

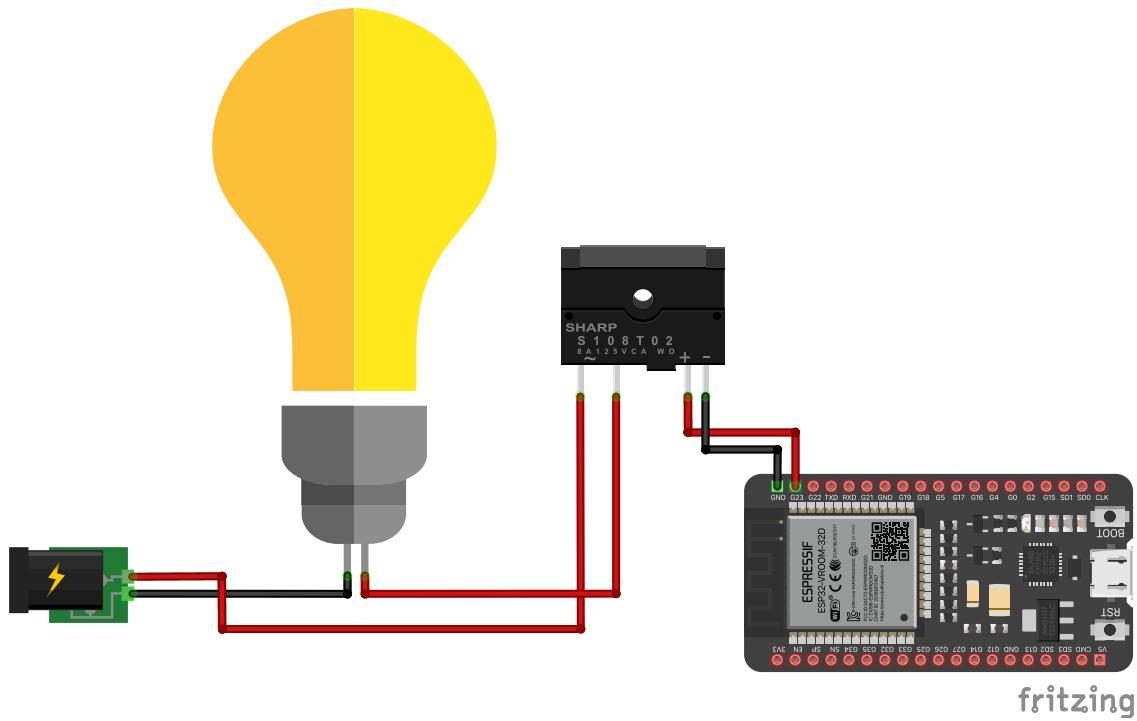
adott tag adatait: hogy, ha ez megegyezik az adatbázisban jelen levő azonosító egyikével, akkor – jelenlegi rendszerben – az RGB LED<sup>8</sup> zölden kezd villogni, ellenkező esetben pedig pirosan fog villogni.<sup>4</sup>

<b>RFID-RC522</b>	<b>ESP-WROOM-32</b>
SDA/SS	GPIO 0
SCK	GPIO 4
MOSI	GPIO 23
MISO	GPIO 16
GND	GND
RST	GPIO 27
3,3 V	5V

### **2.3. táblázat.** ESP-WROOM-32 és RFID-RC522 kapcsolása

**Kapcsoláshoz magyarázat:** MOSI - GPIO 23 a MISO - GPIO 16 az SDA - GPIO 0 és a RST - GPIO 27 a két eszköz közötti kapcsolatért felelős, az SCK - GPIO 4 pedig az órajelet adja meg. A 3,3 V - 5V a tápellátás, a GND - GND pedig a földelés.

### 2.1.8. Szilárdtest relé



**2.4. ábra.** Az ESP-WROOM-32 bekötése szolidtest-relével és egy izzóval

A szilárdtest relé egy elektronikus kapcsolóeszköz. Az elektromechanikus reléhez hasonlóan képes be- vagy kikapcsolni az átfolyó áramot, ha külső vezérlőjelet adnak

<sup>8</sup> Red Green Blue Light Emitting Diode, magyarul Piros Zöld Kék Fényt Kibocsátó Dióda

át a vezérlőn. Az elektromechanikus reléhez képest a szilárdtest relék azonban nem tartalmaznak mozgó alkatrészeket, ezért nem hangosak és hosszabb élettartalmúak.

A félvezetők elektromos és optikai tulajdonságait használják fel a kapcsolás végrehajtására, és teljes leválasztást biztosítanak a vezérlőáramkör és a terhelési áramkör között.[16]

Ebben a rendszerben úgy lett alkalmazva, hogy egy ESP-WROOM-32 és egy izató van csatlakoztatva egy szilárdtest reléhez. Az ESP-WROOM-32 rákapcsolódva a Raspberry Pi-ra képes azt a felhasználó a felületen keresztül ki- vagy bekapcsolni.<sup>4</sup>

Számos más olyan eszköz esetében lehet alkalmazni, amit ki- vagy be szeretnénk kapcsolni, mint például egy ventilátor.

## 2.2. Alkalmazott szoftverek, keretrendszerök, és programozási nyelvek

### 2.2.1. C++ és az Arduino IDE

A Bjarne Stroustrup által 1979-ben kifejlesztett C++ egy általános célú programozási nyelv, ami lényegében a C programozási nyelv továbbfejlesztett változata. A főbb kiengészítések közé tartozott az objektumorientált programozás, és a hiba- és végrehajtás kezelés.[17]

A mikrokontrollerek legtöbbjét C++-al szokták programozni, amihez az Arduino IDE az egyik legnépszerűbb választás, ami számos kiegészítőkkel, és mikrokontrollerek programozásához használatos funkciókkal rendelkezik, legyen ez a kód feltöltése, vagy a soros monitoron megjelenített információk kiírása. Külső könyvtárak is hozzáadhatóak a különböző eszközök támogatása érdekében.

### 2.2.2. HTML és CSS

A HTML<sup>9</sup> egy leírónyelv, amely szövegek leírására és megjelenítésére szolgál. Számos funkciója van, mint a bekezdések, táblázatok, vagy képek beszúrása. Többnyire weboldalak elkészítésénél használják. Mezőket „<>”-en belül határozzuk meg és lezárni ehhez hasonlóan „</>”-el kell. Mindezt annak megfelelően kell párosítani, hogy mit is írunk: legyez ez például egy bekezdés, ami „<p><sup>10</sup> </p>” aminek a két tag közé kell beírni a szöveget.

Mindez kevés arra, hogy a szöveg kinézetét változtassuk, ezért is társítják melléje a CSS-t, azaz Cascading Style Sheet-et<sup>11</sup>, ami HTML vagy XML nyelvek által meghatározott megjelenés formázására szolgál. Ezeket a stílus leírásokat „<style></style>”

<sup>9</sup> Hyper Text Markup Language, magyarul Hiper Szöveg Leíró Nyelv

<sup>10</sup> „p” az angol paragraph (bekezdés) szóra utal

<sup>11</sup> Magyarul: Kaszkádolt Stílus Lap

tag-ek közé szoktuk leírni. Megvan ennek is a sajátos meghatározási módja, hogy mit- és hogyan alkalmazunk.

A webalkalmazásnál máshogyan állítom be a kinézetet, amit a 2.2.7 részen említek, hogy hogyan is működik a Tailwind.

### 2.2.3. PHP

A PHP<sup>12</sup> egy általános felhasználású szkriptnyelv, és fordító, amely szabadon elérhető és széles körben használt webfejlesztés során. A nyelvet elsősorban szerveroldali szkriptek készítésére használják, bár használható parancssori szkriptekre is, és korlátozott mértékben alkalmazásokra.[18]

C programozási nyelv jellegű a szintaxisa, és rendkívül engedékeny programozási nyelv. Amikor PHP-ban akarunk valamit írni, akkor azt .php-re végződő fájlban írjuk, aminek a belső tartalma „<?php”-vel kezdődik, és „?>”-vel végződik. Ezek között írható le a kódunk. Legtöbb esetben HTML-el van társítva.

### 2.2.4. JavaScript

A JavaScript az egy objektum orientált programozási nyelv, ami elsősorban interaktív elemek létrehozására biztosít lehetőséget weboldalakon vagy alkalmazásokon, azonban ez átterjedt sok más környezetre és használati területre is.

A JavaScript megtanulása és futtatása felettébb egyszerű, és rendkívül híres és gyakran használt programozási nyelv.[19]

A webalkalmazásomban is a felületen sok helyen JavaScript-et, azon belül is a jQuery-t használom funkciók kezelésére, legyen az például egy lámpa felkapcsolásának kezdeményezése.<sup>4</sup>

### 2.2.5. jQuery

A jQuery az egy ingyenes és nyílt forráskódú, gyors, kisméretű és funkciókban gazdag JavaScript könyvtár. Sokkal egyszerűbbé teszi az olyan dolgokat, mint a HTML elemek elérése és kezelése, az eseménykezelés, és az animáció egy könnyen használható API-val, amely számos böngészőben működik. A sokoldalúság és a bővíthetőség kombinációjával a jQuery emberek millióinak JavaScript-írási módját változtatta meg. [20]

### 2.2.6. Chart.js

Chart.js egy népszerű, nyílt forráskódú JavaScript könyvtár, amely lehetővé teszi a fejlesztők számára, hogy könnyen létrehozzanak reszponzív és testre szabható diagramokat vagy grafikonokat a weben. Többféle diagram típust támogat.

---

<sup>12</sup> PHP: Hypertext Preprocessor, magyarul: PHP: hiperszöveg előfeldolgozó

A könyvtár a HTML5-re épül, amely lehetővé teszi a gördülékeny animációkat és az interaktivitást. A Chart.js egy egyszerű API-t kínál, amely lehetővé teszi a fejlesztők számára a diagramok egyszerű beállítását és stílusának testreszabását. Több beépített testreszabási lehetőséget is biztosít, mint például a diagramszínek, címkék, és jelmagyarázatok.[21]

Chart.js nagy előnye, hogy a hivatalos oldalukon részletesen van dokumentálva, hogy hogyan kell telepíteni, és mit hogyan is lehet használni, példákon keresztül.

A Chart.js a egy adott szobában lévő hőmérséklet és páratartalom szenzor adatok grafikonon való megjelenítésére használja a weboldal.

## 2.2.7. Tailwind CSS

A Tailwind CSS egy népszerű CSS keretrendszer, amely előre definiált osztályokat biztosít a webalkalmazások stílusának egyszerűsítéséhez és optimalizálásához. Ezek olyan osztályok formájában vannak meghatározva, amik leírják a kívánt vizuális hatást, a helyett, hogy egyedi CSS stílusokat írnának az egyes elemekre.<sup>13</sup>

Például egy bekezdéshez így tudjuk azt megadni azt, hogy a szövege jobbra zárt és félkövér legyen: *< p class="text-right font-bold" > ez egy bekezdés </ p >*

A Tailwind stílus alkalmazása lehetővé teszi a fejlesztők számára a gyors, összehangolt és egységes felület létrehozását, miközben javítja a kinézet gyors alakíthatóságát és skálázhatóságát, azonban ezzel rontva a HTML fájl átláthatóságát.

Széles választékban kínál előre elkészített osztályokat a közös UI<sup>15</sup> komponensekhez egy konfigurációs fájlban, mint például gombok, űrlapok, grid rendszerek és tipográfiák, valamint a bonyolultabb elrendezésekhez és pozicionálási feladatokhoz is. Ezzel a konfigurációs fájllal biztosítja a fejlesztőknek, hogy a beépített osztályokat testre szabhassák és, hogy saját egyedi osztályokat is létre hozhassanak.

Fejlesztést pedig hivatalos dokumentációval segítik, mivel mindenre van nekik példa és leírás.[22]

## 2.2.8. dbDiagram.io

A dbdiagram.io egy webes eszköz, amely lehetővé teszi a felhasználók számára az adatbázis tervezés létrehozását egy letisztult vizuális felületen keresztül.

Az eszköz támogatja különböző adatbázis rendszereket, mint például a MySQL. Hasznos funkciókat kínál, mint például az automatikus SQL kódgenerálás, és a diagramok importálása vagy exportálása különböző formátumokba.[23]

<sup>13</sup> Mindezt build-elés, azaz felépítés idejében átalakítja CSS kódját, és a felesleges elemeket kihagyja.

<sup>14</sup> text-right, azaz szöveg-jobb, font-bold, azaz szöveg-félkövér

<sup>15</sup> User Interface: azaz Felhasználói Felület

Szakdolgozatom során ezt használtam arra, hogy elkészítsem az adatbázis tervét, és majd a 3.2 szakaszon lesz róla egy mellékelt kép is ami a 3.1. ábrán látható, hogy hogyan is lett minden kialakítva.

### 2.2.9. Font Awesome

A Font Awesome egy népszerű nyílt forráskódú ikon könyvtár, amely skálázható vektor ikonokat<sup>16</sup>kínál, amelyek testre szabhatóak az oldalukon, és CSS alkalmazásával is. A könyvtár több ezer ikont tartalmaz, amik széles körű kategóriákat fednek le, például márka vagy közösségi média ikonokat. Lehetőség van fizetős és ingyenes ikonok használatára is.

A Font Awesome egy rugalmas és könnyen használható megoldást kínál az ikonok hozzáadásához weboldalakhoz vagy alkalmazásokhoz, lehetővé téve a fejlesztőknek, hogy feldobják projektjeik kinézetét. A könyvtár folyamatosan frissül új ikonokkal és funkciókkal.[24]

**Az általam használt ikonok, a rendszerben:**

-  „<i class="fa-solid fa-arrow-left"></i>”
-  „<i class="fa-solid fa-clock-rotate-left"></i>”
-  „<i class="fa-solid fa-plus"></i>”
-  „<i class="fa-regular fa-pen-to-square"></i>”
-  „<i class="fa-solid fa-trash-can"></i>”
-  „<i class="fa-solid fa-rotate-right"></i>”
-  „<i class="fa-solid fa-ban"></i>”
-  „<i class="fa-solid fa-gear"></i>”
-  „<i class="fa-solid fa-key"></i>”
-  „<i class="fa-solid fa-id-card-clip"></i>”

### 2.2.10. MySQL

A MySQL egy nyílt forráskódú adatbázis-kezelő rendszer, amelyet széles körben használnak webalkalmazásokhoz. Lehetővé teszi a felhasználók számára, hogy relációs adatbázisokat hozzanak létre, kezeljenek és tartsanak karban. Több platformon is működik és sok programozási nyelvvel is kompatibilis.

---

<sup>16</sup>Ez azt jelenti, hogy egy algoritmus segítségével minden ugyanolyan minőségű lesz a kép, bár mekkorára is nagyítjuk.

A XAMPP egy webszerver szoftver, amely tartalmazza a MySQL adatbázis-kezelő rendszert, és a PHP-t is, ami telepíthető a helyi számítógépre webfejlesztéshez és teszteléshez, mint ahogyan én is alkalmaztam lokális fejlesztés során.

### 2.2.11. PlantUML

A PlantUML egy ingyenes és nyílt forráskódú eszköz, amely egyszerű szövegszintaxist használ a különböző formátumokban létrehozott UML<sup>17</sup> diagramok készítéséhez. Többféle diagramtípus támogatása mellett használható szoftverfejlesztéshez és rendszertervezéshez.

Ennek segítségével rajzoltam le azt, hogy NodeMCU és a Raspberry Pi hogyan is kommunikál a rendszeren belül.<sup>4</sup>

### 2.2.12. Fritzing

A Fritzing egy nyílt forráskódú szoftver, aminek a segítségével elektronikai eszközök kapcsolási és sematikus rajzait, de akár szimulációt is lehet benne készíteni.

Ez egy olyan program, aminek a letöltéséért cserébe fizetni kell, ezzel is támogatva a program fejlesztését és karbantartását. Letölthetők hozzá külső könyvtárak, ezzel is növelte a lehetőségeket ilyen rajzok létrehozásában.

Ennek használatával hoztam létre a 2.1 .szakaszban a kapcsolási rajzokat.

**Fejlesztés során használt külső könyvtárak:**

- A 2.1. ábrán az ESP-WROOM-32, az ESP32-CAM, és az FTDI Adapter[25]
- A 2.2. ábrán az ESP-WROOM-32 és a DHT22 szenzor [25]
- A 2.3. ábrán az ESP-WROOM-32[25] és az RFID-RC522[28]
- A 2.4. ábrán az ESP-WROOM-32[25] az izzó[26] és a szilárdtest relé[27]

### 2.2.13. Laravel

A Laravel egy ingyenes és nyílt forrású PHP webalkalmazás-keretrendszer, amelyet skálázható és nagy méretű webalkalmazások építésére használnak. Egy elegáns szintaxist, erős eszközöket és moduláris csomagolási rendszert biztosít, hogy a fejlesztők tiszta és karbantartható kódot írhassanak. A Laravel követi az MVC<sup>18</sup> architekturális mintát, és beépített funkciókkal rendelkezik, mint például az azonosítás, a route-olás és gyorsítótár.[29] A fejlesztők dolgát azzal könnyítik meg leginkább, hogy hivatalos dokumentációja van, ahol minden kis részletre adnak példát és leírást.[30]

---

<sup>17</sup> Unified Modeling Language, magyarul Egységesített Modellező Nyelv

<sup>18</sup> Model-View-Controller: magyarul Modell-Nézet-Vezérlő

### **MVC Keretrendszer:**

Az MVC egy szoftvertervezési minta, amelyet a felhasználói felületek fejlesztéséhez használnak.

### **Az alkalmazást három összekapcsolt komponensre bontja:**

A **modell**, amely az adatokat és a logikát képviseli, a **nézet**, amely megjeleníti az adatokat a felhasználónak, és a **vezérlő**, amely kezeli a felhasználói bemenetet és kapcsolatban áll a modellel és a nézzel. Az MVC keretrendszer, mint például a Laravel, strukturált megközelítést biztosítanak a webalkalmazások fejlesztéséhez, és segítik a fejlesztőket a moduláris és karbantartható kód írásában.

Éppen ezért is esett a választásom a projekt tervezési fázisában a Laravel keretrendszerre. Ez a webalkalmazás még Laravel 9-nél indult el.

## 3. fejezet

# A rendszer felépítése és működése

### 3.1. Raspberry Pi 4B alkalmazása

Legelőször, amikor a Raspberry Pi-t elkezdtem használni arra, hogy szolgáltassa a weboldalt, akkor úgy volt, hogy egy már megadott Wi-Fi-re csatlakozott rá. Ez nem a legjobb megoldás volt, mivel szükséges volt beleégetni a Raspberry config fájljába, hogy mi annak a Wi-Finek az SSID-je és a jelszava.

A Raspberry-t SSH<sup>1</sup> használatával kapcsolódok fel rá, amit egy helyi és egy távoli számítógép közötti biztonságos csatorna kiépítésére fejlesztettek ki. A projekt fel van töltve GitHub-ra[44], ami egy verziókövető rendszer. Innen klónoztam le a Raspberry-re a projektet, és így tartom napra készen ezen.

A projektet a `/var/www/html/` mappán belülre kellett helyezni és apache2 könyvtár segítségével tudtam elérni a webalkalmazást host-olni, azaz szolgáltatni, és elérni is. MySQL-el használom az adatbázist a Raspberry-n is. Miután már többet foglalkoztam a projekttel, akkor rájöttem, hogy a Wi-Fi-re való csatlakozás nem a legoptimálisabb, ezért jött az az ötlet, hogy maga a Raspberry vegye fel a Wi-Fi router szerepét. Ez úgy valósul meg, hogy először is Ethernet kábellel biztosítom a Raspberry számára az internetet. Utána pedig a Raspberry Pi hivatalos oldalán való leírás[45] segítségével teljesen be tudtam állítani Wi-Fi routerként.

**Fontosabb tudni valók ezzel:** Az interface statikus felülete `192.168.200.1`-re lett állítva, ez az első IP cím, ezért sem lehet az ESP-knek 1-re végződő IP véget adni. DHCP<sup>2</sup> szerver 150-es IP végtől dinamikusan állítja az IP címeket, ezért nem lehet az ESP-nek 149-től nagyobb IP véget adni, hogy statikusan és könnyen elérhetőek legyenek az eszközök.

---

<sup>1</sup> Secure Shell

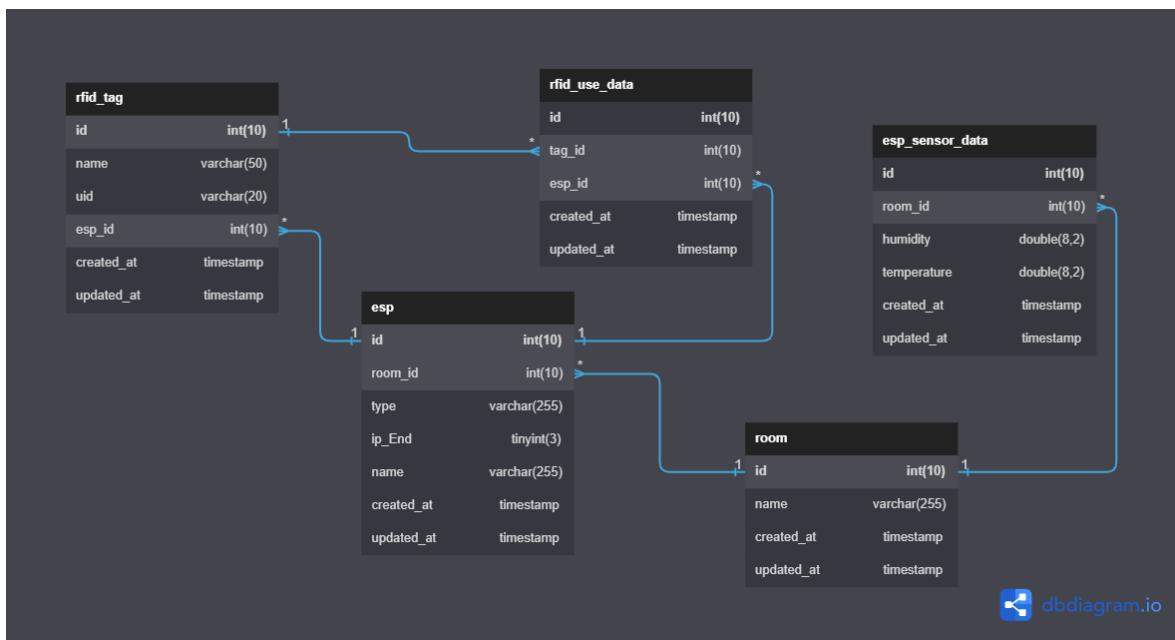
<sup>2</sup> Dynamic Host Configuration Protocol, azaz Dinamikus Gazdagép Konfigurációs Protokoll

## 3.2. Adatbázis felépítése

Miután meg volt ez ötlet és átestem pár tervezési fázison, hogy hogyan is álljon össze a rendszer, jöhetett az egyik legfontosabb feladat: hogy hogyan is álljon össze mindenek az adatbázisa. Ezen belül is az, hogy milyen kapcsolatok és táblák szerepeljenek mindebben.

Mindegyik táblában van plusz két – a *Laravel által automatikus létrehozott* – mező: a „*created\_at*” azaz létrehozás dátuma, és az „*updated\_at*”, ami az utolsó módosítás dátuma.

A táblázatban az idegen kulcsok azért is lettek így kialakítva, hogy a későbbiekben könnyebben lehessen kezelni és azért is, hogy ha például egy szobát töröl a felhasználó, amihez tartoznak különböző eszközök, akkor azokat is törölje vele együtt, ezt nevezük kaszkádolt törlésnek.



**3.1. ábra.** Az rendszer adatbázisa, amit dbDiagram.io segítségével készítettem el

### room, azaz szoba

Mivel egy otthon szobákból áll, ezért létre hoztam ehhez egy táblát, amikhez tudunk csatolni más adattáblát is. Szobáknak van egy olyan mezője, amit a felhasználó adhat meg, az pedig a „*name*”, azaz a szoba neve. Egy szobához tartozhat bármennyi ESP-WROOM-32<sup>3</sup>, azaz eszköz, vagy szobához tartozó szenzoros adat.

<sup>3</sup>Későbbiekben: ESP

## **esp\_sensor\_data, azaz ESP szenzoros adatok**

Amikor a szobához hozzáadunk egy olyan ESP-t, ami szenzoros adatokat továbbít, akkor az ebben a táblában lesz letárolva. Ez olyan mezőket tartalmaz, mint a „room\_id”, azaz az adathoz hozzá tartozó szoba id-je, „humidity”, azaz relatív páratartalom, és „temperature”, azaz hőmérséklet.

## **esp**

A rendszerben ESP-ket használunk arra, hogy különböző eszközök működjenek, ezért is lett felvéve külön táblaként. Egy ESP-nek olyan mezői vannak, mint a „room\_id”, azaz az a szoba id-je, amelyikhez tartozik, „type”, azaz a szenzornak a típusa, ami az alábbi lehet a jelenlegi rendszerben: szenzor, kapcsoló, RFID olvasó, vagy kamera. „ip\_End” az az IP cím utolsó része<sup>4</sup>, és a „name”, azaz az ESP neve.

## **rfid\_tag, azaz RFID címke**

Mivel egy ESP használható RFID kártyaolvasóként, ezért nyilván kell tartani azt is, hogy mely címkék lettek eddig felvéve a rendszerbe. Egy RFID címke az alábbi mezőket tartalmazza: „name”, azaz egy név, „uid”, ami az RFID címének az az adata, amit az olvasó kap, és egy „esp\_id”, azaz annak az olvasónak az azonosítója, amihez tartozik egy címke.

## **esp\_use\_data, azaz RFID címkéknek az előzmény adatai**

A rendszerben úgy lettek alkalmazva az RFID-k, hogy amikor benne van a rendszerben egy adott címke és az használatra kerül, akkor annak a időpontja rögzítésre kerül. Ez az alábbi mezőket tartalmazza: „tag\_id”, azaz a használt RFID címke azonosítója, „esp\_id”, ami a annak az RFID olvasónak az azonosítója, ami beolvasta a címkét.

### **3.3. Laravel működése**

#### **3.3.1. Modellek és migrációk**

Mivel a Laravel az adatbázis adatokat Model<sup>5</sup>-ekkel és Migration<sup>6</sup>-ök segítségével dolgozik, ezért létre kellett hozni ezeket. Model az, amit a vezérlőben alkalmazok, a Migration pedig az adatbázis létrehozásában segít.[31]

---

<sup>4</sup>Ez majd a későbbiekben több értelmet fog nyerni a 3.5. szakasz során

<sup>5</sup>Magyarul: modell

<sup>6</sup>Magyarul: migrációk

Ahhoz hogy a Laravel backend-jében könnyen tudjam kezeln az adatbázisan lévő adatokat, azért létre kell hozni egy modellt, és a hozzá tartozó migrációt amit a terminálban az adott parancssal lehet létrehozni: „*php artisan make:model ModelName --migration*”, ezzel létre jön a migráció és a modell saját fájlja.

### 3.1. Programkód. Esp.php kódja - ami az ESP modellje

```

1 <?php
2 namespace App\Models;
3 use Illuminate\Database\Eloquent\Model;
4
5 class Esp extends Model
6 {
7     // $table azt adja meg hogy melyik adatbázis táblának felel
8     // meg
9     protected $table = 'esp';
10    // $fillable az azt adja meg, hogy melyek azok a mezők
11    // amiket módosíthatunk
12    protected $fillable = array('type', 'ip_End', 'name');
13    // $timestamps pedig egy igaz-hamis változó, ami azt
14    // határozza meg, hogy generáljon-e 'created_at' és '
15    // 'updated_at' mezőket
16    public $timestamps = true;
17 }
```

### 3.2. Programkód. Az ESP migrációs kódja

```

1 <?php
2 use Illuminate\Database\Migrations\Migration;
3 use Illuminate\Database\Schema\Blueprint;
4 use Illuminate\Support\Facades\Schema;
5
6 class CreateEspTable extends Migration {
7     // az up metódus akkor fut le amikor új esp táblát
8     // generál
9     public function up()
10    {
11        // itt megadjuk azt hogy az esp táblát
12        // szeretnénk beállítani
13        Schema::create('esp', function(Blueprint $table)
14        {
15            // automatikusan növekvő azonosító
16            $table->increments('id');
17            // előjel nélküli szoba azonosító
18            $table->integer('room_id')->unsigned()
19            // ;
20            // esp típusa ami szöveges
21            $table->string('type');
22        });
23    }
24
25    public function down()
26    {
27        Schema::dropIfExists('esp');
28    }
29 }
```

```

18         // tinyinteger az 0–255 értékek között
19         // → az ip végződésre
20         $table -> tinyInteger('ip_End') ->
21             → unsigned();
22         // esp neve ami szöveges
23         $table -> string('name');
24         // 'created_at' es 'updated_at' mezők
25         $table -> timestamps();
26     });
27     // a down metódus akkor fut le, hogy ha töröljük az esp
28     // → táblát
29     public function down()
30     {
31         Schema::dropIfExists('esp');
32     }
33 }

```

Hogy ha lefutattjuk ezek után azt a terminálban, hogy „*php artisan migrate:fresh*”, akkor ez alapján a .env fájlon belül megadott adatbázisban létrehozza a migrációkban leírt táblákat.

### 3.3.2. Route-olás

A Route<sup>7</sup> azt adja meg, hogy milyen útvonalon mit tegyen a vezérlő. Erre egy példát úgy tudnék mondani, hogy ha a böngészőbe beírjuk a címét egy oldalnak utána /-jellel hozzáírunk valamit, akkor az azt az útvonalat követné.

Négy féle HTTP kérés típus van az ilyen útvonalaknál: **get**, **post**, **put**, és **delete**. A „**get**” általában akkor használatos, amikor szeretnénk egy (vagy több) rekordot visszakapni, a „**post**”-ot általában akkor szoktuk használni, amikor valamilyen rekordot szeretnénk létrehozni, a „**put**”-ot általában egy létező rekord módosítás kérésnél szokás<sup>8</sup> és a „**delete**”-et amikor rekord törlést szeretnénk kérni.

Például a projektben a „*http://192.168.200.1/settings*<sup>9</sup>” megmondja a vezérlőnek, hogy ilyenkor irányítsa át a felhasználót a beállítások oldalra.

A „*web.php*”-ban a beállítás oldalra való irányítás a 3.3. kód soron látható.

```
Route::get('/settings', [EspController::class, 'index'])->name('Settings');
```

**3.3. Programkód.** Hőmérésklet és páratartalom szenzor oldal útvonala  
A 3.3. programkódban azt láthatjuk, hogy egy „**get**” HTTP kérést kapunk a „*/settings*” útvonalon, és ezzel az „**EspController**” vezérlőnek az „**index**” függvényét hívjuk meg. A „->**name()**”-en pedig meg lett adva egy név, amire hivatkozva tudunk

<sup>7</sup> Magyarul: útvonal

<sup>8</sup> De ez helyettesíthető egyszerűen „**post**”-tal is

<sup>9</sup> Magyarul beállítások

egyszerűbben akár hosszabb útvonalakat is meghívni.

Hogy ha a vezérlőnek változót szeretnénk átadni, akkor azt pedig úgy tudjuk megadni, mint ahogyan a 3.4. kódsonon is látható.

```
Route::get('/chart/{room}', [EspSensorController::class, 'index'])->name('Sensor Data Chart');
```

### 3.4. Programkód. Hőmérésklet és páratartalom szenzor oldal útvonala

A 3.4. kódsonon is látszik, hogy úgy kell megadni egy változót, amit a vezérlőnek adunk át, hogy „`{}`”-k közé beírjuk azt a változó nevet. Ez az útvonal azért felelős, hogy a hőmérésklet és páratartalom szenzor által mért adatoknak külön oldalát nyissa meg, a „`room`” pedig azt adja meg, hogy mi a szoba azonosítója. A Modell nevét át lehet adni útvonal paraméterként, hogyha azonosítót – *ezáltal egy modellt* – szeretnénk kinyerni.[32] A 3.5. kódsonon pedig az látható, hogy a vezérlőben hogyan érjük el az ez által megadott modellt.

```
public function index(Room $room){...}
```

### 3.5. Programkód. „*EspSensorController*” index metódusa

#### 3.3.3. Controller

A Controller, más néven vezérlő felelős a háttérben lévő folyamatok lebonyolításáért, és kezeli a felhasználói bemenetet és kapcsolatban áll a modellel és nézettel.

Érdemes minden osztálynak külön vezérlőt készíteni, mert ezzel átláthatóbb lesz a kód. Ha már létezik egy modellünk, és ehhez szeretnénk egy vezérlőt létrehozni, akkor azt az alábbi terminálba beírt utasítással tudjuk: „`php artisan make:controller ControllerNeve --model=ModelNeve`”[33]

A vezérlőkben ezek a metódusok szoktak létrejönni, amikor generáljuk.

- `index()`: rekordok kilistázására
- `create()` rekord létrehozás form megjelenítésére
- `store(Request $request)` új rekord eltárolására
- `show(ModelNev $modelnev)` megadott rekord megjelenítésére
- `edit(ModelNev $model)` megadott rekordnak a módosító form megjelenítésére
- `update(Request $request, ModelNev $modelnev)` a megadott rekord módosítása az adatbázisban
- `destroy(ModelNev $modelnev)` adott rekord törlése az adatbázisból

Ezeken kívül más metódusokat is szabadon létrehozhatunk. Én például ezt tettem az eszköz kapcsoláshoz, az „`EspController`”-en belül a „`Toggle($esp,$status)`” metódussal, amit majd a 3.5. szakaszon kifejtek.

A 3.6. programkódban pedig egy példa látható, hogy hogyan is adhatunk vissza egy adott rekordot egy adott nézettel.

```
public function edit(Room $room)
{
    return view('room.modify', ['room'=>$room]);
}
```

### 3.6. Programkód. „RoomController” edit metódusa

A 3.6. programkódban az látható, hogy azt a nézetet adj a vissza, ami egy szoba módosítására való, egy adott szoba rekorddal. Amikor visszaadunk egy nézetet, akkor lehetőség van arra, hogy ne csak egy adott rekordot adjunk vissza, hanem több rekordot, vagy rekord listákat is. Arra pedig itt egy példa kód.

```
public function index()
{
    $rooms = Room::all();
    $esps = Esp::all();

    return view('index',
        [
            'rooms' => $rooms,
            'esps' => $esps
        ]);
}
```

### 3.7. Programkód. „RoomController” index metódusa

Ez a „RoomController”-nek az *index()* metódusa, ami azért felelős, hogy a főoldalon meg lehessen jeleníteni az összes szobát és eszközt. Szintén a „Room::all();” az összes szoba rekordot az adatbázisban egy listába gyűjti össze, ami a Laravel Eloquent[34] egyik tulajdonsága közé tartozik, ami egy ORM<sup>10</sup>. Ez könnyebbé teszi az adatbázisban elérhető rekordok elérését Model-ekkel és rajtuk végrehajtható műveletekkel.

#### 3.3.4. Blade

A Laravelnek az egyik sajátossága a „nezetNeve.blade.php” típusú fájloknak elnevezett nézetek, amik lehetővé teszik azt, hogy ilyen blade utasításokat tudjunk végrehajtani. „{{}}” – ilyen dupla kapcsos zárójelek között tudunk például a nézetben megjeleníteni egy rekordot.

Másik ilyen sajátossága a PHP utasítások megadása @-jel leírásával. Például egy

---

<sup>10</sup> Object-Relational Mapper, azaz Objektum-relációs leképező

„ha” kérdés így néz ki Laravel blade utasítással: „@if() @endif”.[35]

Például a 3.3.3. programkódban, ahol egy szobát adunk vissza a módosítás nézetbe, és annak szobának a nevét így érhetjük el: „{{\$room->name}}”

## 3.4. A kezelői felület bemutatása és működése

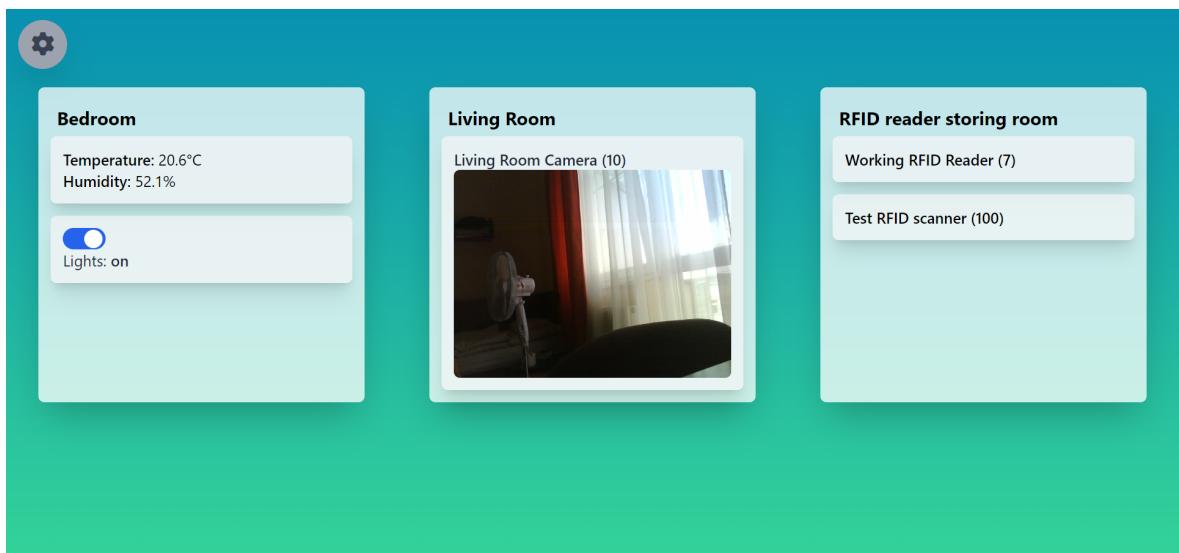
Az alábbi szakaszokon a kezelői felület megjelenését, és működését fogom taglalni. Igyekeztem letisztult design kialakítására. Az összes lap úgy van megoldva, hogy a lehető legtöbb képernyőfajtán megfelelően nézzen ki, azaz mindegyik oldal reszponzív. Mindez a Tailwind CSS reszponzivitási eszközeit használva.[36]

### 3.4.1. A főoldal

A főoldal akkor jelenik meg, hogy ha a Raspberry Pi által szolgáltatott Wi-Fi-re felcsatlakozunk, és hogy ha böngészőnek a kereső sávjába azt írjuk, hogy: „**192.168.200.1**”.

Hogy ha nincs még se szoba, se ezen belül eszköz az adatbázisba felvéve, akkor az a szöveg fogad minket, hogy „*There are no rooms added to the database!*”, ami magyarul azt jelenti, hogy „*Nincsen szoba az adatbázisba felvérve!*”.

Hogy ha már van szoba, és ehhez eszközök felvérve az adatbázisba, akkor a 3.2. képhez hasonlóan néz majd ki a főoldal.



3.2. ábra. A főoldal

Az adatok úgy jelennek meg, hogy a „*RoomController*” „*index()*” metódusából visszakapja a nézet a szobákat és az eszközöket külön listában. A Laravel Blade-nek van egy olyan utasítása, hogy „*@forelse(\$rekordok as \$rekord) @empty @endforelse*”. Ez a hármas egy olyan módosított „*@foreach*”, ami rendelkezik egy olyan utasítás résszel,

(„@empty”) hogy mi a teendő, hogy ha a végig iterálandó lista üres. Ezért is van, az hogy figyelmeztető felirat jelenik meg, hogy ha nincs szoba az adatbázisban.

Ezen az iteráláson belül van még egy ilyen iterálás, ami az eszközökön megy végig, amiben pedig egy olyan „*if()*”, azaz „ha” utasítás van, ami leelenőrzi, hogy az adott eszköz az adott szobához tartozik-e és azon belül azt is, hogy milyen az eszköz típusa. Ezért van az, hogy például egy hőmérséklet és páratartalom adatokat megjelenítő panel máshogyan néz ki, mint egy eszköz kapcsolásra használatos panel.

#### **Toggle, azaz kapcsoló panel:**

Mindegyik kapcsoló[37] típusú panelhez tartozik egy „*onClick()*” metódus, ami azt nézi, hogy rá lett-e kattintva a gombra. Melléje még dinamikusan párosul egy jQuery kód, ami kiírja az oldal betöltésekor<sup>11</sup>, hogy csatlakozni próbál az adott ESP eszközhöz.<sup>12</sup> Ha nem sikerül választ kapni az adott ESP eszköztől, akkor átállítja a kapcsoló feliratát arra, hogy jelenleg nem elérhető, és a kattinthatósági funkciót is elveszi a gombról.

#### **Hőmérséklet és páratartalom panel:**

Mindegyik szenzor panelhez tartozik két dinamikusan hozzárendelt metódus. Az egyik legelőször is lekéri a szobához tartozó legutóbbi adatot az adatbázisból, majd a második azt végzi el, hogy 10 másodpercenként lekérje a legújabb szenzor adatot. Hogy ha sikerül ez, akkor azokat az adatokat megjeleníti, mint ahogyan azt a 3.2. képen is láthatjuk. Hogy ha nincs szenzor adat, akkor kiírja, hogy nincs jelenleg nem elérhető adat az adott szenzor által.

Hogy ha rákattintunk a szenzor panelre, akkor átvisz minket egy másik oldalra, ahol az elmúlt 24 órában rögzített óránkénti átlagos hőmérséklet és páratartalom adatokat nézhetjük meg.

**Kamera panel:** A főoldalon előben lehet látni az adott kamerának a közvetítését. Ehhez a panelhez több dolog is társul: vagy egy „*onLoad()*” metódus, ami azt nézi, hogy a kép betöltött-e, ami annyit tesz, hogy a kapcsolódási kísérlet feliratot átállítja a szenzor nevére és az IP végződésére. Amikor betöltődik az oldal<sup>11</sup>, akkor a panelra kiíródik, hogy kapcsolódni próbál az adott kamerához. Ezt egy olyan „*EventListener*”, azaz esemény hallgató tartozik, ami azt nézi, hogy a kapcsolódási kísérlet sikertelen volt-e. Ilyenkor a felirat visszaállítódik arra, hogy kapcsolódni próbál, és meg is ismétli a kapcsolódást. Ezek után beállítódik az, hogy ez a kapcsolódási kísérlet 15 másodpercenként megismétlődjön.

Hogy ha rákattintunk a kamera képére, akkor átvisz minket egy másik oldalra, ahol nagyban láthatjuk annak az egy kamerának a közvetítését. Ennek a jQuery logikája ugyanaz, mint a főoldalon.

**Átjutás a beállításokra:** Az oldalon, hogy, ha a bal felső sarokban rákattintunk a fogaskerék ikonra, akkor átjutunk a beállítások oldalra.

---

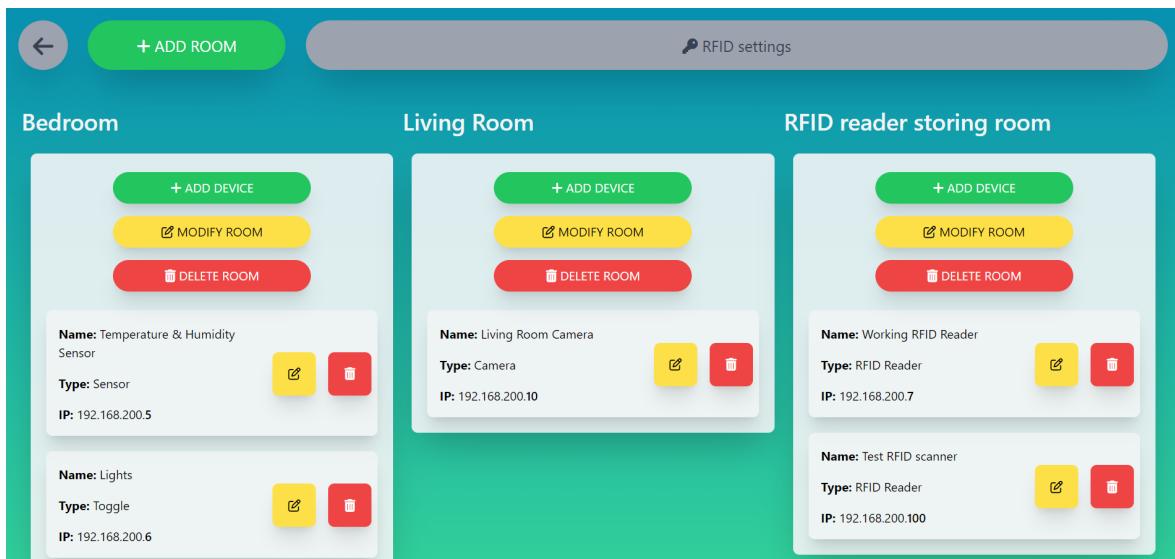
<sup>11</sup> Amit a „*\$(document).ready(function(){...})*” mond meg, hogy betöltött-e az oldal

<sup>12</sup> Ez később majd a 3.5.2 részen lesz elmagyarázva.

### 3.4.2. Beállítások

A beállítások oldal a főoldalon már alkalmazott módon listázza ki a szobákat és a hozzá tartozó eszközöket: egymásba ágazott „*@forelse*” utasítás-sal és egy „*@if*”, ami leellenőrzi, hogy az adott eszköz az adott szobához tartozik-e. Ahogyan a 3.3. képen is látható.

Tudni kell azt, hogy szobát kell hozzáadni az adatbázisba, hogy eszközt tudjunk felvenni hozzá. Ennek az az egyetlen előfeltétele, hogy két szoba ugyanazzal a névvel nem lehet létre hozni. Ha szobát akarunk hozzáadni a rendszerhez, akkor az oldal tetején található „Add Room”<sup>13</sup> gombra kell kattintani, ahol megadhatjuk, hogy a szobának mi legyen a neve.<sup>14</sup>



3.3. ábra. Beállítások oldal

Alapvetően minden szobához tartozik három gomb: „*Add Device*”, azaz Eszköz Hozzáadása, „*Modify Room*”, azaz Szoba Módosítása, és „*Delete Room*”, azaz Szoba Törlése.

Legelőször is, amikor hozzá szeretnénk adni egy új eszközt egy adott szobához, akkor a 3.4. képhez hasonló oldal fog minket fogadni.<sup>15</sup>

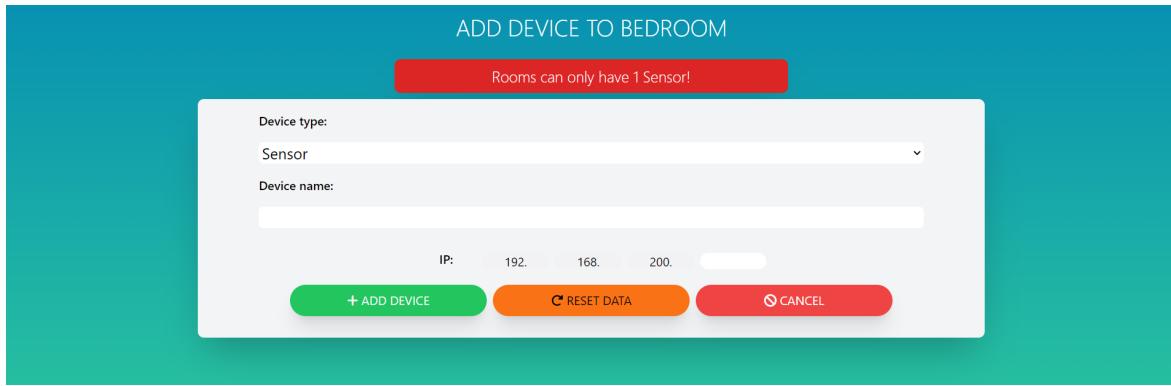
Itt megadhatjuk az eszköz típusát, – amit a 3.2. szakaszban leírtam, hogy milyen típusú eszközöket adhatunk a rendszerbe, – egy tetszőleges nevet, és az ESP szenzornak az IP-jének az utolsó tagját. Hogy ha törölni szeretnénk az oldal mezőnek eddig bevitt adatait, akkor a „*Reset Data*”<sup>16</sup> gombra nyomhatunk, és akkor újból üres lesz

<sup>13</sup> Azaz Szoba Hozzáadása

<sup>14</sup> Természetesen a vezérlőben le van kezelve, hogy nem lehet üresen hagyni az adott mezőt, vagy hogy már létezik egy ilyen szoba az adatbázisban. Mindez le van kezelve a szoba módosításnál is.

<sup>15</sup> A piros keretben lévő hibaüzenetet kivéve: – ez csak azért van így megjelenítve, hogy bemutassam, hogy ezt így jelzi az oldal, hogy ha egy oldalhoz több mint egy szenzort szeretnénk hozzáadni.

<sup>16</sup> Azaz Adatok Visszaállítása



**3.4. ábra.** Eszköz hozzáadása oldal, ahol még egy szenzort akarunk rakni egy szobába

minden mező. Hogy ha meggondoltuk magunkat, és mégsem akarunk egy új eszközt hozzáadni a szobához, akkor a „Cancel”, azaz Mégse gomb-ra kattintva visszakerülünk a beállítások panelre. Hogy ha minden szükséges adatot beírtunk a mezőkbe, akkor az „Add Device”, azaz Eszköz hozzáadása gombra kattintva megkezdődik az eszköz hozzáadásának kísérlete az adatbázisba. Mindezt a vezérlőben leellenőrzöm, hogy ennek megfelelően viselkedik az oldal. Ha hiba van, akkor azt megjeleníti, hogy ha pedig sikerül a hozzáadás, akkor pedig visszairányít a beállítások panelre és értesít az oldal, hogy sikeres volt az eszköz hozzáadása az adott szobába.

A határok ezekre az adatokra kliens és vezérlő oldalon is le vannak természetesen kezelve.

#### Ezek az esetek az alábbiak:

- minden mező megadása kötelező.
- Egy szobába csak egy szenzort lehet felvenni.<sup>17</sup>
- Az eszköz neve maximum 50 karakter lehet.
- Az ESP IP végződése 2 és 149 között kell hogy legyen.

Minden beállítások oldalon elérhető funkció véghezvitelé után értesíti a felhasználót egy feliraton, hogy mit történt: legyen az például egy új szoba hozzáadásának, módosításának, vagy törlésének a sikeressége.

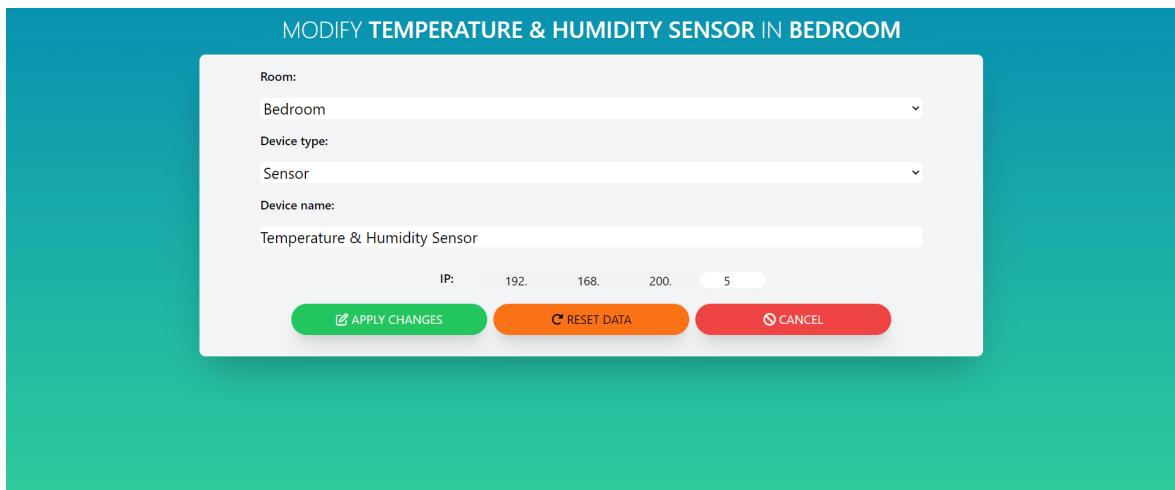
Minden eszköznek az adatai megjelennek a paneljén, mint ahogyan az a 3.3. képen is látható. Ehhez társul minden eszköznek a paneljén egy módosító és egy törlés gomb. Amikor a módosításra megyünk, akkor minden adatot meg tudunk egy adott eszköznek változtatni: legyen ez az eszköz típusa, neve, IP vége, vagy az, hogy melyik szobához tartozik.

Hogy ha törölni szeretnénk szobát vagy eszközt, akkor legelőször is megkérdezi a felhasználót az oldal egy értesítésben, hogy biztos-e abban, hogy törölni akarja az adott

<sup>17</sup> Ennél a hibánál azt az üzenetet kapjuk, mint ahogyan a 3.4. képen is látható

rekordot. Mindezt a 3.2. szakaszban leírtam, hogy miért is nagy döntés, hogy törölne a felhasználó egy szobát vagy eszközt.

Az eszköz módosítási oldal a 3.5. képen látható, hogy hogyan is néz ki.



**3.5. ábra.** Eszköz módosítás oldal

Ezen a részen ugyanúgy le vannak kezelve az esetek, mint a hozzáadásnál is. Itt a „Reset Data” gomb az eredeti adatokra állítja vissza a mezőkben lévő adatokat. Még annyi különbséggel, hogy itt már kiválaszthatjuk, hogy melyik szobába is szeretnénk átállítani az adott eszközt, már, hogy ha a felhasználó szeretné. Ha végeztünk a módosítással akkor az „Apply Changes”, azaz Módosítások Alkalmazása gombra kattintva az adatok ellenőrzésre kerülnek, mint ahogyan az eszköz hozzáadásnál történt.

A beállítások fülből lehet átmenni az RFID beállítások oldalra, hogy ha rákattintunk az oldal tetején lévő „RFID Settings”<sup>18</sup> gombra.

### 3.4.3. RFID beállítások

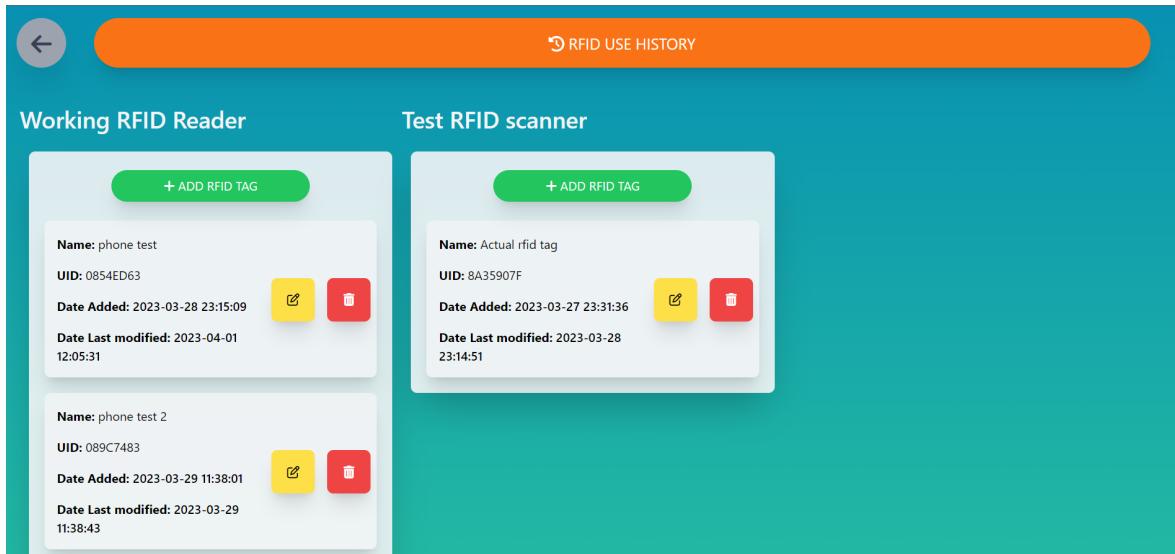
Az RFID beállítások oldalon már a rendszerbe felvett RFID olvasókhöz tudunk felvenni RFID tag-eket, és ezen belül elérhető az RFID használati táblázat.

Ezen az oldalon a beállítások oldalhoz majdnem ugyanúgy van kilistázva, hogy milyen RFID olvasó és hozzá tartozó RFID címkek, és azok adatai, mint ahogyan a 3.6. képen látható.

**Egy RFID tagnek az alábbi adatait lehet látni:**

- Name – Taghez hozzáadott név
- UID – Az adat amit a tag tartalmaz
- Date Added – A tag az adatbázisba felvételének dátuma

<sup>18</sup> Azaz RFID beállítások



**3.6. ábra.** RFID beállítások oldal

- Date last modified – A tag utolsó módosításának dátuma

Hogy ha egy új tag-et szeretnénk hozzárendelni egy RFID olvasóhoz, akkor az annak megfelelő „Add RFID Tag”<sup>19</sup> gombra kell kattintani, ami átvisz minket az RFID oldal létrehozására alkalmas oldalra, amit a 3.7. képen is láthatunk.

#### Az alábbiak a feltételek egy RFID címke létrehozásakor és módosításakor:

- minden mező kitöltése kötelező.
- Egy bizonyos UID-vel rendelkező tag-et csak egyszer lehet hozzárendelni egy adott olvasóhoz.
- RFID tag neve maximum 50 karakter lehet.

Mint ahogyan az alap beállítások oldalon, itt is visszajelzést kapunk vagy helyileg a form-on, hogy ha valami hiba adódik valamelyik adattal vagy az RFID beállítások oldalon kapunk visszajelzést a műveletről, és annak sikerességéről.

Ezen az oldalon azt láthatjuk, hogy van egy mező, amibe írhatunk, ami az RFID tag-hez tartozó név<sup>20</sup> és van egy másik mező, de abba nem tudunk írni. Ez azért van, mivel úgy lett megoldva a tag hozzárendelése az olvasóhoz, hogy rá kell kattintani a „Read Tag”<sup>21</sup> gombra, és ezután van 8 másodpercünk<sup>22</sup>, hogy hozzáéríntsük az RFID olvasóhoz az adott címkét, amit hozzá szeretnénk rendelni.

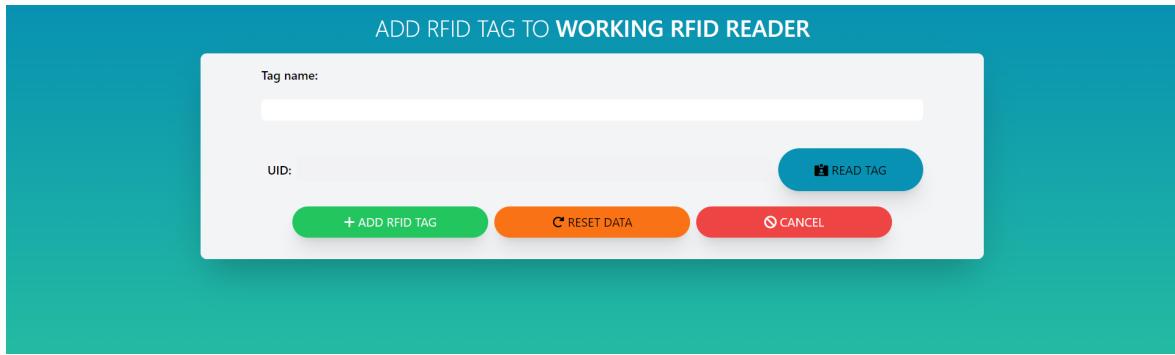
Ha sikerrel járunk, akkor a címke UID-je megjelenik a megfelelő szövegdobozban. Ellenkező esetben, hogy ha az RFID olvasó offline állapotban van, vagy nem lett tag hozzáéríntve az olvasóhoz, akkor értesít az oldal, hogy a művelet sikertelen volt.

<sup>19</sup> Azaz RFID címke hozzáadása

<sup>20</sup> Tag name

<sup>21</sup> Azaz Címke beolvasása

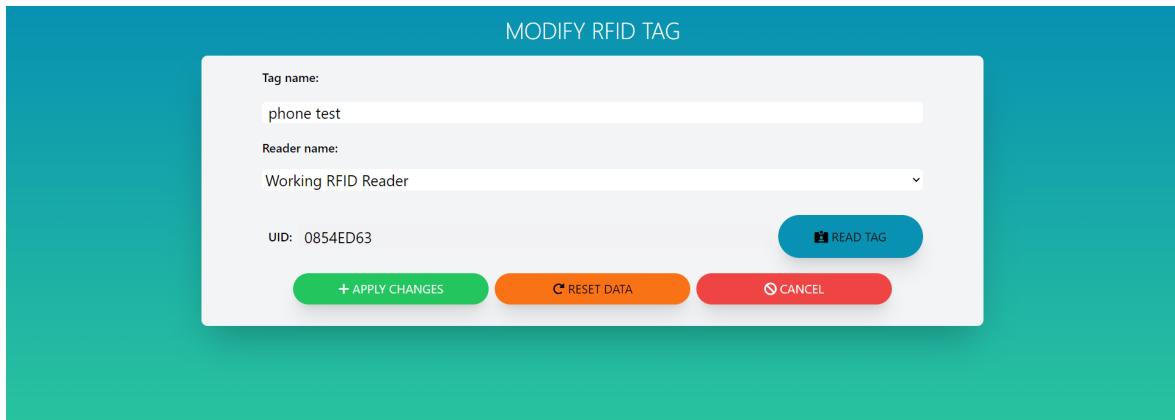
<sup>22</sup> Azért adtam ennyi időt rá, mert ennyi idő alatt kellően van időnk ezt a műveletet elvégezni.



**3.7. ábra.** RFID tag hozzáadása oldal

Itt is ugyanúgy jelen van az a három gomb, mint az eszköz, és a szoba hozzáadása oldalon, ami a mentés, az adatok visszaállítsa, és a mégse.

Az RFID tag módosítására is van lehetőség, hogy ha az adott tag paneljén rákattintunk a módosítás gombra. Ilyenkor lehetőségünk van az RFID olvasó módosítására is<sup>23</sup>, amit láthatunk a 3.8. képen is. Amikor megváltoztatjuk az adott tag-et, akkor a módosítás dátuma annak megfelelő lesz.



**3.8. ábra.** RFID tag módosítása oldal

Van lehetőségünk visszalépni a sima beállítások oldalra, hogy ha a bal felső sarokban a vissza nyílra kattintunk. Az „RFID Use History” gomb-ra kattintva pedig áttérünk az RFID használati táblázat oldalra.

#### 3.4.4. RFID használati táblázat

Az RFID-s rendszer úgy működik<sup>24</sup>, hogyha fel van véve egy RFID tag az adatbázisba az adott olvasóhoz, akkor az eszköz alapértelmezett állapota az olvasó állapot<sup>25</sup> van, és amikor benne van az adatbázisban a beolvasott tag UID-je, akkor zölden villog az

<sup>23</sup> Természetesen akkor az annak megfelelő RFID olvasóhoz kell majd érinteni a címkét, hogy ha az uid-t szeretnénk megváltoztatni.

<sup>24</sup> Mint ahogy részben említve lett a 2.1.7. szakaszon

<sup>25</sup> Ezt jobban majd a 3.5.4. szakasz fogom kifejteni.

ESP-hez kötött RGB LED, és felvételre kerül az rfid\_use\_history adatbázis táblába, hogy mikor, és melyik RFID tag volt beolvasva melyik olvasóval.

Ordinal number	Tag uid	Tag name	Date used
1	8A35907F	Actual rfid tag	2023-03-28 23:15:12
2	8A35907F	Actual rfid tag	2023-03-28 23:15:17
3	8A35907F	Actual rfid tag	2023-03-28 23:16:02
4	8A35907F	Actual rfid tag	2023-03-28 23:17:03
5	8A35907F	Actual rfid tag	2023-03-29 11:37:11
6	8A35907F	Actual rfid tag	2023-03-29 11:37:15
7	8A35907F	Actual rfid tag	2023-04-01 01:07:48
8	8A35907F	Actual rfid tag	2023-04-01 01:07:55
9	8A35907F	Actual rfid tag	2023-04-01 01:08:08

**3.9. ábra.** RFID tag hozzáadása oldal

Hogyha erre az oldalra megyünk, akkor egy táblázat fog minket fogadni, – mint ahogyan a 3.9. képen is látható – vagy is inkább annyi táblázat, amennyi olvasót vettünk be a rendszerbe. Ezeknek a táblázatoknak 4 oszlopa van: **Ordinal number**, azaz sorszám, **Tag uid** azaz címke azonosító, **Tag name**, azaz a címkéhez hozzáadott név, és **Date used**, azaz a használat dátuma.

Ezen a táblázaton nyomon tudjuk követni, hogy mikor is volt használva az adott RFID címkénk.

A táblázatba az adatok úgy kerülnek hozzáadásra, hogy először végig iterálunk egy `@foreach`-el az olvasókon, majd ezen belül végig iterálunk az összes használati alkalmat, és annak megfelelően le van ellenőrizve, hogy ahhoz az olvasóhoz tartozik-e az adott használati alkalom. Hogy ha hozzá tartozik, akkor pedig hozzáadja az adott táblázathoz egy sorban azt a rekordot és annak az adatait.

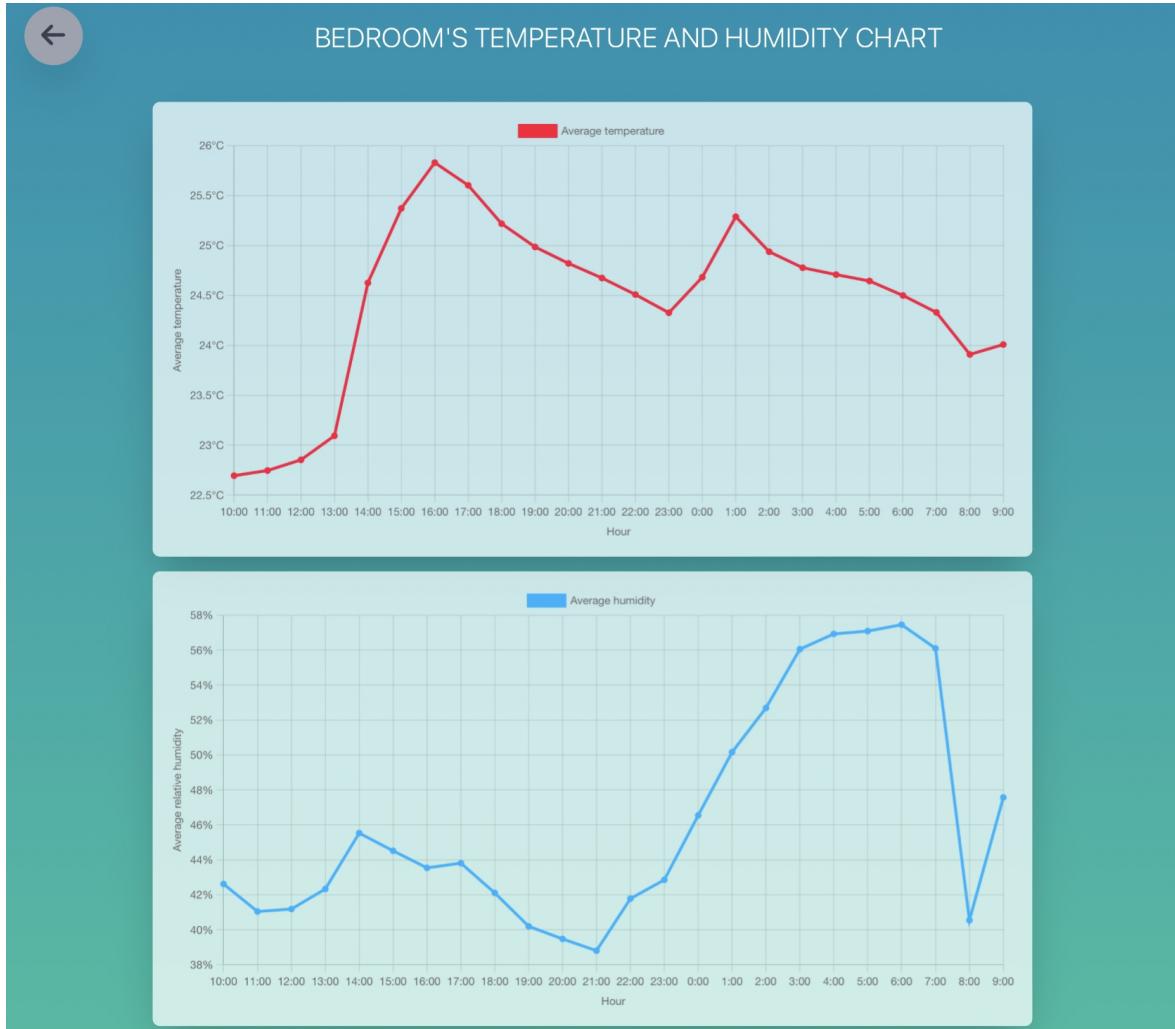
Elsődlegesen nyomon követés céljából lett az oldal megvalósítva, hogy ha komolyabban meg tudnám oldani az RFID használatát, például, hogyha egy zárhoz tudnám hozzácsatlakoztatni az RFID kártyaolvasós ESP-t, de ez anyagiak miatt nem került sajnos megvalósításra a zárszerkezet alkalmazása.

Szintén van lehetőség visszalépni az RFID beállítások oldalra, hogy a bal felső sarokban rákattintunk a vissza nyíl ikonra.

### 3.4.5. Hőmérsékleti és páratartalom előzmények - ChartJS

Amikor hozzáadásra kerül a rendszerbe egy hőmérséklet és páratartalom szenzor, akkor az az által olvasott adatokat meg tudjuk nézni egy grafikonon. Amikor egy szenzoros

panelre kattintunk a főoldalon. Ha rákattintunk, akkor a 3.10. képen látható oldalhoz hasonlót fogunk látni.



**3.10. ábra.** Hőmérséklet és páratartalom szenzor által mért adatok

Az oldal a Chart.js JavaScript könyvtár használatával lett megoldva. Ez mind úgy valósul meg, hogy a vezérlőben lekérjük az elmúlt 24 órányi adatnak az óránkénti átlag mennyiségét és azt továbbítjuk az oldalnak, és ezáltal lehet megjeleníteni a Chart.js használatával a grafikon.

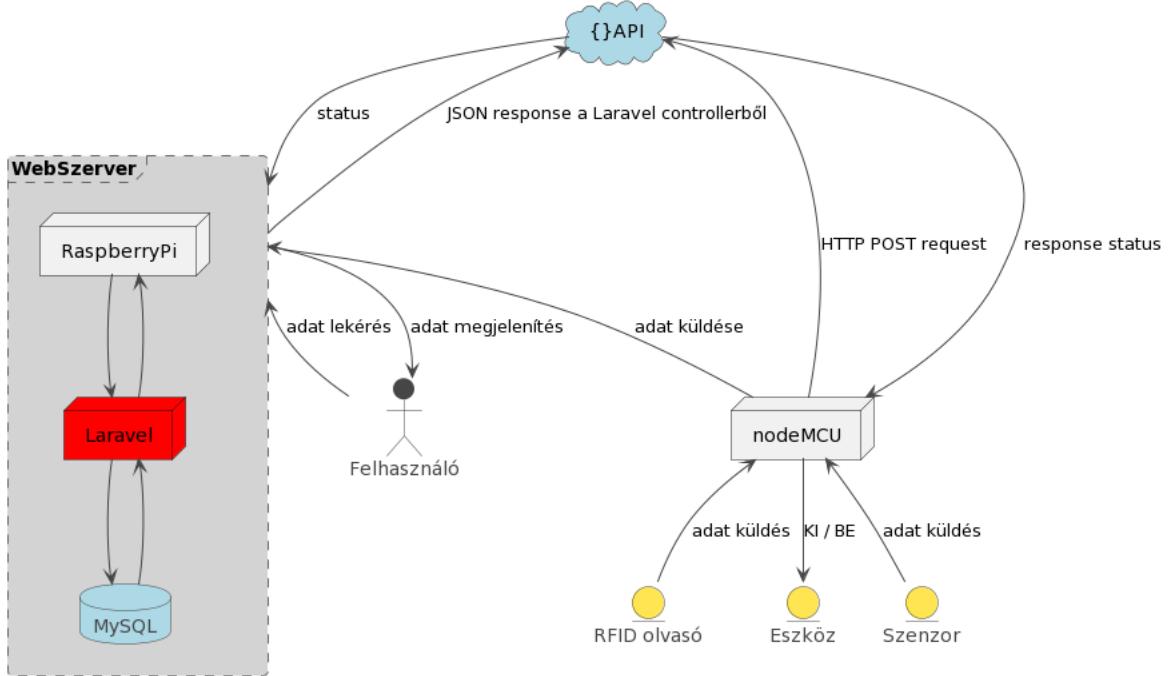
A felső grafikon a hőmérséklet adatokat mutatja, míg az alsó grafikon a relatív páratartalom adatokat.

Itt is lehetőség van visszamenni a főoldalra a bal felső sarokban lévő vissza nyílra kattintva.

### 3.5. Eszközök kommunikációja a webszerverrel

Mivel az alkalmazott eszközök *ESP-WROOM-32*-esekkel van megoldva, ezért valahogyan ki kellett gondolni, hogy ezeket hogyan is alkalmazzam, és hogyan is működjenek.

Itt ezen a részen ezt részletezem. Az eszközöknek az adat folyamata a 3.11. ábrán szemléltetem, amit plantUML-lel készítettem el.



**3.11. ábra.** Eszközök kommunikációja Laravellel

### 3.5.1. Hőmérséklet és Páratartalom szenzor

Legelső eszköz, ami beintegrálásra került a rendszerbe, az a hőmérséklet és páratartalom szenzor volt. Amikor az ember okos otthonot képzeli magának, az első dolog ami eszébe jut ilyenkor az az, hogy az adott szoba hőmérsékletét nyilván tudjuk tartani.

Első probléma amibe ütköztem az az volt, hogy az ötlet az remek, de hogyan is fogom ezeket az adatokat elküldeni a Laravel oldalnak, hogy azt a vezérlőben el tudjam menteni adatbázisba.

Először is kell egy külső könyvtár ahhoz, hogy a DHT22-es szenzort alkalmazni tudjuk.[39] Utána pedig arra volt szükség, hogy hogyan is küldjünk Laravel számára adatokat ESP-ról.[40]

Ahogyan a leírásban néztem, szükséges lesz egy API<sup>26</sup> útvonalra, amit a Laravel api.php fájljában található meg. Mivel tudom azt, hogy rekordot akarok elmenteni, ezért ennek a HTTP kérésnek POST-nak kell lennie. Ez az útvonal *192.168.200/esp* lett. Ezen az útvonalon pedig az „*EspSensorController*” „*store*” metódusát fogom használni, hogy elmentse a szenzoruktól érkező adatokat.

A leírás úgy alkalmazta az ESP-t, hogy meg kellett neki adni azt, hogy rácsatlakozzon a RaspberryPi által szolgáltatott Wi-Fi-re. Ezért meg kell neki adni, hogy mi

<sup>26</sup> Application Programming Interface, azaz Alkalmazás Programozási Felület

a Wi-Fi neve, és jelszava. Még a robotika órán tanultakból tudtam azt, hogy be lehet állítani azt, hogy mi legyen az ESP lokális statikus IP címe, mi legyen a gateway és a subnet mask. Mindez azérét volt szükséges, mivel az alkalmazás IP cím végződés alapján kommunikál az ESP-kel. (Például nekem a szenzoros ESP-nek az az IP címe, hogy *192.168.200.5*.)

Tudom azt a leírásból, hogy a Laravelnek a *192.168.200.1/api/esp* címre kell küldeni az adatokat, hogy működjön azzal az útvonallal.

Ezután már csak annyi kell, hogy lemérje a DHT szenzor a hőmérséklet és páratartalom adatokat a „*dht.readHumidity()*” és „*dht.readTemperature()*” metódusokkal, majd ezt felkészítve HTTP kliens segítségével összeállítva elküldje a Laravel-nek.

Amikor az ESP elküldi az adatotkat, akkor kap vissza egy payload-ot, vagyis cso-magot, ami alapján lehet tudni, hogy a folyamat sikerrel járt-e.

Legelső verzióról még statikusan be volt égetve a kódba, hogy az 1-es id-jű szobának küldi el az adatokat. Miután gondolkodtam ezen, hogy hogyan lehetne azt hogy dinamikusan annak a szobának legyenek elmentve azok az adatok, amihez fel van véve, ahhoz azt a megoldásra gondoltam, hogy a szoba id-je helyett azt küldi el, hogy mi az IP-jénél az utolsó része. Itt a vezérlő lekérdezi, hogy akkor ennek alapján melyik szobához tartozik, majd ennek megfelelően kerülnek elmentésre az adatok. Ezek az adatküldések 10 másodpercenként történnek meg, hogy a lehető legfrissebb szenzor adatokat láthassuk.<sup>27</sup>

A főoldalon, hogy láthassuk ezeket az adatokat, ahhoz jQuery-t használtam, ami azt végzi el, hogy egy „*\$.getJSON()*” metódussal eléri a Laravel web.php-jában az „*/esp/getLatest/szobaIDje*” útvonalat, ami alapján a vezérlő visszaadja a legutóbbi adatot, ami az adatbázisban található. Ez a függvény első lefutása után beállítja azt, hogy 10 másodpercenként kérje le ezzel metódussal ugyanúgy az adatot, minden a „*setInterval()*” metódus használatával.

### 3.5.2. Eszköz kapcsoló

A fő céljaim között az volt, hogy eszközöket – például világítást is – tudjak kapcsolni ki vagy be a rendszerben. Ehhez kellett egy Async<sup>28</sup> Webszerver[41] használata.

Az ESP működését úgy gondoltam ki az Async Webszerver leírása alapján, hogy legyen három kérés lehetőség az ESP felé: „***status***”, azaz állapot, „***on***” azaz be, és „***off***”, azaz ki.

Amikor kap egy kérést az ESP Async Webszerver, akkor mindenkor küld vissza egy állapotot, hogy milyen állapotban van – legyen az **1**, azaz be–, vagy **0**, azaz kikapcsolt

<sup>27</sup> Számítások alapján, hogy ha egy teljes éven keresztül működik a rendszer, akkor is 1 szenzornak az adatai az adatbázisban 70megabyte körüli, ami felettebb alacsony memória igényű

<sup>28</sup> Azaz Aszinkron

állapot JSON formátumban. Itt is természetesen ugyanúgy lett megadva a Raspberryre való csatlakozás, mint ahogyan a hőmérséklet-páratartalom szenzornál is.

Ha azt a HTTP GET kérést kapja az Async Webserver, hogy „/*status*”, akkor csak visszaküldi, hogy jelenleg milyen állapotban van az eszköz, amit vezérel. (Bekapcsolás esetén mindenig kikapcsolt állapottal kezd az ESP.) Amikor „/*on*” HTTP GET kérést kap az Async Webserver, akkor átváltja az állapot változót 1-re, a megadott jelnek az állapotát magas-ra állítja, – ezzel jelet küldve a szilárdtest relének, hogy átfolyhat rajta az áram, ezzel bekapcsolva a hozzá csatolt eszközt – és visszaküldi JSON formátumban, hogy az állapota „1”, azaz bekapcsolt állapotban van. JSON formátumban ez úgy néz ki, hogy `{"status":1}`.

Ehhez hasonlóan zajlik le, amikor azt a HTTP GET kérést kapja az Async Webszerver, hogy „/*off*”, annyi különbséggel, hogy az állapotot átállítja 0-ra, a szilárdtest relének a jelet alacsonyra állítja, – ezzel kikapcsolva az eszközt – és JSON formátumban azt küldi vissza, hogy „0”.

Laravel oldalon jQuery használatával vannak a kérések megoldva. Mindegyik kapcsoló típusú elemhez dinamikusan hozzá van rendelve az „*onClick()*” metódussal, hogy hívja meg a „*toggle()*”, azaz kapcsolás metódust. Amikor betölt az oldal, akkor lefut egy „*getStatus()*”, azaz állapot ellenőrzés, ahol jQuery volt használva, ami azt végezi el, hogy egy „*\$.getJSON()*” metódussal eléri a Laravel web.php-jában a „/*getStatus/espIPvége*” útvonalat, ami vezérőnek azt a feladatot adja, hogy küldjön HTTP GET kérést az adott ESP által szolgáltatott Async Webszerver felé. A kapott JSON válasz alapján pedig átállítja a kapcsolót be- vagy kikapcsolt állapotra a felületen, és a feliratát is annak megfelelően átírja, ha nem sikerül választ kapni valamilyen hiba miatt, vagy mert offline állapotban van az ESP, akkor letiltásra kerül a kapcsolható gomb a felületen és a feliraton az jelenik meg, hogy jelenleg nem elérhető az adott eszköz.

A „*toggle()*” metódus azt nézi, hogy a felületen található kapcsolót be- vagy ki kapcsoltuk, ez alapján éri el a Laravel web.php-ban a „/*esp/toggle/espIPvége/állapot*” útvonalon keresztül a „*Toggle()*” metódust, ami annyit tesz, hogy HTTP GET kérést küld a megadott ESP-nek azzal, hogy be- vagy ki szeretnénk kapcsolni a rákapcsolt eszközt. A biztonság kedvéért ilyenkor lekéri még egyszer azért, hogy milyen állapotot is vett fel az eszköz.

### 3.5.3. Kamera

A kamera implementálásánál voltak problémák. Először is hogyan legyen megjelenítve a kamera által szolgáltatott videó? Erre az lett a megoldás, hogy egy HTML kép-en belül kapja folyamatosan az aktuális képet – azaz `<img>`, aminek az src, azaz forrását a dinamikusan megadott IP vég által éri el az oldal.

Azt, hogy hogyan szolgáltasson webszerverként élő képet az ESP32-CAM, az egy

forrás-ból[42] származik, amibe annyi módosítást végeztem el, hogy az IP-je statikus legyen az ESP32-CAM-nek, hogy biztosítsam ezzel a könnyű elérést.

Második probléma az volt, hogy legelőször úgy volt megoldva, hogy lekérdezi az oldal egy „`$.getJSON()`” metódussal, hogy a kamera elérhető-e. Miközben csatlakozik, akkor pedig azt a feliratot jelenítené meg az oldal, hogy próbál csatlakozni az adott eszközre. Ebből az a probléma adódott, hogy amikor betölt a kép előbb, mint hogy választ kapjon a metódus, akkor megjelenik a szolgáltatott felvétel, de a csatlakozási kísérletnek felirata is maradt ugyan úgy.

Ezekre sikerült elterveznem egy olyan megoldást, ami megfelelően működik akkor is, hogy ha később lesz online a kamera, vagy egyből csatlakozik, vagy ha menet közben megszakadna a kapcsolat a kamerával.

Így a kamerához több metódus tartozik a főoldalon a jól működő megoldáshoz: egy beépített jQuery „`onLoad()`” metódus, ami akkor fut le, hogyha a kép betölt, egy beépített esemény hallgató, azaz „`EventListener`”, ami azt ellenőrzi, hogy hiba adódott-e a kép elérése közben, egy általam megírt „`imageExists()`” metódus, ami „`$.getJSON()`” lekérdezéssel azt ellenőrzi, hogy elérhető-e a kamera, egy „`error()`” metódus, ami átállítja a feliratot arra, hogy hiba keletkezett, majd a képet újból beállítja, és egy „`success()`” metódus, ami átállítja a csatlakozási kérelemről a feliratot arra, hogy mi az eszköz neve és az IP végződése.

Az az egy hátránya ennek a megoldásnak, hogy egyszerre csak egy eszközről lehet megtekinteni a szolgáltatott felvételt.

### 3.5.4. RFID kártya olvasó

Az RFID kártya olvasót egy forrás[38] és egy dokumentáció[15] alapján alkalmaztam, hogy hogyan is olvas egy RFID kártya olvasó, és hogyan is áll össze. Emellett az előző ESP eszközök beállításai alapján – legyen az az Async Webszerver az állapot átállítás, és az adat átküldése Laravel API-ra alapján – állítottam össze.

Az RFID olvasónak két állapota van; – Az egyik az, amikor figyeli, hogy hozzá van-e érintve egy RFID címke, akkor küldi a Laravel API-jára a beolvasott RFID UID-jét, és a vissza kapott payload alapján – Ami vagy „OK”-ot, amikor szerepel az adatbázisban az adott UID, vagy „FAIL”-t, amikor nem szerepel az adott UID az adatbázisban – menti el az adatbázisba az adatot.

Mint ahogyan a hőmérséklet-páratartalom szenzoros eszköznél van az API részt hívja meg csak itt a „`/rfid/`” útvonalon az „`RfidUseDataController`” „`store`” metódusát, ami legelőször lekérdezi, hogy van-e ilyen címke azzal az UID-vel, amit beolvasott, majd ha ez létezik, akkor elmenti az „`esp_use_data`” táblába, hogy mikor lett beolvasva az a címke, – majd mint ahogyan már említettem – „OK” válasz esetén zölden fog villogni a hozzá kötött RGB LED. Ha nem szerepel az adatbázisban az UID, akkor

nem kerül lementésre semmi, hanem visszaküldi az ESP-nek hogy „FAIL”, és pirosan fog ebben az esetben villogni a rákötött RGB LED.

A másik állapot akkor van, amikor a 3.7. képen vagy a 3.8. képen látható „*Read Tag*”-ra kattintunk. Ehhez a gombhoz van egy „*onClick()*” metódus, ami azt nézi, hogy rá lett-e kattintva. Olyankor meghívja a „*GetUID()*” metódust, ami az alábbiakat végzi el: először is ellenőrzi, hogy az oldalon melyik olvasó lett kiválasztva, majd egy „*\$.getJSON()*” metódussal meghívja a web.php-n lévő */getTag/olvasóID/* útvonalon az „*EspController*”-ben lévő „*getTag()*” metódust, ami megkapja az adott olvasó ID-je alapján, hogy mi az IP végződése, ami alapján küld annak az ESP-nek az Acync Webszerverére azt a HTTP GET kérést, hogy „*/read*”, ami átállítja az ESP állapotát, hogy JSON formátumba küldje vissza a beolvasott UID-t. Erre be lett állítva, hogy 8 másodperc legyen arra, hogy ezt a műveletet elvégezze a felhasználó. Így az ESP-től kapott UID-t a vezérlő továbbítja vissza a „*getTag()*” metódus „*\$.getJSON()*”-be ezt. Ez alapján írja be az adott UID szöveges mezőbe az adott címke UID-jét.

## 4. fejezet

# Tesztelések

Egy szoftver – vagy bármilyen termék – tesztelése az egyik legfontosabb dolog, hogy garantálni tudjuk a lehető legjobb működést. Minél több teszt van elvégezve, annál több hibát találunk meg. Ha tesztelés közben hiba jelentkezik, azt orvosolni kell minél előbb. Ezért is törekedtem arra fejlesztés közben is arra, hogy minden funkciót rendszeresen teszteljek, amire automatizált és manuális teszteket is végre hajtottam.

### 4.1. Cypress automatizált tesztelések

A Cypress az egy nyílt forráskódú JavaScript alapú feketedobozos – vagy specifikáció alapú – tesztelő keretrendszer. A feketedobozos tesztelési forma az, ahol a forráskód nincs felhasználva tesztelés során. A Cypress lehetővé teszi azt, hogy egy webalkalmazás felületét gyorsan és konzisztensen le tudjuk tesztelni.

A Cypress használatát azzal könnyítik meg a fejlesztőknek, vagy tesztelőknek, hogy hivatalos dokumentációban biztosítanak használathoz példát és leírást.[43]

A Cypress telepítéséhez az alábbi kódot kell a terminálon belül beírni: „*npm install cypress –save-dev*”, majd ha meg akarjuk nyitni a Cypress tesztelői felületet, akkor pedig „*npx cypress open*”-t kell beírni. Ilyenkor ki kell választani azt, hogy „*E2E Testing*”<sup>1</sup>, majd hogy milyen böngészőben szeretnénk lefuttatni ezeket a teszteket. – Google Chrome, Mozilla Firefox, Electron, vagy Microsoft Edge-et használva. Ha kiválasztottuk a böngészőt, akkor meg mutatja, hogy a */cypress/e2e* mappán belül milyen teszt fájlok találhatóak. Ha kiválasztjuk a kívánt tesztelési fájlt, – ez esetben a „*testing.cy.js*”-t akkor a tesztek elkezdenek lefutni.

**Ezzel az alábbiakat teszteltem:**

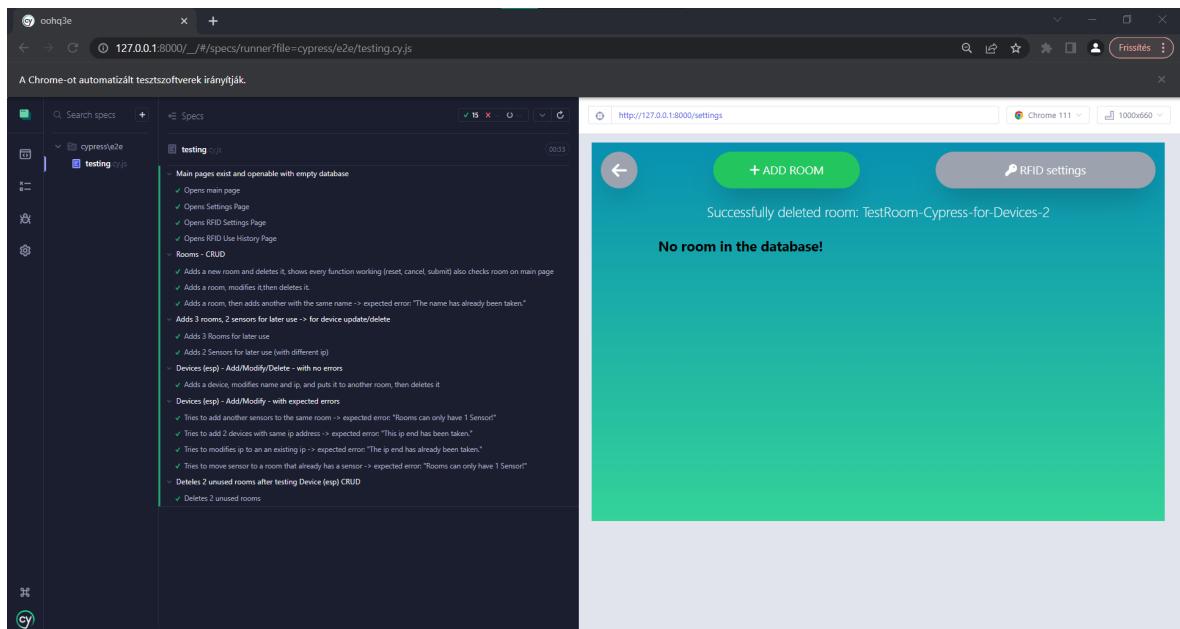
- Üres az adatbázis, minden fő oldalt megnyit, és azok léteznek.

---

<sup>1</sup> End-to-End Testing, ami olyan teszt, ami felhasználói viselkedést szimulál szoftveres környezetben

- Szobákkal CRUD<sup>2</sup> műveletek helyes adat megadásokkal és elvárt hibajelzésekkel.
- Hozzáad 3 szobát és két szenzort későbbi használatra.
- Eszköz hozzáadása és módosítása megfelelően.
- Eszköz hozzáadása és módosítása elvárt hibajelzésekkel.
- Majd a végén törli a használt szobákat – ezzel az eszközöket is.

A 4.1. képen pedig az látható, hogy az összes teszt sikeresen lefutott.



**4.1. ábra.** A lefutott Cypress tesztek

## 4.2. Manuális tesztelések

Olyan tesztelések végezem manuálisan, amelyeket nem lehetséges automatizált teszteléssel elvégezni: A 4.1. táblázatban az RFID olvasó és a kapcsoló eszköz teszteléseit, a 4.2. táblázatban pedig a kamera teszteléseit jegyeztem fel.

<sup>2</sup> Create Update Delete, azaz Létrehozás, Módosítás és Törlés

Teszt leírása	Elvárt eredmények	Tapasztalatok
Az RFID címke hozzáadás és módosítás oldalon, ha „Read Tag”-re kattintunk és címkét nem érintünk az olvasóhoz, és letelik a 8 másodperc.	Az oldal jelzi, hogy nem sikerült a beolvasás.	Az oldal egy értesítésben jelezte, hogy a beolvasás nem sikerült, és próbáljuk újra.
Az RFID címke hozzáadás és módosítás oldalon, ha „Read Tag”-re kattintunk és érintünk címkét az olvasóhoz 8 másodpercen belül.	Az oldalon a UID mezőben megjelenik az adott címke UID-ja.	A UID mezőben sikeresen megjelenik a címkéhez hozzátartozó adat.
RFID olvasóhoz hozzáérintünk egy olvasóhoz nem felvett címkét.	Olvasóhoz kötött RGB LED pirosan kezd villogni.	Az olvasóhoz kötött RGB LED pirosan villog miután megkapta a választ.
RFID olvasóhoz hozzáérintünk egy hozzátartozó címkét.	Olvasóhoz kötött RGB LED zölden villog, ehez tartozó adatok láthatóak az RFID használati előzményeknél a megfelelő RFID olvasó táblázatában.	Az olvasóhoz kötött RGB LED zölden villogott, miután megkapta a választ, az RFID használati előzmény oldalon a megfelelő adatok jelennek meg a hozzá tartozó olvasó táblázatában.
Kapcsoló csatlakozott a szerverhez. - Főoldal	A kapcsolónak a paneljén lévő felirat megjelenik a megfelelő állapottal.	A kapcsolónak a paneljén a felirat megjelent a kapcsolási állapotnak megfelelően.
Kapcsoló nem csatlakozott szerverhez. - Főoldal	A kapcsoló paneljén megjelenik a felirat, hogy jelenleg nem elérhető.	A kapcsolónak a paneljén a felirat sikeresen megjeleñíti, hogy az eszköz jelenleg nem elérhető - a gomb most már nem kapcsolható.
Kapcsolóval bekapcsoljuk az eszközt.	A kapcsoló paneljén a felirat és a gomb jelzi, hogy be lett kapcsolva az eszköz. Az eszköz bekapcsolt állapotba kerül.	A kapcsoló paneljén a felirat és a gomb sikeresen jelzi, hogy be lett kapcsolva az eszköz. Az eszköz szintén sikeresen bekapcsolt állapotba került.
Kapcsolóval kikapcsoljuk az eszközt.	A kapcsoló paneljén a felirat és a gomb jelzi, hogy ki lett kapcsolva az eszköz. Az eszköz kikapcsolt állapotba kerül.	A kapcsoló paneljén a felirat és a gomb sikeresen jelzi, hogy ki lett kapcsolva az eszköz. Az eszköz szintén sikeresen kikapcsolt állapotba került.

**4.1. táblázat.** Manuális tesztelések RFID-ra és kapcsolóra

Teszt leírása	Elvárt eredmények	Tapasztalatok
Kamera csatlakozik a szerverhez - Főoldal.	A megadott kamera panelen megjelenik a kamera élő képe.	A megadott kamera panelen sikeresen megjelenik a kamera élő képe.
Kamera nem csatlakozik vagy lecsatlakozott a szerverről - Főoldal	A megadott kamera panelen kiírja, hogy nem sikerült a kapcsolódás, és újra próbálozik.	A megadott kamera panelen sikeresen kiírja, hogy nem sikerült a kapcsolódás, és újra próbálozik.
Kamera nem csatlakozik vagy lecsatlakozott a szerverről - Adott kamera kép oldala.	A megadott kamera panelen megjelenik a kamera élő képe.	A megadott kamera panelen sikeresen megjelenik a kamera élő képe.
Kamera nem csatlakozik a szerverhez - Adott kamera kép oldala	A kamera panelen kiírja, hogy nem sikerült a kapcsolódás, és újra próbálozik.	A kamera panelen sikeresen kiírja, hogy nem sikerült a kapcsolódás, és újra próbálozik.

**4.2. táblázat.** Manuális tesztelések kamerára

## 5. fejezet

# Rendszer telepítése

Ezen a részen azt szeretném megmutatni, hogy egy Raspberry Pi-re hogyan is tudjuk elérni azt, hogy minden megfelelően működjön, és használni lehessen. Terminál utasításokon keresztül mondjuk el, hogy mit is csinálunk.

**Először is győződjünk meg arról, hogy a legfrissebb a rendszerünk.**

```
-> sudo apt update && sudo apt upgrade -y
```

### 5.1. Raspberry beállítása mint Wi-Fi hozzáférési pont

Itt azokat írom le, ahogyan nekem is alkalmazva lett a Raspberry Pi-n, hogy lehessen majd csatlakozni a Raspberry-hez, mint Wi-Fi hozzáférési pont.

```
-> sudo apt install hostapd  
-> sudo systemctl unmask hostapd  
-> sudo systemctl enable hostapd  
-> sudo apt install dnsmasq  
-> sudo DEBIAN_FRONTEND=noninteractive apt install -y  
    netfilter-persistent iptables-persistent  
-> sudo nano /etc/dhcpcd.conf
```

Fájl végére beilleszteni:

```
interface wlan0  
static ip_address=192.168.200.1/24  
nohook wpa_supplicant  
  
-> sudo nano /etc/sysctl.d/routed-ap.conf
```

Beírni az alábbit:

```
# Enable IPv4 routing
net.ipv4.ip_forward=1

-> sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
-> sudo netfilter-persistent save
```

```
-> sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
-> sudo nano /etc/dnsmasq.conf
```

Hozzáadni az alábbit a file-hoz:

```
interface=wlan0 # Listening interface
dhcp-range=192.168.200.150,192.168.200.200,255.255.255.0,24h
# Pool of IP addresses served via DHCP
domain=wlan      # Local wireless DNS domain
address=/smart.pi.home/192.168.200.1
# Alias for this router
```

```
-> sudo rfkill unblock wlan
-> sudo nano /etc/hostapd/hostapd.conf
```

Hozzáadni az alábbit a file-hoz:

```
# ország kód -> HU - magyar
country_code=HU
interface=wlan0
# Wi-Fi neve:
ssid=raspberrySmarthome
hw_mode=g
channel=7
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
# Wi-Fi jelszava:
wpa_passphrase=huu1vi9doL
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

```
-> sudo systemctl reboot
```

## 5.2. Webalkalmazás telepítése

Én az alábbi leírás alapján telepítettem az alkalmazást Raspberry Pi-re.

```
-> sudo apt install apache2
-> sudo apt install php libapache2-mod-php
php-mbstring php-xmlrpc php-soap php-gd
php-xml php-cli php-zip php-bcmath
php-tokenizer php-json php-pear

-> sudo apt install mariadb-server
-> sudo mysql_secure_installation
-> curl -sS https://getcomposer.org/installer | php
-> sudo mv composer.phar /usr/local/bin/composer
-> sudo chmod +x /usr/local/bin/composer

-> sudo git clone https://github.com/OOHQ3E/smarthome/
-> sudo chgrp -R www-data /var/www/html/smarthome/web/oohq3e
-> sudo chmod -R 775 /var/www/html/smarthome/web/oohq3e/storage
-> sudo nano /etc/apache2/sites-available/laravel-project.conf
```

Az alábbit kell így:

```
<VirtualHost *:80>
DocumentRoot /var/www/html/smarthome/web/oohq3e/public
```

```
<Directory /var/www/html/smarthome/web/oohq3e>
AllowOverride All
Require all granted
</Directory>
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

```
-> sudo a2enmod rewrite
-> sudo systemctl restart apache2
```

Vissza kell lépni a \var\www\html\smarthome\web\oohqe-be

```
-> composer install
-> sudo apt install npm
-> npm install
-> cp .env.example .env
```

Beállítani az adatbázist úgy .env fájlban, ahogyan a rendszeren be lett állítva már, és azokat az adatokat megadni

```
-> sudo mysql -u root  
-> CREATE DATABASE smarthome;  
kilépni CTRL+C-vel
```

```
-> php artisan key:generate  
-> php artisan migrate:fresh  
-> npm run build
```

[46]

# Összegzés

Ahogy befejezem a szakdolgozat projektet, elégedettséget éreztem. Az okos otthon rendszer fejlesztése nehéz volt, de a végtermék figyelemre méltó. A rendszerhez felület létrehozáshoz Laravel, az eszköz kezelésre NodeMCU, Wi-Fi hozzáférési pontként és a projekt gerinceként pedig egy Raspberry Pi szolgál.

A NodeMCU-kat sokféle speciális célú eszközként használtam fel. Ilyen például a kamera, az RFID-olvasó használat rögzítéssel, a hőmérséklet és páratartalom szenzor, amely valós idejű adatot jelenít meg, és grafikon oldalt tartalmaz a szenzor adatokhoz. Végül de nem utolsó sorban az eszköz kapcsoló. Ezek az eszközök párhuzamosan működnek a rendszerben, és a felhasználói felületen keresztül használhatóak és érhetők el.

Az út nehéz volt, de nagyon értékes tapasztalatokkal gazdagított. A rendszer ki-alakítása közben megtanultam a felhasználóbarát felület létrehozásának csinját-bínját, valamint a hardver és szoftver zökkenőmentes integrálását. A NodeMCU és a Laravel nagyon szépen működik együtt, mert mindenkor elősegíti a gyors fejlődést és rugalmas keretet kínál a jövőbeli bővítéshez. A rendszer sokoldalúságát a NodeMCU eszközök demonstrálják, rendszerbe integrálásuk pedig az okos otthon rendszerekben rejlő lehetőségeket mutatja be.

Az okos otthoni rendszerek iránti igény növekszik, és büszke vagyok arra, hogy hozzájárulhattam ehhez a területhez.

Mégis hosszú út áll előttem. A jövőben az MQTT beépítésével és a különböző okos eszközök támogatásával bővíteni fogom a rendszer funkcionalitását és alkalmazhatóságát. A további kényelem érdekében automatizációt és távoli hozzáférést tervezek megvalósítani a rendszerhez, amely lehetővé teszi a felhasználók számára, hogy a világ bármely pontjáról kezelhessék okos otthon eszközüket.

Végezetül remélem, hogy a szoftvert egy nyílt forráskódú okos otthon automatizálási rendszerként adjam ki, amit bárki használhat. A felhasználók jelenleg angol nyelven használhatják a rendszert, de a jövőben további nyelvekre is tervezem a támogatást.

# Irodalomjegyzék

- [1] JEFF BLANKENBURG: *Define Your Appliance Category for a Better Customer Experience*  
URL: <https://developer.amazon.com/blogs/alexa/post/a89f7243-08a0-4c73-8fc5-eb604a93f437/define-your-appliance-category-for-a-better-customer-experience>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [2] JASON WISE: *How Many People Use Alexa in 2023? (U.S. Amazon Statistics)*  
URL: <https://earthweb.com/alexa-users/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [3] GOOGLE HOME NEST TERMÉKEI  
URL: [https://store.google.com/gb/category/connected\\_home](https://store.google.com/gb/category/connected_home)  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [4] HANDWIKI: *Engineering:Xiaomi Smart Home*  
URL: [https://handwiki.org/wiki/Engineering:Xiaomi\\_Smart\\_Home](https://handwiki.org/wiki/Engineering:Xiaomi_Smart_Home)  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [5] WHAT IS OPEN SOURCE  
URL: <https://opensource.com/resources/what-open-source>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [6] ERIC BROWN: *Home Assistant: The Python Approach to Home Automation*  
URL: <https://www.linux.com/topic/embedded-iot/home-assistant-python-approach-home-automation/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [7] HOME ASSISTANT - INTEGRATIONS  
URL: <https://www.home-assistant.io/integrations/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [8] OPENHAB HIVATALOS OLDALA  
URL: <https://www.openhab.org/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.

- [9] THE EPIC STORY OF THE RASPBERRY PI  
URL: <https://raspberrytips.com/raspberry-pi-history/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [10] ESPRESSIF SYSTEMS: *ESP32-WROOM-32 Datasheet*  
URL: [https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf)  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [11] OFFICAL ESP32 DEVICES DOCUMENTATION  
URL: <http://esp32.net/#Info>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [12] ANAT ZAIT: *NODEMCU - A PERFECT BOARD FOR IOT*  
URL: <https://www.circuito.io/blog/nodemcu-esp8266/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [13] AOSONG ELECTRONICS Co.,LTD: *Digital-output relative humidity & temperature sensor/module DHT22 (DHT22 also named as AM2302)*  
URL: <https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [14] U.S. FOOD AND DRUG ADMINISTRATION: *Radio Frequency Identification (RFID)*  
URL: <https://www.fda.gov/radiation-emitting-products/electromagnetic-compatibility-emc/radio-frequency-identification-rfid>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [15] HANDSON TECHNOLOGY: *RC522 RFID Development Kit*  
URL: <http://www.handsontec.com/dataspecs/RC522.pdf>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [16] JAYESH UPADHYAY: *How do Solid State Relays work?*  
URL: <https://www.circuitbread.com/ee-faq/how-do-solid-state-relays-work>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [17] ROMAIN JUILLET: *Essentials of Programming in C++*  
URL: <https://www.bocasay.com/essentials-programming-c/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.

- [18] ROBERT SHELDON: *DEFINITION: PHP (Hypertext Preprocessor)*  
URL: <https://www.techtarget.com/whatis/definition/PHP-Hypertext-Preprocessor>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [19] SUSAN SIMKINS: *Quick Start to JavaScript: Volume 1*  
URL: <https://www.pluralsight.com/courses/quick-start-javascript-1-1870>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [20] JQUERY  
URL: <https://jquery.com/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [21] CHART.JS HIVATALOS DOKUMENTÁCIÓ OLDALA  
URL: <https://www.chartjs.org/docs/latest/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [22] TAILWIND CSS HIVATALOS DOKUMENTÁCIÓ OLDALA  
URL: <https://tailwindcss.com/docs>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [23] DBDIAGRAM.IO OLDALA  
URL: <https://dbdiagram.io/home>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [24] FONTAWESOME - FREE ICONS  
URL: <https://fontawesome.com/search?o=r&m=free>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [25] ACHIM PIETERS *Fritzing – New Parts*  
URL: <https://www.studiopieters.nl/fritzing/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [26] ALFRED DAGENAIS *Fritzing Components - Light bulb*  
URL: <https://github.com/alfreddagenais/fritzing-components/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [27] FRITZING FORUM *SSR-40va solid state relay*  
URL: <https://forum.fritzing.org/t/ssr-40va-solid-state-relay/16832>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.

- [28] AMONTANES *RFID-RC522*  
URL: <https://fritzing.org/projects/mfrc522>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [29] LARAVEL - OVERVIEW  
URL: [https://www.tutorialspoint.com/laravel/laravel\\_overview.htm](https://www.tutorialspoint.com/laravel/laravel_overview.htm)  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [30] OFFICIAL LARAVEL DOCUMENTATION  
URL: <https://laravel.com/docs/10.x>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [31] ELOQUENT: GETTING STARTED  
URL: <https://laravel.com/docs/9.x/eloquent>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [32] ROUTING  
URL: <https://laravel.com/docs/9.x/routing>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [33] CONTROLLERS  
URL: <https://laravel.com/docs/9.x/controllers>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [34] ELOQUENT: GETTING STARTED  
URL: <https://laravel.com/docs/9.x/eloquent>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [35] BLADE TEMPLATES  
URL: <https://laravel.com/docs/9.x/blade>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [36] CORE CONCEPTS: RESPONSIVE DESIGN  
URL: <https://tailwindcss.com/docs/responsive-design>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [37] TAILWIND CSS TOGGLE - FLOWBITE  
URL: <https://flowbite.com/docs/forms/toggle/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [38] CONNECT RFID TO PHP & MYSQL DATABASE WITH Nodemcu ESP8266  
URL: <https://iotprojectsideas.com/connect-rfid-to-php-mysql-database-with-nodemcu-esp8266/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.

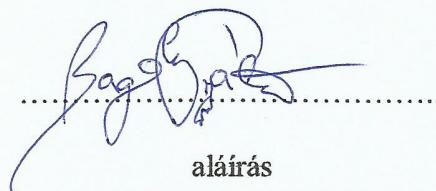
- [39] ARDUINO.CC *Adafruit: DHT sensor library*  
URL: <https://www.arduino.cc/reference/en/libraries/dht-sensor-library/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [40] ARDUINO TO LARAVEL COMMUNICATION  
URL: <https://www.instructables.com/Arduino-to-Laravel-Communication/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [41] ASYNC WEB SERVER FOR ESP8266 AND ESP32  
URL: <https://github.com/me-no-dev/ESPAsyncWebServer/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [42] ESP32-CAM-ARDUINO-IDE  
URL: <https://github.com/RuiSantosdotme/ESP32-CAM-Arduino-IDE/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [43] OFFICAL CYPRESS DOCUMENTATION  
URL: <https://docs.cypress.io/guides/overview/why-cypress>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [44] SZAKDOLGOZAT GITHUB REPOSITORY-JA  
URL: <https://github.com/00HQ3E/smarthome/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [45] RASPBERRY CONFIGURATION FOR WI-FI  
URL: <https://www.raspberrypi.com/documentation//computers/configuration.html#setting-up-a-routed-wireless-access-point>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.
- [46] HOW TO INSTALL LARAVEL ON UBUNTU 18.04 WITH APACHE  
URL: <https://www.hostinger.com/tutorials/how-to-install-laravel-on-ubuntu-18-04-with-apache-and-php/>  
LINK UTOLJÁRA ELLENŐRIZVE: 2023.04.13.

## NYILATKOZAT

Alulírott BABOY GÁBOR, büntetőjogi felelősségem tudatában kijelentem, hogy az általam benyújtott, Programozható elektronikák alkalmazásai című szakdolgozat (diplomamunka) önálló szellemi termékem. Amennyiben mások munkáját felhasználtam, azokra megfelelően hivatkozom, beleértve a nyomtatott és az internetes forrásokat is.

Tudomásul veszem, hogy a szakdolgozat elektronikus példánya a védés után az Eszterházy Károly Katolikus Egyetem könyvtárába kerül elhelyezésre, ahol a könyvtár olvasói hozzájuthatnak.

Kelt: Eger, 2023. év április hó 14 nap.



aláírás