Szakdolgozat

Szaniszló Gábor

Debrecen

2022

Debreceni Egyetem

Infomatikai Kar

Okosotthon beágyazott rendszer és annak biztonsága

Témavezető

Oláh Norbert

tanársegéd

Készítette

Szaniszló Gábor

Mérnökinformatikus Bsc

# Köszönetnyilvánítás

Köszönetet szeretnék mondani témavezetőmnek, Oláh Norbertnek, amiért türelmével és szakmai hozzájárulásával segítette munkám elkészültét.

Továbbá köszönetet szeretnék szüleimnek, barátnőmnek és barátaimnak, akik támogattak a szakdolgozatom írása alatt.

Tartalomjegyzék

[1. Bevezető 1](#_Toc120604333)

[2. Felhasznált eszközök és technológiák 3](#_Toc120604334)

[2.1. A fejlesztői környezet 3](#_Toc120604335)

[2.2. Firebase 4](#_Toc120604336)

[2.2.1. Firebase Hosting 4](#_Toc120604337)

[2.2.2. Firebase Real Time Database 4](#_Toc120604338)

[2.2.3. Firebase Cloud Storage[KIEG firebase doksi] 5](#_Toc120604339)

[2.2.4. Firebase Authentication 5](#_Toc120604340)

[2.2.5. Firebase Real Time Database Security Rules 6](#_Toc120604341)

[2.3. Dolgok Internete 7](#_Toc120604342)

[2.3.1. IoT biztonság 8](#_Toc120604343)

[2.4. Felhasznált JavaScript könyvtárak 10](#_Toc120604344)

[2.4.1. Firebase JavaScript SDK 10](#_Toc120604345)

[2.4.2. MDB 10](#_Toc120604346)

[2.4.3. Highcharts JS 11](#_Toc120604347)

[2.5. RFID technológia [21] 11](#_Toc120604348)

[2.5.1. RFID frekvenciasávok 12](#_Toc120604349)

[2.6. Felhasznált hardvereszközök 13](#_Toc120604350)

[2.6.1. Espressif ESP32 13](#_Toc120604351)

[2.6.2. BME280-M kombinált szenzor [datasheet] 17](#_Toc120604352)

[2.6.3. PIR mozgásérzékelő szenzor 18](#_Toc120604353)

[2.6.4. MFRC522 RFID olvasó [datasheet] 18](#_Toc120604354)

[2.6.5. P2520-12 Elektromágnes 18](#_Toc120604355)

[2.6.6. MQ-135-M levegő minőség érzékelő modul 19](#_Toc120604356)

[2.6.7. SA-17 lángérzékelő modul 19](#_Toc120604357)

[2.7. Összegzés 20](#_Toc120604358)

[3. A fejlesztett rendszer és a felhasználói felület bemutatása 21](#_Toc120604359)

[3.1. A rendszer megtervezése, előkészítése 21](#_Toc120604360)

[3.2. A rendszer felépítése 24](#_Toc120604361)

[3.2.1. Beléptető rendszer 25](#_Toc120604362)

[3.2.2. Környezeti szenzorrendszer 28](#_Toc120604363)

[3.2.3. Fénykezelő rendszer 29](#_Toc120604364)

[3.2.4. Felhasználói felület 30](#_Toc120604365)

[3.2.5. Firebase Realtime Database és Cloud Storage biztonsági szabályok 31](#_Toc120604366)

[4. Összegzés 33](#_Toc120604367)

[5. Irodalomjegyzék 35](#_Toc120604368)

# Bevezető

Szakdolgozatom témájául az okosotthonok megvalósítását választottam beágyazott rendszerekkel, illetve ezeknek a szisztémáknak biztonsági szempontból történő megközelítését.

Manapság az okosotthonok igen nagy népszerűségnek örvendenek. Napjainkban számos ilyen rendszerrel találkozhatunk, rengeteg termék létezik az otthonunk okosítására. Az okosotthon rendszerek nagyban meg tudják könnyíteni mindennapjainkat, egyszerűbbé tehetik azokat, azonban, ha nincs megfelelő hangsúly helyezve a biztonságra, akkor kockázatot jelenthetnek a felhasználók számára. Az Avast felmérései alapján [6] az okosotthonok 40,8%-a tartalmaz olyan eszközt, ami sérülékenynek tekinthető. Ezért is szerettem volna ezzel foglalkozni.

Célomul tűztem ki egy olyan rendszer kiépítését beágyazott rendszerek segítségével, amely teljesíti a feljebb említett előnyöket, illetve megfelelően biztonságosnak tekinthető. Több mikrokontrollerből álló rendszert fejlesztettem, ami tartalmaz egy RFID-val működő beléptető rendszert, amely webes felületről is vezérelhető, világításvezérlést, illetve különböző szenzorokat, mint például hőmérséklet, nyomás, gázérzékelő, lángérzékelő, melyeknek az adatait a felhasználó elérhet az eszközeiről. A felhasználó látja a felületen a rendszer állapotjelzéseit, értesül a be- és kilépésekről. A webes felület távolról is elérhető, figyelmet fordítva a biztonságra, hogy ne férjen hozzá olyan, akinek nincs engedélye. A mikrokontrollerek között stabil és biztonságos kommunikáció biztosítása fontos. A webes felületet a Google Firebase szolgáltatja, ami lehetővé teszi saját weboldal szolgáltatását az internet felé. Ezen kívül tartalmaz más hasznos szolgáltatást fejlesztők számára, mint például felhasználó azonosítás, NoSQL adatbázisok, real time database, egy felhőből szolgáltatott adatbázis, amiben a tárolt adatok JSON formátumban vannak eltárolva, és valós időben vannak szinkronizálva a klienseknek.

A munka elvégzése a rendelkezésre álló eszközök megvizsgálásával kezdődött. Ez magába foglalja a szükséges eszközök kiválasztását, beszerzését és az információgyűjtést. Ezután következett a rendszer megtervezése. Mikrokontrollerek közül a NodeMCU ESP32S típusú kontrollerre esett a választásom, a széleskörű elérhető források, jól dokumentáltság, sokszínű funkcionalitása (például beépített WiFi és Bluetooth), alacsony fogyasztása, illetve az ára miatt. Ezek az eszközök gyűjtik be az adatokat a szenzorokból és továbbítják a Firebase alkalmazás valós idejű adatbázisa felé, amely frissíti a felhasználó felületén. A felületen gombnyomással vagy akár egy kapcsoló segítségével irányíthatjuk a rendszert. Ilyenkor az adatbázisban tárolt állapot megváltozik. A mikrokontrollorek csatlakoztatva vannak az internetre, össze vannak kötve a felhasználóval, be vannak jelentkeztetve, az eszközök értesülnek az állapotváltozásokról.

# Felhasznált eszközök és technológiák

Ebben a részben bemutatom a cél eléréséhez felhasznált eszközöket, technológiákat és szolgáltatásokat.

## A fejlesztői környezet

A fejlesztés során a programkódokat a Visual Studio Code integrált fejlesztői környezetben írtam. A Visual Studio Code ingyenes és nyíltforráskódú. Alapfunkcióit tekintve kielégíti a fejlesztői igények alapjait. Az alapfunkciókat lehetőségünk van bővítmények telepítésével kiegészíteni. Nagyban megkönnyíti és felgyorsítja a fejlesztést a beépített okos kódkiegészítés. A weblap elkészítéséhez nem használtam bővítményt.

A mikrokontrollerek programozásához a PlatfromIO elnevezésű bővíményt vettem segítségül, amelyet kifejezetten beágyazottszoftver-fejlesztésre terveztek. A dolgozat írásakor 1439 különböző eszközt, 25 keretrendszert és 48 platformot támogat. A bővítmény rendelkezik a Visual Studio kódon belül integrált grafikus felülettel, ahonnan új projekteket hozhatunk létre, kezelhetjük azokat, könyvtárakat adhatunk hozzájuk. Rendelkezik beépített terminállal és soros port monitorral, amellyel olvashatunk az eszközök soros kimenetéről.A képen szöveg, monitor, képernyőkép, képernyő látható

Automatikusan generált leírás

. ábra A PlatformIO Home panelje a Visual Studio Code fejlesztői környezetben

## Firebase

A Firebase egy eszköz és szolgáltatás, amelyet a Google hozott létre, ami az alkalmazásom szerves részét és tulajdonképpen a hátterét, backend-jét képezi. A Firebase felhasználási módjai szerteágazóak. Felhasználhatjuk akár Android vagy iOS alkalmazások fejlesztéséhez, backend alkalmazásokhoz, vagy mint az én esetemben webalkalmazásokhoz is.

Választásom erre a szolgáltatásra azért esett, mert eszközeit egyszerű és gyors implementálni, legyen akár szó felhasználó azonosításról, bejelentkezésről, adatkezélésről, adatbázis hozzáférésről.

A később leírt és projektem által felhasznált mikrokontrollerekhez is készült olyan könyvtár, amit arra használhatunk, hogy összekössük az eszközt a Firebase különböző szolgáltatásaival egyszerűen.

A következő alpontokban szeretném bemutatni azon részeit, amiket használtam az alkalmazásom megvalósításához.

### Firebase Hosting

A Firebase Hosting webes tartalom szolgáltatására való, beépített funkciója a Firebase rendszernek, amely modern webes alkalmazásokhoz szánt. Segítségével hatékonyan és gyorsan tudunk webapplikációkat, statikus és dinamikus tartalmat szolgáltatni az Interneten. [7]

### Firebase Real Time Database

A Firebase Real Time Database egy felhő alapú adatbázis. Azonban ebben az esetben az adatok más módon vannak tárolva, mint ahogy az például egy relációs adatbázisban megszokhattuk. A Firebase Real Time Database-ben az adatok JSON formátumban vannak eltárolva [8], és valós időben vannak szinkronizálva minden csatlakozott kliens felé. A következő példakód szemlélteti a JSON formátum felépítését:A képen szöveg látható

Automatikusan generált leírás

2. ábra JSON formátum

### Firebase Cloud Storage

A Firebase Cloud Storage olyan alkalmazásokhoz készült, amelyeknek szükségük van arra, hogy felhasználó által generált tartalmak, mint a fotók vagy videók, tárolására és kiszolgálására. A Firebase SDK a Cloud Storage-t használó alkalmazások számára lehetővé teszi a fájlok biztonságos feltöltését és letöltését a hálózat minőségétől függetlenül.

Az adatok egy úgynevezett Google Cloud Storage bucket-ben tárolódnak. Ennek a bucket-nak van egy URL-je, amit meg kell adnunk az alkalmazásunkban, hogy elérhesse a tárhelyet. Létrehozásnál ki kell választanunk egy földrajzi régiót, ahol a fizikai szerver lesz amelyen tárolódnak az adatok. Ezt később nem változtathatjuk meg.

Egy Cloud Storage bucket csak fájlokat tartalmaz, mappákat vagy könyvtárakat nem. Azonban megadhatunk olyan fájlneveket, amelyek tartalmaznak „/” jelekkel elválasztott prefixet. Példa: */data/photo.png*. Így ezeknek az egyedi fájlnevek használatakor úgy tűnhet mintha olyan könyvtárútvonalak lennének, mint amilyenek a fájlrendszereknél megszokhattunk. A Firebase adminisztratív grafikus kezelőfelületén mappaként és a fájlok azokba rendeződve jelennek meg. [13]

A biztonságot és különböző jogokat és engedélyeket, mint például ki írhatja és olvashatja az adatok, hasonlóképpen, mint a Realtime Database esetén biztonsági szabályok megírásával adhatjuk meg.

### Firebase Authentication

Gyakori első lépés az alkalmazásunk biztonságossá tételénél a felhasználóink azonosítása valamilyen módon [13]. A Firebase Authentication lehetővé teszi az alkalmazásunkban, hogy akár több módon is be tudjuk azonosítani egy felhasználót, ezáltal biztosítva a rá szabott és hozzá tartozó adatokhoz a hozzáférést. Használhatunk hagyományos egyszerű e-mail cím és jelszó alapú, vagy akár harmadik féltől származó azonosítást, mint például Google, Apple, Facebook, Twitter, Github, Microsoft, Yahoo. Lehetséges még telefonszám alapján történő azonosítás, amikor SMS-t kap a felhasználó. Ezen kívül még arra is kapunk lehetőséget, hogy saját autentikációs módot használjunk, ha az alkalmazásunk rendelkezik ilyen beépített tulajdonsággal. Ezen módszerek közül a fejlesztő szabadon választhat ki egyet vagy akár többet igényeinek megfelelően.

Az azonosítás a Firebase Authentication használatakor a felhasználótól be kell kérni az őt igazoló adatait, ami lehet bármilyen e-mail cím és jelszó kombináció, vagy akár bármilyen fent említett szolgáltatóhoz tartozó azonosító adatai. Ezután az igazoló adatokat ellenőrzi a Firebase Authentication backend szolgáltatás, és válaszol az eredménnyel. Sikeres bejelentkezés után a felhasználó hozzáférhet adatokhoz, vagy változtathat rajtuk, a fejlesztő által megadott módon.

### Firebase Real Time Database Security Rules

A Firebase Real Time Database Security Rules a valós idejű adatbázis hozzáférési és módosítási szabályait foglalja magába. Ezek a szabályok határozzák meg, hogy ki tudja olvasni és írni az adatbázisban lévő adatokat, hogyan vannak ezek az adatok strukturálva [13]. A szabályok automatikusan érvényesítve vannak a Firebase szerverein minden esetben. Minden olvasási és írási kérelem esetén ellenőrizve vannak a szabályok, és csak akkor hajtódik végre a kérés, ha azok megengedik.

Alapértelmezetten a szabályok nem engedik meg a felhasználói hozzáférést a valós idejű adatbázishoz. Erre amiatt van szükség, hogy ameddig nem implementálunk saját szabályokat és megfelelő azonosítást, addig védve legyen adatbázisunk.

A felhasználók beazonosítása csak egyik része a biztonságnak [13]. Amikor tudjuk ki a felhasználó valamilyen módon irányítanunk kell az ő hozzáférésüket az adatbázisban tárolt adatokhoz. A Realtime Database Security Rules megengedik, hogy külön szabályokat hozzunk létre felhasználóknak.

A szabályokat egy JavaScript-szerű szintakszis szerint lehet leírni és négy fajtája van:

|  |  |
| --- | --- |
| Szabálytípusok | |
| .read | Meghatározza, hogy az adat olvasható-e a felhasználó által |
| .write | Meghatározza, hogy az adat írható-e a felhasználó által |
| .validate | Meghatározza, hogy az adatbázisba írandó helyesen formázott érték hogyan néz ki, van-e gyerek attribútuma, milyen az adattípusa. |
| .indexOn | Meghatároz egy indexelendő elemet, ami által lehetséges a rendezést és lekérdezést |

*1.táblázat*

A következő példában egy olyan szabályleírást láthatunk, amely megengedi bárkinek, hogy olvasson a */foo/* útvonalon, de senkinek, hogy írja [13]:



3. ábra Firebase biztonsági szabályok példa [13]

A szabályok tartalmazhatnak beépített változókat [13], mint például az *auth*, amely olyan attribútumokkal rendelkezik, mint egy egyedi felhasználó azonosító, a bejelentkezési metódus és Firebase ID token, ami a felhasználóhoz tartalmazó adatokat tartalmaz.

## Dolgok Internete

A dolgok internete, avagy a jobban ismert angol nevén Internet of Things egy olyan eszköz (vagy eszközök csoportja), amelyek szenzorok segítségével képesek információt feldolgozni, és azt az Interneten keresztül megosztani mást eszközökkel. Ez a technológia gyors ütemben fejlődött az elmúlt években, és ez folytatódik tovább [9]. Fogalmazhatunk úgy is, hogy intelligens eszközök halmaza, amelyek adatokat gyűjtenek. Ilyen intelligens eszközök lehetnek akár hétköznapi háztartási eszközök, mint például egy mosógép vagy egy kávéfőző, orvosi eszközök, vagy akár viselhető eszközök. mint egy okosóra. [10] Az elektronikai eszközök, valamint a távoli vezérlés lehetőségének rohamos fejlődése, és ennek eredményeként ezek árainak csökkenése lehetővé tette a dolgok internetének, és azon belül az okosotthonok elterjedését nem csak ipari, de akár otthoni és hétköznapi felhasználói környezetben is.

A dolgok internete egyik gyakori alkalmazása az úgynevezett okosotthonok. Az okosotthon ezeknek az IoT vagy Internet of Things technikai megoldásoknak az együttese, amelyek az általuk begyűjtött adatokkal és a házban működő rendszerek irányítását lehetővé téve biztonságosabbá és kényelmesebbé teszik a benne lakók életét. [11]

Az okosotthon rendszerekben a kommunikációra általában kétféle megoldást használnak. Az egyik a mikrohullámú Wi-Fi technológia, a másik a rádiófrekvenciás RFID technológia. Alkalmazásomban mindkét technológiát alkalmaztam.

Az okosotthon és egyéb Dolgok Internete rendszerek elemeit képzik az érzékelők, amelyek a környezet állapotát, változását mérik; szabályzó rendszer, ami egy vagy mikrokontrollerből, vagy számítógépből áll, amelyek biztosítják a rendszer vezérlését; aktív elemekből, amelyek végrehajtják a szabályzó rendszer utasításait, ilyenek például a motorok, kapcsolók.

Az IoT megoldások magukba foglalják az Internethez való csatlakozást is, ezért nagyon fontos az ilyen rendszereket adatvédelmi és egyéb biztonsági szempontokból is védeni.

### IoT biztonság

Mivel a dolgok internete nagyon szerteágazó tud lenni, ezért a biztonsága is szintén az. A technológia fejlődésével ez csak fokozódik [12]. Ezért fontos, hogy lássuk, hogy az ilyen rendszerek milyen sebezhetőségekkel rendelkeznek.

Minél több módon kapcsolódnak az eszközeink, annál több van a veszélyt jelentő szereplőknek problémát okozni. Az olyan protokollok, mint a HTTP (hypertext transfer protocol) vagy API (application program interface – alkalmazásprogramozási interfész) csak két példa, amelyek a támadóknak lehetőséget adhatnak kárt tenni.

Az IoT eszközökhöz nem csak szigorúan az Internethez kapcsolódó eszközök tartoznak. A Bluetooth technológiával kommunikáló eszközök is ide tartoznak. A kapcsolat miatt biztonságra van szükségük. Az előbb említettek miatt létrejövő félreértések hozzájárulnak a dolgok internetéhez kapcsolódó személyes adatok megsértéséhez, rendszerekhez nem engedélyezett hozzáférésekhez.

Alábbi pontokba összeszedve fejtek ki kiberbiztonsági és adatvédelmi szempontokat, amelyek kihívást jelentenek az IoT biztonságban, mint iparágban. Ezeknek a nem figyelembevétele veszélyt jelentenek a felhasználók vagy akár szervezetek adat- és pénzügyi biztonságára [12].

**1. Távoli elérhetőség**

Az IoT eszközök nagy támadási felülettel rendelkeznek, ennek a fő oka az Internet elérhetőség. Ez a lehetőség bármennyire is fontos és hasznos, mindazonáltal megengedheti a támadóknak, hogy távolról rosszindulatúan hozzáférjenek az eszközökhöz. Ez az egyik oka, hogy a különböző adathalászati vagy úgynevezett phishing támadások hatékonyak tudnak lenni.

**2. Körültekintés hiánya a különböző ágazatokban**

Ahogy a technológia fejlődött egyre több ipari és alapvető ágazat terjesztette ki a használt rendszereit intelligens eszközökkel, hogy modernek, hatékonyabbak és költségkímélőek legyenek. Ilyen ágazatok például az autóipar vagy az egészségügy. Azonban ez a változás nagyobb függőséget hozott létre a technológiával, mint bármikor ezelőtt.

A technológiától való függőség több lehetőséget ad a támadók kezébe, így az IoT eszközök használata is csak felerősíti ezt. Eleinte az autóipari vagy egészségügyi gyártók nem voltak minden esetben felkészítettek arra, hogy belefektessék a szükséges mennyiségű pénzt és erőforrást ezen eszközök biztonságossá tételére [12].

E körültekintés hiánya fölöslegesen tett ki sok szervezetet és gyártót a megnövekedő számú kiberbiztonsági veszélyforrásnak.

**3. Erőforrás korlátok**

A körültekintés hiánya nem az egyetlen probléma volt, amivel szembe kellett nézni ezekben a frissen digitalizált iparágakban. Egy másik nagy gondot jelentő kihívás a dolgok internete eszközök erőforrás korlátjai. Nem minden IoT eszköznek van meg a számítási kapacitása, hogy képes legyen integrálni tűzfalakat vagy antivírus szoftvereket.

A következő kérdés, hogy akkor hogyan védhetjük meg IoT rendszereinket és eszközeinket a támadásoktól.

A fentebb említett problémák közül a legtöbb leküzdhető lehet megfelelő felkészültséggel, különös figyelmet fordítva rájuk a kutatási és fejlesztési folyamat elején legyen akár szó fogyasztói, vállalati vagy ipari IoT fejlesztésről. A fejlesztőknek azonban figyelembe kell venniük a biztonsági sebezhetőségeket a fejlesztés minden szakaszában, nem csak a tervezési időszakban.

Az egyik módszer a biztonság növelésére a kliens-szerver kapcsolatok esetén a nyilvános kulcsú infrastruktúra (angol nevén: public key infrastructure, röviden PKI). A PKI lehetővé teszi, hogy kulcspárokkal titkosíthassunk tartalmat [15], illetve, hogy kezeljünk digitális tanúsítványokat. Ez segít abban, hogy megvédjük az egyszerű szöveges információkat, és ezek a weboldalakra küldésénél privátok maradjanak.

A hálózat biztonsága is nagyon fontos IoT és más kiberbiztonsági szempontból. Mivel az IoT hálózatok fizikai és digitális elemeket is magukba foglalnak, az IoT biztonságnak mindkettővel foglalkoznia kell. Az IoT hálózatok védelméhez tartozik a megfelelő port biztonság, a porttovábbítás (ismertebb angol nevén port forwarding vagy tunnelling) kikapcsolása és portok csak olyan esetben való megnyitása, amikor szükséges; tűzfalak használata, behatoló detektáló és megelőző rendszerek; nem engedélyezett IP (Internet Protocol) címek blokkolása, a legfrissebb szoftverek használata.

A következő szempont az alkalmazásprogramozási interfészek biztonsága. Az API-k a legtöbb modern weboldal fontos elemei. A támadók felhasználhatják ezeket, ha a fejlesztők nem elég figyelmesek. Annak biztosítása elengedhetetlen, hogy csak engedélyezett eszközök, felhasználók és alkalmazások kommunikálhassanak az API-kal, hogy az adatok védve legyenek az IoT eszközök és a backend szolgáltatások közötti kommunikáció során. [12]

## Felhasznált JavaScript könyvtárak

A JavaScript nyelvnek kikerülhetlen és helyettesíthetlen szerepe van a webes fejlesztésekben, a HTML és a CSS mellett. Ezt az is mutatja, hogy 2022-ben a világon létező 1,8 milliárd weboldalnak a 98%-a Javascriptet használ, mint kliens oldali programozási nyelv [16].

Az alkalmazásom fejlesztése során használt könyvtárakat szeretném bemutatni ebben a paragrafusban.

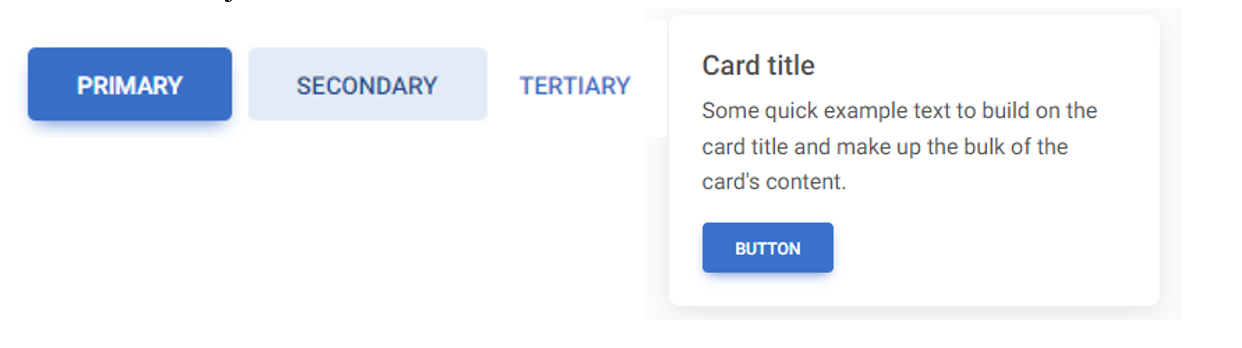
### Firebase JavaScript SDK

A Firebase JavaScript SDK egy implementáció olyan kliens oldali könyvtáraknak, olyan alkalmazások számára, amelyek a Firebase szolgáltatásait használják. A Firebase JS SDK leegyszerűsíti ezen olyan szolgáltatások használatát, mint a Firebase Authentication, Real Time Database, Cloud Storage.

### MDB

Teljes nevén Material Design for Boostrap vagy MDB egy olyan nyílt forráskódú UI (User Interface) csomag, ami minőségi, kész felhasználói felület komponenseket tartalmaz, amiket használhatunk weblapunk kinézetéhez. Az MDB támogatja az alap JavaScript-et, de ezen kívül még más JavaScript keretrendszereket is, mint az Angular, React, Vue és ahogy az elnevezés is sugallja a Bootstap-et.[17]

Az MDB az elemek kinézetét a Google által fejlesztett Material Design úgynevezett formanyelvére (angolul design language) építi. A Google elérhetővé tette külső fejlesztők számára formanyelv használatához szükséges API-kat, ezáltal lehetővé téve az olyan könyvtárak létrejöttét, mint az MDB. [17]

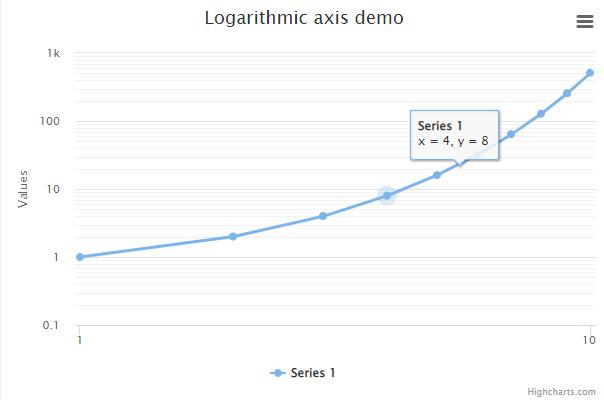


4. ábra MDB-vel készült gomb és card komponensek [17]

### Highcharts JS

A Highcharts JS egy interaktív diagram könyvtár. A könyvtár rendelkezik a legtöbb eszközzel, amelyre szükség van adatvizualizációhoz. JavaScript-re és TypeScript-re épül. A Highcharts támogat a JavaScript-en kívül még számos népszerű programozási nyelvet, mint a .Net, PHP, Python, R, Java, valamint iOS-t és Android-ot is.[18]

A diagrammokat tartalmazó könyvtár kombinálható mást könyvtárakkal, például, amivel Gantt-diagrammokat, térképeket lehet megjeleníteni.



5. ábra Highcharts-al készült diagram [18]

## RFID technológia

A Radio Frequency Identification (RFID) magyarul rádiófrekvenciás azonosítást jelent. Az RFID technológia egy automatikus azonosító eljárás, amely rádióhullám technológián alapul. [28]

Az RFID négy alkotóelemből áll: jeladóból, ami egy chipből és egy antennából áll, leolvasó berendezésből, amely szintén rendelkezik egy vagy több antennával. A technológia különlegessége, hogy a jeladó képes bármiféle tápellátás nélkül működni. A leolvasó elektromágneses mezőt generál és amint a jeladó az antenna hatósugarában van, a mező feszültséget indukál, ennek hatására a chip olvasható és írható [28]. A jeladó attól függően, hogy mennyi energiát vesz fel, vált a bekapcsolt és a kikapcsolt állapot között, ezt a leolvasó bináris nullának vagy egynek értelmezi. A jeladó képes tárolni tehát digitális adatot, mint például egy azonosító szám.

Az RFID technológia sok iparágban használt. Például, az autógyártásban a gyártósoron elhelyezett leolvasókkal és a legyártott elemeken elhelyezett jeladókkal követni lehet a gyártási folyamatot, raktárakban követhetők a tárolt dolgok, vagy akár állatok azonosítására és mozgásuk követésére [29]. Használják még boltokban, amikor a termékeket látják el jeladókkal a lopások meggátolása érdekében.

### RFID frekvenciasávok

Az RFID technológiával kapcsolatban három frekvenciatartománnyal találkozhatunk: az LF, HF vagy UHF. Ezek az alacsony frekvenciát, a magas frekvenciát és az ultramagas frekvenciát jelentik.

Manapság az ipari alkalmazásokban a legelterjedtebb az UHF, a magas átviteli sebesség és a nagy hatótávolság miatt [28]. Ultramagas frekvencia használata esetén azonban ügyelni kell arra, hogy az RFID címkék nem takarhatók el fémmel, mert akkor nem érzékelhetők.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Frekvenciasáv | Hatótávolság | Zavarhatóság (fém) | Átviteli sebesség | Irányítottság |
| LF: 120 - 150 KHz | kevesebb, mint 1 m | erős | alacsony | nem szükséges |
| HF: 13.56 MHz | kb. 1.5 m | erős | közepes | nem szükséges |
| UHF: 433 MHz | kb. 20 m | nagyon erős | magas | szükséges |

*2. táblázat*

## Felhasznált hardvereszközök

Ebben a fejezetben bemutatom milyen eszközöket használtam az okosotthon rendszerem megépítéséhez.

### Espressif ESP32

Az ESP32 egy alacsony költségű, alacsony fogyasztású egylapkás (angolul: system on chip, röviden SoC) mikrokontroller család beépített Wi-Fi és dual-mode Bluetooth támogatással. Az ESP32 mikrokontrollerek rendelkezhetnek egy Tenselica Xtensa LX6 egy- vagy kétmagos mikroprocesszorral, vagy egy Tenselica Xtensa LX7 kétmagos mikroprocesszorral, amelyek 32 bitesek [21]. 320 kilobájt RAM-al, és 448 kilobájt ROM-al rendelkezik.

Az ESP32 mikrokontrollereket az Espressif Systems kínai cég fejleszti és a TSMC (Taiwanese Semiconductor Manufacturing Company Limited) gyártja. Az utódja az ESP8266 mikrokontrollernek.

A képen szöveg, elektronika, áramkör látható

Automatikusan generált leírásAz ESP32-t kifejezetten hordozható eszközökhöz és IoT alkalmazásra fejlesztették. Az Espressif szerint az EPS32 képes megbízhatóan funkcionálni ipari körülmények között is. Az üzemi hőmérséklet -40°C-tól 125°C-ig terjed. Az ESP32 802.11 b/g/n szabványú Wi-Fi kapcsolattal rendelkezik [23].

6. ábra ESPWROOM-32 modul ESP32-D0WDQ6 chip-el

Periféria interfészek [21]:

* 34 programozható általános felhasználású be- és kimenet (angolul: general purpose input/output, röviden GPIO), amiket felhasználhatunk szenzorokkal, más eszközökkel való kommunikációra
* SPI interfészek, azaz Serial Peripheral Interface, ami egy szinkron soros kommunikációs interfész specifikáció.
* I2C (Inter-Integrated Circuit) interfészek – szintén soros kommunikációs busz.
* I2S interfészek – egy digitális soros busz interfész standard, amit digitális audió eszközök összekötésére használnak.
* UART (Universal Asynchronous receiver-transmitter) – aszinkron soros kommunikációra szolgáló eszköz
* PWM (angolul: pulse width modulation) – impulzusszélesség-moduláció
* Hall effektus szenzor, ami képes detektálni a mágneses tereket.
* Kapacitív érintés érzékelők

Az ESP32 fejlesztésénél nem csak a sok hasznos funkcióra és sokoldalú használatra helyeztek hangsúlyt, hanem a biztonságra is. A következő biztonsági funkciókkal rendelkeznek [21]:

* Az összes IEEE 802.11 standard biztonsági protokoll támogatott, mint a WPA, WPA2, WPA3.
* Flash titkosítás, ami arra szolgál, hogy titkosítsuk az ESP32 chip-en kívüli flash memóriát [25].
* Secure boot, azaz biztonságos bootolás, ami meggátolja, hogy a saját kódomon kívül más kód fusson a chip-en. A flash memóriából betöltött adat ilyenkor minden induláskor ellenőrizve van.
* Kriptográfiai hardveres gyorsítás: AES, SHA-2, RSA, elliptikus görbe kriptográfia, véletlenszám generátor.

A Secure boot és a flash memória titkosítás egymástól függetlenül és működhet, de egy biztonságos környezethez mindkettőt használatba kell vennünk. [24] [25]

A Secure Boot használata eseten két 256 bites kulcs generálódik, a „secure bootloader key” és a „secure boot signing key”. A „secure signing key” egy kulcspár, amelynek a publikus kulcsa belefordítódik a szoftver bootloader-ébe és a boot egy részfolyamatának a megerősítésére szolgál [17]. A privát kulcsot titokban kell tartanunk. Két opciónk van, amikor a Secure Bootot használunk. Az egyik az úgynevezett „One-time Flash”, azaz egyszeri írás, amikor a továbbiakban nem lehet új programot írni a mikrokontrollerre, ilyenkor a kulcsok sosem tárolódnak a chipen kívül. Ez az opció kész termék esetén ajánlott. A másik opció a „Reflashable”, amikor hozzáférünk a kulcsokhoz, és ezzel a módszerrel tudunk újból írni a mikrokontrollerre.

Amikor a flash titkosítás be van állítva, a firmware, amit írunk a mikrokontrollerre, egyszerű szövegként íródik fel, majd az első indulásnál titkosítódik az adat. Ennek eredményeként a fizikai kiolvasása a mikrokontroller flash memóriájának sem vezetne sikerre, hiába próbálnánk megtudni milyen adatok vannak benne, mivel csak a titkosított szöveggel találnánk szembe magunkat.

Az ESP32 alapú mikrokontrollerek nemcsak sok lehetőséggel és nagy eszköztárral rendelkeznek, hanem költséghatékonyak is.

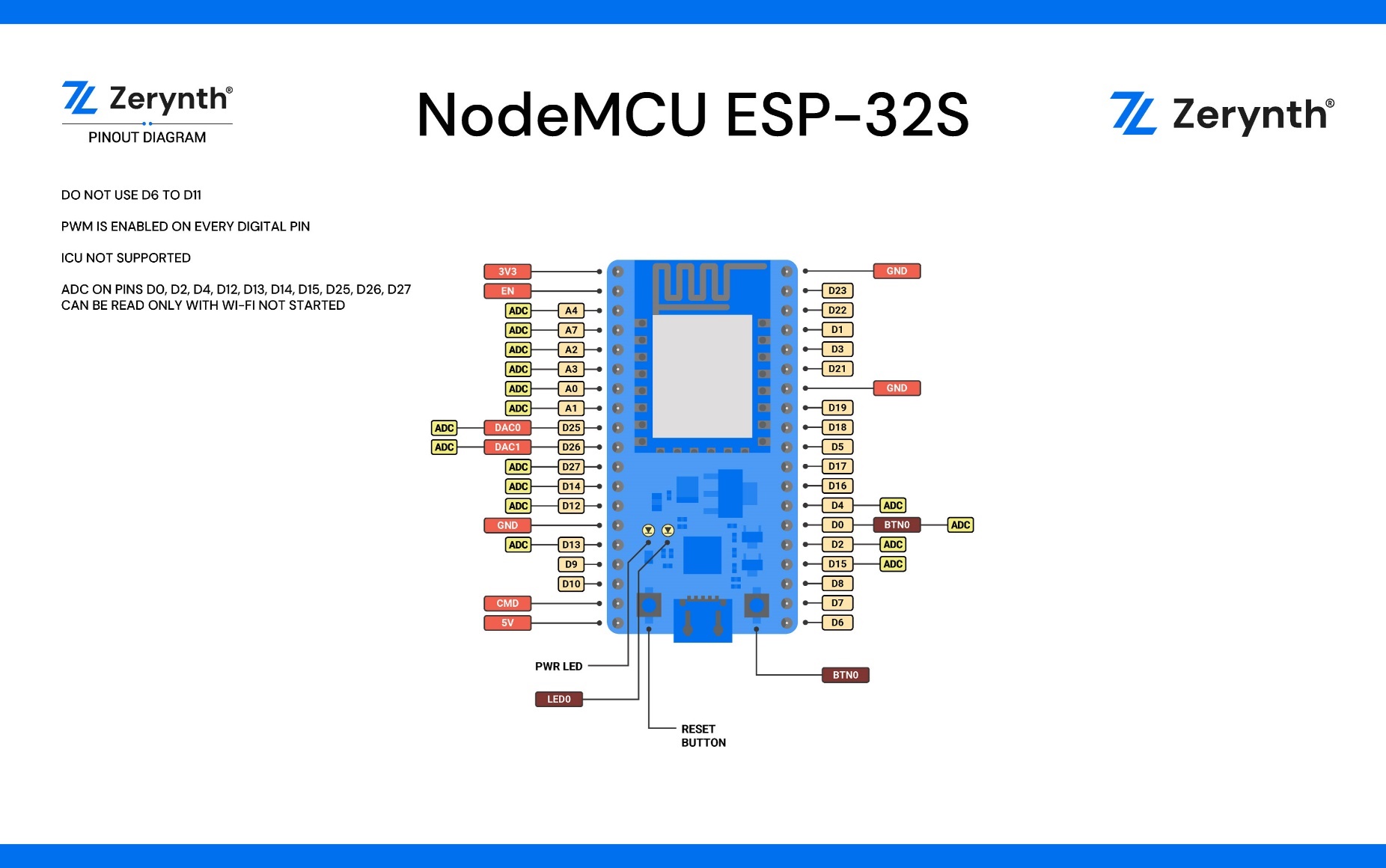
A projektemben az egyik EPS32 chipel rendelkező mikrokontroller a NodeMCU ESP32S, ami egy ESP-WROOM-32 modullal rendelkezik.

A képen elektronika, áramkör látható

Automatikusan generált leírás

7. ábra NodeMCU ESP32S

A fejlesztői board-on lévő EPS-WROOM-32 egy ESP32-D0WDQ6 chippel van ellátva. Ennek két processzormagja van, amelyek önállóan vezérelhetőek. A processzor órajele állítható 80MHz-től 240MHz-ig. A chipben van egy alacsonyfogyasztású társprocesszor is, amely a főprocesszor helyett használható olyan feladatok ellátása esetén, amelyek nem igényelnek nagy számítási kapacitást. Az eszköz támogatja a Wi-fi, Bluetooth és egyéb technológiákat, amelyeket más ESP32 alapú eszközök is.



8. ábra NodeMCU ESP-32S pin kiosztása [19]

A fenti ábrán látható az eszköz pin kiosztása. Ezek segítségével tudunk a chippel kommunikálni. Képes rajtuk keresztül analóg jelet digitálissá alakítani (ezek a képen ADC feliratú pinek), vagy akár fordítva.

A másik ESP32 eszköz, amit használtam az ESP-32CAM. Az egy olyan EPS32 modul, amelyhez tudunk egy kamerát csatlakoztatni. Az OV2640 és OV7670 típusú kamerákat támogatja. A projektemben felhasznált eszköz egy OV2640 kamerával rendelkezik, ami 2 Megapixeles. Az eszköz rendelkezik micro SD kártya foglalattal, így bővíthetjük a tárhelyét, ahol tárolhatunk képeket, videókat.

A képen elektronika, áramkör látható

Automatikusan generált leírás

9. ábra ESP-32CAM eszköz csatlakoztatott kamerával

Az ESP32 eszközök támogatják az Over The Air (OTA) frissítéseket, amely azt jelenti, hogy nem kell fizikailag hozzáférnünk az eszközhöz, hogy frissítsük a programkódot. Ilyenkor van lehetőségünk használni az application rollback funkciót is, ami arra hivatott, ha az új alkalmazásban kritikus hibák vannak.

### BME280-M kombinált szenzor

A BME280 egy digitális páratartalom, hőmérséklet és nyomás mérő szenzor. A BME280-M az ezt a szenzort tartalmazó modul neve. A szenzort a Bosch fejlesztette és gyártja. Rendelkezik I2C és SPI interfésszel. Az eszköz 1,71 V és 3,6 V közötti feszültséggel működőképes. A páratartalom és a nyomás érzékelő külön külön be- és kikapcsolható.

Működési intervallumok:

|  |  |  |
| --- | --- | --- |
|  | Minimum | Maximum |
| Hőmérséklet | -40 | +85 |
| Nyomás | 300 hPa | 1100 hPa |
| Páratartalom (relativ) | 0 % | 100 % |

*3. táblázat*

A képen szöveg, elektronika látható

Automatikusan generált leírásA képen szöveg, elektronika látható

Automatikusan generált leírásMaga a szenzor rendkívül kompakt, területe 2,5 \* 2,5 mm2, magassága 0,93 mm. A páratartalom érzékelő gyors válaszidőt biztosít az alkalmazásokhoz széles hőmérsékletskálán. A nyomásmérő, úgy, mint a páratartalom érzékelő is, magas pontosságot ígér. Ezek felhasználják a hőmérsékletszenzort értékeik pontosítására. A hőérzékelő felhasználható a környezeti hő meghatározására is. Alkalmazási módjai lehetnek: fitness monitoring, azaz a kiszáradástól vagy magas hőről való figyelmeztetés; Dolgok Internete; GPS navigáció javítása (time-to-first-fix javítása); időjárás figyelés; otthon automatizálás. [30]

11. ábra BME280 szenzort tartalmazó modul

10. ábra BM80 szenzor [30]

### PIR mozgásérzékelő szenzor

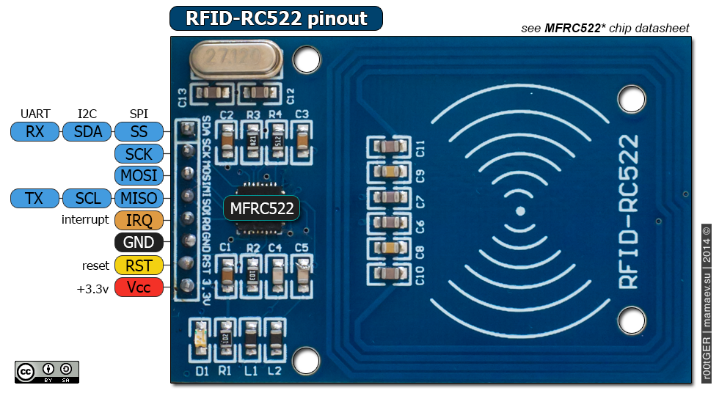
A passzív infravörös érzékelő (röviden PIR érzékelő) képes mérni a látómezőjében lévő tárgyakról érkező infravörös fényt. Ezeket a szenzorokat használhatjuk mozgásérzékelésre. A szenzor, amikor változás érzékel, a digitális kimenetét magasra állítja.

Gyakran alkalmazzák a PIR érzékelőket automata lámpák, riasztók esetén.

### MFRC522 RFID olvasó

Az MFRC522 egy RFID jelet olvasó modul. A modul három interfésszel rendelkezik: SPI, UART és I2C. A szükséges tápfeszültség 3,3V.

Támogatja az írást és olvasást 13,56 MHz-en. Képes az olvasó és író antennájával kommunikálni olyan jeladó eszközökkel, amelyek nem rendelkeznek aktív áramkörrel. Tehát olyan egységek olvasását és írását is képes elvégezni, amelyek nem igyényelnek tápellátást, mint például egy egyszerű RFID chippel ellátott kártya vagy úgynevezett RFID tag, amelyeket gyakran láthatunk beléptető rendszerek használatánál. [19]



12. ábra MFRC522 RFID olvasó

### P2520-12 Elektromágnes

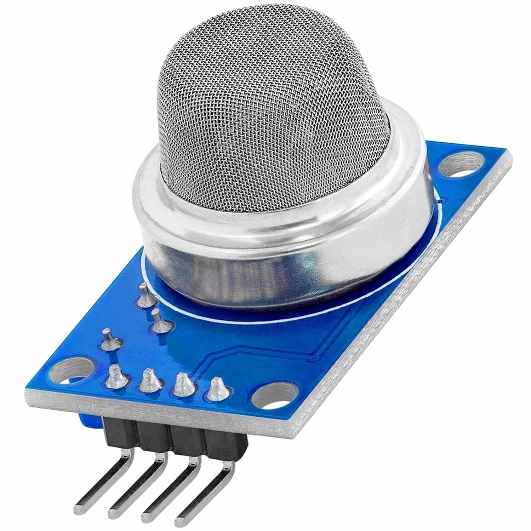
Az elektromos ajtó nyitó-záró megoldásoknál gyakran használnak elektromágnest. Ennek bemutatásához egy P2520-12 elektromágnest használtam, aminek a teherbírása 8 kilogramm, ezzel csak demonstrálni szeretném, hogy hogyan működik egy ilyen rendszer. Melyek be tudnak zárni egy ajtót, azok jóval nagyobb teherbírással rendelkeznek és méreteikben is nagyobbak.

### MQ-135-M levegő minőség érzékelő modul

Az MQ-135 egy komplex levegő minőség szenzor, ezt tartalmazza az MQ-135-M névvel jelzett modul.

A modulban lévő érzékelő alkalmas bizonyos gázok kimutatására a levegőben. Képes érzékelni többek között az ammóniát (NH3), benzol és alkohol gőzöket, széndioxidot, füstöt. Pontos koncentrátumot nem mér, viszont egy előre beállított érték fölött riaszt, ekkor a digitális kimenetén logikai egy jelenik meg. A riasztási küszöböt a modulon található trimmer potméterrel kell kalibrálni. A modul analóg kimenetén közvetlen elérjük a szenzor kimenetét. Tápfeszültsége 5V.

A szenzor első használatakor egy úgynevezett beégetési vagy előmelegítési folyamatot kell végrehajtani, amikor az érzékelőt 24-48 órára egy tiszta levegőjű helyiségben bekapcsolt állapotban hagyjuk. Ez egy egyszeri procedúra, viszont a szenzornak minden más bekapcsolásnál kell körülbelül fél perc, ameddig megfelelően felmelegszik. A szenzor tulajdonképpen a felülétén lévő ellenállás változását képes érzékelni, amelyet a rá becsapódó részecskék okozzák. [20]



13. ábra MQ-135-M modul

### SA-17 lángérzékelő modul

Az SA-17 egy lángérzékelő modul. 5V a bemenete és két kimenete van, egy analóg és egy digitális.

A modul egy infravörös szenzor segítségével érzékeli a lángot. Hátránya ennek, hogy ezt az érzékelőt nem csak lángra, hanem más infravörös jelre is érzékeny, mint például egy televíziónak a távirányítója.



14. ábra SA-17 lángérzékelő modul

A modult arra használom, hogy beépítve a mikrokontrollert és egyéb eszközt tartalmazó dobozba, jelezni tudjon, ha esetleg valamilyen elektromos hiba miatt tűz keletkezik.

## Összegzés

A 2. fejezetben bemutattam a technológiákat, megkíséreltem széleskörűen és érthetően körülírni azokat, ezzel szemléltetve, hogy a rendszerem hátterében a hardvereszközökkel együtt mik állnak és mire épül fel. Ezek és az ehhez hasonló technológiák és hardvereszközök képezik az okosotthon rendszerem alapjait, később a fejlesztés során ezeknek az összeségét felhasználva építettem meg az említett rendszert, és tettem működőképessé, a felhasználói felülettől a mikrokontrollerek firmware-jén át a szenzorokig.

# A fejlesztett rendszer és a felhasználói felület bemutatása

A következő fejezetben bemutatom, hogy az előző fejezetben részletezett technológiákból és eszközökből hogyan épül fel és működik az okosotthon rendszerem.

## A rendszer megtervezése, előkészítése

A rendszer megtervezésénél fontosnak gondoltam, hogy ne legyen túlkomplikált, mivel így könnyű kiküszöbölni a fejlesztés során felbukkanó esetleges hibákat, megtalálni az olyan akadályokat, amelyeket valamilyen okból nem tudunk áthidalni rendelkezésünkre álló megoldásokkal.

A tervezés legelején nem felhő alapú háttérszolgáltatásban gondolkodtam, mint a Firebase, amit végül a kész rendszer elkészítéséhez használtam, hanem egy olyan szervert képzeltem el, ami egy Raspberry Pi egykártyás számítógépen futott volna folyamatosan. Itt az első probléma, amibe ütköztem, hogy szerettem volna, hogy a rendszer távolról is elérhető legyen, a státuszinfók látszódjanak, a vezérlés lehetséges legyen ne csak az otthoni helyi hálózatról, amelyre az eszköz csatlakozik. Ahhoz, hogy ez lehetséges legyen, a Raspberry Pi portját meg kellene nyitnunk az Internet felé, ami biztonsági szempontból veszélyes tud lenni. Ahogy, említettem, nem szerettem volna ezt a rész túlkomplikálni, ezért más megoldást kellett keresnem. Ez ellen a megoldás ellen szólt az is, hogy a Raspberry Pi-nak a költsége magasnak számít a többi eszközhöz képest, míg a Firebase használata csak hosszú ideig tartó működés után kerülne annyiba, mint ez a számítógép.

Így jutottam végül arra a döntésre, hogy a rendszerem hátterében a Firebase szolgáltatásait fogom használni. Azért esett erre a választásom, mivel a biztonságot is könnyű volt vele megoldani. A Firebase segítségével elérhetővé válik a felhasználói felület bárhonnan, anélkül, hogy szükség lenne backend kódot írnom, így csak a frontend résszel kellett foglalkoznom. A felület jeleníti meg a felhasználó felé az állapotinformációkat, innen vezérelhetők a fények és az ajtó. Az alkalmazás az adatokat, amiket megjelenít a 2. fejezetben említett Firebase szolgáltatásokat használja. Amikor változás történik, például történik egy szenzoradat-feltöltés az ESP32 által, azonnal frissül, ez igaz, mind a kijelzett legutóbbi mérésre, vagy akár a hőmérséklet, páratartalom és nyomás grafikonokra. A mikrokontrollerek az adatokat a Firebase Realtime adatbázisba töltik fel közvetlenül, és innen olvassák azt, ha a felületről vezérlünk valamit, ugyanis a frontend-ről változtatott értékek is itt kerülnek frissítésre.

Mikrokontrollerek szempontjából szintén nem az ESP32 platform volt az első elképzeléseim között, hanem a RPI-PICO-W, amit a Raspberry gyárt, viszont ahonnan a többi szükséges eszközt szereztem be több hetet kellett volna várnom, hogy ahhoz a pár darabhoz hozzájussak amire szükségem lenne. Egy ilyen rendszer fejlesztése szempontjából a hardvereszközök kiválasztásánál tehát azt a tényezőt is figyelembe kell vennünk, hogy ezek mennyire elérhetőek a piacon. Mivel már volt szerencsém ESP32-vel dolgozni más fejlesztés során, elérhető volt és árban is kedvező volt számomra ezért úgy döntöttem megfelelő lesz számomra. Az ESP32 biztonsági szempontból is jó választásnak bizonyult a titkosítható flash memóriája miatt.

Próbáltam egy olyan rendszert építeni, ami egy felhasználó számára hasznos a mindennapi életben, megkönnyíti azt, illetve, hogy akár az okostelefonjáról vagy számítógépéről elérhetővé váljanak olyan adatok és lehetőségek, amelyek otthonával kapcsolatosak. Ezzel szerettem volna demonstrálni, hogy a modern világban egy ilyen rendszer hogyan tud működni biztonságosan és felhasználói szempontból egyszerűen.

A szenzorok és más eszközök kiválasztásánál szintén figyelembe kellett vennem elérhetőségüket megvásárlás során. Például a fejlesztés elején csak olyan szenzormodult tudtam beszerezni, amely csak hőmérsékletet és nyomást mér, később be tudtam szerezni a BME280-m modult, amely tud páratartalmat is, így végül ez került be a rendszerbe.

Költségek figyelembevétele miatt van olyan eszköz, amely valós környezetben nem állná meg teljes mértékben a helyét, ezt demonstrációs céllal alkalmazom a rendszeremben. Ilyen az elektromágnes, amellyel az ajtó nyitást és zárást oldom meg. Egy olyan elektromágneses ajtózár, amely képes ténylegesen megtartani egy ajtót többször drágább, mint amit én használtam, de ezen eltérésen kívül a működésük szinte azonos.

Az okosotthon rendszerek egyik előnye, hogy bővíthetők. Például ennél a rendszernél, ha szeretnénk a hőmérséklet alapján a fűtést állítani, kiegészíthető lenne egy olyan rendszerrel, ami ezt a funkcionalitást ellátja. Ennek tudatában terveztem a felhasználó felület kinézetét is, így, ha hozzáadódik későbbi fejlesztés során egy vagy több funkció, a felület eddigi kinézetén nem kell változtatni. Így úgymond a felület modulokra oszlik funkciónként. Ezt a MDB könyvtár design csomagja által tartalmazott kártyák teszik lehetővé. Ezt a könyvtárat amiatt választottam, mivel a Material Design-ra épül, amely modern letisztul és egyszerűen megérthető kinézetet biztosít az alkalmazásunknak, így a felhasználó könnyűszerrel megtanulja intuitívan a felület használatát. Ez a megközelítés véleményem szerint fontos egy felület kinézetének a tervezésénél, mivel kényelmesebbé is tesszük ezáltal az interakciókat, ezzel valamennyire ösztönözve a felhasználót az alkalmazásunk használatára.

Fejlesztői környezet szempontjából a már fentebb kifejtett Visual Studio Code integrált fejlesztői környezetet használtam a PlatformIO bővítménnyel kiegészítve. Egy másik elterjedt környezet mikrokontrollerek programozására az ArduinoIDE. Az eszközök, amelyekkel programozhatjuk és olvashatjuk a mikrokontrollerek kimenetét itt is elérhetőek, viszont ebből hiányoznak olyan funkciók, mint a kódkiegészítés vagy a hibakeresés, ezért döntöttem inkább a Visual Studio Code mellett, mivel ezzel fejlesztési időt tudtam spórolni. A könyvtárakat, terminálbeállításokat, és egyéb konfigurációkat a PlatformIO bővítmény által létrehozott projektkönyvtáron belül található *platformio.ini* fájlban kell megadnunk a megfelelő formában. A programkódot tartalmazó forrásfájlokat a szintén a projektkönyvtárban található */src/* alkönyvtárba kell helyeznünk.

Az ESP32 mikrokontrollerek programozása esetén van lehetőségünk két keretrendszert is használni. Az egyik az Espressiff által fejlesztett ESP-IDF, ami egy olyan fejlesztői csomagot biztosít, amely megfelelő bármilyen alkalmazás fejlesztésére az ESP32 típusú platformokon C és C++ nyelven. A másik az Arduino keretrendszer, amely szintén C/C++ nyelvre épül, mikrokontroller specifikus kiegészítésekkel. Az utóbbira esett a választásom, mivel ezzel már van korábbi tapasztalatom, és egyszerűbbnek találom, mint az ESP-IDF. Az ESP-IDF előnye azonban az, hogy ESP32 specifikusabb. Például a flash memória titkosítását és a Secure boot-ot az ESP-IDF segítségével tudjuk elvégezni. Az ESP-IDF-hez tartozik egy parancssoros kliens alkalmazás is, amelyek az előbb említettek megvalósításához és egyéb konfigurációkhoz is használhatjuk. [23] Egy egyszerű program így néz ki az Arduino keretrendszerben:

**#include** <Arduino.h>

**void** **setup**()

{

Serial.begin(**9600**);

}

**void** **loop**()

{

Serial.println("Hello world!");

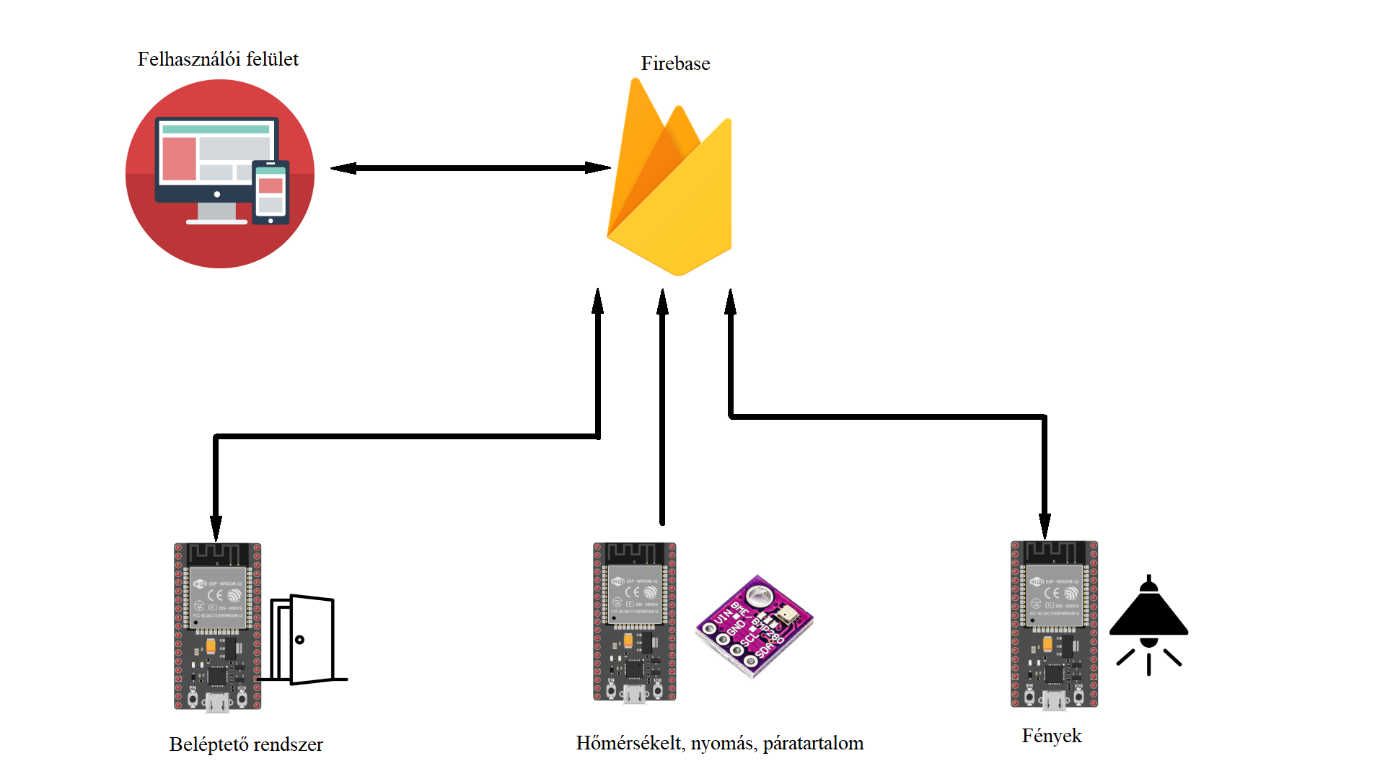
delay(**1000**);

}

Ez a program 9600 bit/másodperccel elindítja a soros kommunikációt majd egy másodpercenkét kiír a soros portra. A program *setup()* eljárása indulásnál egyszer fut le. A loop() eljárás folyamatosan hívódik meg, tehát addig fut míg az eszköz be van kapcsolva.

## A rendszer felépítése

A rendszer több részből tevődik össze. A felhasználói felületből, a Firebase által biztosított backendből, amihez tartozik a Realtime Database használata, az autentikáció és a Firebase Storage, a frontendből, amely pedig a felhasználói felületet jelenti, ez kommunikál a backend-del. Ezeken kívül tartalmazza a beléptető rendszert, a hőmérséklet, nyomás és páratartalom mérő egységet, a fényeket vezérlő egységet. Ezek továbbítják az adatokat a Firebase backend felé, vagy figyelik az adatbázisban történő változást és ez alapján cselekednek.

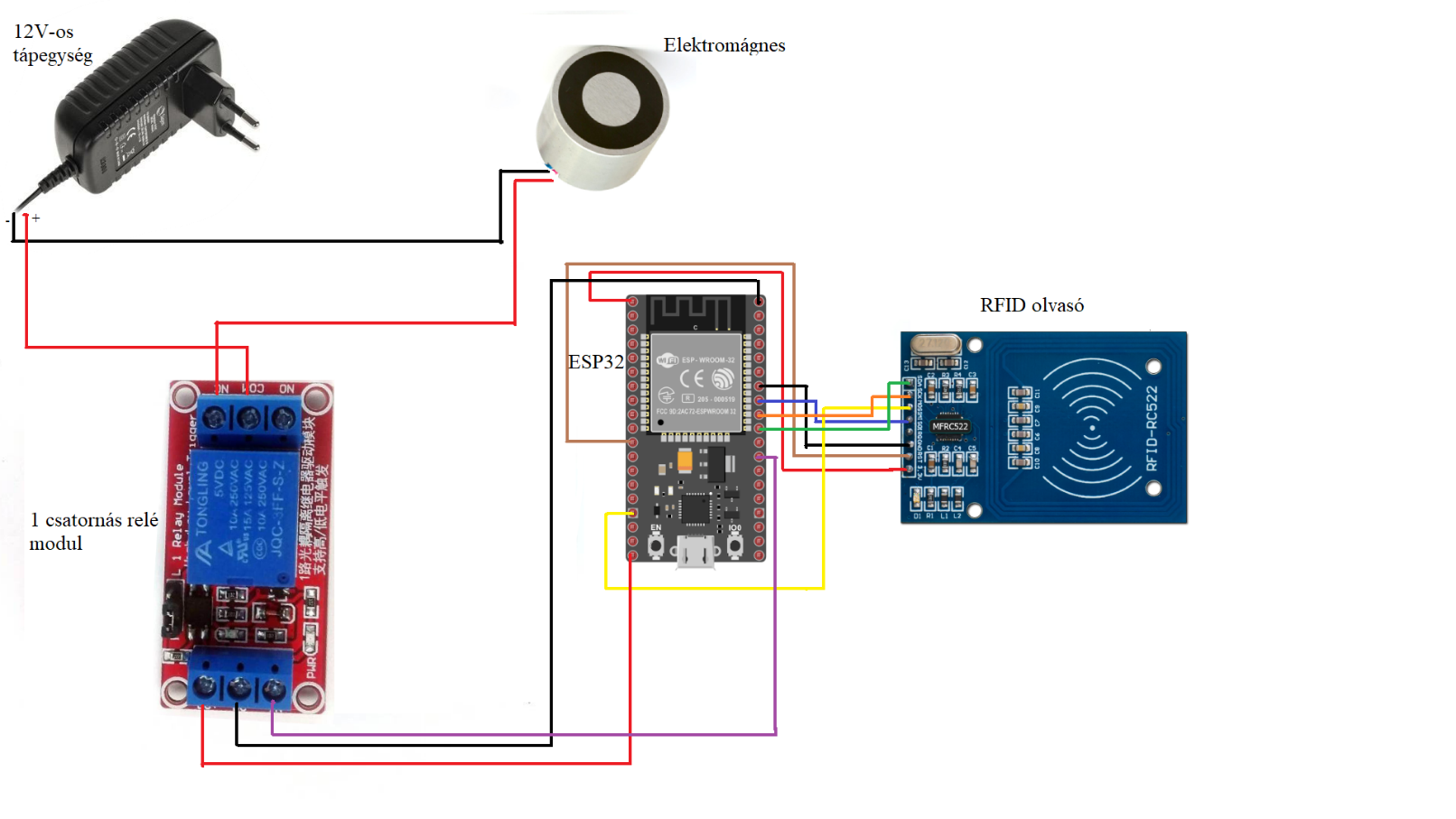


15. ábra A rendszer sematikus felépítése

Ezeket az egységeket szeretném bemutatni részletesen.

### Beléptető rendszer

A beléptető rendszert szeretném szintén több részletre felosztani. Az első része az a mikrokontroller, ami figyeli az adatbázisban történő változásokat, amelyek a felhasználói felületről történnek, ezeknek megfelelően cselekedik. Ez vezérli a beléptető rendszer többi elemét, amik: az elektromágnes, az RFID olvasó, és a ESP32 CAM kamerával ellátott mikrokontroller, amely, ha engedélyezve van a felhasználói felületről, a hozzá csatlakoztatott mozgásérzékelő jelzésére képeket készít, azokat feltölti a Firebase adatbázisába és küld egy emailt a felhasználónak a képekkel és időponttal.

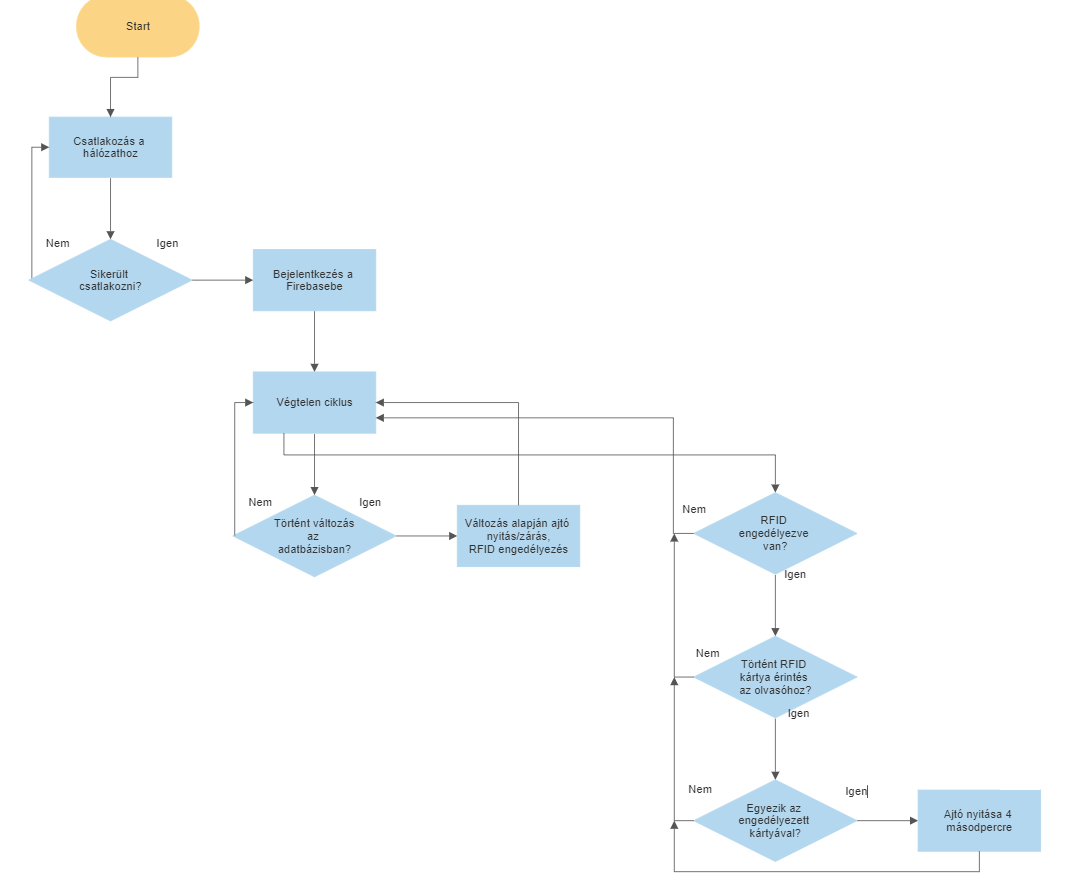


16. ábra Az elektromágnest és RFID olvasást vezérlő rész kapcsolási rajza

A fenti ábrán láthatjuk, hogyan kapcsolódnak az elektromágnest és az RFID olvasást kezelő egység részei. Láthatunk baloldalt egy 1 csatornás relé modult. Erre amiatt van szükség, mivel az elektromágnes tápfeszültsége 12V és ezt az ESP32 mikrokontroller nem tudja ellátni. A relé modul segítségével be és ki kapcsolható eszközöket tudunk vezérelni kevesebb feszültséggel, mint amennyire az eszköznek szüksége van. A relé tulajdonképpen egy programozható kapcsoló. A bemeneti részén (az ábrán a relé modul aljánál) találunk három csatlakozót, amelyek a következő feliratokkal vannak ellátva: DC+, DC- és IN. A DC+ csatlakozót 5V feszültséghez kell csatlakoztatnunk kell adnunk az eszköznek. A DC- csatlakozót földeléshez kell kötnünk. A harmadik IN feliratú csatlakozónál adjunk meg a bemenetet a relé modulnak, ez alapján kapcsol át. A kimeneti részén szintén 3 csatlakozó található NO, COM és NC feliratokkal. A COM a közös csatlakozót jelenti, az NO a „normally on”, az NC a „normally closed” rövidítése. A COM-ba kell bekötnünk a tápegység pozitívját. Az COM és az NO között akkor jön létre csatlakozás, ha a bemeneti IN csatlakozón 5V-ot vagy magas jelet adunk, ilyenkor van nyitva a relé. Ha 0V-t vagy alacsony jelet adunk, akkor nincs csatlakozás, az eszköz, ami be van kötve erre a kimenetre ki van kapcsolva. Az NC kimenet ennek ellentétes módon működik. Általában ez a kettő kimenet közül az egyiket használjuk, attól függően, hogy mikor szeretnénk, ha az eszköz bekapcsolna: ha adunk jelet vagy amikor nem adunk jelet.

A projektemben az NC módot használom, így, ha a mikrokontroller olyan utasítást kap a felhasználói felület felől vagy valaki az engedélyezett RFID tag-et vagy kártyát érint az olvasóhoz, akkor magasra állítja a kimenetet és kikapcsol az elektromágnes, tehát kinyílik az ajtó. Alapesetben alacsony a kimenet, az elektromágnes be van kapcsolva, az ajtó zárva.

A mikrokontroller induláskor megpróbál csatlakozni egy előre megadott Wifi hálózathoz, amint sikerült, bejelentkezik a Firebase-be szintén előre beállított API kulccsal és email cím jelszó párossal. Sikeres bejelentkezés után a Firebase visszaad egy egyedi azonosítót, ami a felhasználót azonosítja. Ez alá az egyedi azonosító alatt találhatóak az adatbázisban a vezérléshez szükséges adatok. Ha változás történik ezekben az adatokban a Firebase értesíti az eszközt erről. Ilyenkor egy úgynevezett callback függvény hívódik fel. Ez függvény megkapja az adatot, majd feldolgozza. A felületről gombnyomással tudjuk nyitni és zárni az ajtót, illetve egy kapcsolóval be tudjuk állítani, hogy engedélyezett az RFID-s belépés vagy nem. Ha engedélyezzük, akkor, ha közel rakunk egy kártyát vagy tag-et az RFID olvasóhoz, az olvasó jelez a kimenetén, hogy adatot kapott és kiküldi az olvasott értékeket, ezek bekerülnek egy bájt tömbbe. A program összehasonlítja a kapott adatot egy előre meghatározott bájtsorozattal. Ha egyezés van, akkor engedélyezett a belépés, ilyenkor négy másodpercre kikapcsolja az elektromágnest, majd visszakapcsolja. Ha nem egyezik, akkor nem történik semmi.



17. ábra Ajtó nyitó program folyamatábrája.

A fenti ábrán látható a program leegyszerűsített folyamatábrája, szemléltetve az azelőtt leírt működést.

A programban az RFID kezelést az MFRC522 olvasóhoz tartozó ESP32 kompatibilis könyvtárat használtam, ami olyan kész függvényekkel segíti a fejlesztést, mint például a *PICC\_IsNewCardPresent()*, ami, ahogy a neve is sugallja, igaz értéket ad vissza, ha az olvasó RFID tag-et érzékel és negatívat, ha nem. A program indításakor inicializálódik az SPI interfész, ezt használja majd az olvasó, amit a *PCD\_Init()* eljárással tudunk inicializálni.A beléptető rendszer alá sorolom a kamerával rendelkező ESP32 modult is. Az ESP32 CAM modul szintén csatlakozik induláskor a hálózathoz. A mikrokontrollerhez csatlakoztatva van egy mozgásérzékelő szenzor. A felhasználói felületről egy kapcsolóval állítani tudjuk, hogy a kamera készítsen-e felvételeket, amikor a mozgásérzékelő jelez és ezeket feltöltse a Firebase Cloud Storage adatbázisába, majd elküldje őket emailben. A készített képeket microSD kártyára is időbélyeggel menti. A mozgásérzékelő és a mikrokontroller összeköttetése viszonylag egyszerű, a modulnak 3 csatlakozója van: egy tápbemenet, amelyet 3,3V-al láttam el, egy földelés és a kimenet, amelyen digitális magas jelet ad, ha érzékel. Ezen a kimeneten figyelem a változást, ha a felhasználói felületről engedélyezve van.

Az email küldést az ESP Mail Client könyvtárral oldottam meg, amely forráskódja megtalálható az egyik legnépszerűbb online kód és verziókezelő oldalon a Github-on. A fejlesztője ugyanaz, aki a Firebase könyvtárat is készítette az ESP32-höz. A könyvtár lehetővé teszi, hogy csak szöveges vagy akár csatolmányokat is tartalmazó emaileket küldjünk az ESP32 mikrokontrollerekről SMTP (Simple Mail Transfer Protocol) vagy IMAP (Internet Message Protocol) szerver segítségével. A programomban a Gmail szolgáltató SMTP szerverét használtam, amelyhez regisztráltam egy új felhasználót. Ahhoz, hogy az ESP32 tudjon emaileket küldeni ezzel a Gmail felhasználóval, generáltatnunk kell egy úgynevezett App Passwordot, azaz alkalmazás jelszót, amely 16 karakterből áll. Ahhoz, hogy ezt megtegyük be kell kapcsolnunk a kétfaktoros azonosítást a fiókbeállításoknál. Ez az App Password mindenképp szükséges, hogy tudjunk emailt küldeni. A programunkban a könyvtár által felkínált módon megadjuk a Gmail SMTP szerver adatait, bejelentkezési adatainkat, és így képesek vagyunk a kijelölt célcímre üzeneteket küldeni.

### Környezeti szenzorrendszer

A környezeti szenzorrendszer szintén tartalmaz egy mikrokontrollert, ami fogadja a hozzá csatlakoztatott szenzormodulok értékeit. Ezeket az értékeket később továbbítja a Firebase adatbázis felé. A szenzorrendszereket az előző fejezetben már bemutattam.

Az egyik rendszer, amelyet tartalmaz, az a BME280-M modul. A modul három szenzort tartalmaz: hőmérséklet, páratartalom és nyomás. A szenzormodullal való munkához az Adafruit Sensor és Adafruit BME280 nevű könyvtárakat használtam. A BME280-M I2C kommunikációt használ. A program induláskor inicializálja az I2C intrefészt. Ezután csatlakozik a WiFi hálózathoz hasonlóan a beléptető rendszer programjához, majd sikeres csatlakozás után beállítja az időt, majd bejelentkezik a Firebase-be és elvégzi a szükséges konfigurációkat. Ekkor a Firebase visszaad egy egyedi azonosítót, az olvasott szenzoradatokat ez alá az azonosító alá írja majd. A mikrokontroller 30 percenként olvas adatot és tölti fel. Az adatokat időbélyeggel látja el, ez alá kerül majd a hőméréklet, nyomás és páratartalom érték. A Firebase Realtime Databaseben következőképen néz ki:

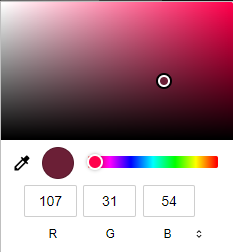


18. ábra A Firebase adatbázisban a BME280 szenzoradatok hierarchiája

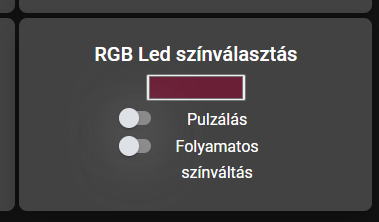
### Fénykezelő rendszer

A fénykezelő rendszerben a fogadja az utasításokat a felülettől, hogy lámpákat kapcsolja fel vagy le. Ezeket az értékeket az ESP32 szintén a Realtime Database-ben törénő változások okán kapja meg.

A rendszer nemcsak a ki és bekapcsolható fényeket, hanem egy programozható RGB LED szalagot is, amelynek pontos típusa WS2812B. Az RGB a red green blue, azaz a piros zöld és kék rövidítése, ami arra utal, hogy a szalagon található LED-ek ennek a három szín kombinációjából állít össze minden más színt. Programozhatósága abból ered, hogy minden LED-et a szalagon külön-külön tudunk színt és fényerőt állítani. A felhasználói felület lehetőséget ad, hogy beállítsunk egy folyamatos színt, pulzálást, vagy folyamatos színváltást, amely végighalad a színskálán.



19. ábra Szíválasztó bemenet



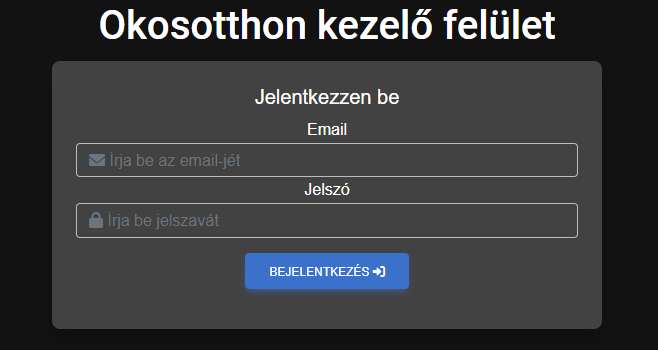
20. ábra RGB Led színválasztó elem a felhasználói felületen

A további fények egyszerű kapcsolókkal vezérelhetők a felületről.

### Felhasználói felület

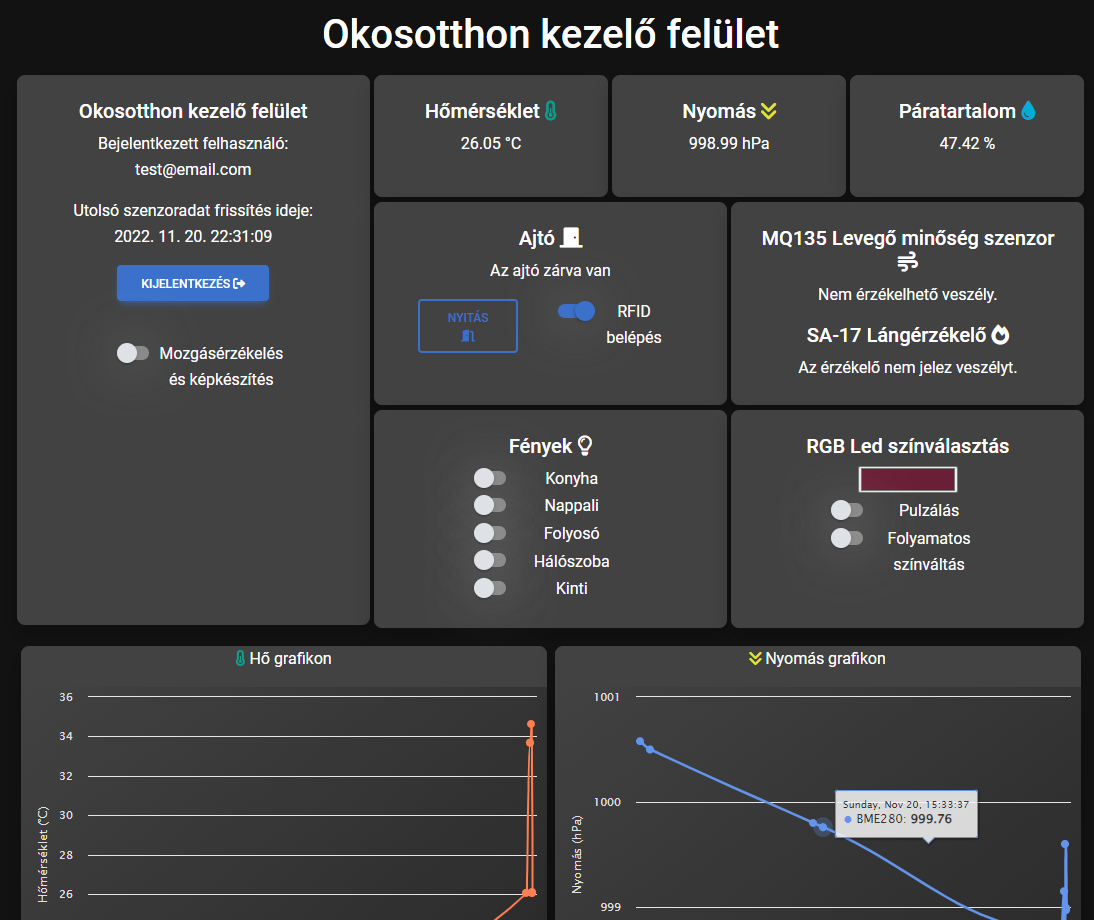
A felhasználói felületet olyan stílusban szerettem volna megvalósítani, annak érdekében, hogy kézreálló és egyszerű legyen a használója számára. Ezt lehetővé tették azok az elemek, amelyek az MDB könyvtár biztosított számomra. A felületet igyekeztem úgy elkészíteni, hogy egyaránt asztali környezetben nagyobb képernyőről, vagy akár mobileszközről is teljesen használható legyen.

A felhasználót első használatkor egy bejelentkező panel fogadja, ahol meg kell adnia email címét és a jelszavát.



21. ábra Bejelentkező panel

Bejelentkezés után válik elérhetővé az okosotthon kezelőfelülete. Egy rácsrendszert alkottam kártyákból, amelyek megjelennek a különböző szenzoradatok, az ezen adatokból készül grafikonok, gombok és kapcsolók.



22. ábra Felhasználó felület bejelentkezés után

Ahogyan látható, a felület bal oldalán látható nagyobb elemben találunk pár adatot, mint a belépett felhasználó email címe, és hogy mikor történt az utolsó feltöltése a BME280-M modul szenzoradatainak. Ezek az adatok legfelül jelennek meg. Ez alatt találhatóak az ajtó és fénykezelő panelek, illetve a levegőminőség szenzor és a lángérzékelő visszajelzései.

Legalul található a három grafikon, amelyekhez a Highcharts JS könyvtárat használtam. A Highchats segítségével könnyedén tudtam grafikonokat rajzolni és személyreszabni őket, hogy illedjenek az alkalmazáshoz. A három grafikon a hőmérséklet, a nyomás és a páratartalom utolsó tíz feltöltött értékét ábrázolja, tehát az utóbbi 5 órában mért mennyiségeket.

### Firebase Realtime Database és Cloud Storage biztonsági szabályok

A Firebase adatbázisai védelmére a biztonsági szabályok szolgálnak, melynek alapjait a 2. fejezeben kifejtettem. Annak érdekében, hogy a felhasználó csak a saját adataihoz férjen hozzá, illetve, hogy más se férjen hozzá jogosulatlanul bármilyen adathoz az adatbázisban a következő szabályokat alkalmaztam:

A képen szöveg látható

Automatikusan generált leírás

23. ábra Alkalmazott biztonsági szabályok

Ezek a szabályok azt jelentik, hogy csak az az egyedi azonosító alatt tud olvasni és írni adatot, amely megfelel a sajátjának. Így bejelentkezetlen felhasználó sem tud hozzájuk férni, akár a webes felületről, akár http kérésekkel próbálkozik.

A Cloud Storage esetén hasonló szabályokat alkalmaztam, szintén ezzel az elvvel, hogy csak a saját egyedi azonosító alatt lévő adatok az elérhetőek.

# Összegzés

Ezeknek a mikrokontrollereknek a programozása, hasonlít arra, mintha más szoftvereket írnánk, de bizonyos dolgokban el tud térni. A beágyazott rendszerekre történő fejlesztés olyan kihívásokat von maga után, amelyek például egy web- vagy egy asztali alkalmazás esetén nincsenek. Ami hiányzott a fejlesztés során az, amit más helyen megszokottnak tekinthető hibakereső használata, amely meg tudja állítani azon a ponton a programot, ahol szeretnénk vagy akár soronként végighaladni rajta egyesével, közben felfedve az esetleges hibákat. Az ESP32 esetében ez csak egy külön erre fejlesztett hardvereszköz csatlakoztatásával lehetséges, amelyhez nekem nem volt hozzáférésem. Ez talán egy kicsit megnehezítette a mikrokontrollerek fejlesztési folyamatát és az esetlegesen felmerülő hibák elhárítását. A másik aspektus a beágyazott rendszer fejlesztésénél, amely eltér például az asztali alkalmazásoktól, az az, hogy a rendszert meg is kell építeni, össze kell kötni vezetékekkel, megtervezni figyelmesen, hogy melyik kábel, melyik csatlakozóra megy, mert ha véletlenül rosszul csináljuk, tönkre is tehetjük eszközünket. Egy eszközömet korábban már hasonló figyelmetlenségből sikerült is használhatatlanná tennem. A titkosítás és biztonsági lépések során szintén véthetünk olyan hibákat, amelyek nem, vagy csak nagyon nehezen visszafordíthatók. Erre is fokozott figyelmet kell fordítani. Ezen nehézségek tudatában is érdekesnek tartom az ezekkel történő dolgozást, mert olyan kihívást tár elém, amelyet, ha sikerül áthidalni igazi heuréka érzés tud lenni.

A weblap fejlesztés számomra viszonylag idegen terület volt számomra. Össze kellett raknom egy olyan webalkalmazást, amely nem csak designos, valamilyen úton módon képes kommunikálni az IoT rendszerrel. Ennek a segítségére vált a Firebase, aminek a használata során megértettem, hogy a modern szoftverfejlesztésben miért olyan elterjedt ilyen vagy ehhez hasonló felhőszolgáltatásokat használnia projektünk hátterében. Meglepően egyszerű volt elkezdeni vele a munkát, részletes dokumentációja révén. Szintén könnyen tudtam a mikrokontrollerekkel is összekapcsolni a Firebase projektemet, ami pedig jóval megrövidítette azt az időt, amelyet az alkalmazás háttérben lévő részének fejlesztésével töltöttem volna, így tudtam több figyelmet fordítani a mikrokontrollerekre, illetve a felület funkcionalitására. A Firebase biztonsági szempontból is segítette munkámat, mivel viszonylag kevés kóddal is effektív védelmet biztosít adatainknak. A Firebase segítségével áll össze egy teljessé a rendszerem, mivel összeköti a webalkalmazást és a mikrokontrollereket.

A fejlesztés alatt sikerült elmélyítenem az előző bekezdésekben említett témákkal kapcsolatban. A rendszer jelenlegi állapotában stabilan képes működni, és ellátni minden tervezett funkcióját. Ennek ellenére még nagyon sok potenciális fejlesztést lehetne eszközölni ehhez az okosotthon rendszerhez. Bővíthető lenne még sok más funkcióval, új szenzorokkal, új aktív rendszerekkel, hiszen a Firebase képes ennél meg jóval több adatot kezelni, mikrokontrollerek és modulok lehetőségének tárháza pedig szinte végtelen, szinte csak a kreativitás és az anyagiak szabhatnak határt a fejlesztésben.

Az okosotthon fejlesztése során rengeteg pozitív élménnyel gazdagodtam és rengeteg olyan tudást szereztem, amelyet később hasznosíthatok. Ez erősítette azt az érzést bennem, hogy később akár hobbiként, vagy akár szakmámban is szeretnék beágyazott rendszerekkel vagy hasonlóval foglalkozni.

Az projekt kódjai elérhetőek a következő linken: <https://github.com/gab0sz/SzaniszloGaborSzakdolgozat>

# Irodalomjegyzék

1. Biljana L. Risteska StojkoskaKire V. Trivodaliev, 2017: *A review of Internet of Things for smart home: Challenges and solutions;*

Journal of Network and Computer Applications, DOI:10.1016/j.jnca.2017.08.017

1. Zhi-Kai Zhang, Michael Cheng Yi Cho, Chia-Wei Wang, Chia-Wei Hsu, Chong-Kuan Chen, Shiuhpyng Shieh, 2014: *IoT security: ongoing challenges and research opportunities;*

IEEE, ISBN: 978-1-4799-6833-6

1. Laurence Moroney, 2017: *Definitive Guide to Firebase;*

California: Apress, ISBN: 978-1-4842-2943-9

1. Agus Kurniawan, 2019: *Internet of Things Projects with ESP32: Build exciting and powerful IoT projects using the all-new Espressif ESP32.;*

Packt Publishing Ltd

1. Ron Weinstrein, 2005: *RFID: a technical overview and its application to the enterprise*

IT professional

1. Avast, 2019: *Smart Home Security Report*  <https://cdn2.hubspot.net/hubfs/486579/avast_smart_home_report_feb_2019.pdf> (Letöltve 2022.09.30.)
2. Firebase Hosting Dokumentáció

<https://firebase.google.com/docs/hosting> (Letöltve 2022.11.06.)

1. Firebase Realtime Database Dokumentáció

<https://firebase.google.com/docs/database> (Letöltve 2022.11.06)

1. 5G and its impact on the Internet of Things

<https://www2.stardust-testing.com/en/5g-and-impact-on-iots> (Letöltve 2022.11.07)

1. A look at examples of IoT devices and their business applications in 2022

<https://www.insiderintelligence.com/insights/internet-of-things-devices-examples/> (Letöltve 2022.11.07)

1. Mit jelent, ha „intelligens” egy épület?

<https://www.iasautomatika.hu/intelligens-epulet> (Letöltve 2022.11.07)

1. IoT security (internet of things security)

<https://www.techtarget.com/iotagenda/definition/IoT-security-Internet-of-Things-security> (Letöltve 2022.11.08)

1. Understand Firebase Realtime Database Security Rules

<https://firebase.google.com/docs/database/security> (Letöltve 2022.11.11)

1. Use conditions in Realtime Database Security Rules

<https://firebase.google.com/docs/database/security/rules-conditions> (Letöltve 2022.11.11)

1. Public Key Infrastructure

<https://learn.microsoft.com/hu-hu/windows/win32/seccertenroll/public-key-infrastructure?redirectedfrom=MSDN> (Letöltve 2022.11.14)

1. JavaScript Usage Statistics to Watch Out for in 2022

<https://radixweb.com/blog/top-javascript-usage-statistics> (Letöltve 2022.11.14)

1. Material Design for Bootstrap 5 & Vanilla JavaScript

<https://mdbootstrap.com/docs/standard/> (Letöltve 2022.11.14)

1. Highcharts hivatalos weboldal

<https://www.highcharts.com/> (Letöltve 2022.11.14)

1. MFRC522 Datasheet

<https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf> (Letöltve 2022.11.15)

1. MQ-135 – Gas Sensor for Air Quality

<https://components101.com/sensors/mq135-gas-sensor-for-air-quality> (Letöltve 2022.11. 14)

1. ESP32 Technical Reference Manual

<https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf> (Letöltve 2022.11.15)

1. ESP32 Series Datasheet

<https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf> (Letöltve 2022.11.15)

1. Espressif hivatalos weboldal

<https://www.espressif.com/en/products/socs/esp32> (Letöltve 2022.11.15)

1. API Guides: Secure Boot

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/secure-boot-v1.html#secure-boot-image-signing-algorithm> (Letöltve 2022.11.15)

1. ESP32-CAM

<https://www.hestore.hu/prod_10041737.html> (Letöltve 2022.11.15)

1. API Guides: Flash Encryption

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/flash-encryption.html> (Letöltve 2022.11.15)

1. NodeMCU ESP32

<https://esphome.io/devices/nodemcu_esp32.html> (Letöltve 2022.11.15)

1. Mi is az az RFID-technológia? Meghatározás, jelentés, működés

<https://l-mobile.com/hu/industrie-40/mi-az-rfid-technologia-meghatarozas-jelentes-mukodes/> (Letöltve 2022.11.14)

1. Wildlife animal tracking using RFID and GSM technology

<https://www.ijser.org/researchpaper/WILDLIFE-ANIMAL-TRACKING-USING-RFID-AND-GSM-TECHNOLOGY.pdf> (Letöltve 2022.11.15)

1. BME280: Combined Humidity and Pressure Sensor

<https://cdn.sparkfun.com/assets/e/7/3/b/1/BME280_Datasheet.pdf> (Letöltve 2022.11.15)