Matematikai és Informatikai Intézet

Okos otthon hub és irányítóközpont

Készítette
Lovász Ákos
Programtervező informatikus BSc

Témavezető
Dr. Tajti Tibor
Egyetemi adjunktus

Eger, 2021

Tartalomjegyzék

Bevezetés

Tanulmányaim folyamán számos technológiával ismerkedtem meg, melyek mindegyike rengeteg lehetőséget tárt fel előttem, viszont a szakmai gyakorlatom során kiemelkedően megragadta a fantáziámat az Andoid fejlesztés és a hardverprogramozás összekapcsolása által kialakult rendszerek lehetősége.

Az Android alkalmazások fejlesztése iránt mindig is érdeklődtem, egy-egy kisebb alkalmazást gyakorlásként már készítettem ezt megelőzően, de komolyabban itt kezdtem vele foglalkozni, megismerkedni a vele járó sajátosságokkal.

Az ilyen jellegű eszközök kapcsolata és kommunikációja már korai gondolataimban is az okos otthonok felépítésére emlékeztetett, ezért is gondoltam megfelelő táma választásnak.

A döntést követő kutatás során szembetűnő hátránya volt az okos otthon rendszereknek, hogy a legtöbb „márkás" megoldás elsősorban drága és csak felületes hozzáférést tesznek lehetővé, melyet teljes mértékben a rendszer gyártója határoz meg.

Az alternatív, olcsóbb rendszerek bár nyíltabb hozzáállással próbálnak előnyt szerezni, viszont sokszor erősen a technikai oldalába mélyednek, így egy átlagos felhasználónak bonyolultnak, nehezen kezelhetőnek tűnhetnek. Ezen felül gyakran futhatunk olyan problémába, hogy az általunk választott rendszerben lévő hiányosságokat csak más gyártótól származó eszköz nyújtana megoldást, viszont különböző gyártók eszközei nagyon ritkán kompatibilisek egymással.

Ezeket az észrevételeket figyelembe véve egyértelműnek tűnt, hogy van lehetőség egy olyan rendszer kivitelezésére, ami elsősorban olcsóbb, de ugyanakkor nem túlbonyolított, felhasználóbarát marad. Fontos a nyitottság, a bővíthetőség, és a széleskörű kompatibilitás lehetősége, hogy a felhasználó biztos lehessen abban, hogy a jövőben felmerülő hiányosságok egyszerűen pótolhatók.

1. fejezet

A rendszer alapjai

A rendszer két fő komponensből áll. A kiszolgáló, mely egy Orange Pi Zero egykártyás számítógép, amin fut a Node-RED, egy olyan webes felületet biztosító szolgáltatás, mely grafikusan kezelhető komponensek összekapcsolásával teszi lehetővé a renszer működését befolyásolni, és a Mosquitto MQTT bróker, ami lehetővé teszi a Node-RED[1] és az Androidos alkamazás közötti kommunikációt.

1.1. A kiszolgáló hardver

Az eszköz egy nyílt forráskódu egykártyás számítógép, amin Armbian[9] (ARM processzor architektúrára specializált Debian) operációs renszer fut, de lehetőséget nyújt Ubuntu, vagy akár Android operációs rendszer telepítésére is. Ezen fut a Node-RED felület és a Mosquitto MQTT bróker, melyeket a helyi hálózaton bármely eszköz el tud érni, csupán az eszköz IP címét kell ismernie.

1.2. Az Android alkalmazás

Az Androidos alkalmazás célja a felhasználónak hozzáférést nyújtani az összes elérhető eszközhöz, azok állapotát megjeleníteni és felületet biztosítani azok irányítására, állapotuk megváltoztatására.

A kiszolgálóval való kommunikációt az Eclipse nyílt forráskódú Paho[6] Androidos kliens oldali MQTT implementációját használatba véve valósítja meg az alkalmazás.

A kommunikáció két irányú, azaz nem csak az alkalmazás tudja az okos eszközöket irányítani, hanem fogad üzeneteket a kiszolgálótól, így tud naprakész információt prezentálni az eszközök állapotáról a felhasználó számára.

A felület rugalmasságából adódóan az alkamazás nem kizárólag okos otthon kezelésére alkalmas, bármilyen MQTT protokoll alapú rendszeren való kommunikációra képes, viszont miven a fejlesztés során az okos otthonok kezelése volt az elsődleges

1.1. ábra. Orange Pi Zero[3]

1.2. ábra. A Node-RED-hez és az MQTT brókerhez használt eszköz

szempont, így erre a célra használva a legoptimálisabb a felhasználói élmény.

1.3. Node-RED

A Node-RED egy nyílt forráskódú, „flow” alapú programozási eszköz az IBM Emerging Technologies[12] által fejlesztve az OpenJS Foundation[13] részeként. Ez egy Node.js alapú fejlesztési eszköz, aminek a felületét egy böngészőn keresztül lehet elérni ahol „node”-okat elhelyezve a felületen egy funkcióhálózatot létrehozva lehet úgymond programozni. Ez a funkcióhálózat egy „Deploy” gomb hatására bekerül a futási környezetbe, így effektíve az eddigi viselkedést felülírva, változtatásainkat elmentve.

1.4. MQTT

Az MQTT (Message Queueing Telemetry Transport)[4] egy OASIS szabványú kommunikációs protokoll, melyet IoT (Internet of Things) eszközök kommunikációjához fejlesztettek ki. A protokoll alapja a „Publish/Subscribe” alapú kommunikáció, azaz egy eszköznek lehetősége van egy adott „topic”-ra üzenetet továbbítani, vagy feliratkozni, azaz az adott „topic”-on beérkező üzeneteket megkapni. Ezek az üzenetek egy brókeren keresztül érik el céljukat, mivel a bróker tárolja hogy mely eszkör milyen témára iratkozott fel, ez alapján tudja a megfelelő klienseknek továbbítani a megfelelő üzenetet. Mivel az MQTT IoT eszközök kommunikációjához készült, így fejlesztése alatt különös figyelmet fordítottak az erőforrások megspórolásához, ezért ez a protokoll

nagyon kevés erőforrást vesz igénybe, szinte bármilyen eszköz használatba tudja venni.

1.5. Okos eszközök

Okos eszköznek számít bármilyen berendezés, mely rendelkezik hálózati kommunikációra képes alkatrészekkel, ebben az esetben egy MQTT protokollt használatba vevő eszköz. Az MQTT protokoll rugalmasságából adódóan már a renszer tesztelése során is több eszköztípust vettem használatba, pédául Androidos tableteket, telefonokat és ESP D1 Mini mikrokontrollereket.

2. fejezet

Hardver

2.1. Orange Pi Zero

Egy egykártyás számítógép az Orange Pi felhozatalából a Zero model, ami beépített WiFi modullal, Ethernet porttal, 512MB RAM-al és egy 4 magos ARM processzorral ellátva tesztjeim alapján egy kisebb ház okos eszközeinek ellátására elegendő, viszont természetesen van lehetőség erősebb hardveren futtatni a kiszolgáló szolgáltatásokat. Ezek a szolgáltatások Linux és Windows operációs rendszert futtató hardverek bármelyikén képesek futni, így lehetőségünk van akár régi, már nem használt androidos telefonon, vagy ellentétsen, csak erre a célra kitűzött szervergépet kialakítani. Természetesen a végletek között rengeteg lehetőségünk van igényeink szerint választani kiszolgáló hardvert, például egy Orange Pi Zero, Raspberry Pi 4, vagy Seeed Odyssey. Az Orange Pi Zero-ra esett a választásom alacsony ára mellett nyújtott lehetőségei tárháza miatt.

2.2. Okos eszközök

2.2.1. ESP 8266

Okos eszköz fejlesztéséra egy rendkívül olcsó lehetőség az ESP D1 Mini mikrokontroller, ami Arduino nyelven programozható, WiFi moduljának és a hivatalos Arduino MQTT könyvtárnak köszönhetően viszonylag egyszerűen okos otthon eszközzé lehet alakítani.

2.2.2. Androidos eszköz

Mivel az Androidos alkamazások bizonyos könyvtárai és fejlesztői eszközei az alkalmazott Android verziótól függnek, így az alkalmazás fejlesztése során kitűzött minimum Android verziót el kell érnie a használni kívánt eszköznek. Ebben az esetben a minimum támogatott Android verzió a Marshmallow, azaz Android 6.0. Ez a legrégebbi Android

megjelenés amit bevett szokásként támogatnak a modern alkamazások, ennél korábbi verziót támogatni nehézkes, általában nem éri meg, mivel a felhasználók kevesebb mint 1%-a használ olyan eszközt, ami 6.0-nál régebbi Androidot futtat.

2.2.3. Egyéb eszközök

Minden olyan okos eszköz integrálható a rendszerbe, amely képes MQTT protokollon keresztül kommunikálni egy helyi hálózaton.

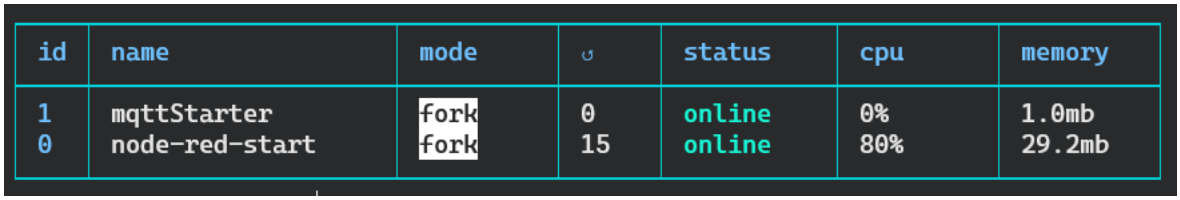
3. fejezet

Szoftver

A teljes rendszer olyan módon van felépítve, hogy a szerver oldali programoktól elvárt, hogy állandó futás mellett lehetőséget nyújtsanak bővítésre, konfigurációra és változtatásokra.

3.1. PM2

Az állandó futást a PM2[14] nevű folyamatvezető program biztosítja, akár egy váratlan újraindítás vagy áramkimaradást követően az operációs rendszer indulásával ezek a programok is indulnak, megfelelő konfigurációval.



id	name	mode	↺	status	cpu	memory
1	mqttStarter	fork	0	online	0%	1.0mb
0	node-red-start	fork	15	online	80%	29.2mb

3.1. ábra. PM2[14]

3.2. ábra. A kiszolgálók automatikus indítására használt eszköz

Ez a terminálban futtatható Node.JS eszköz egyszerűen telepíthető NPM vagy Yarn csomagkezelőkkel, és azonnal használatba vehető. Lehetőséget nyújt nem csak Node.JS alkalmazások futtatására, de egyéb futtatható fájlok kezelésére is, például shell scriptek futtatására.

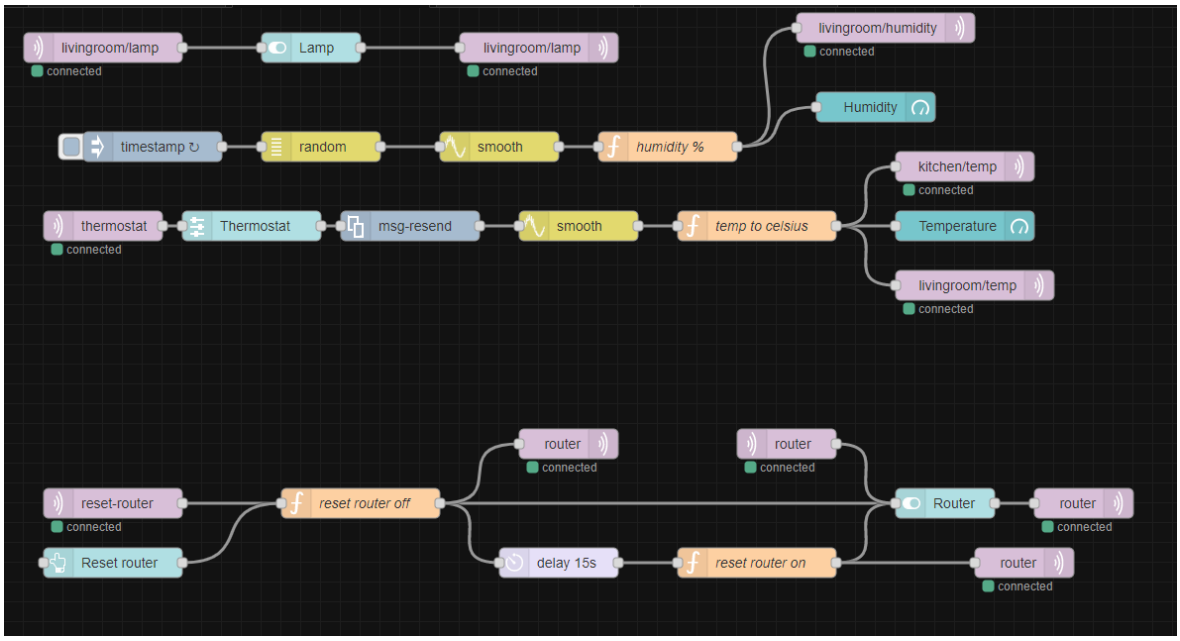
3.2. Node-Red

3.2.1. Flow

A Node-Red rendszerben csomópontok elhelyezésével és azok összekötésével lehet egy folyamatábrára hasonlító szerkezetet kiépíteni, ami a program viselkedését befolyásolja.

Minden csomópont egy funkciót lát el, ehez mérten van a csomópontnak egy vagy több bemeneti és/vagy egy vagy több kimeneti pontja, amin keresztül kommunikál a többi csomóponttal.

Egy „flow” a Node-Red felületén a benne lévő csomópontok és azok kapcsolatát foglalja magában. Minden flow reprezentálhat egy házban egy-egy szobát, nagyobb rendszereknél pédául egy-egy épületet, vagy több emeletes épületekben egy-egy emeletet. Ez felhasználható rendszerezési, kategorizálási vagy szerepköri felosztásra is, teljes mértékben a felhasználótól függ.



3.3. ábra. Minta ház nappali flow

3.4. ábra. Egy minta ház kiépítésében a nappaliban található eszközök irányítása

Az MQTT csomópontok belső konfigurációja csupán két dolgot vár el felhelyezése során: az MQTT topic, amire az adott csomópont feliratkozik, és az MQTT bróker címe. A bróker címét minden MQTT csomópont megosztja, így a kiszolgáló címének változásakor elég egy helyen megváltoztatni a címet, az összes csomópont megkapja az új címet és újra tud csatlakozni.

Implementáltam minden szobába egy okos lámpát, hőmérőt és páratartalom mérőt. Ezen felül a nappaliban és a hálószobában vagy egy-egy termosztát, melyeket függetlenül irányíthatunk. Mivel ez a minta ház csak szimuláció, így a szemléltetés érdekében a hőmérsékletváltozást egy időintervallum alatt fokozatosan változtatom a termosztát állítást követően, így reprezentálva a való világban a ház fokozatos felfűtését/lehűtését. Szintén szemléltetési céllal a páratartalom mérők véletlenszerű értékeket vesznek fel, mivel egy bemutató példa házban nem szükséges a valóságnak megfelelő értékeket szimulálni, a rendszer szemléltetése a lényeg.

Az okos lámpa egy egyszerű kapcsolóként jelenik meg a bemutató házban, ez ter-

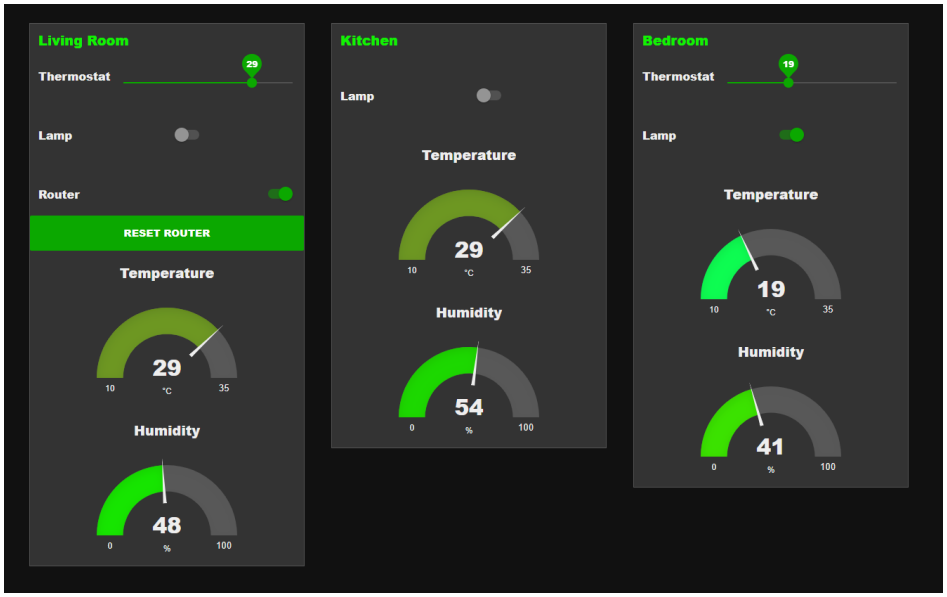
mészetesen egy okos kapcsolóval ellátott lámpát reprezentál, melyet irányíthatunk az Androidos alkalmazásból, vagy a webes felületről.

Ezen felül felhelyeztem egy internet router újraindító gombot, ezzel adva példát egy egygombos funkió ellátására, mivel egyszerűbb egy újraindítás gombot megnyomni, mint kapcsolóként kezelni és a felhasználóra bízni, hogy ne kapcsolja vissza idő előtt a router-t, így megelőzve az esetleges késés által előidézett időkimaradás eltörlését, mivel az MQTT sorban küldi az üzeneteket, ha a rendszer lelassul egy pillanatra, lehet, hogy két üzenet, amit a felhasználó egymástól eltérő időpintban küldött el, mégis egyszerre érkezik a kiszolgálóhoz.

Ez a webes szerkesztőfelületet a helyi hálózaton a kiszolgáló eszköz IP címén, a konfigurációban megadott portszámon (alapbeállítás szerint 1880) lehet elértni. Az itt végzett változtatások csak akkor kerülnek a futási környezetbe, ha a „Deploy” feliratú gombot megnyomjuk, különben minden változtatás csak a felületen történik, átmenetileg van mentve, hogy közben a szolgáltatás az előző verzióval tovább tudjon futni, így állandó elérést és irányítást biztosítva a már felcsatlakozott okos eszközök számára.

3.2.2. Dashboard

A Node-Red lehetőséget nyújt egyéb csomópontok telepítésére, melyeket felhasználók vagy egyéb entitások fejlesztettek. Egy ilyen csomópont kollekció a Node-Red Dashboard[2], mely által lehet készíteni egy webes felületet, amin keresztül lehet szemléltetni, irányítani a rendszer elemeit.



3.5. ábra. Node-Red Dashboard[2]

3.6. ábra. A Node-Red Dashboard, szobákra bontva, példa eszközökkel ellátva

Ez nem olyan rugalmas, mint az Android alkalmazás, mivel minden elemet kézzel kell felvenni a felületre, majd minden konfigurációját és interakcióját a flow szerkesztőben kell kézileg megadni, minden elemet külön fel kell programozni.

3.3. MQTT

3.3.1. Broker

MQTT brókernek az Eclipse Mosquitto[5]-t választottam, mivel egy jól ismert alapítvány által fejlesztett, teljesen nyílt forráskódú multi-platform projekt, melyben különös figyelmet fordítottak az erőforrásokkal való spórolásra, így egykártyás számítógépeken való alkamazását megkönnyítve.

Szintén előnyt jelentett választásom során a jó minőségű dokumentáció, mely alapján viszonylag zökkenőmentes volt a telepítés és a konfiguálás.

3.3.2. Konfigurálás

Mivel a konfigurációs fájlban meg kell adni a brókernek, hogy mely címen fog csatlakozási kérelmeket és üzeneteket kapni, készítettem egy olyan shell scriptet, mely minden indításnál egy új, naprakész konfigurációs fájlt készít a bróker indítása előtt.

```
echo -n "listener 1883 " > mqtt.conf; ip -4 addr show eth0 |
    ↪ grep -oP '(?<=inet\s)\d+(\.\d+){3}' >> mqtt.conf echo $"
    ↪ allow_anonymous true" >> mqtt.conf
```

Ez a shell script generál egy olyan konfigurációs fájlt, ami a következőket állítja be:

- 1883-as porton várja az üzeneteket
- Lekérdezi a kiszolgáló számítógép IPv4 címét, amit egy regex szűrőn keresztül formázom megfelelő módon és a portszám után illesztem be, előírás szerint.
- Engedélyezi az anoním kapcsolatokat. Mivel ez a rendszer egy helyi hálózatra van tervezve, nem jelent biztonsági kockázatot az autentikáció nélküli kapcsolat, ezt kiváltja az MQTT ID rendszer.

3.4. Android

3.4.1. Activity

Az Android „aktivitásokra” bontja a kódot, melyek a hozzájuk tartozó „töredékek” alappilléreként szolgál. Fő célja az aktivitásokra bontásnak az erőforrások megspórolása, minden aktivitás egy specifikus feladatkört lát el, miközben a másik aktivitások

a háttérben leállnak, amíg nem lépnek újra használatba. Az én esetemben egy fő aktivitás látja el az alapvető feladatait az applikációmnak, mivel ennek a fő aktivitásnak a kiszolgálóval való kapcsolattartás és a felhasználói adatok tárolása, így több aktivitásra bontani ezt csak feleslegesen bonyolítana mind a program logikáján, mind az átláthatóságán.

Az Android rendszer alapjáraton 4 fő „szálat" különböztet meg:

Main Thread

A fő szál. Ez az alkalmazás indításakor az operációs rendszer által nyitott szál, mely felelős az események kezeléséért, felhasználói felület megrajzolásáért és egyéb elemek létrehozásáért.

Ui Thread

A felhasználói felületi szál, ami felelős a felhasználóval való kapcsolat fenntartásáért és a felületi eseménykezelésért mint például egy gomblenyomás. Fontos, hogy a UI szálat sosem szabad megakasztani hosszú folyamatokkal, például hálózati csatlakozás indításával, mivel ilyenkor a felület teljes egészében leáll, a felhasználó felé nem reszponzív, nem tud semmilyen eseményt kezelni amíg a folyamat ami megakasztotta a szálat be nem fejeződik.

Worker Thread

Az egyéb többszálas folyamatokat kezelő szál. Ez a szál nyújt megoldást a UI szál megakasztására, ezt kell használni hosszabb, nem azonnal ellátható események kezelésére. Felmerül használata során viszon az a probléma, hogy a felhasználói felületet csakis a UI szálról lehet frissíteni.

Binder Thread

A kötött szolgáltatások távolról meghívható metódusokat tartalmaznak. Ha a végrehajtott metódusra történő felhívás ugyanabban a folyamatban van, amelyben a kötött szolgáltatás fut, a metódus a hívószalagban hajtódik végre. Ha azonban a hívás egy másik folyamatból származik, akkor a metódus egy olyan szálban kerül végrehajtásra, amelyet a rendszer ugyanabban a folyamatban tart, mint az meghívó.

3.4.2. Fragmentek

A fragment az része az alkalmazás felhasználói felületének, ami saját felosztását definiálja és kezeli. Saját életciklussal rendelkezik, viszont egyedülálló elemként nem használhatóak, mindig kell lennie egy tulajdonosának, ami lehet egy másik fragment vagy

activity. Ehez a tulajdonoshoz kötve jelenhet meg egy fragment felülete hozzá csatolva vagy részévé válva.

A fragmentek célja hogy egy activityn belül külön választott felhasználói felületet prezentálhassunk, egymástól független elemekre bontva, így megkönnyítve a felhasználói felület felépítésének procedúráját és az erőforrások megtakarítását.

A fragmentek ideális használati köre egy activityn belüli navigációs elemek által elérni kívánt felületek prezentációja. Mivel egy fragment életciklusa akkor ér véget, amikor a felhasználó egy másik fragmentre váltással felülírja azt, hosszútávú adattárolásra nem alkalmas, bár erre is vannak beépített megoldások, például a fragmentet tartalmazó activiy szintjén tárolni az adatokat, vagy a fragmentek belső „instance bundle" változójával kezelni, bár az utóbbi típusmegközései miatt igencsak korlátozó módszer, de több módszer kombinálása sem kizárt, így mindent olyan módon lehet kezelni, ahogyan az adott esetben optimális.

3.4.3. Felhasználói felület

fragment prezentációja

3.4.4. Kapcsolat

mqtt xd

3.4.5. Kártyák

tipusok, felépítés

3.4.6. Kommunikáció

kártyák, connect, disconnect, background task

3.4.7. Adattárolás, Profilok

per user kártyák, username, address prefrecnce, profil txt, Topic:Type:Data.Subdata

3.5. Tesztelés

4. fejezet

A renszer működése

Itt írom le a kész rendszer működését,

4.1. Első indítás

hátugye nem minden létezik first start

4.1.1. Szerver

mostly preconfigured de a flow lehet kell noderedbe meg mqtt config idk, meg ezeket telepíteni

4.1.2. Android

alkalmazást telepíteni, először nincs user de onnan minden magától megy

4.2. Telefon csatlakoztatása kiszolgálóhoz

ip

4.3. Okos eszközök kezelése az alkalmazásban

sub to topics, makes cards

4.4. Okos eszközök kezelése a webes felületen

node-red ui, több vele a szarakodás mint androidon so nem előnyös, része az üzemeltetésnek

5. fejezet

Továbbfejlesztési lehetőségek

user auth: bár mqtt-nél nincs sok értlme, maybe parentala controls cloud service for out of home control andoid auto-looks for the broker, this might be expensive and hard

Köszönetnyilvánítás

Köszönöm a vscode-nak hogy van,

Köszönöm magyarországnak hogy jobban teljesít

Köszönöm a covidnak, hogy átmentem nummatból

Köszönöm anyádnak hogy meleg

Köszönöm faszkivannak hogy nézte a streameket

Irodalomjegyzék

[1] Node Red forrás,
https://nodered.org

[2] Node Red Dashboard forrás,
https://flows.nodered.org/node/node-red-dashboard

[3] Orange Pi forrás,
http://www.orangepi.org/

[4] MQTT protokol forrás,
https://mqtt.org

[5] Mosquitto bróker forrás,
https://mosquitto.org

[6] Paho Android MQTT implementáció,
https://www.eclipse.org/paho/index.php

[7] Material design forrás
https://material.io/components/

[8] Android fejlesztői dokumentáció
https://developer.android.com/guide

[9] Armbian operációs rendszer
https://www.armbian.com/

[10] IoT based Smart Environment Using Node-Red and MQTT
Deepthi, B. & Kolluru, Venkata Ratnam & Varghese, George & Narne, Rajendraparasad & Srimannarayana, Nerella. (2020). IoT based Smart Environment Using Node-Red and MQTT. Journal of Advanced Research in Dynamical and Control Systems. 12. 10.5373/JARDCS/V12I5/20201684. https://www.researchgate.net/publication/342327250_IoT_based_Smart_Environment_Using_Node-Red_and_MQTT

[11] Android™Notes for Professionals book
https://books.goalkicker.com/AndroidBook/

[12] IMB Emerging Technologies
https://emerging-technology.co.uk/

[13] OpenJS Foundation
https://openjsf.org/

[14] PM2 Process Manager
https://pm2.keymetrics.io/