**UNIVERSITATEA „SAPIENTIA” DIN CLUJ-NAPOCA**

**FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,**

**TÎRGU-MUREȘ**

**SPECIALIZAREA CALCULATOARE**

# DEZVOLTAREA UNUI SISTEM DE MANAGEMENT AL ENERGIEI PENTRU CASELE INTELIGENTE

## PROIECT DE DIPLOMĂ

**Coordonator științific:**

**Conf.dr.ing. Kutasi Dénes Nimród**

**Ș.l.dr.ing. Szabó László Zsolt**

**Absolvent:**

**Szeibert Xavér**

**2023**

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Specializarea: **Calculatoare**

**Viza facultății:**

## LUCRARE DE DIPLOMĂ

| Coordonator științific: | Candidat: **Szeibert Xavér** |
|---|---|
| **Conf. dr. ing. Kutasi Dénes Nimród** | Anul absolvirii: **2023** |
| **ș.l. dr. ing. Szabó László Zsolt** | |

**a) Tema lucrării de licență:**
DEZVOLTAREA UNUI SISTEM DE MANAGEMENT AL ENERGIEI PENTRU CASELE INTELIGENTE

**b) Problemele principale tratate:**
- Studiu bibliografic privind sistemele de management al energiei in casele inteligente
- Proiectarea unui smartmeter de energie electrica, implementare practica
- Realizarea unui software de logare a consumului de energie de la consumatori, posibilități de decizii și comenzi de cuplare/decuplare consumatori

**c) Desene obligatorii:**
- Principiul de funcționare al sistemului proiectat
- Schema electronica al smartmeterului
- Schema de principiu al comunicației dintre actori

**d) Softuri obligatorii:**
-Program pe microcontroler pentru măsurarea energiei
- Program de achiziții date, statistici

**e) Bibliografia recomandată:**

[1] Jasim, Nabaa Ali, and Haider ALRkabi. "Design and Implementation a Smart System for Monitoring the Electrical Energy based on the Internet of Things." Wasit Journal of Engineering Sciences 10, no. 2 (2022): 92-100.
[2] Macheso, Paul Stone, and Doreen Thotho. "ESP32 Based Electric Energy Consumption Meter."


**f) Termene obligatorii de consultații: săptămânal**
**g) Locul și durata practicii:** Universitatea „Sapientia” din Cluj-Napoca,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Primit tema la data de: 31.03.2022
Termen de predare: 27.06.2023

Semnătura Director Departament

Semnătura coordonatorului

Semnătura responsabilului programului de studiu

Semnătura candidatului
Szeibert

**Declarație**

Subsemnata/ul Szebert Xavér, absolvent a specializării Calculatoare, promoția 2023 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

TÎRGU-MUREȘ,

Data: 26.06.2023.

Absolvent

Semnătura

# Dezvoltarea unui sistem de management al energiei pentru casele inteligente

## Extras

În lumea modernă și în rapidă dezvoltare, tehnologia stă în spatele tuturor nevoilor noastre, hobby-urilor și locurilor de muncă, pe care o utilizăm în mod confortabil și fără probleme. În aceste vremuri, odată cu creșterea inflației, prețul tuturor nevoilor a crescut, în special al energiei electrice. Prin urmare, dacă nu acordăm atenție consumului nostru zilnic de energie și nu facem schimbări în obiceiurile noastre de utilizare a energiei, putem întâmpina ușor probleme financiare.

Scopul lucrării mele este dezvoltarea unui sistem IoT (Internet of Things) care permite măsurarea și monitorizarea ușoară a consumului de energie al dispozitivelor electronice din gospodărie. În plus, sistemul poate monitoriza și producția de energie a panourilor solare instalate în gospodărie. Datele sunt stocate într-o bază de date, permițând realizarea de analize și statistici ulterioare. Aceasta oferă o imagine mai precisă asupra costurilor de energie și ajută la optimizarea consumului și valorificarea surselor de energie regenerabilă.

Diversele unități comunică între ele prin intermediul protocolului MQTT, unde publicatorii (echipamentele de uz casnic, panourile solare) trimit datele de eșantionare brokerului (Raspberry Pi), unde datele sunt procesate, stocate într-o bază de date MongoDB, iar datele prelucrate sunt trimise prin intermediul unui serviciu REST API către o aplicație mobilă.

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM**
**MAROSVÁSÁRHELYI KAR**
**SZÁMÍTÁSTECHNIKA SZAK**

# ENERGIA MENEDZSMENT RENDSZER FEJLESZTÉSE INTELLIGENS OTTHONOKHOZ

**DIPLOMADOLGOZAT**

**Témavezető:**
**Dr. Kutasi Dénes Nimród, egyetemi docens**
**Dr. Szabó László Zsolt, egyetemi adjunktus**

**Végzős hallgató:**
**Szeibert Xavér**

**2023**

# Kivonat

A modern és gyorsan fejlődő világban már minden szükségletünk, hobby és munka mögött egy elektromos technológia áll, amiket kényelmesen, gondok nélkül használunk. A mostani időkben az infláció emelkedésével minden szükségletnek az ára megemelkedett, legfőképpen az elektromos áram, ezáltal, hogy ha nem figyelünk oda a mindennapi energia fogyasztásunkra és nem változtatunk az energia felhasználási szokásainkon, akkor könnyen anyagi gondokba ütközhetünk.

A dolgozatom célja egy olyan IoT (Internet of Things) rendszer kialakítása, amely lehetővé teszi a háztartáson belüli elektronikai berendezések fogyasztásának könnyű mérését és nyomon követését. Emellett a rendszer képes monitorozni a háztartásban telepített napelemek termelését is. Az adatokat egy adatbázisban tárolom, ami lehetővé teszi a későbbi statisztikák és elemzések készítését. Ezáltal pontosabb betekintést lehet nyerni az energiaköltségekbe, valamint segít optimalizálni a fogyasztást és kihasználni a megújuló energiaforrásokat.

A különböző egységek MQTT protokollon keresztül kommunikálnak egymásközt, a publisherek(háztartási berendezés, napelemek) küldenek mintavételezett adatokat a brokernek(Raspberry pi), ahol az adatok fel vannak dolgozva, el lesznek tárolva egy MongoDB adatbázisban és a kész adatok egy REST API szoláltatáson keresztül el lesznek küldve egy mobilos applikációra.

***Kulcsszavak:*** IoT, Intelligens teljesítmény mérő, MQTT, Energia Menedzsment

# Abstract

In the modern and rapidly evolving world, there is electric technology behind all our needs, hobbies, and work, which we use comfortably and without problems. In current times, with the increase in inflation, the prices of all necessities have risen, especially electricity. Therefore, if we do not pay attention to our daily energy consumption and do not make changes in our energy usage habits, we can easily encounter financial difficulties.

The aim of my thesis is to develop an IoT (Internet of Things) system that enables easy measurement and monitoring of energy consumption of electronic devices within a household. Additionally, the system is capable of monitoring the production of solar panels installed in the household. The data is stored in a database, allowing for future statistical analysis and insights. This provides a more accurate understanding of energy costs and helps optimize consumption while leveraging renewable energy sources.

The different units communicate with each other through the MQTT protocol, where publishers (household appliances, solar panels) send sampled data to the broker (Raspberry Pi), where the data is processed, stored in a MongoDB database, and the processed data is sent to a mobile application via a REST API service.

***Keywords***: IoT, Smart Energy meter, MQTT, Energy Management

## Tartalomjegyzék

## Ábrák jegyzéke

## Táblázatok jegyzéke



## Egyenletek jegyzéke

# 1. Bevezető

## 1.1. Témaválasztás indoklása

A technológia orientált világban manapság már nehezebben lehet követni, hogy hány elektronikai eszközünk van a háztartásunkban, legfőképpen ezeknek az eszközöknek energia fogyasztását. Nagyobb és sokkal jobban felszerelt háztartásokban egészen lehetetlen, hogy saját magunk tudjuk karbantartani elektronikai berendezéseinket, hogy a sokszoros fogyasztásuk ne hagyjon jelentős nyomot anyagi helyzetünkben.

A tanulmányaim során érdeklődtem és foglalkoztam az okosotthonok rendszerezésével foglalkozó technológiával, amit szakmai körökben az IoT (Internet of Things) néven ismerünk. Ez a technológia lehetővé teszi különböző eszközök összekapcsolását egy közös platformon, az információk összegyűjtését és akár különböző feladatok végrehajtását is biztosítja.

A dolgozatom során részletesen bemutatom az IoT rendszer kialakítását, a kommunikációt MQTT protokollon keresztül, az adatok feldolgozását és tárolását egy MongoDB adatbázisban, valamint az adatok mobilalkalmazásba történő továbbítását egy API segítségével. Ezenkívül részletesen ismertetem a rendszer működését, az energiafogyasztás mérése és monitorozása mellett a napelemek termelésének figyelemmel kísérését és az energiahasználat optimalizálását.

## 1.2. Célkitűzések

A dolgozatom hatékony megvalósításához, célokat tűztem ki, amelyek meghatározzák a projektem különböző komponenseinek megalkotását, megfogalmazását:

- Az eszközök pontos teljesítménymérésének biztosítása
- Sajátos broker és szerver tervezése az adatok tárolására és kommunikáció biztositására
- Mobilapplikáció készítése az energia menedzsment figyeléséhez
- Statisztika készítése és megjelenítése a rendelkezésre álló adatokkal

# 2. Elméleti megalapozás és szakirodalmi tanulmány

## 2.1. Szakirodalmi tanulmány

A szakirodalomban leírt mérési módszerek szerint [1] az energiafogyasztás mérésére más hasonló szenzorok mellett a ZMPT101B feszültségérzékelő és az SC-013 áram érzékelők használatosak. Ezekkel az érzékelőkkel lehet mérni a fogyasztóra eső feszültséget és áramot, és az EmonLib könyvtár segítségével kiszámítható az aktív és reaktív teljesítmény. Az adatokat gyakran a Blynk alkalmazáson keresztül jelenítik meg, amelyet az ESP32 mikrovezérlőre telepíthető könyvtárral lehet integrálni, így a felhasználó könnyedén láthatja a mért adatokat és figyelemmel kísérheti az energiafogyasztást.

Másik megközelítés a PZEM-004T energia mérő [2] használata, amely méri az áramerősséget, feszültséget és az aktív teljesítményt. Az adatokat az ESP32 vezérlő I2C protokoll segítségével kapja meg a PZEM-004T modultól. Ezután a mikrovezérlő MQTT kommunikációval továbbítja az adatokat a Raspberry Pi szerver felé, ahol további feldolgozásra és tárolásra kerülnek.

A hagyományos módszerek az aktív és reaktív teljesítmények méréshez [3] egy standard feszültségmérő kiszámítja a feszültség és az áramerősség négyzetes középértékét és a teljesítmény faktorját. A hagyományos megoldás szerint szükséges, hogy direkt megmérjük frekvenciát, a fáziseltolást a feszültség és az áramerősség között.

$$\varphi = w \cdot t_{\varphi} = 2 \cdot \pi \cdot f \cdot t_{\varphi} \Rightarrow \cos \varphi$$

*1. egyenlet Fázisszög számítás*

$$P_{(W)} = V \cdot I \cdot \cos \varphi$$

*2. egyenlet Aktív teljesítmény számitás*

$$Q_{(VAR)} = V \cdot I \cdot \sin \varphi$$

*3. egyenlet Reaktív teljesítmény számitás*

A szakirodalom szerint [3] létezik egy alternatív módszer, ahol (P) az aktív teljesítmény kifejezhető a (pM) pillanatnyi teljesítmény csúcsértékének és (S) a látszólagos teljesítménynek a különbségével. A reaktív (Q) teljesítmény és a fázisszög (ϕ) meghatározható a pillanatnyi csúcsteljesítmény és a látszólagos teljesítmény értékeivel.

$$P_{(W)} = p_M - S$$

*4. egyenlet Aktív teljesítmény számítás – új módszer*

$$|Q|_{(VAR)} = \sqrt{( p_M \cdot (2 \cdot S - p_M))}$$

*5. egyenlet Reaktív teljesítmény számítás - új módszer*

$$\cos \varphi = pM/S - 1$$

*6. egyenlet Fázisszög számítás - új módszer*

A következtetés szerint az új módszer az aktív és reaktív teljesítmény számítására előnyösebb a hagyományos képletekhez képest. Ennek oka, hogy az új módszer lehetővé teszi a pillanatnyi csúcs és látszólagos teljesítmények közötti különbség közvetlen kiszámítását, anélkül, hogy szükség lenne a fázisszög és frekvencia közvetlen mérésére.

## 2.2. Elméleti alapok

### 2.2.1. Energia menedzsment

Az áramenergia fogyasztása könnyen pénzügyi problémákká tudnak alakulni, ha nincs szabályozva a háztartásban az energia menedzselés. A menedzseléshez [4] 5 főbb szempontot figyelembe kell venni:

- Meg kell állapitani az energiafogyasztók forrását, ezáltal könnyen lehet követni, hogy melyik készülék fogyaszt a legtöbbet.
- Villanyszámlákat gyűjteni és azoknak árait figyelni kell, ezzel is jobban be lehet tervezni, hogy akár rövid távon is takarékosabb legyen a fogyasztás.
- Figyelni kell az árammérő adatokat, ezzel meghatározva, hogy mely készülékek fogyasztanak a legtöbbet vagy a legkevesebbet.
- Ki kell használni azokat a lehetőségeket, hogy spórolni tudjunk az áramfogyasztással
- Nyomon kell követni az eddigi lépéseinket és ezáltal összesíteni lehet, hogy döntéseink jó vagy rossz irányba haladnak.

### 2.2.2. Energia Hálózati típusok

Két különböző energetikai hálózatot [5] tudunk megkülönböztetni, azt a rendszert, ahol a kitermelt áramnak nem biztosítunk átjárást az energia szolgáltatónak, azt „Off-Grid" hálózatnak hívjuk, és azt a rendszert, ahol az áram oda-vissza tud terjedni és rá van kapcsolva a szolgáltató energiahálózatára, azt „On-Grid" hálózatnak nevezzük.

### 2.2.3. Napelemes rendszer

Egy napelemes rendszer kialakítása egy háztartásnál bonyolult folyamat és megvalósítása nagy beruhást igényel.

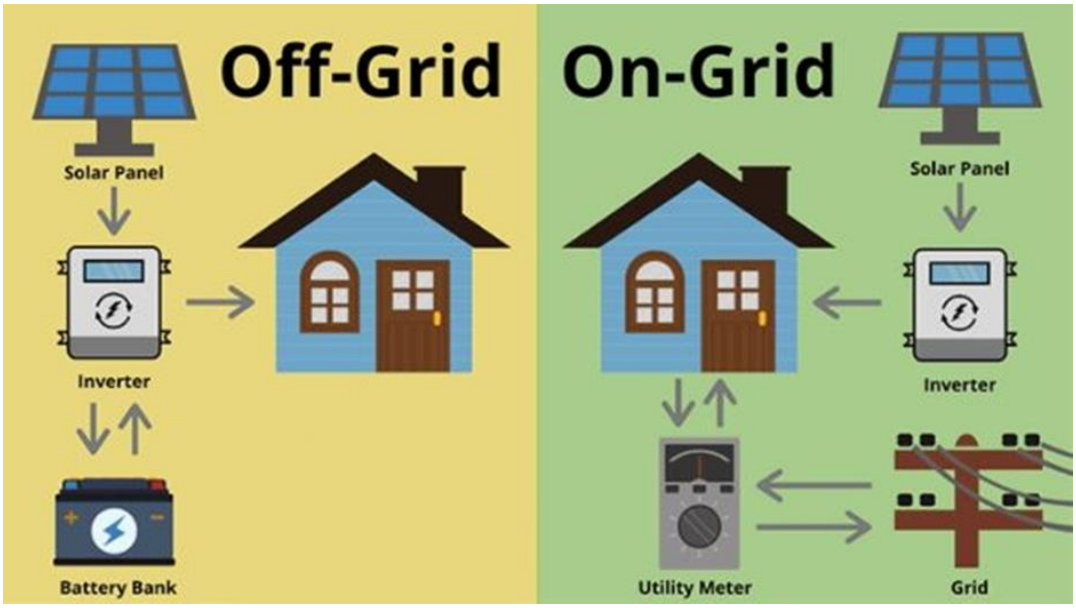
Megtervezési folyamata függ a város napelem engedélyeiről és az energiaszolgáltatás szabályzataitól. Az optimális és leghatékonyabb áramtermeléshez nagyon függ a tető architektúrájától, a legalkalmasabb, hogy ha napelemek befedik a tetőzet teljes felületét. A napelemek különböző gyártótól különböző méretben és fajtában elérhetőek.

A napelemek elhelyezkedése [6] is függ, legérdemesebb olyan tetőkre telepíteni, amelyek felületük Délre néznek az északi szélességben vagy Északra a déli szélességben. Hatékonyabb viszont drágább megoldást nyújt, ha a napelemek motrok segítségével folyamatosan úgy változtatják helyzetüket, hogy merőlegesen álljanak a napsugarakra.

A napelemek egyenáramot termelnek és a háztartási eszközök váltóáramot igényelnek, ezért egy DC to AC áramátalakítót kell telepíteni. Hybrid illetve smart inverterek esetén követni tudjuk egyenként a napelemek állapotát, áramtermelési értékeiket.

On-Grid rendszerek esetében, az áramhálózatunkat egy kétirányú villanyórára csatlakoztatjuk, hogy egyszerre tudjuk mérni a generált és a szolgáltató által biztosított energiát.

Off-Grid rendszerek esetében, hálózatunkat elégséges egy egyszerű villanyórára csatlakoztatni, hogy nyomon tudjuk követni a generált árammennyiséget. Ajánlatos energia bankokat is telepíteni, hogy felhős és esti időszakokban biztosított legyen az áramellátás.




*1. ábra https://urjanet.com/blog/top-5-tips-successful-energy-management/*

### 2.2.4. Elektromos teljesítmény mérése

Az elektromos hálózaton a váltófeszültséget általában a szinuszos feszültség mérése segítségével mérik. Ezt a mérést az áramlási transzformátorok végzik, amelyek gyakran ferromágneses anyagból készülnek. Az áramlási transzformátor mágneses mezőt hoz létre az áramkörben átfolyó áram alapján. Ez a mágneses mező lehetővé teszi a szenzor számára, hogy érzékelje és mérje a feszültséget.

Az áramlási transzformátorok fontos szerepet játszanak az elektromos hálózatokban történő feszültségmérésben, mivel lehetővé teszik a magas feszültségű áramkörök és a mérőeszközök közötti biztonságos izolációt. Emellett lehetőséget nyújtanak a mérések nagyobb pontosságára és skálázhatóságára is, mivel különböző méretű és kapacitású áramlási transzformátorokat lehet alkalmazni a különböző feszültség- és áramértékek méréséhez.

Az áramtranszformátorok gyakran használt eszközök az elektromos hálózatokban az áramerősség mérésére [7]. Ezek az eszközök átalakítják az áramot alacsonyabb értékekre, ami biztonságosabb és könnyebben mérhető. Az áramkörben egy söntellenással együtt alkalmazzuk őket, ami lehetővé teszi az áramerősség mérését a söntellenáson eső feszültség alapján.

Az áramtranszformátor egy ferromágneses anyagból készült eszköz, amelynek feladata, hogy mágneses mezőt hozzon létre az áramkörben átfolyó áram alapján. Ez a mágneses mező aztán átalakítja az áramot alacsonyabb értékekre, amelyek könnyebben mérhetők.

Az egyenáramú rendszerekben a teljesítmény [8] egyszerűen kiszámítható a feszültség és az áram szorzatából. Váltóáramú rendszerekben azonban a teljesítmény meghatározása bonyolultabb. Itt figyelembe kell venni a feszültség és az áram időnkénti irányváltozását.

A teljesítmény mérése során általában az RMS (effektív) értéket használjuk, amely a feszültség és az áram időnkénti változását figyelembe véve adja meg a valós teljesítményt. Ha a rendszerben reaktív elemek (pl. induktancia vagy kapacitás) is vannak jelen, akkor a feszültség és az áram időben eltérhet egymáshoz képest. Ebben az esetben a teljesítmény kiszámításához figyelembe kell venni a fázisszöget és az energiafaktort.

### 2.2.5. Elektromos energia számítása

Az energiafelhasználás tervezése során fontos számítani az átlagos energiafogyasztást. Ehhez az időegységen belüli átlagos teljesítményt (P) kell kiszámolnunk, majd ezt szorozzuk össze az időtartammal (t), amelyet általában órában mérünk. Az így kapott szorzat az energia (E) értékét adja kilowattórában (kWh).

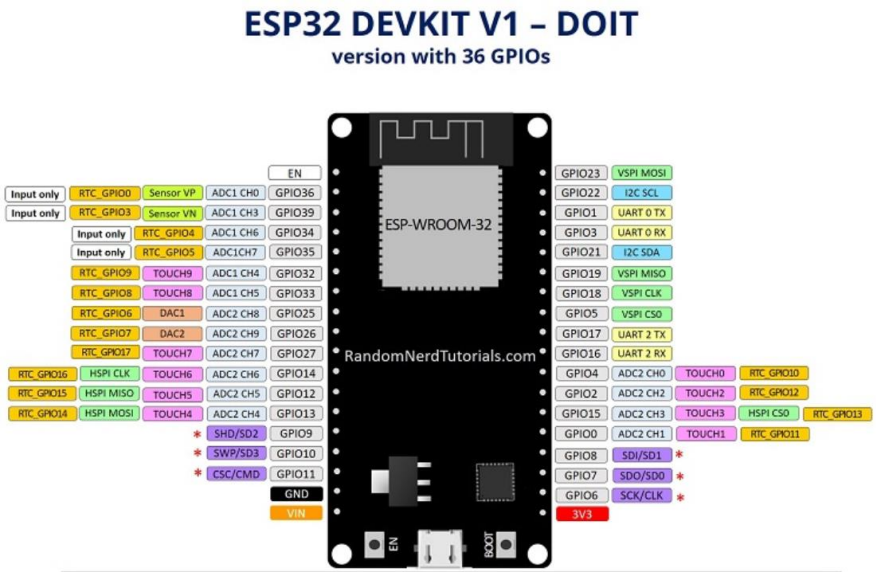
$$E_{(kWh)} = P \cdot t$$

*7. egyenlet Energia számitás*

## 2.3. Felhasznált technológiák

### 2.3.1. ESP32

A teljesítmény méréséhez egy Devkit V1 típusú ESP32[1] mikróvezérlőt [9] használtam, amely egy alacsony árú, fogyasztású eszköz, amely rendelkezik Wi-Fi és Bluetooth vezeték nélküli technológiával, amely lehetővé teszi a vezeték nélküli kommunikációt más eszközökkel és hálózatokkal. A mikróvezérlő előnye, hogy rendelkezik számos perifériákkal és interfészekkel, amelyek lehetővé teszik a kommunikációt más eszközökkel. Ide tartoznak az analóg-digitál átalakító (ADC), I2C, SPI, PWM modul, UART és a GPIO bemenetek.




*2. ábra ESP32 pinout referencia*

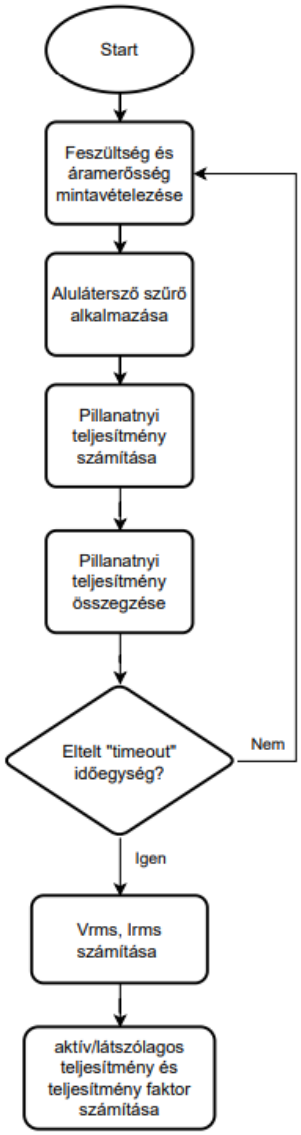
### 2.3.2. EmonLib könyvtár – teljesítmény mérés

A teljesítmény mérése során használtam az OpenEnergyMonitor [10] által fejlesztett EmonLib.h könyvtárat. Az EmonLib.h-ban található "CalcVI()" függvényt alkalmaztam, amely lehetővé teszi a teljesítmény számítását a „timeout" időintervallum alatt. A „timeout” paraméter meghatározza az időtartamot, amely alatt történik a mintavételezési folyamat. Ez a függvény

[1] https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

többszörös mintavételezést végez az analóg bemeneteken annak érdekében, hogy rögzítse a feszültség és az áramerősség értékeit. A mintavételezett adatokra egy digitális aluláteresztő szűrőt alkalmaz, hogy kivonja az 1.65 V offset-et, hogy a jel 0 értekhez legyen igazodva.

A megszűrt adatokkal kiszámítja a pillanatnyi teljesítményt a pillanatnyi feszültség és áramerősség szorzatával, ezeket majd átlagolva megkapjuk az aktív teljesítményt. A látszólagos teljesítmény számításához, meg szorozza a feszültség és az áramerősség négyzetes középértékével. A teljesítmény faktort meghatározza az aktív teljesítményt a látszólagos teljesítménnyel való osztással.




*3. ábra Teljesítmény számitás folyamata*

```
realPower = sum_inst_power  / numberOfSamples;
apparentPower = Vrms * Irms;
powerFactor = realPower / apparentPower;
```

*4. ábra Teljesítmény értékek számítása*

### 2.3.3. MQTT protokoll

A kommunikáció megvalósításához az ESP32 mikrokontroller és a szerver között az MQTT (Message Queuing Telemetry Transport) [11] protokollt használtam. Az MQTT egy könnyű súlyú kommunikációs protokoll, amely különösen alkalmas IoT (Internet of Things) alkalmazásokban való használatra. Ez a protokoll skálázható és megbízható, ugyanis támogatja a QoS (Quality of Service) szinteket, melyek lehetővé teszi az üzenetek garantált kézbesítését.

### 2.3.4. Raspberry Pi

A Rendszerem központi feldolgozó egysége és szervere egy Raspberry Pi[2] 4B [12] számitógép. Ez az eszköz egy erős processzorral és nagy memóriával rendelkezik az adatok hatékony feldolgozására és stabil működésére.

Az eszközön fut egy sajátos MQTT broker, amely fontos szerepet játszik az adatok közvetítésére és továbbitására a különböző eszközök között. Itt vannak lekezelve a témák (topics) amelyekre az eszközök feliratkoznak és elküldik az üzeneteket. Igy lehet lekezelni, hogy melyik eszköz küldi az üzenetet és ezáltal tovább feldolgozni az adatot.

*5. ábra Raspberry Pi logo*

[2] https://www.raspberrypi.com

### 2.3.5. Flask keretrendszer - webszerver

A szerveren Flask[3] webserver fut [13], amely lekezeli a REST API kérések kiszolgáltatását. A Flask egy könnyű és rugalmas webes keretrendszer Python nyelven, amely lehetővé teszi webalkalmazások fejlesztését és futtatását. A Flask számos beépített funkciót és eszközt biztosít, amelyek segítségével könnyedén kezelhetjük az HTTP kéréseket, kezelhetjük az URL-ek útvonalvezetését


*6. ábra Flask logo*

### 2.3.6. MongoDB

A fogyasztók és termelők adatainak eltárolására a MongoDB-t[4] [14] választottam, amely egy NoSQL típusú dokumentum-alapú adatbázis. A MongoDB rendkívül rugalmas és skálázható, lehetővé teszi az adatok hatékony tárolását és feldolgozását.

Az adatok dokumentum formájában vannak eltárolva, JSON szerű struktúrát alkalmaz. Ez lehetővé teszi a projekt hatékony fejlesztését, mivel a dokumentumok könnyen módosíthatóak és felhasználhatóak.


*7. ábra MongoDB logo*

[3] https://flask.palletsprojects.com

[4] https://www.mongodb.com

### 2.3.7. REST API szolgáltatás

A szerver és a mobilalkalmazás közötti kommunikáció megvalósításához az REST API-t (RESTful Application Programming Interface) [15] használtam, ami egy olyan szolgáltatás, mely lehetővé teszi a szoftverkomponensek közötti adatátvitelét. A REST API meghatározza a végpontokat, a kérések formátumát, válaszok struktúráját. Ez megoldja az absztrakciót a komponensek között, elrejtve egymástól a működési hátterüket.

A REST (Representational State Transfer) API (Application Programming Interface) egy elterjedt és széles körben használt módszer a webszolgáltatások kommunikációjára. A REST alapelvei és tervezési mintái lehetővé teszik a különböző szerverek és kliensek közötti adatcserét és interakciót. A REST API lehetővé teszi a kliensek számára, hogy HTTP protokollon keresztül kéréseket küldjenek a szervereknek, és válaszokat kapjanak tőlük. Az API-n keresztül a kliensek információkat kérhetnek, adatokat küldhetnek, vagy műveleteket hajthatnak végre a szerveren található erőforrásokkal.

### 2.3.8. Android Stúdió

Az Android Stúdió[5] [16] egy integrált fejlesztői környezet (IDE), amelyet az Android alkalmazások fejlesztésére használnak. A Stúdió számos funkciót és eszközt kínál a hatékony Android alkalmazásfejlesztéshez. Kotlin programozási nyelvet használtam a mobilalkalmazás fejlesztéséhez, amely a modern Android alkalmazások egyre népszerűbb választása. Az Android Stúdió és a Kotlin együttműködése segített a mobilalkalmazásom fejlesztésében, valamint a felhasználóbarát és teljesítményorientált Android alkalmazás készítésében.

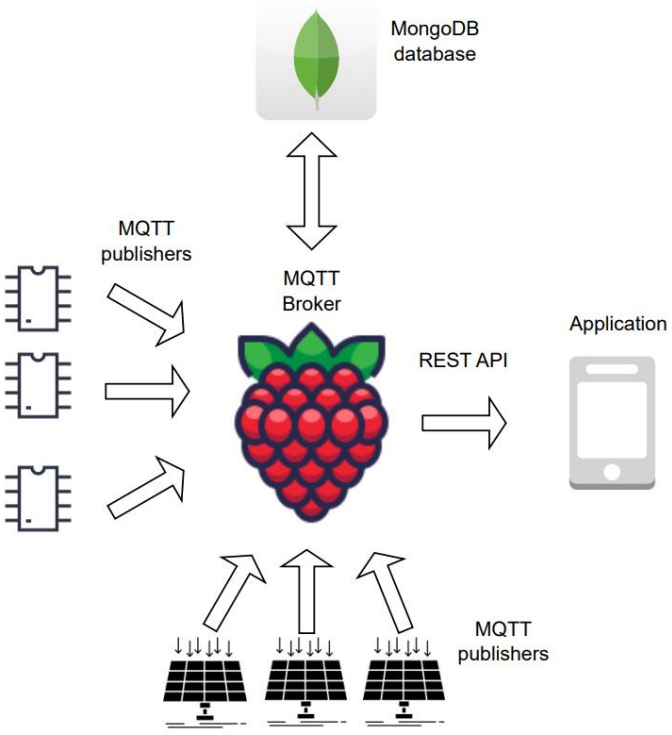
*8. ábra Android Studio logo*

[5] https://developer.android.com

# 3. A rendszer specifikációja és architektúrája

## 3.1. Rendszer architektúra

A dolgozat kivitelezése öt fő komponenst tartalmaz, amelyek mindegyikének saját, meghatározott szerepköre van. Ezen komponensek együttműködése kulcsfontosságú a projekt sikeréhez. A diagram (9.ábra) segítségével lehet követni, hogy a komponensek hogyan kapcsolódnak egymáshoz és hogyan van megoldva a kommunikáció közöttük.




*9. ábra Rendszer architektúra diagrammja*

## 3.2. Komponensek szerepkörei

1. ESP32 fogyasztók:
    - Csatlakozás a Wi-Fi hálózatra
    - Kapcsolat létesítése az MQTT broker-el
    - Áramerősség és feszültség mérése
    - Teljesítmény kiszámítása
    - Mért adat kiszámítása és továbbítása a szerverhez
2. ESP32 termelők:
    - Csatlakozás a Wi-Fi hálózatra
    - Kapcsolat létesítése az MQTT broker-el
    - Áramerősség és feszültség mérése
    - Teljesítmény kiszámítása
    - Mért adat kiszámítása és továbbítása a szerverhez
3. Raspberry Pi – MQTT broker:
    - Kapcsolat létesítése a Publisherekkel
    - Üzenetek fogadása a Publisherektől
    - Publisherek karbantartása szálakban
    - Adatok feldolgozása tovább küldésre
    - Adatok tárolása a MongoDB adatbázisba
4. Raspberry Pi – Flask REST API
    - GET fogyasztó/termelő kérés kiszolgáltatása
    - GET statisztika kérés kiszolgáltatása az adatbázisból
    - POST eszköz hozzáadás kérés kiszolgáltatása
    - DELETE eszköz törlés kérés kiszolgáltatása
    - UPDATE eszköz kérés kiszolgáltatása
5. MongoDB
    - Átlag teljesítmény fogyasztás tárolása adott időn belül
    - Átlag teljesítmény termelés tárolása adott időn belül

6. Mobilapplikáció
    - Aktív fogyasztók/termelők listázása
    - Fogyasztók/termelők hozzáadása/eltávolítása
    - Fogyasztók megváltoztatása
    - Statisztikák lekérése adott dátumra

A komponensek részletes leírását a 3.4-es fejezetben kerülnek bemutatásra.

## 3.3. Rendszerkövetelmények

### Funkcionális követelmények

- A hardver eszköz konnektorba helyezés követően kap tápellátást
- Elérhető internethálózat esetén az eszköz folyamatosan méri a teljesítményt
- Meghatározott idő alatt elmenti az átlagfogyasztást az adatbázisban
- Kimutatja az eszközök állapotát az alkalmazásban
- Statisztikákat mutat ki adott intervallumon belül

### Nem-funkcionális követelmények

- Python 3.10+, MQTT könyvtárak, Flask könyvtár, Android 13
- Raspberry Pi 4B/4GB, ESP32 mikrovezérlő
- Felhő alapú MongoDB adatbázis elérés
- Európai standard 16-20 A, 220-250 V konnektor
- Stabil internetkapcsolat

### Kritikus-biztonsági követelmények

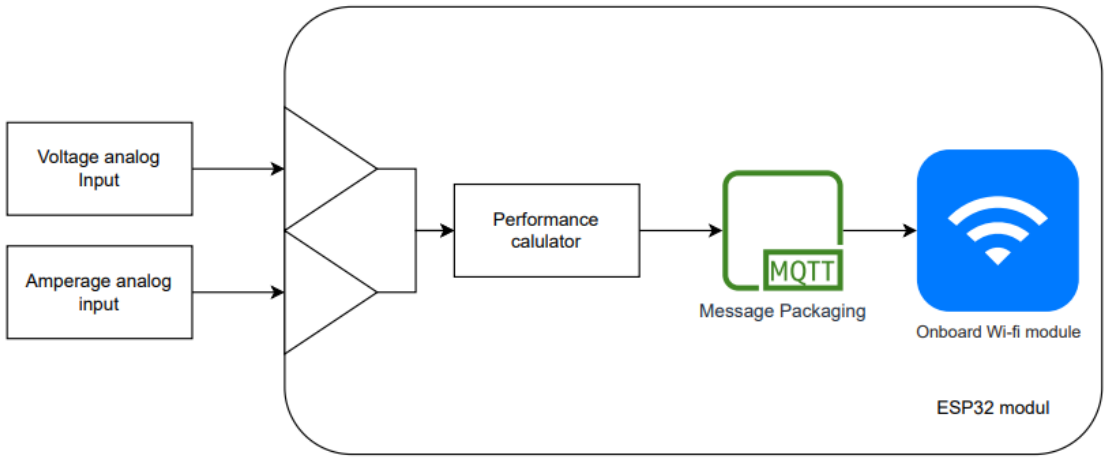
- Magas 220-250 V feszültség
- Magas 16-20 A áramerősség
- Áramkör tervezése és tesztelése labor körülményekben végzendő
- Magasárammal tapasztalt szakértői felügyelet

**Elérhetőségi követelmények**

- Elérhető energiaellátás esetén akadálymentesen elérhetőek az eszközök
- Folytonos energiatáplálás alatt a szervertől származó adatok elérhetőek
- Stabil internet sebesség alatt, az eszközök folytonosan kommunikálnak

## 3.4. ESP32 mikróvezérlő

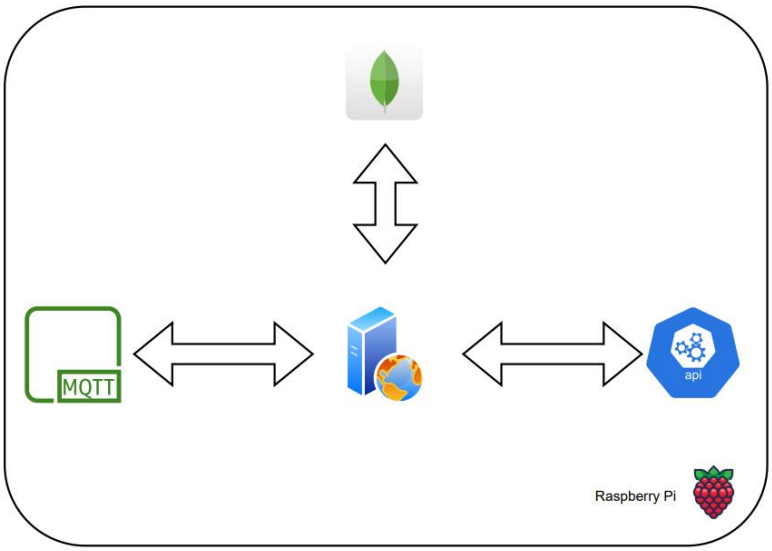
A teljesítmény kiszámításához az ESP32 mikrovezérlő használja a 33. és 36. analóg-digitális konverter bemeneteit, hogy mérje a szenzoroktól érkező adatokat. A „EmonLib.h" könyvtár által biztosított függvények segítségével kiszámítja a pillanatnyi teljesítményt. Az ESP32 beépített Wi-Fi moduljával az internetes hálózatra csatlakozva a PubSubClient könyvtár segítségével az MQTT protokollon keresztül tovább küldi az adatokat a szerverre.




*10. ábra ESP32 Blokkdiagramm*

## 3.5. Raspberry Pi

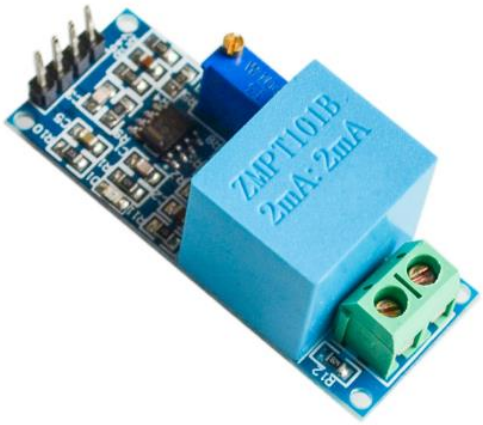
Minden üzenet, adatfeldolgozás a szerveren történik meg. A Raspberry Pi fogadja az MQTT protokollon keresztül az adatokat, strukturált adattá alakítja és továbbítja a REST API szolgáltatáson keresztül a mobilalkalmazásnak. Meghatározott idő alatt egy átlagot számol az összteljesítmény értékből és ezeket elmenti egy MongoDB adatbázisban.




*11. ábra Raspberry pi Blokkdiagramm*

## 3.6. ZMPT101B feszültségmérő szenzor

A feszültség méréséhez a ZMPT101B feszültségérzékelő modult használtam [17], amely tartalmazza a ZMPT101B feszültség transzformátort. Pontos méréseket biztosit akár 250V váltó áramú feszültségig, nagyon alkalmas a feszültség és a teljesítmény mérésére. Könnyen használható és tartalmaz egy potenciométert az ADC kimenet korrigálására.



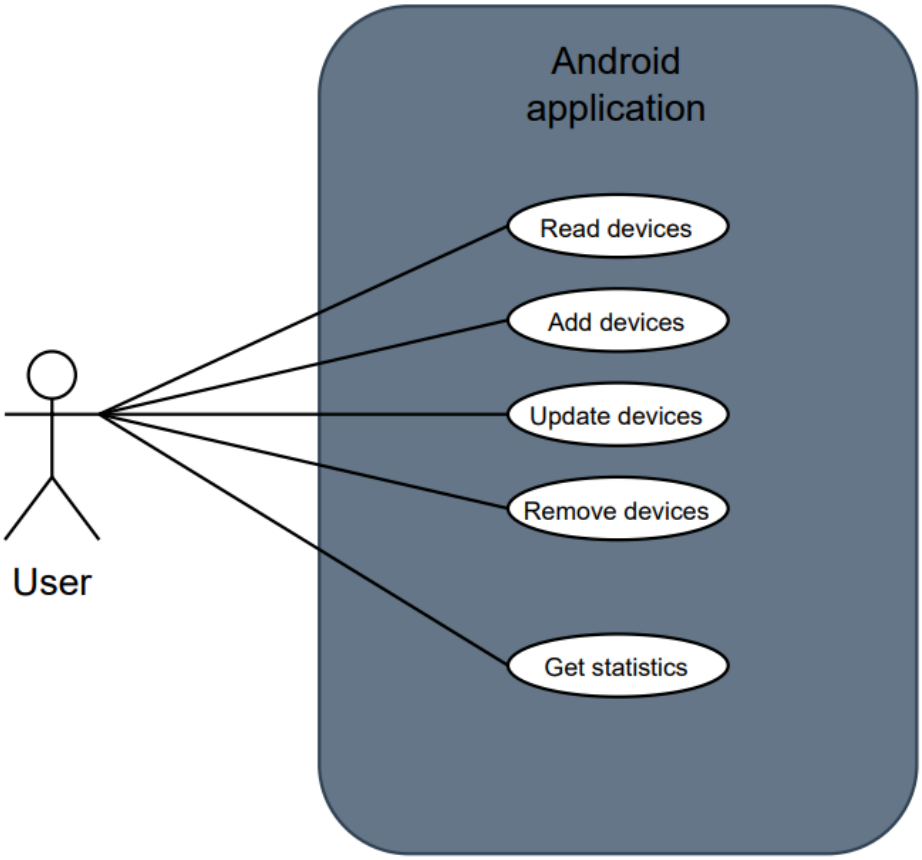

*12. ábra ZMPT101B feszültség szenzor*

## 3.7. Áramtranszformátor

Az áramerősség mérésére egy HWCT 20A-20mA áramtranszformátort használtam. A transzformátor kialakítása szerint a primer oldalon 1000 menet található, míg a szekunder oldalon csak 1 menet van. Az arányok alapján, ha a primer oldalon mért áram 20 amper, akkor a szekunder oldalon 20 milliamper áramot mérünk.


*13. ábra HWCT 20A/20mA transzformátor*

## 3.8. Felhasználói követelmények




*14. ábra Alkalmazás use-case diagrammja*

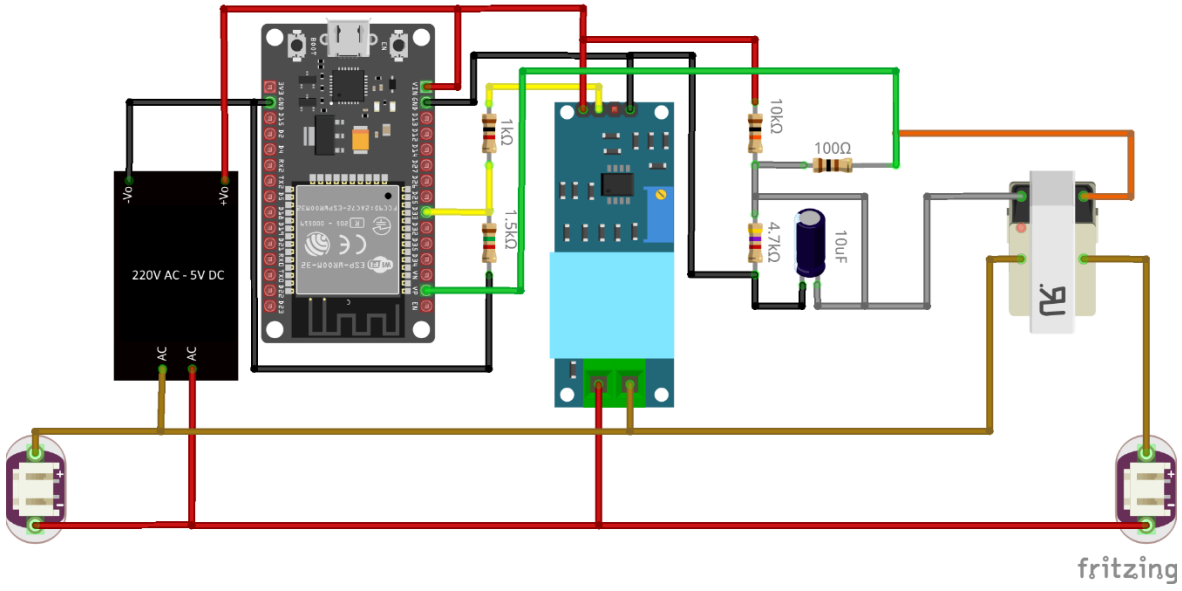
*1. táblázat Alkalmazás use-case magyarázat*

| | |
|---|---|
| Eszközök követése | A felhasználó követheti az eszközök állapotát |
| Eszközök hozzáadása | A felhasználó új eszközt tud hozzáadni |
| Eszközök frissítése | A felhasználó megváltoztathatja az eszközök paramétereit |
| Eszközök eltávolítása | A felhasználó eltávolíthatja az eszközöket |
| Statisztikák lekérése | A felhasználó lekérheti a statisztikákat az átlag termelésről/ fogyasztásról |

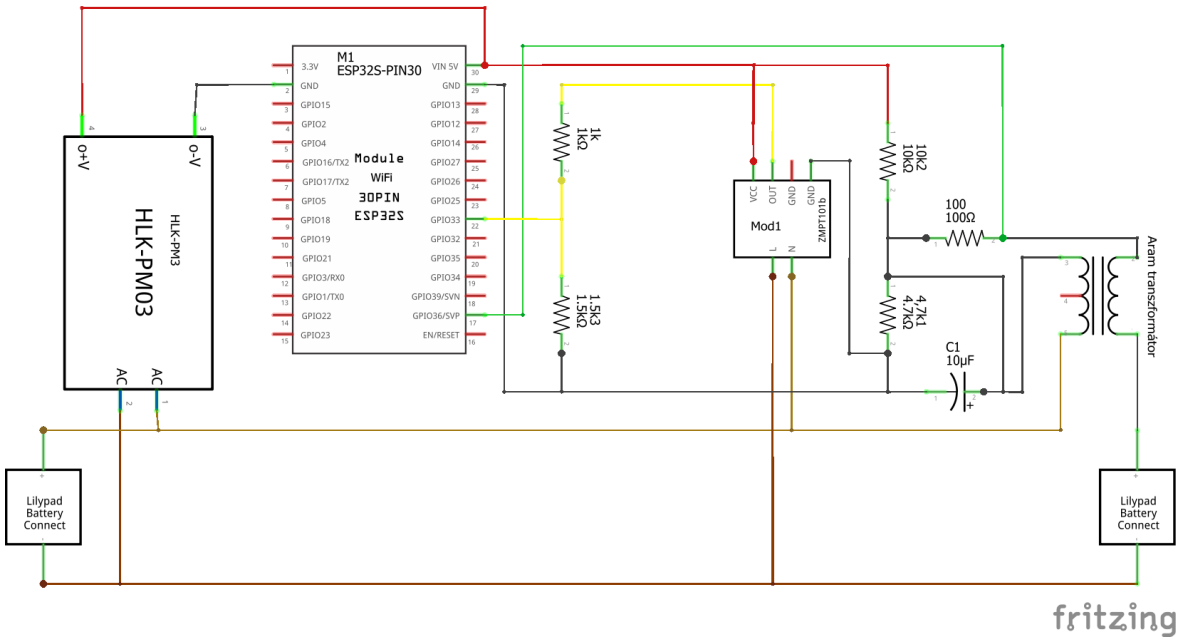
# 4. Részletes tervezés

## 4.1. Áramkör megtervezése

Az elektromos teljesítmény méréséhez terveztem egy áramkört, amelyben az ESP32 mikrovezérlő segítségével képes vagyok mérni a pillanatnyi feszültséget és áramerősséget. A mikrokontroller folyamatos működését biztosítandó, beépítettem egy 220V AC - 5V DC transzformátor modult az áramellátáshoz. A feszültségméréshez a ZMPT101B nevű váltóáramú feszültség szenzort használtam, amely a 220-250V váltóáramú feszültséget kapja bemenetként, és 0-5V AC jelet ad ki kimenetként. Mivel az ESP32 analóg-digitális bemenetei csak 3,3V feszültséget támogatnak, ezért egy feszültségosztót alkalmaztam egy 1,5k és 1k ohm-os ellenállással. Az áramerősség méréséhez pedig egy 20A-20mA áramtranszformátort használtam, ahol egy sönt ellenálláson mértem a rá eső feszültséget. A jel értékét az ESP32-on mérés céljából egy 10k és 4,7k ohm-os feszültségosztóval, valamint a zajszűrés érdekében egy 10uF kondenzátorral csökkentettem.




*15. ábra Áramkör diagram*




*16. ábra Áramkör schematic*

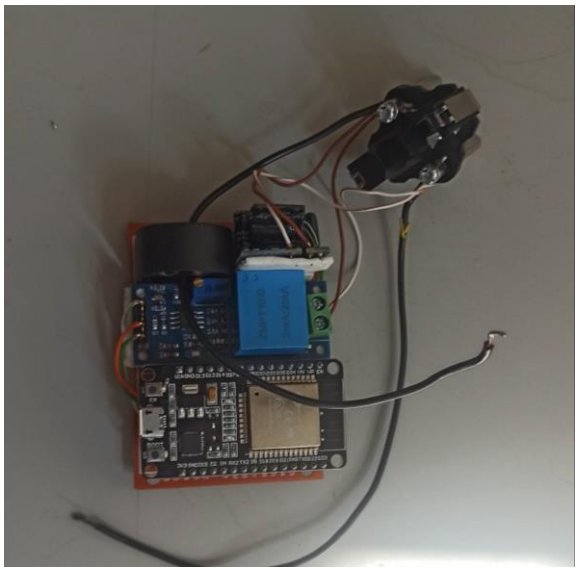
## 4.2. Áramkör összeszerelése

Az eszköz tokozásához egy konnektorba helyezhető dobozra volt szükségem, amely megfelelő helyet és védelmet biztosít a belső komponenseknek. A tervezett méretek alapján egy 64x129x57 mm-es dobozt választottam, amely elegendő teret biztosított az áramkör összeszereléséhez és megfelelően illeszkedett a konnektorhoz.

Az áramkör összeszerelésekor gondosan ügyeltem az alkatrészek megfelelő elhelyezésére és csatlakoztatására. Az alkatrészeket egy próbanyákra forrasztottam, figyelembe véve kapcsolási rajzot.

*2. táblázat Megrendelt alkatrészek*

| **Termék neve** | **Darabszám** |
|---|---|
| ESP32 DEVKIT V1 | 2 |
| ZMPT101B AC Feszültség szenzor | 2 |
| AC áram transzformátor 20A-20mA | 2 |
| Baretta mini mama | 2 |
| Baretta mini tata | 2 |
| 220v to 5v AC-DC transformator | 2 |
| Ellenállás | 2x100Ω, 2x1kΩ, 2x1,5kΩ, 2x4,7kΩ, 2x10kΩ |
| Kondenzátor | 2x10uF |
| Doboz/tokozás | 2 |



*17. ábra Összeszerelt áramkör*

## 4.3. ESP32-n futó szoftver

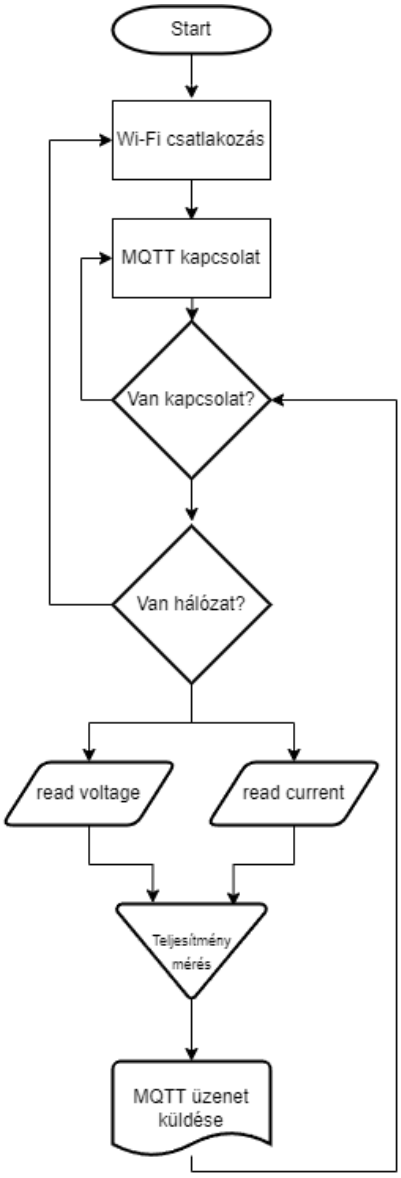
Miután elkészült a hardver, folytattam a szoftver fejlesztésével. Az ESP32 mikrovezérlőn be kellett állítanom, hogy a rendszer induláskor automatikusan csatlakozzon a Wi-Fi hálózathoz. Ha a csatlakozás sikeres volt, akkor a mikrovezérlőbe épített kék LED felkapcsolódik, jelezve a sikeres kapcsolódást. A hálózatra való csatlakozás után, a mikróvezérlő a megszabott IP cím alapján kapcsolatot létesít a Raspberry Pi számitógépen futó MQTT broker-el.

A mikrovezérlő a szerverrel való kapcsolódás után a 33-as és 36-os analóg-digitál konverter bemeneteken mintavételezi a pillanatnyi feszültség és áramerősség értékeit. Az áramerősség esetében meg kell adni a bemeneti pin és a kalibrációs értékeket, feszültség esetében pedig a bemeneti pin, kalibrációs és fáziseltolás értékeket. A kalibrációs értékek szükségesek, hogy a mikróvezérlő bemeneti 0-4095 skála közötti értékeket függvényen belül átalakítsa és kiszámítsa a teljesítmény értékeket. A fáziseltolás értéke három fázisú rendszerek esetében használja fel, viszont az egyfázisú rendszeremnél nem befolyásolja a kimeneti értékeket, igy meghagytam az alapértéknél, a függvény helyes működése céljából. A CalcVI() függvénynek megadott 5 másodperc időegység alatt mikrovezérlő kiszámítja a teljesítmény értékeket, és ezeket az adatokat MQTT protokoll segítségével továbbítja a szerver felé.

A kommunikáció során az MQTT témában az "ESP32/producerx" formátumot használjuk, ahol az "x" értéke különböző lesz minden egyes eszköz esetében. Ez az egyedi azonosító lehetővé teszi a szerver számára, hogy azonosítsa, melyik eszköz küldi az üzeneteket.

```
emon1.voltage(voltagePin, 206.8,1.7);
emon1.current(currentPin, 2.5);
emon1.calcVI(20, 2000);
```

*18. ábra Teljesítmény számolás EmonLib.h könyvtárral*

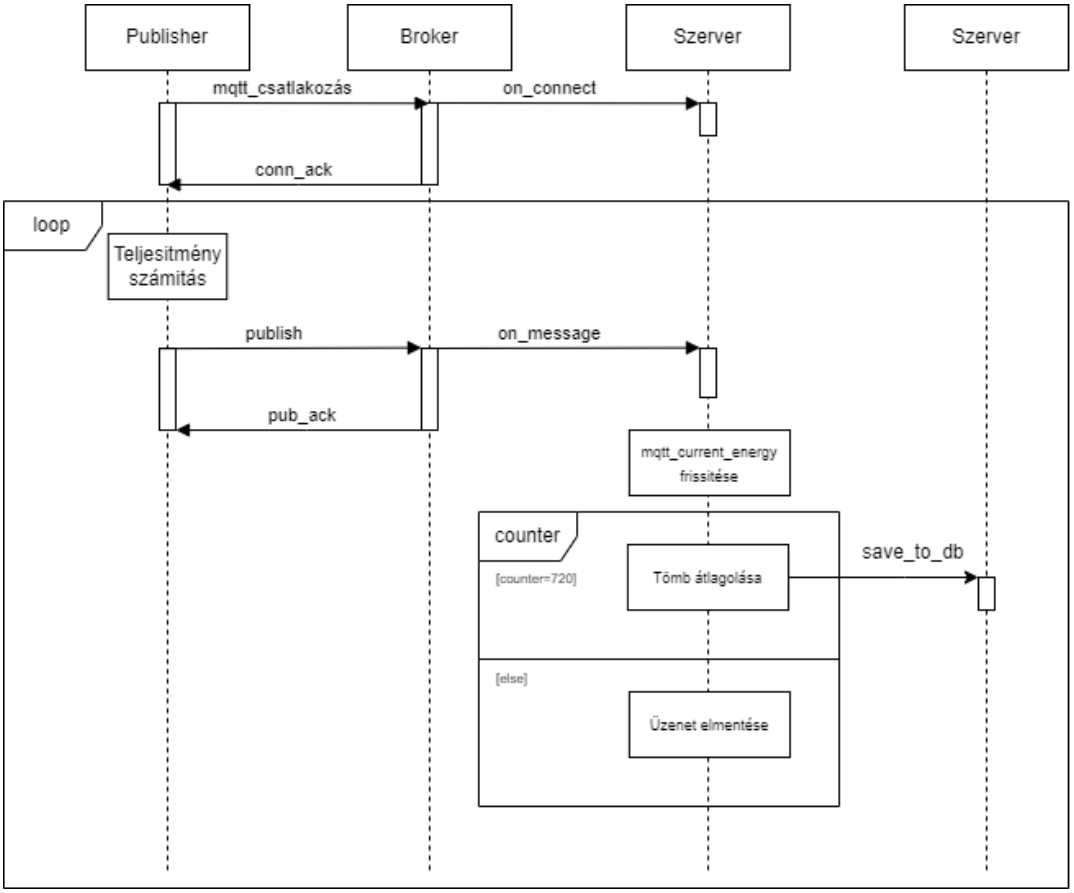



*19. ábra Esp32 mikróvezérlő folyamatábrája*

## 4.4. Raspberry Pi szervergépen futó szoftver

A szerver oldalon a MQTT üzenetek kezeléséhez szükség van egy MQTT brókerre. Ehhez a projekt során egy sajátos megoldást valósítottam meg a Raspbian OS rendszerre telepíthető Mosquitto MQTT bróker [18][19] segítségével. Miután a kapcsolat létrejött és az üzenetek sikeresen megérkeztek, a szerver oldali kód Python környezetben került implementálásra.

A Python kódban a Paho MQTT kliens könyvtárat használtam az MQTT üzenetek kezelésére. A készülék kliensek létrehozásakor az mqttClient_letrehozas() függvény hívódik meg, amely inicializálja a készüléket és beállítja a on_connection() és on_message() függvényeket. A on_connection() függvény jelez a szervernek, hogy az eszköz sikeresen csatlakozott a brókerhez. A on_message() függvény pedig kezeli a brókertől érkező üzeneteket, elmenti egy ideiglenes tömbbe és továbbítja az "mqtt_current_energy" gyűjteményhez, hogy az adatok elérhetőek legyenek a mobilalkalmazás számára. Ezenkívül az üzeneteket egy órás időközönként gyűjti össze, amit egy számláló segítségével valósít meg. Ha a számláló eléri a 720 értéket, akkor az adatokat átlagolja és elmenti a MongoDB adatbázisba.




*20. ábra MQTT üzenetfeldolgozás*

A szerveroldalon egy Flask keretrendszer fut a háttérben, amely felelős a REST API kérések kiszolgálásáért. A szerverprogram kezdetén egy külön szálban elindítunk egy webszervert, hogy a kéréseket párhuzamosan kezelhessük a MQTT kliens szálakkal. A Flask keretrendszer lehetővé teszi, hogy az URL útvonalakhoz rendeljünk függvényeket a app.route dekorátor segítségével. Ez azt jelenti, hogy definiálhatunk különböző útvonalakat (pl. "/devices", "/producers") és hozzájuk rendelhetünk olyan függvényeket, amelyek a kérésre válaszolnak. A GET kérések esetében válaszként visszaküldjük a jelenleg elérhető eszközök gyűjteményét.

```
@app.route('/devices', methods=['GET'])
def GET_devices():
return jsonify(mqtt_current_energy)
```

*21. ábra GET kérés*

Amikor a mobilalkalmazás POST kérést küld, JSON formátumban érkeznek az adatok, amelyek meghatározzák az új eszköz létrehozásához szükséges információkat. Az adatok között szerepelhetnek olyan mezők, mint a deviceName (eszköz neve), deviceID (topic neve) és deviceType (eszköz típusa). Ezeket az adatokat felhasználva frissítjük a jelenleg elérhető eszközök gyűjteményét (mqtt_current_energy), majd egy új szálat indítunk a MQTT kliens számára.

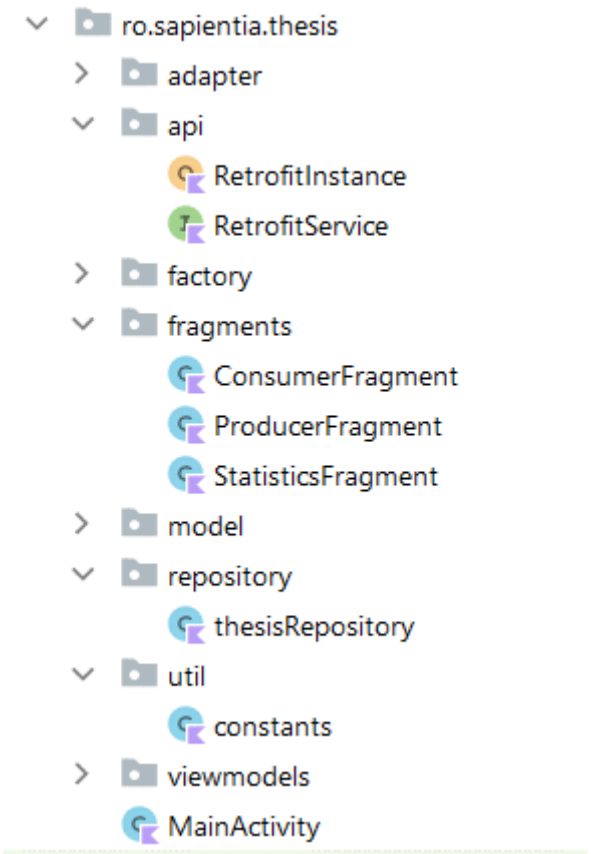
```
@app.route('/add_device',
methods=['POST'])

def POST_device():

    data = request.get_json()

    device_name =
data.get('deviceName')

    device_ID = data.get('deviceID')

    device_image =
data.get('deviceImage')

    ……

    response = {

        "message": "Received
successfully"

    }

    return jsonify(response)
```

*22. ábra POST kérés*

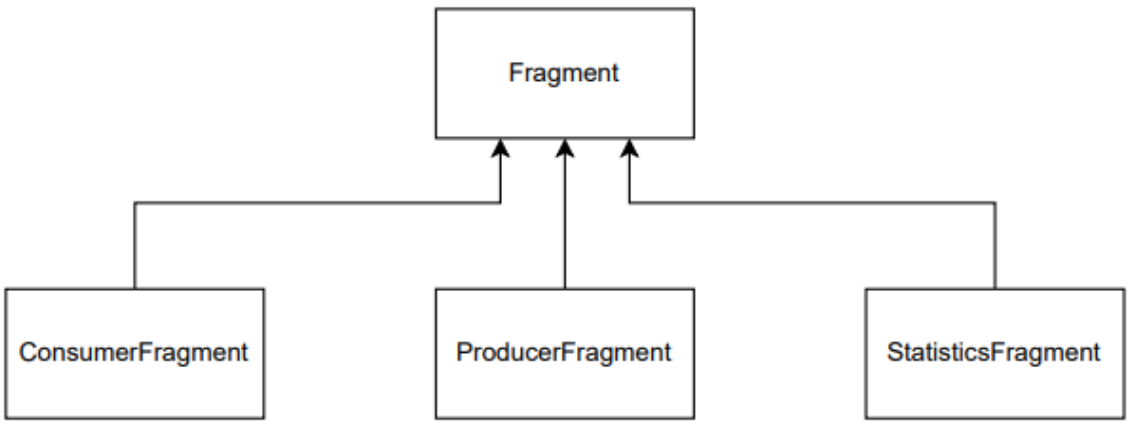
## 4.5. Mobilalkalmazás

A backend megírása után a következő lépés a mobilalkalmazás tervezése. Megterveztem, hogy milyen legyen a struktúrája, a különböző feladatkörökkel foglalkozó komponensek külön csomagokban legyenek elhelyezve.

Az alkalmazás egy fő Activity-t fog tartalmazni, amelyen belül három fragment kerül elhelyezésre, mindegyik különböző funkcionalitással rendelkezik. A fragmentek közötti navigációhoz egy Bottom Navigation komponenst használunk.




*24. ábra Mobilalkalmazás struktúrája*

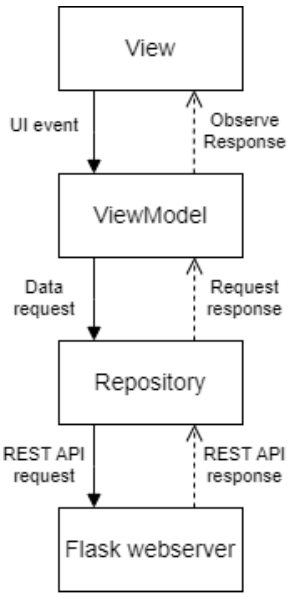



*23. ábra Fragmensek blokkdiagramja*

Az alkalmazásunknak hozzáférésre van szüksége a backend adataihoz, valamint kommunikálnia kell vele. Ehhez a manifest állományban engedélyeznünk kell az internet-hozzáférést az alkalmazás számára. A REST API-kérések elküldésére a Retrofit szolgáltatás felel, amely az adott IP-címre vagy URL-címre küldi a kéréseket. Az üzeneteket JSON formátumban továbbítjuk a kérések során.

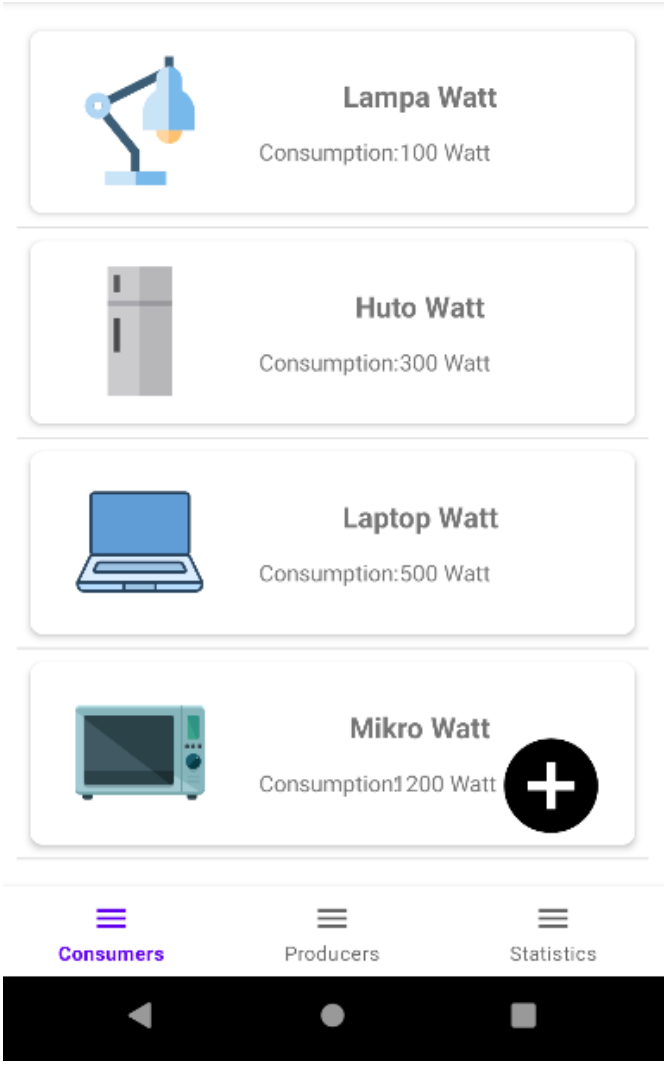

*25. ábra Alkalmazás - szerver kommunikáció diagram*

Az adatok valós idejű figyeléséhez a ViewModel használja a REST API kéréseket a repository osztályon keresztül, hogy lekérje az aktuális adatokat. Ezek az adatok egy MutableLiveData listában lesznek eltárolva, ami dinamikusan frissülhet. Az elmentett adatokat a ViewModel, továbbítja a View komponensnek, amely felel a vizuális felület megjelenéséért.



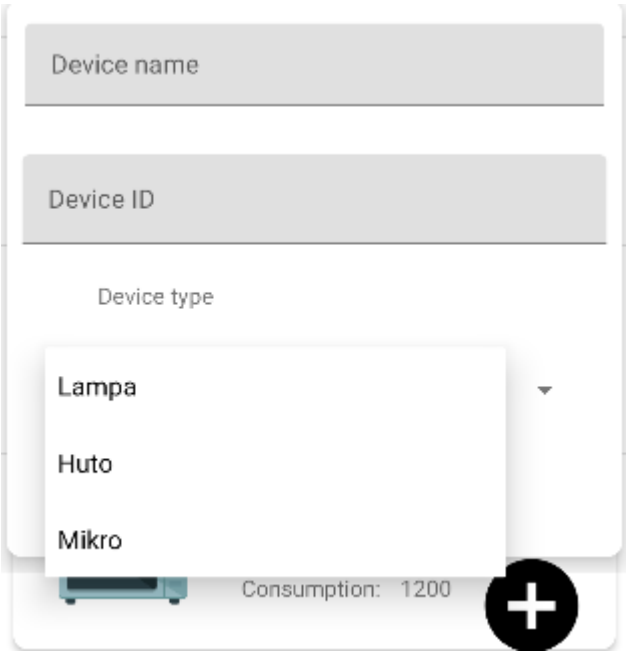

*26. ábra MVVM architektúra*

A fogyasztók és termelők fragmensek keretén belül egy RecyclerView lista elem található, ahol meg lehet figyelni az elérhető eszközöket, ennek nevei, képe és a teljesítmény értéke.




*28. ábra Eszközök listázása*

Újabb eszköz hozzáadásához a plust gombra rákattintva, a vizuális felület előhoz egy ablakot, ahol megadhatjuk az új eszköz nevét és ID-ját, illetve típusát. Az ID az megfogja határozni, hogy milyen MQTT topicra fog feliratkozni.

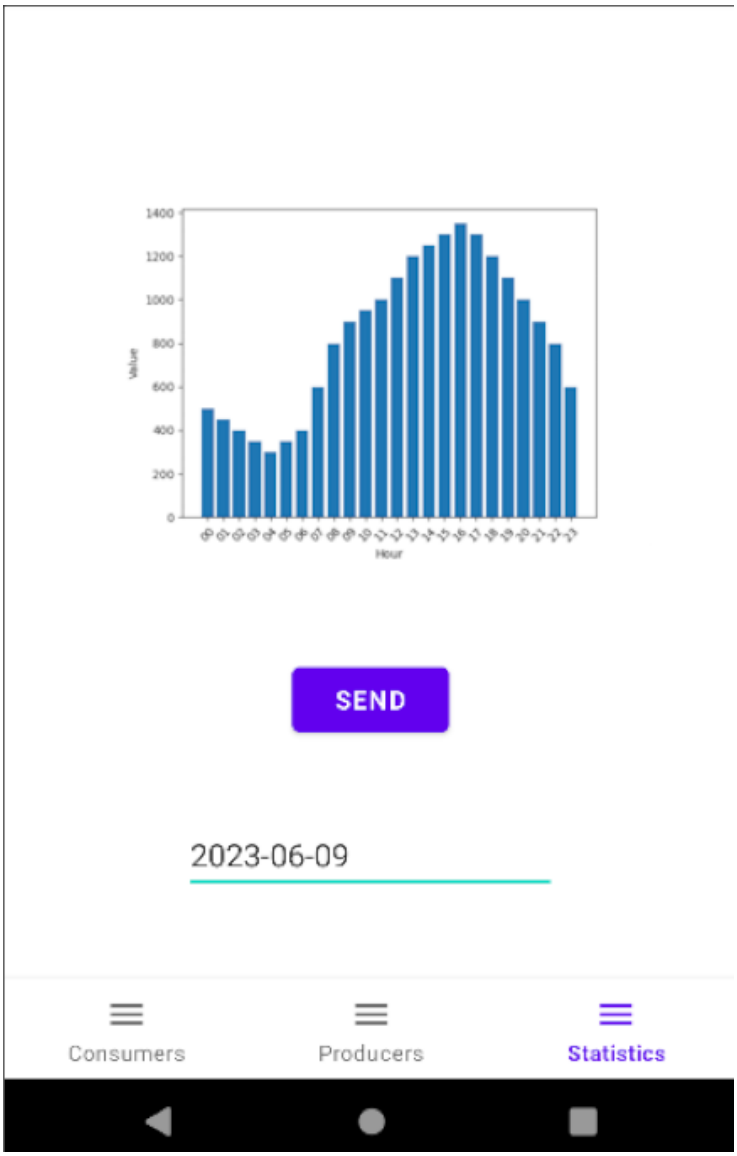



*29. ábra Eszközök hozzáadása*

Az eszköz módosítása és eltávolítása esetében, az adott lista elemre hosszan nyomva a vizuális felület előhoz egy popup ablakot, ahol kiválaszthatjuk a kívánt műveletet. Módosítás esetében, megváltoztathatjuk az eszköz nevét, ID-ját és típusát, törlés esetében eltávolítja a listáról.


*30. ábra Eszközök módosítása / eltávolítása*

A statisztikák lekérésére a StatisticsFragment-en belül meg kell adni egy adott dátumot „YYYY-MM-DD” formátumban, hogy le lehessen kérni arra a napra az átlag fogyasztást, ezt a szervertől egy kép formátumban kapja meg, és megjeleníti a vizuális felületen.




*31. ábra Statisztika lekérés*

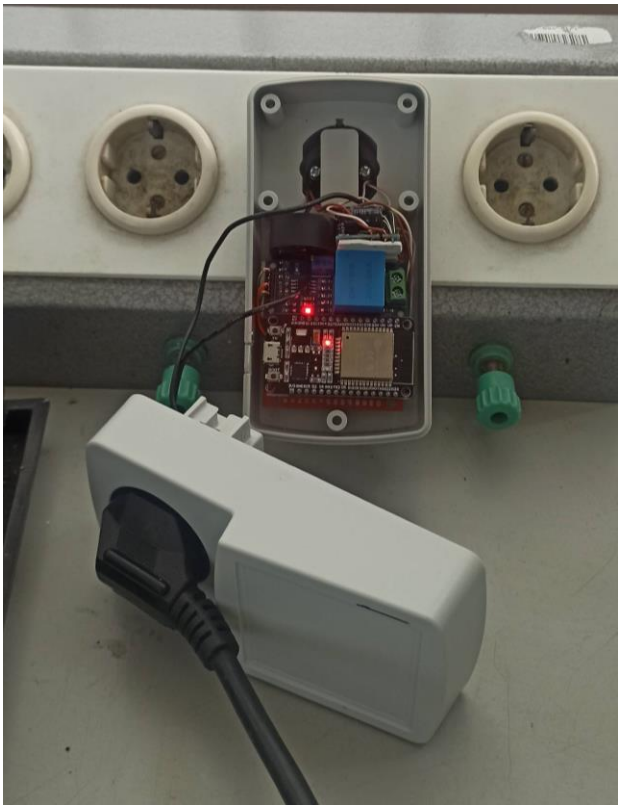
## 4.6. Verziókövetés

A dolgozatomnak a kódbázisának a verziókövetését a GitHubon valósítottam meg, amely a következő címen elérhető: https://github.com/xaverboss/Sapientia-thesis-2023-Energy-management-system . A main branchben található az esp32 mikrovezérlőnek és az mqtt/flask szervernek a forráskódja, illetve a teszteléshez és méréshez alkalmazott kódoknak a forrása, a master branch-en található a mobilalkalmazásnak a forráskódja.

# 5. Üzembe helyezés és kísérleti eredmények

## 5.1. Üzembe helyezési lépések

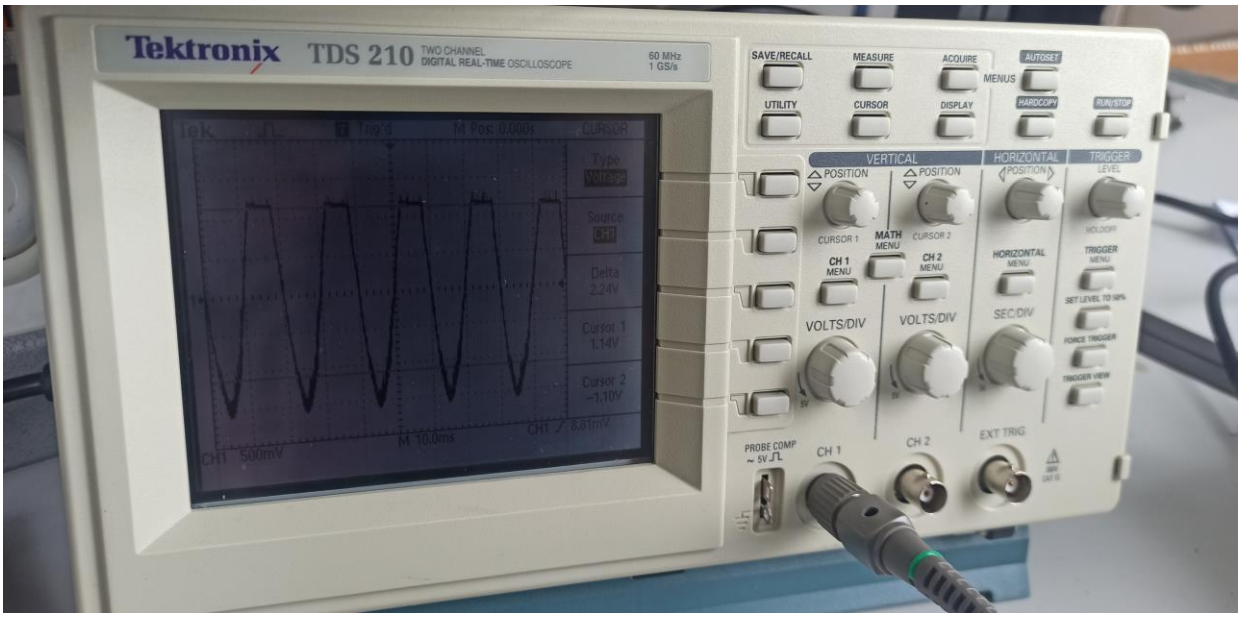
A prototípus rendszerét a beüzemeléshez egy lekapcsolható vagy kihúzható tápellátással kell ellátni, majd a mérendő berendezést csatlakoztatni kell az eszközhöz. Miután az eszköz biztonságosan elhelyezésre került, a tápellátás biztosítékát be lehet kapcsolni, és figyelemmel lehet kísérni az eszköz és a mérendő berendezés működését.



*32. ábra Üzembe helyezés*

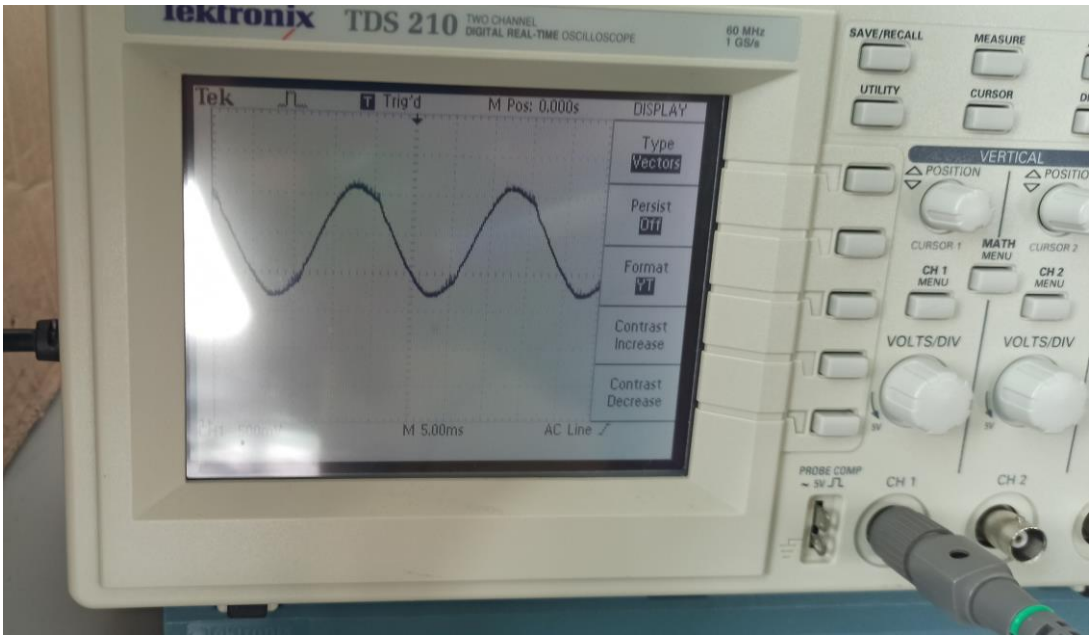
## 5.2. Kísérleti eredmények, mérések

Az okos mérőeszközöm helyes működésének a feszültség és az áramerősség mérése, a labor környezetben használt Tektronix TDS 210 oszcilloszkópot használtam. Kezdetben az oszcilloszkóp szondájával megmértem a ZMPT101B feszültség szenzornak a kimeneti lábait, a helyes mérés tesztelésére. Első körben, ahogyan az (ábra) mutatja, lapított csúcsú szinusz jelet kapunk. Ennek korrigálásához a szenzornak az ADC kimenetét kellet beállítani, amelyet a lapon lévő potenciométerrel lehet finomhangolni. Az oszcilloszkópon meg lehetett figyelni a 220 V szinuszos feszültséget, igy meg lehet győződni róla, hogy a szenzor helyesen mér.



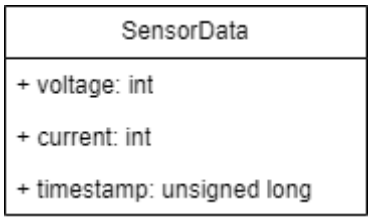

*33. ábra ZMPT101B szenzor mérése*

Az áramerősség szenzor méréséhez az oszcilloszkóp szondáját a söntellenállásra helyeztem. A mérendő berendezés bekapcsolásakor, meg lehetett figyelni a szinuszos jelét, illetve, hogy a jelnek a csúcsai 0-3V skála között van.




*34. ábra HWCT szenzor mérése*

A valós mérések után, már van egy alap, amihez lehet viszonyitani a mikróvezérlőn mérendő nyers adatokat. Egy külön kódot valósítottam meg, ahol a bemeneti pinekre érkező nyers adatokat egy adatosztályba mentettem el.



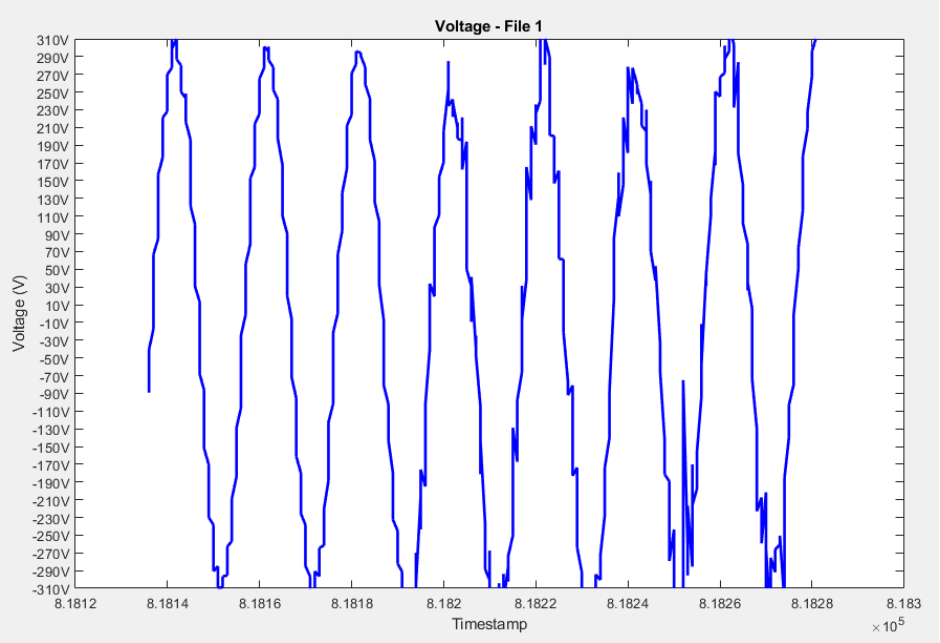

*35. ábra SensorData adat osztály*

A rendszerben 50 Hz-es frekvencia van, amely azt jelenti, hogy 20 milliszekundumonként kell mintavételezni az adatokat. Ezeket az adatokat egy SensorData típusú tömbben lesznek tárolva. A mintavételezés egy teljes percen keresztül történik, és ez idő alatt számos minta kerül rögzítésre. Miután az adatgyűjtés befejeződött, az adatokat az MQTT protokollon keresztül tovább küldöm a szerverre. A szerver oldalon a beérkező adatokat különválasztva egy CSV (Comma-Separated Values) fájlba lesznek elmentve.

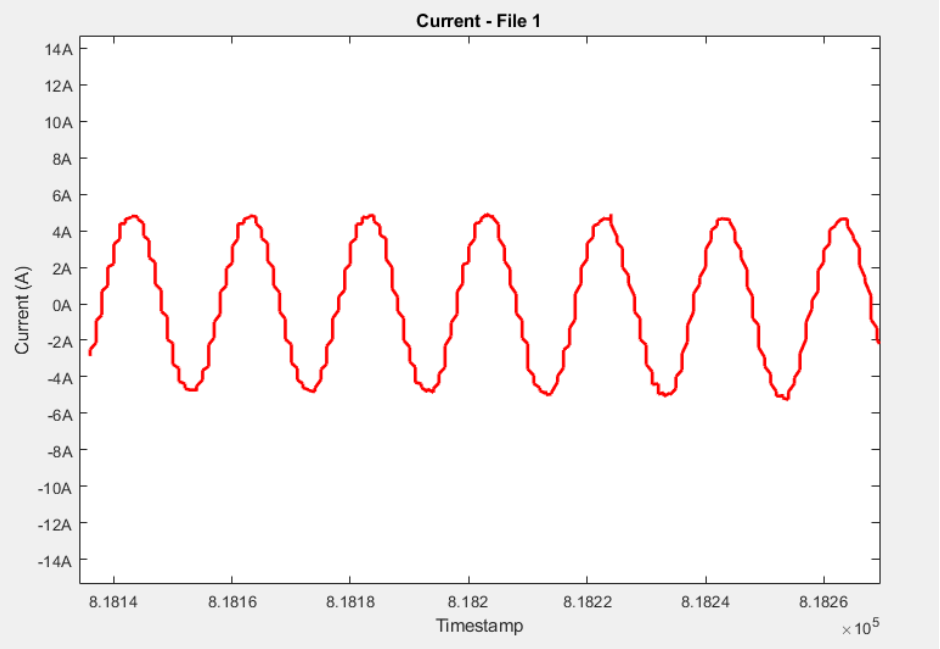
| 1 | Voltage | Current | Timestamp |
|---|---|---|---|
| 2 | 1659 | 1415 | 818136 |
| 3 | 1712 | 1431 | 818136 |
| 4 | 1742 | 1458 | 818136 |
| 5 | 1782 | 1503 | 818137 |
| 6 | 1813 | 1531 | 818137 |
| 7 | 1840 | 1563 | 818137 |
| 8 | 1873 | 1595 | 818137 |
| 9 | 1899 | 1616 | 818137 |
| 10 | 1926 | 1648 | 818137 |
| 11 | 1957 | 1685 | 818138 |
| 12 | 1978 | 1722 | 818138 |

*36. ábra Nyers adatok csv állománya*

A CSV állományban elmentett adatokat egy Matlab script segítségével tudtam ábrázolni külön-külön a feszültség és az áramerősség jeleket. Az 50 Hz frekvenciájú váltóáramú rendszerünkben az ábra kimutatja, hogy a feszültség csúcsértéke eléri a 310V értéket, illetve egy 650 W teljesítményű fűtőtest esetében az áramerősség csúcsértéke eléri a 4,80A értéket.

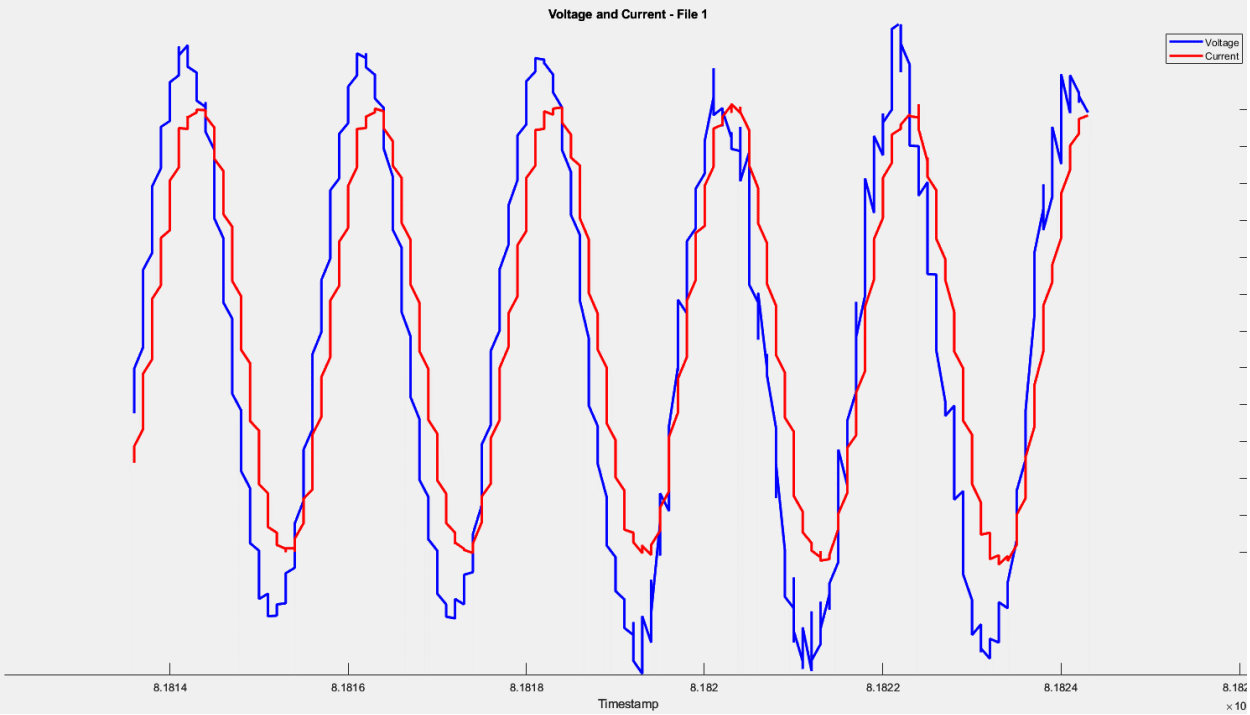



*37. ábra Feszültség csúcsértéke*




*39. ábra Áramerősség csúcsértéke*

Az áram és a feszültség jeleit egy diagramon ábrázolva látható, hogy az áramerősség jele időben el van tolódva a feszültségéhez képest. Ennek következtében, hogy az eszköz képes az induktív jellegű fogyasztókat mérni, illetve a reaktív teljesítmény számitására.




*41. ábra Fázis eltolódás észlelése*

## 5.3. Felmerült problémák és megoldásaik

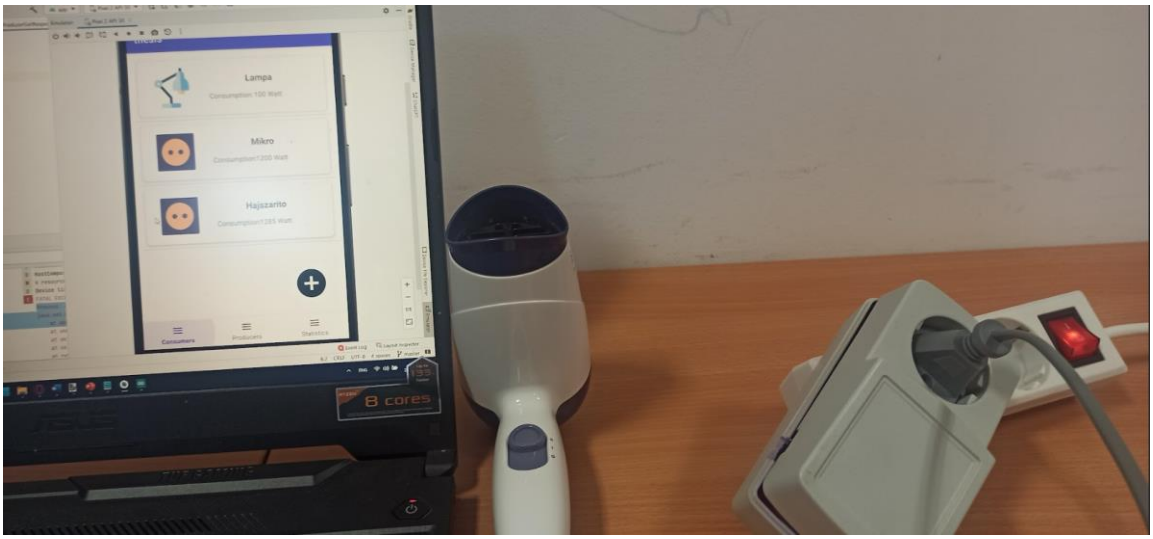
Az általam használt ESP32 Devkit V1 modul kiválasztásakor nem voltam teljesen tisztában az ADC interfészek korlátaival. Az eredeti tervezés során a 25 és 26-os pineken levő ADC2 interfészt használtam a feszültség és áramerősség mintavételezéséhez. Azonban kiderült, hogy amikor a mikrovezérlőn lévő Wi-Fi modult használom, az ADC2 interfészeket lekapcsolja, ami problémát okozott a mérési folyamatban. Úgy döntöttem, hogy a 33-as és 36-os pinekre kötöm a bemeneti jeleket, mivel ezek a pinek az ADC1 interfészhez tartoznak, és nem érintették a Wi-Fi modul működését.

A mikrovezérlőn üzenetküldésre használt MQTT protokoll használatakor, azzal szembesültem, hogy az üzenetek, amelyek meghaladják a 256 byte-ot azok nem érkeztek meg a szerver brokerhez. Az „PubSubClient.h” alap konfigurációk alapján az üzenet csomagoknak a maximális mérete be volt állítva 256 byte-ra és azok a csomagok, amelyek meghaladják ezt a méretet, nem lesznek figyelembe véve. A megoldás az volt, hogy a „PubSubClient.h” könyvtárban kicseréltem MQTT_MAX_PACKET_SIZE értéket 256 megabyte-ra.

A komponensek közötti kommunikáció létrehozásakor nagy gondot okozott az egyetem által biztosított nyílt internet használata, ugyanis a hálózaton vannak beépített védelmi eljárások, amelyek tiltanak bizonyos fejlesztési funkciókat. Ezáltal a saját okostelefonom megosztható hálózatát használtam a fejlesztés során. Legtöbbször nem volt a legjobb megoldás, ugyanis Raspberry Pi IP címe folytonosan változott, igy minden eszköznek külön be kellett konfigurálni, hogy milyen címre csatlakozzon.

# 6. A rendszer felhasználása

A készülék használatához először be kell helyezni a tápforrásba, majd csatlakoztatni kell a mérni kívánt eszközt a készülékhez. Ezután a mobilalkalmazás segítségével figyelemmel lehet követni a pillanatnyi teljesítményfogyasztást.




*42. ábra Rendszer üzembe helyezése*

# 7. Következtetések

## 7.1. Megvalósítások

A dolgozatomban sikerült tervezni és megvalósítani egy okos energia mérőeszközt, amely képes mérni egy háztartási berendezés teljesítményét. Az eszköz továbbítja ezeket az értékeket egy saját MQTT brókerhez és webszerverhez, ahol az adatokat feldolgozhatom és egy adatbázisba menthetem.

A mobilalkalmazásban lehetőség van új eszközök csatlakoztatására és azok kilistázására, így nyomon követhetem a berendezések teljesítményét. Emellett lekérhetek statisztikákat a napi átlagfogyasztásról óránként beosztva.

## 7.2. Továbbfejlesztési lehetőségek

A dolgozat tovább fejlesztéséhez a következő lehetőségek sorolhatóak fel:

- A rendszer telepítése napelemes rendszerekhez
- Relé modulok telepítése, adott teljesítmény értéknél az eszköz lezárja a fogyasztó működését.
- Relé modulok segítségével eldönthetjük, hogy mely eszközök legyenek működésben
- Az okos mérőeszköz méretének csökkenésére, SMD komponensekkel alkalmazott áramkör tervezése
- A megmért adatokkal megalkotott modellel betanított mesterséges intelligenciával és egy időjárás előre jelző API szolgáltatással meg lehet becsülni az átlag termelést/fogyasztást, illetve az esetleges nyereséget/vesztességet

# 8. Irodalomjegyzék


[1] Jasim, Nabaa Ali, and Haider ALRkabi. "Design and Implementation a Smart System for Monitoring the Electrical Energy based on the Internet of Things." Wasit Journal of Engineering Sciences 10, no. 2 (2022): 92-100.

[2] Macheso, Paul Stone, and Doreen Thotho. "ESP32 Based Electric Energy Consumption Meter."

[3] Nobile, Giovanni, Mario Cacciato, and Ester Vasta. 2022. "Measuring Active Power as the Difference between the Peak Value of Instantaneous Power and the Apparent Power" Sensors 22, no. 9: 3517. https://doi.org/10.3390/s22093517

[4] https://urjanet.com/blog/top-5-tips-successful-energy-management/

[5] https://www.viessmann.ro/ro/care-este-diferenta-intre-un-sistem-fotovoltaic-on-grid-unul-off-grid-si-unul-hibrid.html

[6] https://homescape.solar/blog/how-to-plan-solar-panel-installation-for-your-home/

[7] https://en.wikipedia.org/wiki/Alternating_current

[8] https://www.weschler.com/reference/guides/ac-power-measurement-guide/

[9] ESP32 dokumentáció. http://esp32.net

[10] Hudson, Glyn, and T. Lean. "OpenEnergyMonitor." https://docs.openenergymonitor.org.

[11] MQTT dokumentáció - https://www.hivemq.com

[12] Upton, Eben, and Gareth Halfacree. Raspberry Pi user guide. John Wiley & Sons, 2016.

[13] Daniel Gaspar . Mastering Flask Web Development, Second edition, Packt Publishing 2018.

[14] Banker, Kyle, Douglas Garrett, Peter Bakkum, and Shaun Verch. MongoDB in action: covers MongoDB version 3.0. Simon and Schuster, 2016.

[15] Masse, Mark. REST API design rulebook: designing consistent RESTful web service interfaces. " O'Reilly Media, Inc.", 2011.

[16] Android Studio dokumentáció - https://developer.android.com/docs

[17] Abubakar, I., S. N. Khalid, M. W. Mustafa, Hussain Shareef, and M. Mustapha. "Calibration of ZMPT101B voltage sensor module using polynomial regression for accurate load monitoring." Journal of Engineering and Applied Sciences 12, no. 4 (2017): 1077-1079.

[18] Mosquitto broker dokumentáció - https://mosquitto.org/documentation/

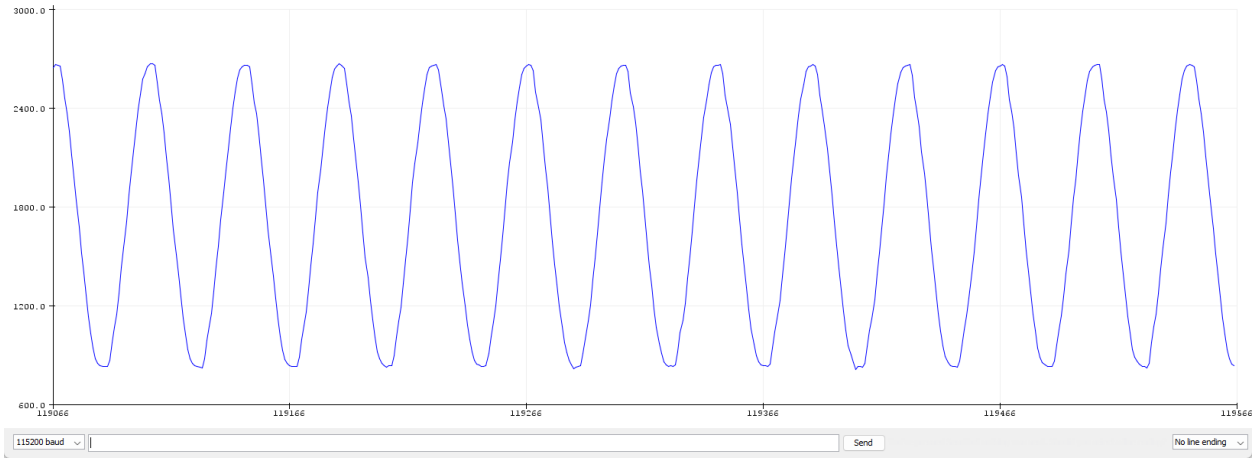
[19] https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/

# 9. Függelék

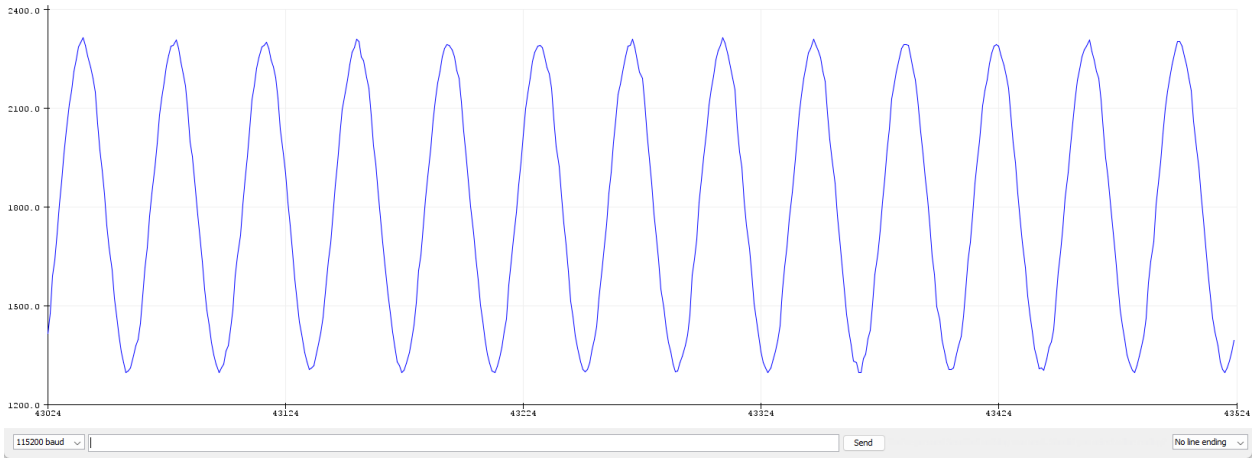
*43. ábra Függelék: Áramkör tokozása*

*44. ábra Függelék: Áramerősség mérése*

*45. ábra Függelék: Rendszer tesztelése labor környezetben*




*46. ábra Függelék: feszültség mintavételezésének ábrája*




*47. ábra Függelék: Áramerősség mintavézelezésének ábrája*

```
Received MQTT message #0: 1244
0
Received MQTT message #4: 1244
4
Received MQTT message #5: 1241
5
Received MQTT message #1: 1241
1
['1230', '1216', '1237', '1241', '1244']
rezso
esp32/energy1
['1230', '1216', '1237', '1241', '1244']
{'device_id': 'rezso', 'timestamp': '2023-06-28 08:49:21', 'topic': 'esp32/energy1', 'value': 1233.6}
Inserted document ID:  649be6012363c5b625377cb5
```

*48. ábra Függelék: logolás a szerveren*

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA

FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ

SPECIALIZAREA CALCULATOARE

Vizat decan

Conf. dr. ing. Domokos József

Vizat director departament

Ș.l. dr. ing Szabó László Zsolt