Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Automatika Intézet

Dobó Krisztián

Idősfelügyelő óra tervezése

PROJEKTMUNKA

KONZULENS
Borsos Döníz

BUDAPEST, 2025

Tartalomjegyzék

1. Bevezetés

Az idősödő népesség számának növekedésével egyre nagyobb igény mutatkozik olyan eszközökre, amelyek képesek valós idejű felügyeletet biztosítani. A technológiai fejlődés lehetőséget biztosít arra, hogy olyan eszközöket készítsünk, amellyel távolról is felügyelhetjük az idősemberek állapotát.

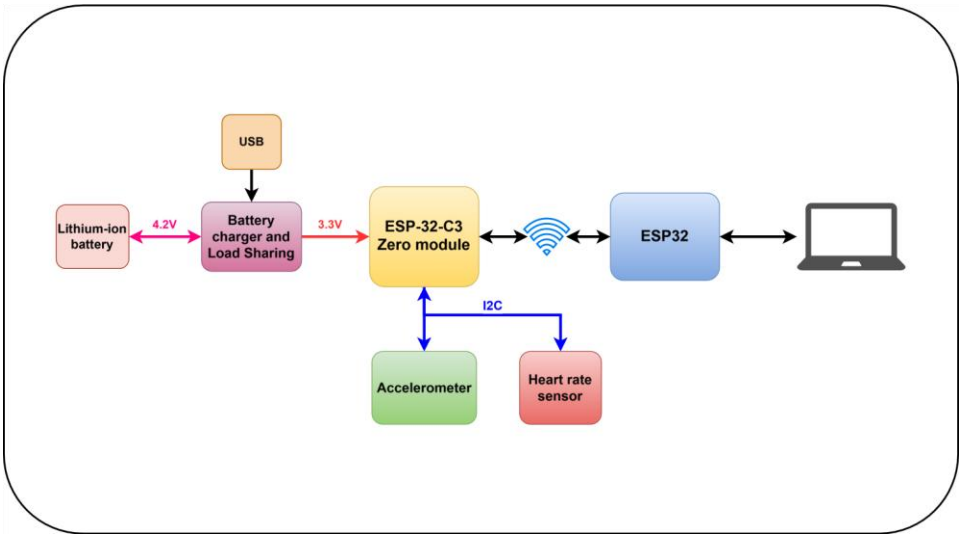
Az idős emberek számára az esés egy gyakori és súlyos következményekkel járó esemény, amely nemcsak fizikai sérüléseket okozhat, hanem csökkentheti a mozgékonyságukat. A technológia fejlődése lehetőséget nyújt az ilyen események időben történő észlelésére és a segítségnyújtás gyorsítására, ezzel is hozzájárulva az idősek biztonságosabb életviteléhez.

Ezen projekt célja egy olyan idősfelügyelő óra tervezése és prototípusának megvalósítása, amely magába integrálja az esése detektálás, pulzusmérés és GPS-alapú nyomkövetés funkciókat. Ezen technológiák együttesen lehetővé teszik az idősek állapotának monitorozását, valamint gyors beavatkozást tesznek lehetővé vészhelyzet esetén. A projekt során különös figyelmet fordítottam az energiahatékonyágra, mivel az akkumulátoros üzemmód kulcsfontosságú egy hordozható eszköz esetében.

A dokumentum további részében bemutatásra kerül a használt hardver, a szenzorok működése, a mérési eredmények bemutatása és a tovább fejlesztési lehetőségek.

2. Teszt áramkör

A projektfeladatomban a mérések elvégzéséhez és a szenzorok teszteléséhez készítettem egy áramkört, amelyen tudtam tesztelni a gyorsulás szenzort és a pulzus mérőt. A szenzor értékek kiolvasásához és az adatok továbbításához egy ESP-32-es mikrovezérlőt használtam. A mérések során az ESP-32 a mért adatokat az Espressif által fejlesztett ESP-NOW-os kommunikációs protokollon keresztül küldtem át, az adatokat egy másik ESP fogadta és küldte tovább a mért eredményeket az adatrögzítő számítógép felé. Az alábbi képen a teszt áramkör blokkvázlata és a mérési elrendezés látható.



1. ábra Teszt áramkör blokk vázlata és mérési elrendezés

2.1. Esés detektálás

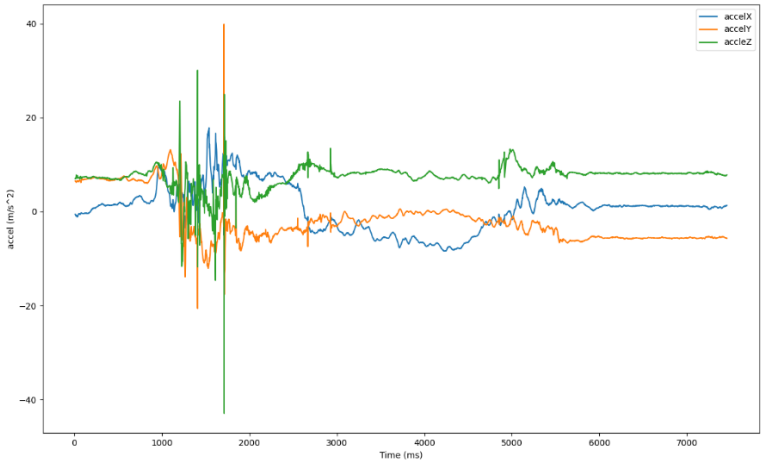
A szakirodalmak többségében az esés detektálásához a gyorsulás szenzor által mért értékekhez valamilyen küszöbértéket rendelnek melynek átlépesekor az adatok feldolgozása során esésként határozzák meg ezen kiugró értékeket. Ennek a módszernek a hátránya, hogy ha a küszöbérték nincs jól megválasztva a kiugró értékek miatt a mért gyorsulásban bármilyen hirtelen, gyors mozgást esésként értékelhet ki az algoritmus. Az esést nem csak egy küszöb érték átlépésével lehet észlelni, hanem valamilyen gépi tanulási módszerrel is. A gépi tanulási módszerben nagy minta adatbázissal kell rendelkezni, hogy belehessen tanítani az adott modellt. Ennek a módszernek a hátránya, hogy nagy tanító adatbázisra van szükség és bonyolult számításokat kell elvégezni egy esés detektálásához, erre a megoldásra van példa a [2] cikkben. A gépi tanulási megoldás a bonyolult számítások miatt egy fogyasztás optimalizált rendszerben nem biztos, hogy praktikus megoldás lehet.

Mivel a küszöbértéken alapuló módszerek nem igényelnek bonyolult számításokat ezért fogyasztás szempontjából is kedvezőbb lehet az ezen a módszeren alapuló esés detektálási módszer.

A projektfeladatom megvalósításában egy 6 tengelyes IMU (Inerciális mérő egység, **I**nertial **M**easurement **U**nit) szenzort használtam, a Bosch Sensortec által fejlesztett BMI270-es gyorsulás mérőt.

A mérések során a gyorsulás mérő mérési frekvenciája 400 Hz-re volt állítva és a giroszkóp ki volt kapcsolva. Mivel az emberi mozgás egy viszonylag lassú mozgás ezért ez a mérési frekvencia elégnek bizonyulhat. Ezen kis mintavételi frekvencia mellet be lehet kapcsolni a szenzor energiatakarékos módját így a teljes működése során a fogyasztása minimális lesz, amely ideális lehet az akkumulátoros rendszerben történő felhasználásra.

A mérés során a fenti konfigurációnak megfelelően 2,5 ms-ként küldött egy megszakítást az ESP felé a gyorsulás szenzor, a szenzornak a megfelelő kimenetén. Minden megszakításkor ki lett olvasva a három gyorsulási érték a szenzorból. A kiolvasott értékek ezután átlettek küldve a másik ESP-nek, hogy az adat gyűjtő tudja rögzíteni a mért adatokat. A *2. ábra* látható esés estén a három tengely által mért gyorsulás értékek.

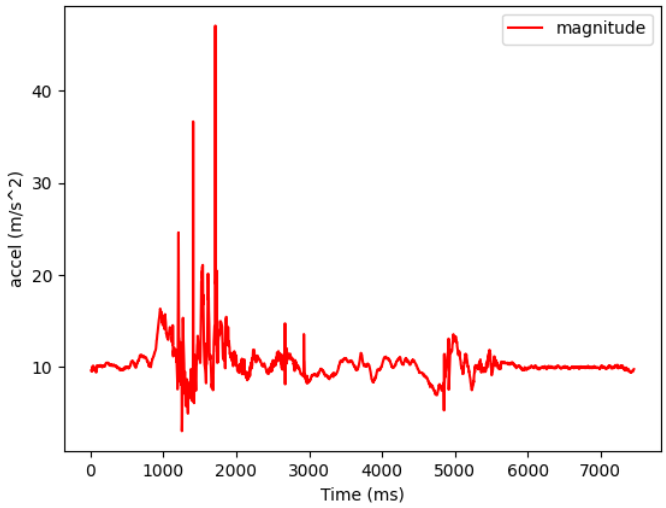


2. ábra Eséskor a három gyorsulási tengely értéke

Az ábrán látszik, hogy ütközéskor a három tengelyen megjelennek kiugró értékek. Mivel a három tengely értékét külön - külön vizsgálni nehéz és fölöslegesen bonyolítja a feldolgozást ezért egy egyszerű számítással a három tengely gyorsulás értékeit összevonom. A következő képlet segítségével kapom meg az összeggyorsulást:

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

ahol a_x, a_y és a_z a gyorsulás szenzor háromtengelyének mért értékei. Ebből az összevont értékből is ütközéskor egy egyértelmű csúcsnak kell meg látszódni az értékekben. A *3. ábra* látható a három tengely összevont értékének ábrázolása:



3. ábra A háromtengely összevont értéke

Esés detektáláshoz meghatároztam egy küszöbértéket, amit, ha az összevont gyorsulási érték átlép akkor eltárolom az átlépés időpontját és ha egy bizonyos időn belül történik egy újabb küszöb átlépés akkor azt esés ként kezelem. A következő függvénnyel detektálom az esést:

1. forráskód Esés detektálás

```
bool fallDetector(float mag)
{
  static uint32_t prevTime = 0, negPrevTime = 0;
  static bool isHigherPozTreshld = false;
  float frequency;
  static uint8_t sampleCounter = 0;
  static uint32_t lastSample = 0;

  if(millis() - lastSample >= 700)
  {
    sampleCounter = 0;
  }

  if(mag >= POSITIVE_TRESHOLD)
  {
    if(!isHigherPozTreshld)
    {
      isHigherPozTreshld = true;

      if(millis() - prevTime <= 200)
```

```
      {
        sampleCounter++;
        if(sampleCounter >= 3)
        {
          sampleCounter = 0;
          return true;
        }
        lastSample = millis();
      }
      prevTime = millis();
    }
  }

  if(mag < POSITIVE_TRESHOLD)
  {
    if(isHigherPozTreshld)
    {
      isHigherPozTreshld = false;
    }
  }
  return false;
}
```

A függvény az összeggyorsulás magnitúdóját figyeli, és 200ms-os időablakon belül három küszöbérték átlépést érzékel esésnek.

2.2. Pulzus mérés

A projektfeladatban egy óra szerű eszközt kell megvalósítani ezért egy olyan pulzus mérő szenzort választottam, amely a visszavert fényből képes meghatározni a pulzus számot. Azért választottam egy ilyen típusú szenzort mert így nem kell külön érzékelőket elhelyezni az adott személy testén a pulzus méréshez.

Ezekben a szenzorokban különböző fényforrásokat szoktak alkalmazni, tipikusan piros, zöld és infravörös fényt. Ez azért van mert minél hosszabb a fény hullámhossza annál mélyebbre hatol be a bőr alá, amellyel pontosabb mérési eredményeket lehet elérni [6]. Ettől függetlenül ezeknek a szenzoroknak a pontossága függ az adott cselekvéstől, a mozgás befolyásolhatja ezeknek a szenzoroknak a mérést [6]. A zöld fénnyel a vér oxigén tartalmát lehet meghatározni [6].

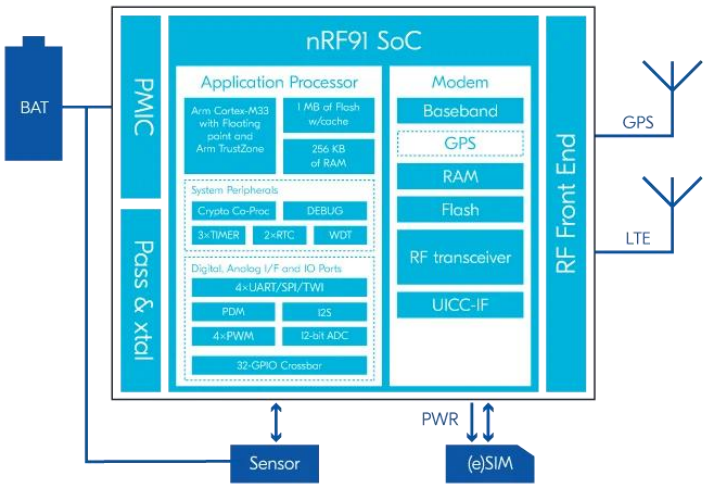
A tesztek során az Analog Devices által készített MAX30102-es pulzus és véroxigén szint mérőjét használtam. A teszt során a SparkFun által készített MAX3010X könyvtárat használtam (https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library). Ennek a könyvtárnak a használatával egyszerűen tudtam kezelni a szenzort. A mérések során tapasztaltam, a fent említett hátrányát a szenzornak miszerint a mozgás nagy mértékben tudja befolyásolni a mért értékeket.

3. Pozíció meghatározása

A projektfeladat kiírásban benne volt, hogy GPS-es nyomkövetést kell megvalósítani. Ennek a megoldására a Nordic Semiconductor által fejlesztett nRF91-es sorozatú SiP-t használtam, a Thingy:91 X-es fejlesztő panelon.

4. ábra Thingy:91 X fejlesztő panel [3]

Azért választottam ezt az eszközt, mert az nRF91 sorozat lehetőséget biztosít a globális műholdas navigációs rendszert (GNSS) használatára, amellyel meghatározható az eszköz pontos földrajzi helyzete. Emellett képes csatlakozni a mobilhálózatra LTE-M kapcsolaton keresztül, így a mért adatokat interneten keresztül továbbíthatjuk egy webszerver felé. Ennek köszönhetően a fejlesztőpanel bárhová magunkkal vihető, miközben folyamatosan kapcsolatot tarthat a szerverrel. Az LTE-M kapcsolat használatához szükség van egy SIM-kártyára, amely lehetővé teszi az eszköz azonosítását és hitelesítését a mobilhálózaton. Az nRF91-es architektúra blokkvázlata az *5. ábra* látszik:



5. ábra nRF91 architektúra blokkvázlata [4]

A blokkvázlatból látszik, hogy az architektúra két fő részre van bontva egy mikrovezérlős részre és egy modem részre. A két rész közötti kommunikációt a Nordic által

készített Modem könyvtár teszi lehetővé. Az alábbi kétfüggvénnyel lehet inicializálni a modem könyvtárat és engedélyezni a GNSS-es földrajzi pozíció meghatározást.

2. forráskód Modem könyvtár inicializálása és GNSS modul aktiválása

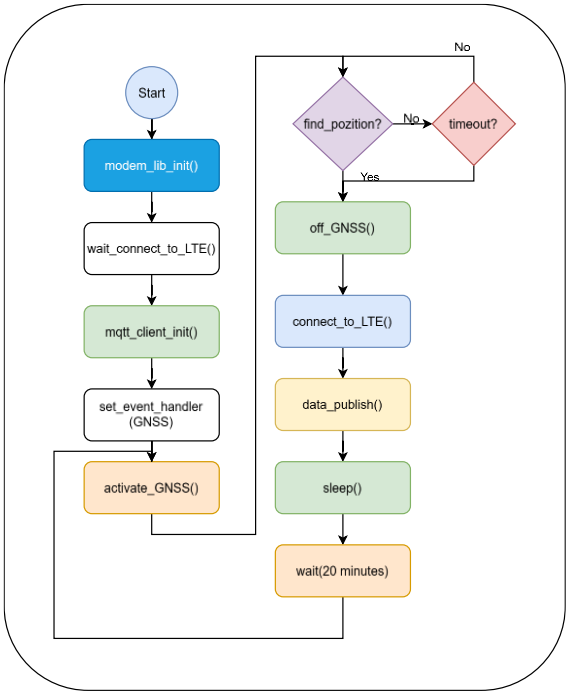
```
nrf_modem_lib_init();
lte_lc_func_mode_set(LTE_LC_FUNC_MODE_ACTIVATE_GNSS);
```

Miután aktiváltuk GNSS modult azután az eszköz a háttérben kiszámolja a pozícióját, ha megtalálta a pozíciót akkor egy flag-el jelzi, hogy kiszámította a pozíciót. A koordináták megtalálásának ideje függ attól is, hogy kültéren vagy beltérben tartózkodunk, valamint függ attól is, hogy mikor volt utoljára pozíció keresve, milyen régen volt az eszköz bekapcsolva. Ez alapján megkülönböztetünk három eshetőséget:

- Hideg indítás (Cold): első használat 2-4 perc is lehet mire talál pozíciót a gyártó szerint. A tesztek az mutatták, hogy ha épületen belül van akkor akár több mint 10 perc is lehet.
- Meleg indítás (Warm): egynapig van az eszköz kikapcsolva a gyártó szerint 45 másodperc alatt számítja ki a pozíciót.
- Forró indítás (Hot): kb. egy órát van kikapcsolva az eszköz a gyártó szerint 22 másodperc alatt számolja ki a pozíciót. [5]

A pozíció keresésnek az egyik feltétele, hogy a fejlesztő panelnek egyidőben legalább 4 műholdat kell látnia megfelelő térerővel, hogy meg tudja oldani a pozíció keresésben használt egyenletrendszereket. Tehát, hogy ha egyszerre több műholdnak is veszi a jelét nem biztos, hogy meg tudja oldani az egyenleteket, ha túl kicsi a befogott jelerőssége [5].

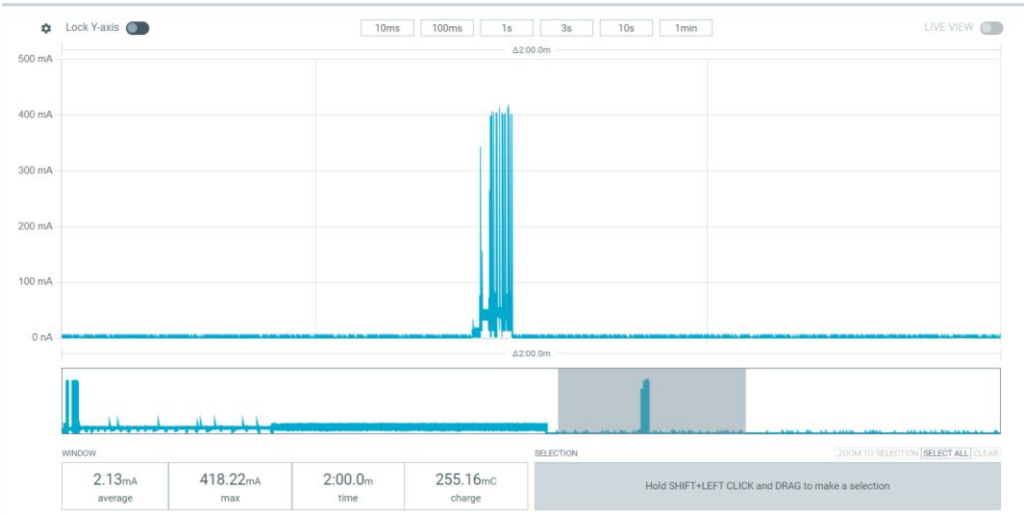
Ezeket felhasználva készítettem egy alkalmazást, amely 20 perces időintervallumokban keresi az eszköz földrajzi pozícióját miután sikerült kiszámítania a pozíciót, csatlakozik az LTE-M hálózatra és a mért adatokat elküldi egy HiveMQ MQTT brokernek. A következő ábrán látható a megvalósított szoftver folyamatábrája:



6. ábra Pozíció keresés folyamatábra

A tesztelés során volt olyan alkalom, hogy 10 - 15 perc után sem sikerült megtalálnia a pozíciót, de az esetek többségében sikeresen ki tudja számítani a koordinátákat akár 3 – 4 perc alatt is. Emiatt 5 perces időkorlát van beállítva a pozíció keresésre, mert ha nem találja a pozíciót, mert le van árnyékolva, a rendszer mindaddig ébren marad, amíg ki nem számítja az új pozíciót. A pozíció keresés közben folyamatosan be kell kapcsolva lennie a GNSS modulnak és mivel ennek a modulnak nagy az áramfelvétele ezért az akkumulátoros üzem közben korlátozni kell azt az időt ameddig aktív a GNSS modul. Miután sikeresen kiszámította a pozíciót vagy túllépte az időkorlátot utána küld egy üzenetet az MQTT brokernek amely tartalmazza az új pozíciót, ha nem sikerült kiszámolnia adott időn belül a pozíciót akkor az alap értékeket küldi a broker fele.

A fejlesztő panelon van lehetőség az nRF91-es modulnak mérni az áramfelvételét működés közben. Az IC áramfelvételét a Nordic Semiconductor által készített Power Profiler Kit II-vel mértem. Az adatlapja szerint képes 200nA-től 1A-ig mérni. A fejlesztő panelon van kialakítva egy csatlakozó, ahová egy kiegészítő panel illeszthető az árammméréshez. Az alábbi képen az látható, hogy adat küldéskor mekkora az áramfelvétele a kommunikációs modulnak.



7. ábra Áram felvétel adatküldés közben

Az LTE-M hálózatra való csatlakozás nagy áramfelvételt igényel 400mA csúcsokkal, de mivel az eszköz az idő nagyobb részében alvó állapotban, ahol az átlag fogyasztás 600µA körül van így ekkora fogyasztás mellett alkalmas lehet akkumulátoros működésre. Pozíció keresés közben az átlag fogyasztása kb. 45 mA szokott lenni. Ha csak a 600µA-es fogyasztást vesszük figyelembe akkor a fejlesztő panel 1350mAh-ás akkumulátoráról több mint 90 napig tudna üzemelni alvó állapotban, de ez az idő működés közben csökkenhet. Egy öt napos mérés során a fejlesztő panel akkumulátora nem merült le a beállított mérési időközzel és az 5perces mérési időkorlát mellett.

4. Küldött adatok mentése

Az MQTT brokernek küldött adatokat egy Python-ban írt script mentette el. A script ugyanarra a brokerre, ugyanazzal a felhasználó névvel és jelszóval csatlakozott. A Thingy:91 X a pozíció adatok mellet az UTC időt is küldi a *nrf-<ID>/GNSS* topicra. A script erre a topicra iratkozott fel és az onnan kapott adatokat mentette el. Egy üzenet az alábbi formában néz ki:

8. ábra Thingy:91 X MQTT üzenet

A Python scriptben figyelem, hogy melyik topicról érkezik az üzenet és ha egy ismertről érkezik az üzenet akkor abból reguláris kifejezések segítségével veszem ki a számomra releváns adatokat.

```
def processMSG(self, msg):
    string = msg.payload.decode("utf-8")

    cleaned_string = string.strip('\x00').strip().strip("'")

    match = re.search(r"Time \(UTC\): ([\d:]+), Latitude: ([\d.]+), Longitude:
([\d.]+)", cleaned_string)

    if match:
        self.Time = match.group(1)
        self.Latitude = match.group(2)
        self.Longitude = match.group(3)
        self.Date = datetime.now()

        print(f"Dátum: {self.Date}, GNSS time: {self.Time}, Latitude:
{self.Latitude}, Longitude: {self.Longitude}")
        self.write_file()
        self.write_to_json()
    else:
        print("Nem sikerült adatokat kinyerni a payloadból: ", cleaned_string)
```

Miután sikeresen kiolvasta az adatokat a kapott üzenetből akkor egy szöveg fájlba elmenti az adatokat az aktuális rendszer idővel. A későbbi feldolgozás miatt elmenti az adatokat egy json fájlba is.

```
{
  "device_id": "nrf-F98C",
  "timestamp": "2025-05-11T14:14:31.906826",
  "time (utc)": "12:14:00",
  "latitude": "46.200796",
  "longitude": "20.107783"
},
```

Ebben a json fájlban a tesztelés ideje alatt maximum egy hétig tarolódnak el az adatok és az egy hétnél régebbi adatokat az újabb mérési eredmények felülírják.

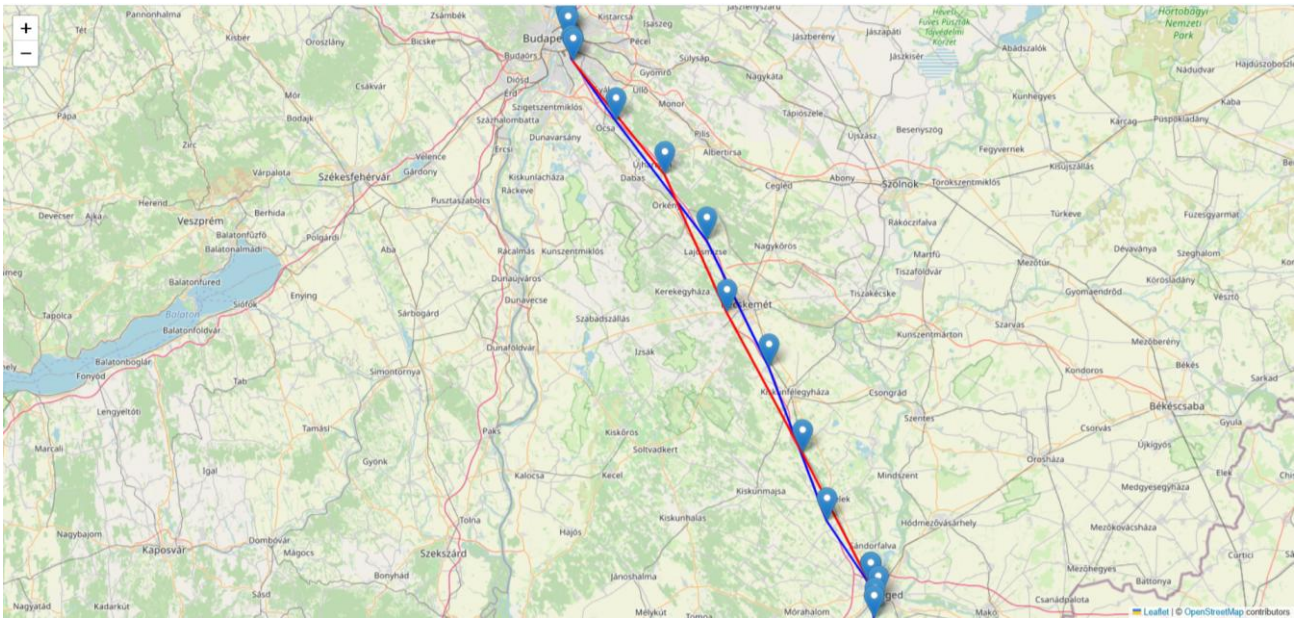
5. Eddig elért eredmények

A teszt mérések során elkészített esést észlelő függvénnyel sikerült a következő szituációkban meghatározni, hogy történt-e esés.

	Esés szám	Érzékelt esés szám
Jobbra esés	10	10
Balra esés	10	8
Előre esés	10	9
Hátra esés	10	8
Leülés	10	1

Az eredményekből látszik, hogy sikerült esést érzékelni különböző esetekben, de az is látszik az eddigi eredményekben, hogy módosítani kell a különböző határértékeket, hogy biztosabban érzékelje az esést. Valamint, hogy ne legyen fals jelzés leülés és egyéb más cselekvés közben.

A nyomkövetés során az elmentett pozíció adatokból sikeresen tudtam készíteni egy útvonalat, amiben látszik, hogy Szegedről milyen útvonalon jutottunk el Budapestre és onnan vissza.



9. ábra Útvonal Szegedről Budapestre

A jelölőkből látszik, hogy nem folyamatos volt a mérés, hanem 20 perces időközönként küldte a Thingy:91 X a pozíció adatokat az MQTT broker felé. A mentett adatokban az is látszik, hogy 4 alkalommal nem sikerült kiszámítania a pozíciót az 5 perces időkorláton belül. Ez

valószínűleg azért történt, mert Budapesten olyan helyen voltam, ahol nem volt kellően erős a befogott jel vagy mert nem látott egyszerre legalább 4 műholdat.

6. Összefoglalás

A projektfeladatom során külön – külön meglett oldva az esés érzékelés és a nyomkövetés. De ahogy az előző fejezetben is írtam van az esés detektálásban hiba. A projekt folytatásaként szeretném megoldani az esés detektálás problémáját. Mivel a BMI270-es szenzor tartalmaz giroszkópot is annak a mérési eredményeit is felhasználva egy stabilabb rendszert lehetne készíteni. Ez a gyorsulás szenzor képes a lépés számolásra ezt a funkcióját is szeretném kihasználni, hogy mérni lehessen vele a felhasználó napi aktivitását is.

A helymeghatározás jól működik, de vannak esetek amikor nem tudja meghatározni a pozíciót ezért valamilyen plusz megoldást kellene bevezetnem, amivel nem lennének kiesett adatok. Ilyen megoldás lehet például, hogy a GPS koordináta helyett az LTE-M hálózathoz való csatlakozáskor elküldené a torony azonosítót így egy nagyobb határon belül, de akkor is lenne pozíció adat.

Folytatásként szeretnék elkészíteni egy áramkört, amelyen egy panelon lenne megvalósítva a nyomkövetés, az esés detektálás és a pulzus mérés. Valamint készíteni szeretnék egy mobil applikációt, amely mutatná az éppen aktuális adatokat és jelezne, ha az eszköz esést detektálna.

Irodalomjegyzék

[1] „Development of a wearable-sensor-based fall detection system” 2014, [online dokumentum]: https://pubmed.ncbi.nlm.nih.gov/25784933/

[2] „Intelligent fall detection method based on accelerometer data from a wrist-worn smart watch” 2019, [online dokumentum]: https://www.sciencedirect.com/science/article/abs/pii/S0263224119303331

[3] https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/Nordic-Thingy91-X-PB-1.0.pdf

[4] https://academy.nordicsemi.com/courses/cellular-iot-fundamentals/lessons/lesson-1-cellular-fundamentals/topic/lesson-1-nrf91-series-sip/

[5] https://academy.nordicsemi.com/courses/cellular-iot-fundamentals/lessons/lesson-6-cellular-fundamentals/topic/lesson-6-gnss/

[6] „A review on wearable photoplethysmography sensors and their potential future applications in health care” 2019, [online dokumentum]: https://pmc.ncbi.nlm.nih.gov/articles/PMC6426305/