ETDK-dolgozat

Mayer Walter

Farkas Szilárd

XXVIII. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK)

Informatika II.: innovatív számítástechnikai termékek, alkalmazások szekció

Kolozsvár, 2025. május 15–18.

Pontos helymeghatározás Ultra-Wideband segítségével

Egy ígéretes technológia alkalmazási lehetőségei

Szerzők:

Mayer Walter

Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, III. év

Farkas Szilárd

Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, II. év

Témavezetők:

dr. Sulyok Csaba, egyetemi adjunktus,

Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar

Szopos Dávid, szoftverfejlesztő,

Codespring

Fazakas Lóránt-Tibor, szoftverfejlesztő,

Codespring

Pontos helymeghatározás Ultra-Wideband segítségével: Egy ígéretes technológia alkalmazási lehetőségei

Mayer Walter, Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, III. év
Farkas Szilárd, Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, II. év

dr. Sulyok Csaba, egyetemi adjunktus,
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar
Szopos Dávid, szoftverfejlesztő,
Codespring
Fazakas Lóránt-Tibor, szoftverfejlesztő,
Codespring

Az Ultra-Wideband (UWB) technológia az utóbbi években egyre nagyobb szerepet kapott a beltéri helymeghatározási rendszerek terén, köszönhetően nagy pontosságának és megbízhatóságának. A dolgozat célja egy olyan UWB-alapú rendszer bemutatása, amely nemcsak elméletben, hanem a gyakorlatban is igazolja a technológia előnyeit más vezeték nélküli megoldásokkal szemben.

A dolgozat elsősorban a kommunikációs technológiák és az IoT rendszerek iránt érdeklődő szakemberek és kutatók számára készült. Részletesen ismerteti a rendszer architektúráját, beleértve a hardver- és szoftverkomponenseket, az adatátviteli megoldásokat, valamint a szerver-kliens infrastruktúrát. Bemutatja, hogyan integrálhatók az UWB-eszközök egy skálázható és hatékony kommunikációs rendszerbe, amely valós idejű helymeghatározást tesz lehetővé.

A fejlesztés során szerzett tapasztalatok alapján elemezi az UWB előnyeit és korlátait, például a pontosságot, az interferenciakezelést és az energiafogyasztást. Emellett kitér a rendszer telepítésének kihívásaira és a lehetséges továbbfejlesztési irányokra, például a nagyobb léptékű alkalmazásokra és az algoritmusok optimalizálására.

Tartalomjegyzék

Bevezető

Az elmúlt 50 év során során a globális helymeghatározó rendszerek (GPS) a katonai eszköztől a mindennapi élet elengedhetetlen részévé váltak. [10] Milliárdok támaszkodnak nap mint nap az ezáltal biztosított navigációra és helymeghatározásra. Bár a GPS forradalmi hatást gyakorolt, számos korláttal is rendelkezik, különösen a jelzavarok terén. [16, 20] Mivel a GPS az szatellitek és a vevők közötti tiszta látóviszonyokon alapul, jelei könnyen megszakadhatnak. Városi környezetekben a magas épületek, alagutak vagy sűrű növényzet akadályozhatja, visszaverheti vagy torzíthatja a jeleket, ami pontatlanságokhoz vagy akár teljes jelvesztéshez vezet. Jóllehet, a GPS elsősorban kültéri használatra lett optimalizálva, így beltéren, ahol a falak és más akadályok tovább gyengítik a jeleket, megbízhatatlan eredményeket produkál. [28, 12]

A beltéri helymeghatározás jelenleg leginkább WiFi, Bluetooth Low Energy (BLE) és RFID alapú rendszerekre támaszkodik. A BLE alacsony fogyasztású és széles körben elérhető [18], de átlagosan csak 4–5 méteres pontosságot kínál, és érzékeny az akadályokra és a többszórt jelhatásokra. [4] A WiFi-alapú módszerek (például fingerprinting vagy CSI-alapú megközelítések) jobb pontosságot érhetnek el, ám jelentős kalibrációs és tárolási overheaddel járnak, illetve nehezen skálázhatók nagy területen. [21, 19] Az RFID költséghatékony és egyszerűen telepíthető, de jellemzően LoS-követelményeket támaszt, és korlátozottan kezel komplex beltéri akadályokat. [24, 25]

Ezzel szemben projektünk egy újszerű megoldást mutat be, amely az Ultra-Wideband (UWB) technológia előnyeit kihasználva ad választ ezekre a kihívásokra. Az UWB centiméteres pontosságot kínál még összetett környezetekben is, mivel kevésbé érzékeny a multipath-interferenciára, és jól kezeli a nem látótávolságú (NLoS) helyzeteket. [13] Emellett alacsony késleltetésű, stabil kommunikációra képes, így ideális olyan alkalmazásokhoz, ahol a precizitás és megbízhatóság kiemelten fontos. Bár elterjedtsége jelenleg korlátozott, technológiai adottságai alapján ígéretes alternatívát jelent a hagyományos beltéri helymeghatározási rendszerekkel szemben. Alkalmazásunk magját a Qorvo DWM1001[1] termékcsalád és az ehhez tartozó szoftveres eszköztár biztosítja, amely hidat képez az UWB-kommunikáció és az absztraktabb szintek között. Dolgozatunk átfogó képet ad az UWB-alapú helymeghatározó rendszerünkről, részletesen ismertetve annak architektúráját a következő felépítés szerint: az 1. fejezet az UWB technológia legfontosabb tulajdonságait

[1] https://www.qorvo.com/products/p/DWM1001-DEV

mutatja be, a 2. fejezet pedig az alkalmazás felépítésére összpontosít, különös hangsúlyt fektetve a firmware funkcióira és a kliens–szerver kettősre. A 3. fejezetben a felhasznált technológiákat részletezzük, míg a 4. fejezet a rendszer gyakorlati alkalmazását vizsgálja. Végül a záró. fejezet kitekintést nyújt a továbbfejlesztési irányokra.

A projekt fejlesztése Kolozsváron, a 2024-es nyári szakmai gyakorlaton kezdődött el, majd tovább folytatódott az egyetemi csoportos projekt tantárgy keretein belül. A szerzők szeretnék köszönetüket nyilvánítani a dolgozat témavezetőjének, dr. Sulyok Csaba egyetemi adjunktusnak, illetve a Codespring mentorainak és munkatársainak, Szopos Dávidnak, Fazakas Lóránt-Tibornak és Szélyes Leventének, továbbá Péterfi Péternek, Muzsi Eriknek, Szabó Leventének, Papp Zoltán-Balázsnak és Mihály Zoltán-Zsoltnak.

1. Az UWB technológia felülnézetből

Az Ultra-Wideband (UWB) egy rövid hatótávolságú vezeték nélküli kommunikációs technológia, amelyet nagy sávszélesség (>500 MHz), alacsony energiafogyasztás, magas biztonság, nagy pontosság és minimális interferencia jellemez[2]. A hagyományos vezeték nélküli kommunikációs módszerekkel ellentétben az UWB széles frekvenciatartományon működik, így rendkívül ellenálló az interferenciával szemben, és kiválóan alkalmas helymeghatározásra. Bár az UWB technológia már az 1960-as évek óta létezik, széles körű kereskedelmi alkalmazása csak 2019-ben kezdődött meg, de azóta már az újabb okostelefonokba is be van építve, lehetővé téve a fejlettebb eszközkövetést. [30].

1.1. Főbb alkalmazási területek

Az UWB technológia egyre nagyobb szerepet kap különböző területeken, többek között:

- *Okos nyomkövető megoldások:* Az UWB technológia különösen hasznos lehet az ipari és logisztikai szektorokban, ahol a precíz eszközkövetés lehetővé teszi a vállalatok számára, hogy egyszerre figyelemmel kísérjék több ezer tárgy helyét. [26]
- *Smartkeys:* Az UWB már alkalmazásra került prémium járműveknél, biztosítva a kulcs nélküli, biztonságos belépést. Az UWB előnye, hogy nem érzékeny annyira a manipulációra, mint más vezeték nélküli megoldások. [26]
- *Smarthome-rendszerek:* Az UWB lehetőséget ad arra, hogy különböző okos eszközöket automatizáljunk. Például, a rendszer érzékeli, ha a felhasználó belép vagy elhagyja az otthont, és ennek megfelelően aktiválhat más rendszereket, legyen szó a világításról vagy háztartási eszközökről. [26]
- *Orvosi alkalmazások:* Például betegek követése és monitorozása. Az UWB alapú viselhető eszközök képesek nyomon követni a létfontosságú jeleket, figyelni a betegek mozgását és távoli egészségügyi szolgáltatásokat biztosítani. [11]
- *Adatátvitel nagy sebességgel:* Az UWB gyors adatátvitelt tesz lehetővé, és együtt tud működni az 5G technológiával, így ideális választás lehet adatintenzív alkalmazásokhoz. [26]

[2] https://www.itu.int/dms_pubrec/itu-r/rec/sm/R-REC-SM.1755-0-200605-I!!PDF-E.pdf

- *Játékok:* A rendszer alacsony késleltetése miatt kiváló választás lehet a versenyszintű játékokhoz, ahol minden egyes pillanat számít. [26]

1.2. UWB helymeghatározási módszerek

Az UWB egyik legnagyobb előnye, hogy olyan környezetekben is működik, ahol a hagyományos helymeghatározási technológiák nem használhatók, de emellett a Bluetooth Low Energy (BLE) technológiával ellentétben az UWB nem zavarja más kommunikációs rendszerek működését, így megbízhatóbb megoldást nyújt bizonyos alkalmazásokhoz. [30] Az UWB eszközök két fő módszert alkalmaznak a helymeghatározásra:

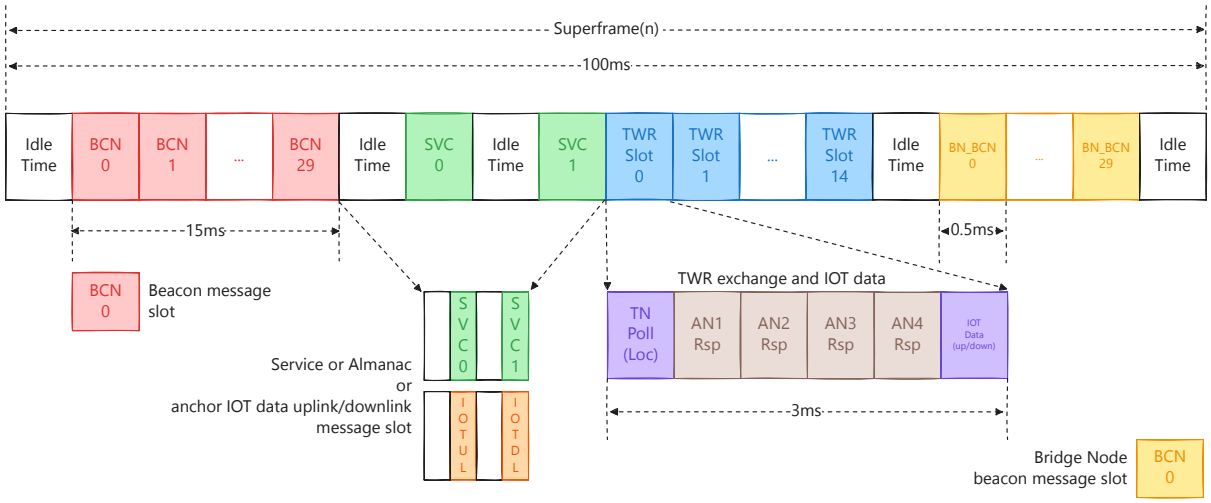
- *Time of Flight (ToF)*: Mérhető a jel adótól vevőig való eljutásának ideje. Ez lehet egyoldalú (Single-Sided) vagy kétoldalú (Double-Sided) mérés.
- *Angle of Arrival (AoA)*: Több antenna segítségével meghatározható az érkező jel szöge a fáziseltérések (PDoA – Phase Difference of Arrival) kiszámításával. Ez lehetővé teszi a 2D vagy 3D pozíciómeghatározást. [30]

Az UWB rendszerekben általában tagek (vevő) és több anchor (adó) vesz részt, amelyek egymással kommunikálva határozzák meg a címkék pontos helyét.

1.3. Az UWB superframe szerkezete

Az UWB kommunikáció egy determinisztikus időosztásos többszörös hozzáférési (TDMA – Time Division Multiple Access) rendszer, amely egy ciklikusan ismétlődő, 100 ms időtartamú szuperkeret (superframe) struktúrára épül. (ld. 1. ábra) Ennek néhány fontos része:

- *30 jeladó üzenet slot (BCN):* Az anchor eszközök ebben a fázisban sugározzák saját pozíciójukat és hálózati státuszukat tartalmazó üzeneteiket. Minden anchor számára egyedileg lefoglalt slot van fenntartva, amelyet a konfiguráció során választ ki vagy kap a hálózattól.
- *2 szerviz slot (SVC):* Ezekben az időrésekben zajlik az Almanac üzenetek küldése, amely a hálózat topológiájára, firmware verziókra és szolgáltatásinformációkra vonatkozó metaadatokat tartalmaz. Ezen kívül IoT adatokat is továbbítanak ezekben a résekben.
- *15 kétirányú távolságmérő slot (TWR):* Ezek a slotok szolgálnak a tag eszközök és anchorok közötti távolságmérés lebonyolítására.



1. ábra. Az UWB szuperkeret szerkezete [8]

- *30 Bridge Node Beacon slot (BNB):* A bridge node-ok ebben a szegmensben hirdetik az elérhető IoT adatok meglétét a tag eszközök számára.
- *Guard/Idle idő:* A szuperkeret végén található időpuffer, amely biztosítja az időzítési hibák elnyelését és a keretek közötti váltás stabilitását. [8]

A pozíciómeghatározás általában az SS-TWR (Single-Sided Two-Way Ranging) módszeren alapul. A tagek és anchorok akár 34 bájtnyi adatot is biztonságosan cserélhetnek. Az adatok kimenete tipikusan Bluetooth-on keresztül történik, vagy egyes esetekben UART (Universal Asynchronous Receiver-Transmitter) interfészen keresztül továbbítják egy csatlakoztatott eszköz felé. [8]

Minden kommunikációs esemény a szuperkeret belső struktúrájához igazodik. Az iniciátor (kezdeményező anchor) vezérli az időzítést, és biztosítja a hálózat szinkronizációját. Az összes többi eszköz (anchorok, tagek és bridge node-ok) a szuperkeret ütemezése alapján funkcionál.

A TDMA működése során a tag eszközök úgy időzítik aktivitásukat, hogy a kiválasztott ranging slotban végezzenek méréseket. A slot foglalást a tag Group Poll üzenettel kezdeményezi, amelyhez a kiválasztott anchorok Response üzenettel válaszolnak. Amennyiben minden válasz megfelelő, a slot foglaltként lesz kezelve, és más tag nem fogja használni.

A rendszer így nemcsak rendkívül pontos helymeghatározást tesz lehetővé, hanem hatékonyan skálázható is, mivel a slotok periodikus kiosztása alapján több tucat tag is működhet interferencia nélkül. Ez a szigorú időbeli szegmentáció biztosítja, hogy a helymeghatározás következetesen és kiszámítható módon történjen, még akkor is, ha a hálózat topológiája dinamikusan változik vagy a környezet zsúfolt. [8]

2. Az alkalmazás felépítése

Ez a szekció részletesen bemutatja az alkalmazás működéséhez szükséges hardveres és szoftveres komponenseket, valamint ezek integrációját. A leírás a fizikai eszközök ismertetésével kezdődik, majd a szoftveres réteg különböző aspektusaira tér ki: a rendszer architektúrájára, a firmware szerepére, a helymeghatározásért felelős UWB eszközökre és algoritmusokra, valamint a hálózati infrastruktúra skálázhatóságára. Ezt követően a szerveroldali komponensek – például az adatmodell és az API – valamint a kliensoldali elemek (adminisztrációs felület, mobilalkalmazás) kerülnek bemutatásra. Végezetül a kliens–szerver közötti kommunikáció mechanizmusai is ismertetésre kerülnek.

2.1. Hardver komponensek

A rendszer gerincét a Qorvo (régebben Decawave) által fejlesztett DWM1001 modulok alkotják, amelyek a DWM1000 UWB adó-vevőn és a Nordic Semiconductor nRF52832 SoC(System On a Chip)-en alapulnak. A DWM1000 végzi a pontos kétirányú távolságmérést (Two-Way Ranging - TWR), míg az nRF52832 biztosítja a Bluetooth Low Energy (BLE) kommunikációt és vezérli az egész modult. [8]

Az általunk felhasznált DWM1001 rendszer architektúrája hierarchikus és moduláris, különböző szerepköröket betöltő komponensekkel, melyek között pontosan definiált kommunikációs interfészek léteznek. A skálázhatóság támogatja az eszközök tömeges telepítését és karbantartását, a decentralizált rendszer pedig redundánsan működtethető, így több hiba is elviselhető anélkül, hogy a rendszer teljes egészében leállna. [8]

2.2. Software komponensek

2.2.1. Rendszer architekúra

Firmware & Szerepkörök

Az összes eszköz ugyanazt a firmware-t futtatja, viselkedésüket a konfiguráció során beállított szerepkör határozza meg, amely a DRTLS (Decawave Real Time Location System) működését valósítja meg (ld. 2. ábra).

Egy DWM1001 modul konfigurálható „anchor” (rögzített referencia pont), „bridge”, illetve „tag” (mobil, lokalizálandó eszköz) szerepkörre.

A firmware az eszközök szerepköreinek megfelelően konfigurálja a modult. A PANS (Positioning And Networking Stack) firmware támogatja az RTLS funkciókat, valamint a Bluetoothon vagy shell interfészen keresztüli konfigurációt.

A firmware kulcsfontosságú feladatai a következők [8]:

- TDMA (Time Division Multiple Access) séma szerinti időzítés (ld. 1.3. fejezet)
- kétirányú távolságmérés (TWR) kezelése, lebonyolítása (ld. 1.3. fejezet)
- Helymeghatározási műveletek (Location Engine)
- Hálózati topológia felismerése
- Ütközésdetekciós és elkerülési algoritmusok
- A firmware-over-the-air frissítés is lehetőséges, ha engedélyezett.

UWB eszközök

A DWM1001 modul a DWM1000 transzceiver által biztosított UWB kommunikációval képes távolság meghatározásra centiméter pontossággal. Az UWB támogatja a LOS (line-of-sight) és korlátozott mértékben a NLOS (non-line-of-sight) kommunikációt is, amely különösen fontos beltéri, zsúfolt környezetben. A DWM1000 fizikai rétege rendkívül ellenálló a multipath zavarásokkal szemben is. [3]

A hálózati struktúra TDMA alapú superframe időosztásos rendszert alkalmaz, amely garantálja a determinisztikus viselkedést és az ütközésmentes adatátvitelt (ld. 1.3. fejezet). A tag eszközök a Beacon és Almanac üzeneteiből tanulják meg a topológiát, majd kiválasztanak 3 vagy 4 anchort a ranginghez. A ranging során **Group Poll** üzenetet küldenek, amire az anchrok **Response** üzenetekkel válaszolnak, majd a tag kiszámolja saját pozícióját a Location Engine segítségével. Ez a folyamat periodikusan ismétlődik, figyelembe véve a tag mozgását, környezeti változásokat és az energiafogyasztási korlátokat is. [8] A rendszer energiahatékonyságát támogatja két üzemmód (Responsive és Low Power), valamint az, hogy az MCU, UWB és BLE komponensek alacsony fogyasztású alvó üzemmódba helyezhetők, amely hosszú élettartamot biztosít akkumulátoros működés esetén is. Ezen kívül a rendszer képes önállóan felismerni a mozgást (pl. beépített gyorsulásmérővel) és adaptívan módosítani a helymeghatározási gyakoriságot. Ez jelentősen javítja az akkumulátor élettartamát anélkül, hogy csökkentené a rendszer pontosságát vagy reakcióidejét. [8]

Gateway architektúra és hálózati integráció

A gateway szerepe, hogy az UWB hálózatról származó adatokat a LAN/WAN irányába irányítsa (ld. 2. ábra). A DWM1001 modul PANS firmware-rel bridge módba konfigurálva képes kommunikálni mind a tag és anchor eszközökkel UWB-n keresztül, mind pedig a host rendszerrel UART interfészen keresztül. A bridge node összegyűjti a hálózati adatokat (pl. pozícióadatok, IoT üzenetek), és továbbítja azokat a host gép felé. Ez MQTT protokollon keresztül történik. Illetve fordítva: a felhőből érkező vezérlő vagy adatüzeneteket továbbítja az UWB-hálózat irányába. Ez HTTP-n keresztül történik. [8]

Az általában Raspberry Pi-n futó szoftverrendszer a következő komponensekből áll [7]:

- *Linux kernel modul:* Alacsony szintű interfész a DWM1001 felé UART-on keresztül.
- *DWM Daemon:* Folyamatosan futó szolgáltatás, amely a bridge node és az alkalmazások közötti kommunikációt kezeli.
- *DWM Proxy:* Absztrakciós réteg, amely REST API-t biztosít a külső kliensek számára.
- *MQTT Broker:* Lehetővé teszi, hogy a DRTLS-ből származó adatokat valós időben MQTT protokoll segítségével továbbítsuk bármely MQTT kompatibilis kliens irányába.
- *HTTP Server:* Webes felületet biztosít konfigurációhoz és adatvizualizációhoz.

A gateway teljes architektúrája úgy lett kialakítva, hogy valós időben képes legyen kezelni a helymeghatározási adatokat, valamint IoT adatokat, miközben biztosítja a skálázhatóságot, megbízhatóságot és alacsony késleltetést. A rendszer támogatja a kétirányú kommunikációt, így a gateway nem csupán adatgyűjtőként, hanem vezérlési csatornaként is funkcionál. Ez lehetővé teszi például azt, hogy az IoT végpontok (mint a tag eszközök) parancsokat kapjanak egy központi szervertől vagy akár egy felhasználói alkalmazástól. [8]

Ezen túlmenően a gateway szerepe kiterjed a hálózat karbantartására is: lehetővé teszi a firmware frissítések indítását OTA (Over-The-Air) módon az egész hálózatra kiterjedően, a bridge node-on keresztül. [8]

Location Engine – helymeghatározó algoritmus és pozíciószámítás

A Location Engine a DRTLS rendszer egyik legösszetettebb és legfontosabb belső komponense, amely a tag eszköz által végzett kétirányú távolságmérések (TWR) eredményeit dolgozza fel és ezek alapján becsüli meg a tag pozícióját a háromdimenziós térben. A modul célja, hogy megbízható, pontos és alacsony késleltetésű pozícióinformációt szolgáltasson,

amelyet később vizualizációs felületek, logikai döntéshozó rendszerek vagy más IoT alkalmazások használnak fel. [8]

A Location Engine működése a következő fő lépésekből áll [8]:

1. *Ranging eredmények gyűjtése:* A tag a TWR szakaszban 3–4 anchorral hajt végre távolságmérést. Ezek az adatok valós idejű időbélyegeken alapulnak, amelyek alapján a repülési idő (Time of Flight, ToF) kiszámítható.
2. *Koordináták összeállítása:* A tag ismeri az anchorok pozícióit, így azokat egy koordinátarendszerbe helyezi (tipikusan Decartes-féle koordináta-rendszerbe).
3. *Pozícióbecslés maximum likelihood módszerrel:* A tag különböző lehetséges helyeket generál, amelyek illeszkednek a kapott távolságértékekhez. Ezek közül kiválasztja a legvalószínűbb pozíciót – ez az ún. maximum likelihood pozíció. [8]
4. *Többszörös kombinációk és hibaszűrés:* Ha négy távolságadat áll rendelkezésre, akkor minden három adatból négy kombináció hozható létre. Ezeket külön-külön is kiértékeli, így növelve a robusztusságot. A kapott pozíciók közül azok kerülnek figyelembe vételre, amelyek hibája a mért távolságokhoz képest alacsony.
5. *Cache mechanizmus:* Amennyiben az aktuálisan használt anchorok megegyeznek az előző ciklusban használtakkal, a rendszer cache-elt adatokat használ a számítás gyorsítására. Ez különösen fontos nagy sűrűségű tag aktivitás esetén.
6. *Mozgóátlag alkalmazása:* A végső pozíció a legutóbbi három számítási ciklus eredményéből képzett mozgóátlag lesz. Ez simítja a zajos méréseket, és vizuálisan stabilabb eredményt ad.
7. *Minőségi értékelés (Quality Score):* A rendszer minden pozícióhoz hozzárendel egy 0–100 közötti minőségi értéket, amely az illeszkedés pontosságát, a mérések megbízhatóságát és a számított pozíció hibáját veszi figyelembe.

A Location Engine robusztussága abban rejlik, hogy dinamikusan tud alkalmazkodni a hálózati környezethez – például NLOS (Non-Line-of-Sight) környezetben is képes ésszerű becslést adni, feltéve, hogy legalább három anchor válasza elérhető és azok távolságadatai nem ellentmondásosak. Multipath hatások kiszűrésére, a távolságadatok torzulásait detektálva szűri ki az irreális pozíciójelölteket. [8]

Skálázhatóság és hálózati méretezés

A DRTLS rendszer hatékony skálázhatósága lehetővé teszi, hogy kis léptékű beltéri alkalmazásoktól egészen ipari méretű, nagy kiterjedésű helyszínekig alkalmazható legyen. A rendszer skálázhatósága mind a tag (mobil), mind az anchor (fix) eszközök tekintetében biztosított, azonban a teljesítmény fenntartása érdekében bizonyos technikai korlátokat és ajánlásokat figyelembe kell venni. [8]

Anchor elrendezés és sűrűség: Egy jól működő RTLS hálózat alapja a megfelelő anchor elhelyezés. A rendszer szempontjából optimális, ha egy tag eszköz egyszerre legalább három, de ideálisan négy anchorral is képes rangingelni. A térbeli elhelyezkedésüknek homogénnek kell lennie az adott térben, hogy a trilateráció matematikai feltételei teljesüljenek. [8]

Az anchorok közötti ajánlott távolság 5–10 méter, de ez refsec:nagyban függ a környezeti viszonyoktól, például falaktól, fémszerkezetektől vagy más rádiófrekvenciás interferenciát okozó objektumoktól. Egy anchor túl nagy területet lefedve több tagot is kiszolgálhat, de ez esetben nő az ütközések esélye a TDMA slotokban, ami pontosságcsökkenéshez vezethet. [8]

Tag sűrűség és kezelhetőség: A rendszer egy TDMA superframe alatt 15 TWR slotot kínál, amelyek mindegyikében egy tag maximum négy anchorral végezhet ranginget. Ennek megfelelően 100 ms-onként nagyjából 15 tag rangingelhet függetlenül – tehát 1 másodperc alatt akár 150 tag is lekérdezhető, feltéve, hogy minden tagnak dedikált slotja van. [8]

Amennyiben a tag populáció meghaladja ezt a számot, a rendszer az eszközöket rotálva, adaptívan kezeli, így a pozíciófrissítési ciklusidő megnőhet. Ezt a kompromisszumot figyelembe kell venni nagy létszámú alkalmazások (pl. gyártósorok, kórházi személyzetkövetés) esetén. [8]

Slot menedzsment és TDMA optimalizálás: A rendszer automatikusan ütközésdetekcióval és újraütemezéssel képes reagálni a túlzsúfolt slotokra. A tag eszközök folyamatosan figyelik az aktuális slot allokációt és szükség esetén más slotra váltanak, amennyiben a kiválasztott anchorok válaszideje nem megfelelő. [8]

A hálózati topológia és slot allokáció állapotát az Almanac szolgáltatás és a szolgáltatási slotokban publikált üzenetek tartalmazzák, így a tag eszközök mindig naprakészen tudnak alkalmazkodni a változásokhoz. Ez lehetővé teszi a dinamikus terheléselosztást és a hálózat öngyógyító képességét is. [8]

Bridge és gateway skálázás: A bridge node-ok és a hozzájuk kapcsolódó gateway rendszerek szintén skálázható módon üzemeltethetők. Egy bridge node tipikusan egy lokális UWB szegmenshez tartozik, és legfeljebb 30–50 anchor vagy tag adatait képes aggregálni egy

gateway-en keresztül. [8]

Nagyobb rendszerekben több bridge node és gateway párhuzamos működtetése javasolt. A gateway-ek központosított MQTT brokerrel kommunikálhatnak, így közös adatbázisban gyűjthetők az adatok, és skálázható backend oldali feldolgozás is kialakítható. [8]

NLOS (Non-Line-of-Sight) viselkedés és környezeti hatások kezelése

Az Ultra-Wideband rendszerek egyik legnagyobb előnye a hagyományos rádiófrekvenciás technológiákkal szemben, hogy fokozottan ellenállóak a multipath hatásokkal és az akadályokkal szemben. Ennek ellenére a DRTLS rendszer működését is jelentősen befolyásolhatják az úgynevezett NLOS (Non-Line-of-Sight) körülmények, amikor a tag és az anchor között nincs közvetlen rálátás. Ilyen környezet alakulhat ki például betonfalak, gépek, fémszerkezetek vagy emberi testek által okozott takarások miatt. [8]

A NLOS hatásainak jellemzői: A NLOS helyzetek leggyakoribb következménye a távolságmérés pontosságának romlása. Mivel a rádiójelek az akadályokról visszaverődnek, a tényleges repülési idő meghosszabbodik, így a rendszer túlbecsüli a tag–anchor közötti távolságot. Ez a hatás jellemzően 0,5–2 méteres hibát is eredményezhet, amely elfogadhatatlan lehet precíziós RTLS alkalmazások esetén. [8]

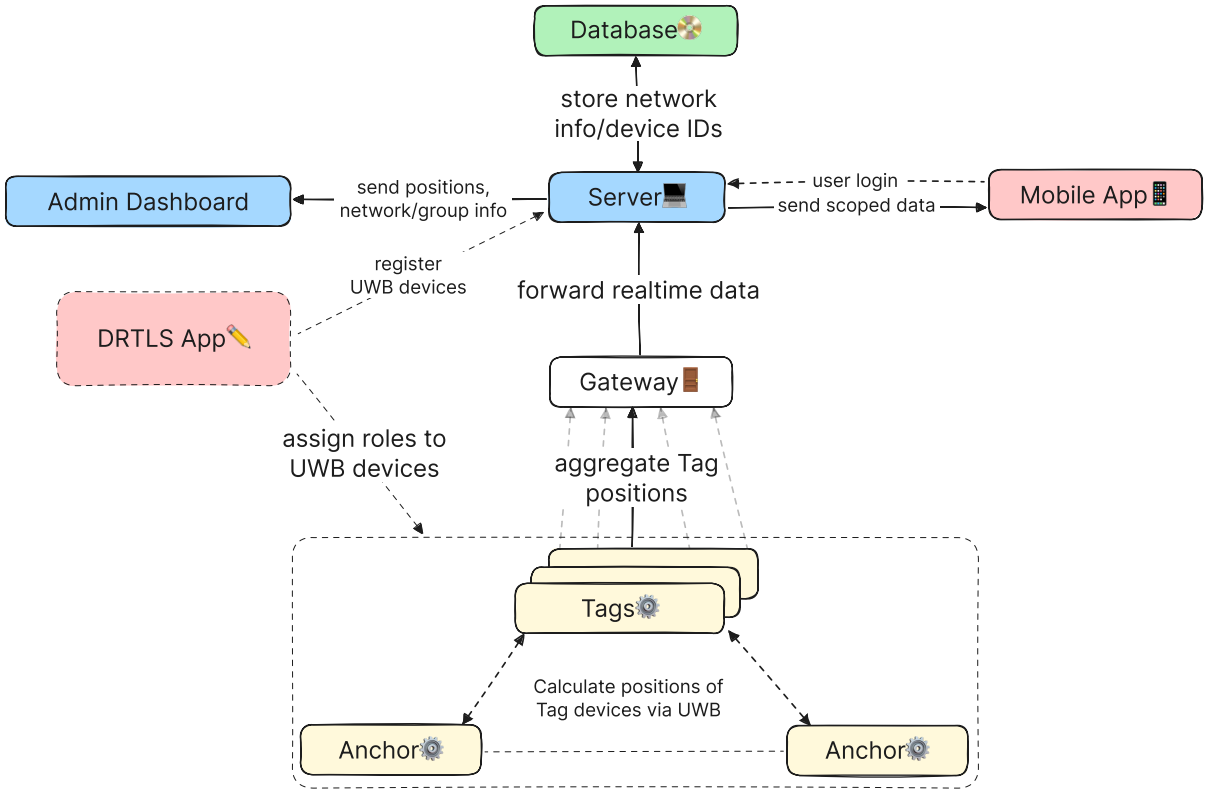
Multipath környezetben több visszavert jel is eljuthat a vevőhöz, amelyek interferenciát okoznak. Bár az UWB spektrum szélessége miatt ezek a hatások részben kiszűrhetők, a pozíciószámítás így is instabillá válhat, ha a NLOS jelenség több anchor-t is érint egyidejűleg. [8]

NLOS detekció és adaptív válaszok: A DWM1001 alapú rendszerek képesek NLOS körülmények detektálására és adaptív módosításokra. A Location Engine az alábbi jelekből következtet arra, hogy egy adott távolságmérés NLOS környezetben történt [3]:

- A távolságmérés hibája meghalad egy előre meghatározott küszöböt (pl. 1,5 méter).
- Az időbélyegek szórása magasabb az elvártnál.
- Az adott anchor-tól származó válasz rendszeresen kimarad vagy jelentős késleltetéssel érkezik.
- Az adott anchor különböző ciklusokban jelentősen eltérő pozícióértékeket eredményez.

Ilyen esetekben a Location Engine dönthet úgy, hogy [3]:

- Figyelmen kívül hagyja az adott anchor mérését a pozíciószámítás során.
- Csökkenti a mérés súlyát az ML (maximum likelihood) pozícióbecslésben.



2. ábra. Rendszer architektúra

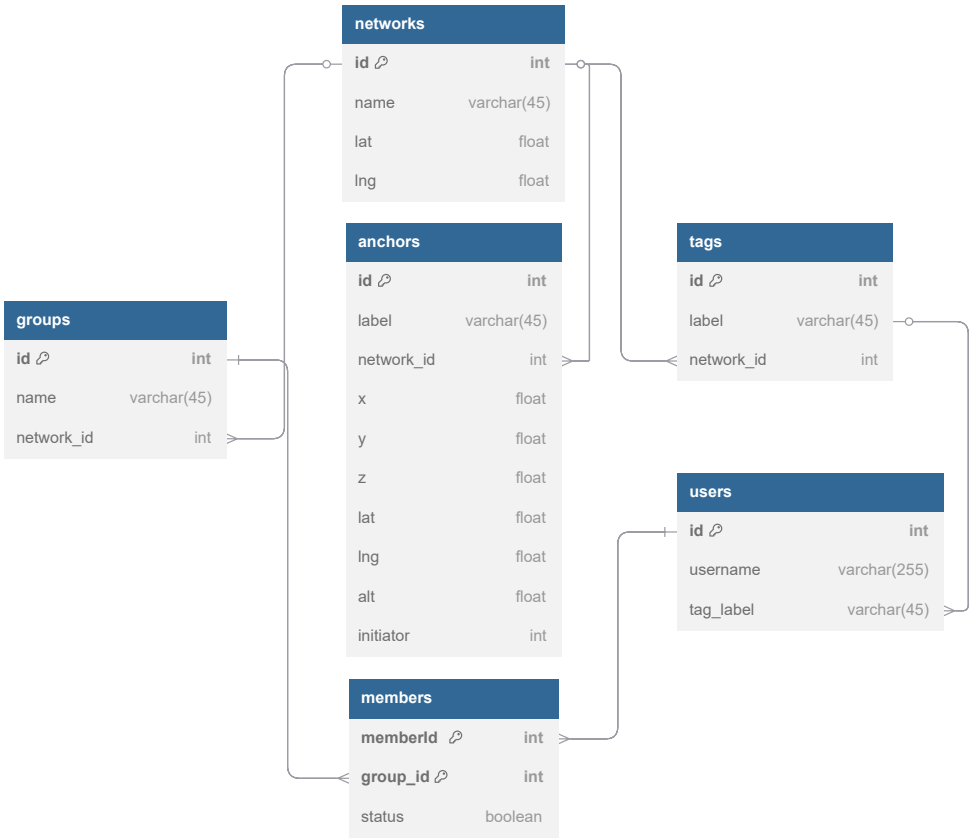
- Alternatív anchor kombinációkat próbál ki, ahol a három legmegbízhatóbb anchor használata élvez prioritást.

2.2.2. Szerver komponensek

Adatmodell

Az itt bemutatott adatszerkezet egy térbeli nyomkövetési és felhasználókezelő rendszer alapját képezi, amely dinamikus környezetekre lett tervezve. Az adatmodell több, egymással összehangolt entitásból áll, amelyek együttesen biztosítják a valós idejű pozíciókövetést, a hálózatokon keresztüli kommunikációt, akár csoportokra lebontva is. A legfontosabb komponensek közé tartoznak a hálózatok, anchorok, tagek, felhasználói fiókok, csoportok és tagsági kapcsolatok (ld. 3. ábra).

A hálózatokat egyedi azonosító, név és földrajzi koordináták jellemzik, amelyek valójában az ún. kezdeményező (initiator) referenciapont (anchor) helyét tükrözik. Mivel a kommunikáció során a tagek pozícióadatai mindig ehhez a ponthoz viszonyítottan kerülnek meghatározásra, e koordináták tárolása elengedhetetlen. Ennek köszönhetően a relatív pozíciók később pontosan visszaalakíthatók földrajzi (hosszúsági és szélességi) koordinátákká. A hálózatokhoz további



3. ábra. Az adatmodell

anchor pontok is tartoznak (legalább három), amelyek a multilaterációs számítások [1] alapját képezik, biztosítva a precíz és konzisztens helymeghatározást.

Minden tag valós idejű helyadatot hordoz, ideértve a számított földrajzi koordinátáit és az adott hálózat referenciapontjához viszonyított relatív pozícióját. Mindegyik tag rendelkezik továbbá egy jelminőségi mutatóval, amely a pozícióbecslés megbízhatóságát tükrözi.

Felhasználók is külön entitásként jelennek meg, akik egyedi azonosítóval, felhasználónévvel és egy hozzájuk rendelt taggel rendelkeznek. E kapcsolatok révén azonosítható, hogy egy-egy pozíció kinek a mozgásához köthető az adott hálózaton belül.

A könnyebb azonosítás és nyomon követés érdekében, különösen nagy felhasználószámú hálózatok esetén, a modell lehetőséget biztosít csoportok létrehozására. Egy csoport egyedi azonosítóval, névvel és egy adott hálózathoz kapcsolódva működik. Az egyes csoportokhoz tartozó felhasználók listája közvetlenül elérhető, míg a tagsági állapotokat külön tábla kezeli. Ez az állapot azt tükrözi, hogy a felhasználó aktuálisan aktív-e a csoportban. A megvalósítás lehetőséget ad a tagság rugalmas kezelésére, valamint az állapotváltozások gyors és hatékony kommunikálására a rendszer többi komponense felé.

API

A rendszer eseményvezérelt, kétirányú kommunikációs modellt alkalmaz, amely valós idejű adatcserét tesz lehetővé a kliens és a szerver között (ld. 2. ábra). Két csatornán keresztül történik az adatáramlás: az egyik oldalon a felhasználók által kezdeményezett események kerülnek továbbításra, a másikon a rendszer válaszai és automatikus frissítései jutnak el a kliensekhez. Ez az architektúra támogatja az alacsony késleltetésű, aszinkron működést.

A felhasználói műveletek események formájában jutnak el a szerverhez, lehetővé téve a regisztrációt, eszközkezelést és csoportműveleteket. Az alábbiakban a legfontosabb események kerülnek bemutatásra.

- `subscribe`: A kliens azonosítására szolgál. A felhasználónév és a címke megadásával (`{ username: string, tagLabel: string }`) biztosítja, hogy a rendszer tudja, kitől származnak a beérkező adatok, és kinek küldjön vissza személyre szabott frissítéseket. Válaszként sikeres regisztráció vagy hibaüzenet érkezhet.
- `registerDevice`: Új tag eszköz regisztrálása a rendszerbe. A kliens megadja az eszköz címkéjét és a hozzá tartozó felhasználót.
- `registerBridge`, `adminSubscribe`: Adminisztratív feladatokra szánt események, melyeket a hidak és adminfelületek használnak.
- `createGroup`, `deleteGroup`, `addMemberToGroup`, `removeMemberFromGroup`: A csoportok kezelését végző események. Például a `createGroup` során egy egyszerű objektum (`{ groupName: string, networkId: number }`) alapján jön létre egy új csoport, automatikusan hozzárendelve a megadott hálózathoz.

A szerveroldali események célja, hogy a kliens valós időben értesüljön az őt érintő hálózati változásokról, rendszerállapotokról és műveletek eredményéről. Ezek az események kulcsfontosságúak a felhasználói felület szinkronizálásában, a vizualizációk frissítésében és a hibakezelés hatékony lebonyolításában:

- `locationUpdate`: Egy tag pozícióadatait tartalmazza: földrajzi koordináták (`lat, lng`), relatív hely (`x, y`), hálózati azonosító és egy minőségi mutató.
- `anchorDiscovery`: Minden új vagy módosított anchor pozícióját küldi el, háromdimenziós koordinátákkal és hálózati hivatkozással.

- `networkDiscovery`, `userDiscovery`: Új hálózatok és felhasználók megjelenésekor küldött események.
- `registerSuccess`: Egy eszköz vagy felhasználó sikeres regisztrációját jelzi. Hibakezelésként szolgálnak a `registerTagNotExistError`, `registerTagAssignedError`, `registerUserHasTagError`, `notRegisteredError` események.
- `groupCreated`, `memberAdded`, `memberRemoved`, `groupDeleted`: A csoportokhoz kapcsolódó változásokat kommunikálják. `groupError`, `groupCreationError`: Csoportműveletek hibakezelése.
- `userGroups` esemény egy adott felhasználó összes csoportját adja vissza, struktúrált formában, lehetővé téve azok gyors és szűrt megjelenítését kliens komponensekben.
- Végül, a különféle rendszerhibák esetén a `subscriptionError`, `mqttError`, `databaseError` és `unknownError` események gondoskodnak a megfelelő hibaüzenetek közléséről.

2.2.3. Kliensek

Admin felület

Az admin felület a rendszer központi vezérlőpultjaként működik, amely élő, interaktív rálátást biztosít a teljes UWB alapú beltéri helymeghatározó rendszerre (ld. 2. ábra). A felület lehetőséget nyújt a hálózatok, tagek, anchorok és csoportok állapotának valós idejű megfigyelésére és kezelésére. Az alkalmazás fejlesztőbarát struktúrával épül fel, külön modulokra bontva az egyes funkcionális elemeket:

- `components/`: A dashboardot felépítő JSX komponenseket tartalmazza (pl. térképek, vezérlőgombok, UI elemek).
- `context/`: React Context definíciók találhatók itt, amelyek biztosítják az állapotmegosztást a különböző komponensek között (SelectedNetwork, RealtimeUWB, stb.).
- `interfaces/`: A projekt közös típusait és interfészeit tartalmazza TypeScript-ben.
- `modals/`: Olyan interaktív felugró ablakokat tartalmaz, melyek felhasználói beavatkozást kérnek (pl. megerősítések, szerkesztési panelek).

```
{hoveredTag !== null &&
anchors.map((anchor, anchorIndex) => {
    const tag = Array.from(tags.values())[hoveredTag];
    const start = new LatLng(tag.lat, tag.lng);
    const end = new LatLng(anchor.lat, anchor.lng);
    const distance = start.distanceTo(end);

    return (
        <Polyline key={`${hoveredTag}-${anchorIndex}`} positions={[start,
        ↪  end]} color="#a3a3a3">
            <Tooltip permanent offset={[0, 0]}>
                {distance.toFixed(2)}m
            </Tooltip>
        </Polyline>
    );
})}
```

4. ábra. Távolságmegjelenítés implementációja

A felület működésének technológiai szempontból különösen izgalmas része a valós idejű kommunikációt biztosító mechanizmus, illetve az azt kiszolgáló állapotkezelő logika. A rendszer egy belső szolgáltatásréteget (`RealtimeProvider`) használ, amely külön kezeli a hálózatok, anchorok, tagek és felhasználók adatait. Ez a réteg folyamatosan figyeli a szerver felől érkező, a 2.2.2 részben tárgyalt formátumú üzeneteket, feldolgozza azokat, és azonnal frissíti a megjelenített adatokat. Ennek köszönhetően a dashboard mindig naprakészen tükrözi a rendszer valós állapotát.

A felhasználói élmény egyik kulcseleme a vizuális térképes ábrázolás. Ezt két fő komponens biztosítja: A `<Map>` komponens a valós koordinátákon alapuló térképi ábrázolást valósítja meg a Leaflet könyvtár segítségével. A 4. kódrészlet különösen érdekes fejlesztői szempontból, ugyanis lehetővé teszi, hogy az egérrel föléhúzott tag és az összes hozzá tartozó anchor közötti távolság vizuálisan is megjelenjen a térképen, vonalakkal és tooltip formájában.

A `<CanvasMap>` ezzel szemben lehetőséget biztosít saját alaprajz feltöltésére, majd erre rárajzolja a pozíciókat. Itt a rajzolási és interakciós logika képezi a legérdekesebb részt, például a `drawTags` függvény (ld. 5. kódrészlet). Ez a rész dinamizmust ad az alaprajzhoz: nemcsak a tageket rajzolja ki az aktuális koordináták alapján, hanem az anchorokhoz tartozó vonalakkal is megmutatja, hogy az adott tag milyen kapcsolatban áll a térben a többi eszközzel.

Mobil alkalmazás

A mobilalkalmazás jelenti a rendszer egyik legfontosabb, központi elemét (ld. 2. ábra).

```
networkTags.forEach((tag) => {
    const r = 100; // méretarány
    ctx.strokeStyle = "#808080";
    anchorPos.forEach((anchor) => {
        ctx.beginPath();
        ctx.moveTo(tag.x * r, tag.y * r);
        ctx.lineTo(anchor.x, anchor.y);
        ctx.stroke();
    });

    ctx.fillStyle = "#34d399";
    ctx.beginPath();
    ctx.arc(tag.x * r, tag.y * r, 5, 0, 2 * Math.PI);
    ctx.fill();
});
```

5. ábra. A <CanvasMap> rajzolási logikája

Átlátható és közvetlen hozzáférést biztosít a rendszer valós idejű állapotához, segítségével a felhasználó közvetlenül követheti nyomon a rendszer által gyűjtött és feldolgozott adatokat.Ezek közé tartoznak többek között a regisztrált tag eszközök pozíciókoordinátái (X és Y), valamint az adott eszköz kapcsolatának minőségi jellemzői .

Az alkalmazás külön `components`, `context`, `interfaces` és `modals` modulokra van bontva az adminfelülethez hasonlóan.

A rendszer egyik összetettebb technikai megvalósítása a tervrajz nézet, amely lehetővé teszi az eszközök pozíciójának valós idejű, interaktív megjelenítését. A skálázás és mozgatás során a térkép mindig visszatér egy ésszerű tartományba. Ehhez rugózó animációt használtunk. Ez a megoldás érintésalapú navigációt használ (ld. 6. kódrészlet): a felhasználó képes nagyítani (pinch-to-zoom) és mozgatni (drag) a rajzot, akár egyidejűleg is.

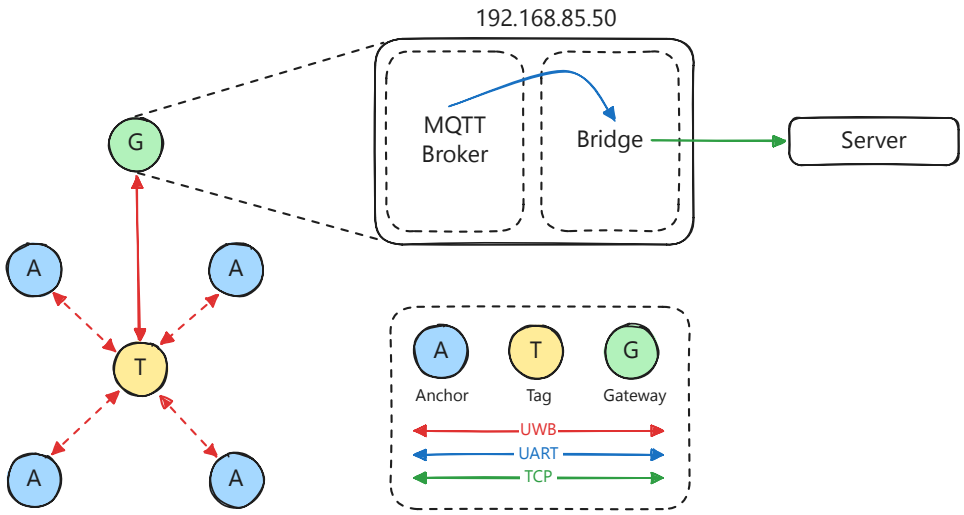
Kommunikáció a szerverrel

Az UWB-alapú helymeghatározást követően (ld. 2.2.1. fejezet), a tagek és anchorok által szolgáltatott pozícióadatok a gateway szerepében működő eszközökhöz kerülnek (ld. 2.2.1. fejezet). Ezek az eszközök az adatokat JSON formátumban továbbítják – minden taghez és anchorhoz külön MQTT topik társul, amelyeken keresztül történik a pozíciófrissítések és konfigurációs üzenetek cseréje.

A gyári rendszerfelépítés alapértelmezetten pull-alapú kommunikációt alkalmaz: a központi szervernek egyenként kell lekérdeznie a gateway-eket. Ez azonban nehézkessé teszi az üzembe helyezést, és jelentős mértékben korlátozza a rendszer skálázhatóságát. Ennek áthidalására egy

```
.onStart(() => {
    scaleGestureContext.value.scale = scale.value - 1;
})
.onUpdate((event) => {
    scale.value = event.scale + scaleGestureContext.value.scale;
})
.onEnd(() => {
    if (scale.value > 2) {
        scale.value = withTiming(2);
    } else if (scale.value < 1) {
        scale.value = withTiming(1);
    } else {
        if (translateX.value > 0 + scaleCanvasWidth) {
            translateX.value = withSpring(0 + scaleCanvasWidth, {}, () =>
            ↪  {
            isRunning.value = true;
            });
        } else if (translateX.value < -(scaledWidth - containerSize.width
        ↪  - scaleCanvasWidth)) {
            translateX.value = withSpring(-(scaledWidth -
            ↪  containerSize.width - scaleCanvasWidth), {}, () => {
            isRunning.value = true;
            });
        }
        ...
})
```

6. ábra. Interaktív nagyítás és mozgatás kezelése



7. ábra. Információáramlás az egyes szereplők között

egyedi, egyszerű szoftvermodul került kidolgozásra, amely minden egyes gateway-en külön MQTT kliensként fut (ld. 7. ábra). Ez a kliens valós időben hallgatja a bejövő üzeneteket, majd azokat push-alapon, TCP socketeken keresztül továbbítja a központi szerver felé (ld. 3.1. fejezet).

Ez a socket-alapú megoldás nemcsak gyorsabb és közvetlenebb adatátvitelt biztosít, hanem lehetőséget nyújt a gateway szintjén történő előfeldolgozásra is – például szűrésre, átalakításra vagy feltételes továbbításra. Ez jelentősen csökkenti a hálózati terhelést, és leegyszerűsíti az új eszközök rendszerbe integrálását.

Az MQTT, amely az OASIS által szabványosított, egy könnyű, eseményvezérelt protokoll, kifejezetten IoT környezetek számára. A publish/subscribe modellre épül, így a résztvevő kliensek nem közvetlenül, hanem egy ún. broker közvetítésével kommunikálnak: az üzeneteket topikokra publikálják, vagy ezekre feliratkozva fogadják. [29]

3. Felhasznált technológiák és eszközök

A rendszer fejlesztése során a megbízhatóság, a skálázhatóság és a fejlesztői élmény maximalizálása érdekében korszerű és elterjedt technológiák kerültek kiválasztásra. Három fő komponensből áll a rendszer: szerveroldali alkalmazásból, kliensoldali felületekből és köztes adatszolgáltató rendszerből.

Az alábbi alfejezetek részletesen bemutatják a választott technológiai stack-et, valamint azokat az eszközöket, amelyek a fejlesztést, és a működést segítik.

3.1. Szerver

Már a 2.2.2 részből ismerős szerveroldal egy Node.js-alapú környezetre támaszkodik, amelyre egy Next.js [15] keretrendszer épül TypeScript nyelven. Noha a Next.js hagyományosan szerveroldali renderelésre is lehetőséget ad, jelen projektben elsősorban a moduláris fájlrendszer-alapú routing, és a fejlesztői élményt javító eszközei (pl. automatikus újratöltés, build optimalizálás) kerülnek kihasználásra. Ezzel egyidejűleg a 2.2.2-ban körvonalazatott valós idejű üzenetcseréért a Socket.IO [22] felel, amely garantálja, hogy a kliens–szerver közötti kapcsolat alacsony késleltetésű és aszinkron maradjon.

A 2.2.2 szekcióban részletezett adatmodell tárolását egy MySQL alapú relációs adatbázis látja el. A szerver és az adatbázis közötti kommunikációt egy Prisma [17] alapú ORM réteg kezeli, amely típusbiztos lekérdezéseket és jól strukturált adatmodelleket kínál. Ezzel a megoldással a lekérdezések típusbiztosak, könnyen karbantarthatóak, miközben a szerveroldali kód egyszerű, deklaratív formában kommunikál az adatbázissal.

A kliensoldali megjelenítésért, így például a 2.2.3-ben ismertetett admin felületért, a React és TypeScript páros felel, amelyre a stílusokat a TailwindCSS [23] szabályrendszere építi rá. A moduláris komponensekkel dolgozó felépítés lehetővé teszi a gyors újrafelhasználást és egységes megjelenést. A térképes vizualizációk kezelése Leaflet [14] segítségével történik.

A 2.2.3 fejezetben bemutatott állapotkezelési mechanizmus a React Context API-ján alapszik. Ezzel a megközelítéssel a hálózatokhoz, anchorokhoz, tagekhez és felhasználókhoz tartozó adatok globálisan elérhetők maradnak, miközben a komponenshierarchián belül kontextusokon keresztül kezelhetők. Ez garantálja, hogy az egyes nézetek (pl. dashboard, térkép) mindig a legfrissebb és legrelevánsabb adatokkal dolgozzanak.

Az adatok validálását egy Zod [31]-alapú sémarendszer végzi, amely egyszerre szolgál

futásidejű validátorként és TypeScript-típusdefinícióként. Ezzel kiküszöbölhető a duplikált típusdeklaráció, és a séma változásait a validáció automatikusan leképezi a statikus típusokban is. A kódbázis minőségbiztosításáért az ESLint konfiguráció felel, amely nemcsak a szintaktikai következetességet, hanem bizonyos hibalehetőségek megelőzését is támogatja.

A szerver határain túl, a 2.2.3 részben szereplő Bridge komponens egy Python nyelvű alkalmazás, amely az egyes gateway-ek helyi MQTT brokereitől érkező adatokat gyűjti össze és továbbítja a központi szerver felé. A Qorvo által biztosított Linux-alapú operációs rendszer nem tartalmazza az ehhez szükséges Python-verziót, ezért a rendszer működtetése érdekében szükséges a Python 3.6 forrásból történő lefordítása, ezzel biztosítva a kompatibilitást és a stabil üzenetcserét.

3.2. Kliens

A mobilalkalmazásunk TypeScript nyelven, a React Native keretrendszerrel készült. Ez lehetővé teszi, hogy egy közös kódbázisból natív iOS és Android alkalmazásokat fejlesszünk, valódi natív élményt biztosítva a felhasználóknak – nem csupán egy webes felületet. [9]

A React Native nem HTML-re, hanem a platform saját natív UI-elemeire renderel. Így az app felülete ugyanazokat az API-kat használja, mint egy natívan írt alkalmazás, míg a logika JavaScriptben (vagy inkább TypeScriptben) íródi. [9]

A komponensek frissülése a klasszikus React logikát követi: a *state* vagy *props* változásakor automatikusan újrarenderelődnek. Ez reszponzív és konzisztens felhasználói élményt biztosít, miközben a háttérfolyamatok külön szálon futnak. [9]

A fejlesztés során az Expo eszköztárat használtuk. Az **Expo CLI**, **Expo Go** és **Expo Snack** eszközök megkönnyítik a fejlesztést, tesztelést és karbantartást, még valós eszközön vagy akár böngészőben is. [5]

A TypeScript típusosságának köszönhetően már fejlesztés közben észlelhetők a hibák, ami segíti a strukturáltabb, karbantarthatóbb kód írását. Az Expo és a React Native CLI automatikusan felismeri a TypeScript fájlokat, így a beállítás is gyors és zökkenőmentes. [27]

4. Az alkalmazás működése

Ez a fejezet részletesen bemutatja az UWB alapú helymeghatározó rendszer üzembe helyezésének lépéseit, valamint az adminisztrátori és mobilalkalmazás-felhasználói felületek működését.

4.1. A rendszer üzembe helyezése

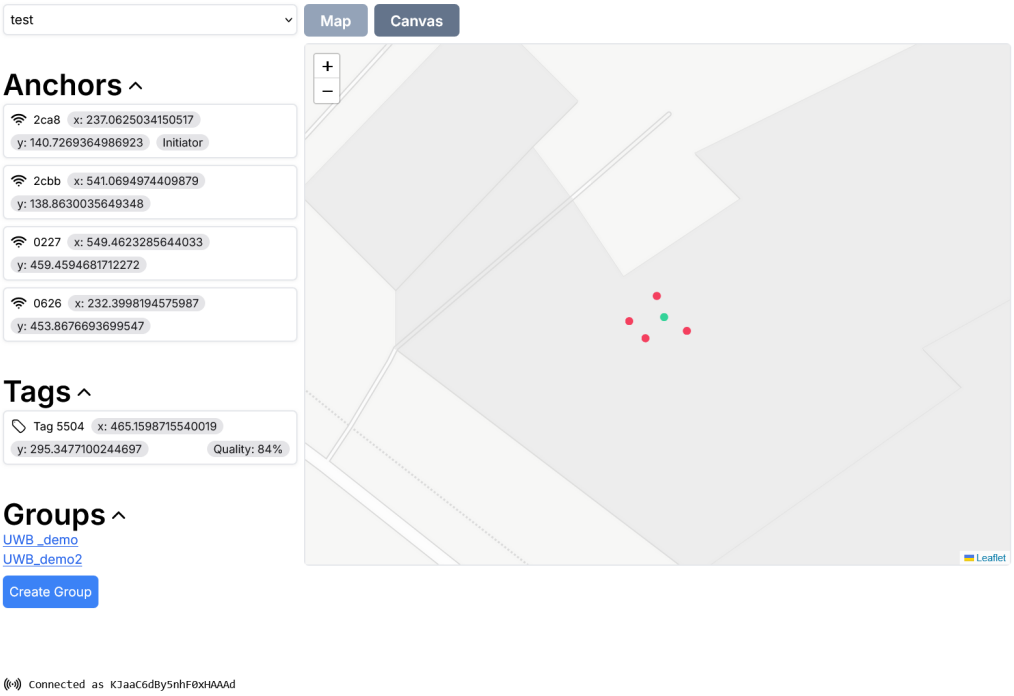
Az UWB hálózat üzembe helyezése viszonylag hosszadalmas és összetett folyamatnak tekinthető. Első lépésként szükség van egy flasher programra – a Qorvo által ajánlott eszköz a Segger [6], amellyel a Qorvo weboldaláról letölthető firmware image fájlokat lehet a chipekre telepíteni. A telepítés megkezdése előtt ki kell választani a megfelelő eszköztípust, jelen esetben az `nRF52832`(ld. 2.1. fejezet) chipet.

Ezt a lépést minden olyan eszközön el kell végezni, amelyet a hálózatban használni kívánunk. Ezen felül az egyik eszközt fizikailag össze kell illeszteni egy Raspberry Pi-vel, létrehozva ezzel a gateway egységet. A gateway konfigurációját egy fájlon keresztül kell elvégezni, amelyben meg kell adni a hálózat PAN ID-ját aminek szeretnénk a részévé tenni.

A sikeres konfigurálás után a gateway eszköz egy beépített webes felületet tesz elérhetővé, amelyen keresztül adminisztrátor felhasználók megtekinthetik az hálózat minden eszközét – egyrészt listázva, másrészt vizuálisan is, egy tetszőleges koordinátarendszerre helyezve. Az egyes eszközök ezen a felületen könnyedén konfigurálhatók a kívánt szerepkörökre (anchor, tag, listener).

Alternatív megoldásként az UWB eszközök rendelkeznek egy ”wakeup” gombbal, amely aktiválja a beépített Bluetooth funkciót. Ilyen esetben a hivatalos Qorvo mobilalkalmazás használatával Bluetooth-kapcsolaton keresztül is elvégezhető a konfigurálás.

A szükséges szerepkörök beállítása után az anchor szerepű eszközöket fizikailag el kell helyezni a térben – például egy teremben vagy épületen belül – olyan pontokon, ahol a lehető legnagyobb eséllyel biztosítható a zavartalan kapcsolat (nem nagyon fedve, nem túl közel a plafonhoz vagy padlóhoz). Ezt követően elindítható az automatikus pozíciókalibráció (auto positioning), amely során az anchor eszközök meghatározzák egymáshoz viszonyított távolságaikat és ebből számolják ki saját koordinátáikat. Az origó ebben a rendszerben az initiator pozíciója, amely alapértelmezetten a (`X=0;Y=0`) pontban van és a tőle jobbra lévő első anchor poziciója (`X=n,Y=0`) lesz, vagyis ez az első két eszköz fogja meghatározni az OX



8. ábra. Térképes nézet

tengelyt. Ezt külön figyelembe kell venni minden egyes alkalommal, és ehhez a limitációhoz alkalmazkodva felépíteni a hálót.

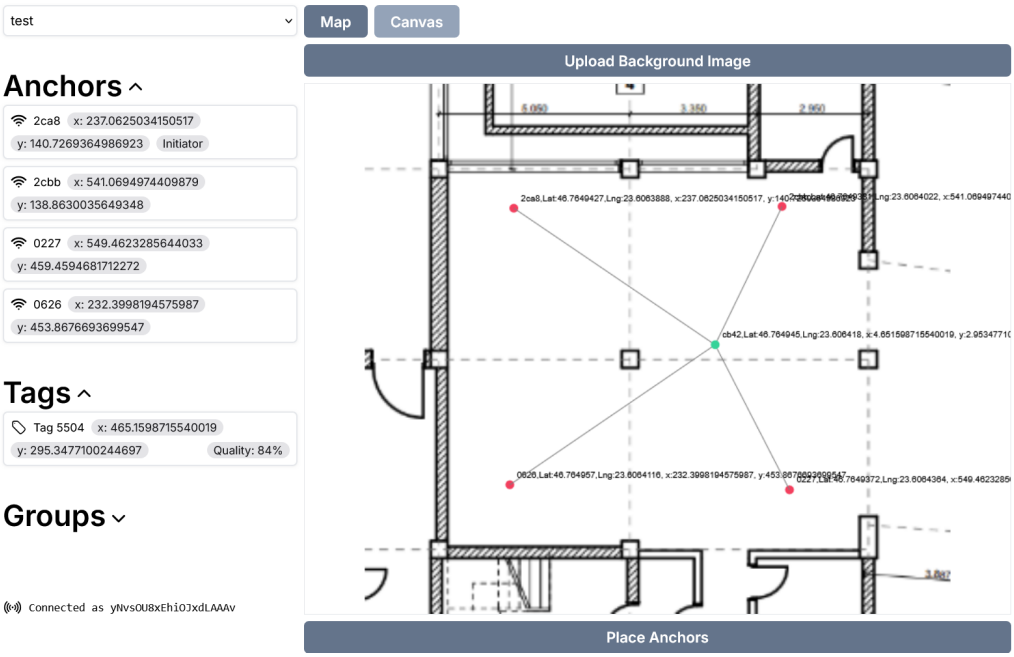
Miután a pozícionálási folyamat lezárult, a tagek automatikusan követhetővé válnak a hálózaton belül, és megkezdi a hasznos pozícióadatok továbbítását a gateway felé.

4.2. Admin felület & mobil felhasználó

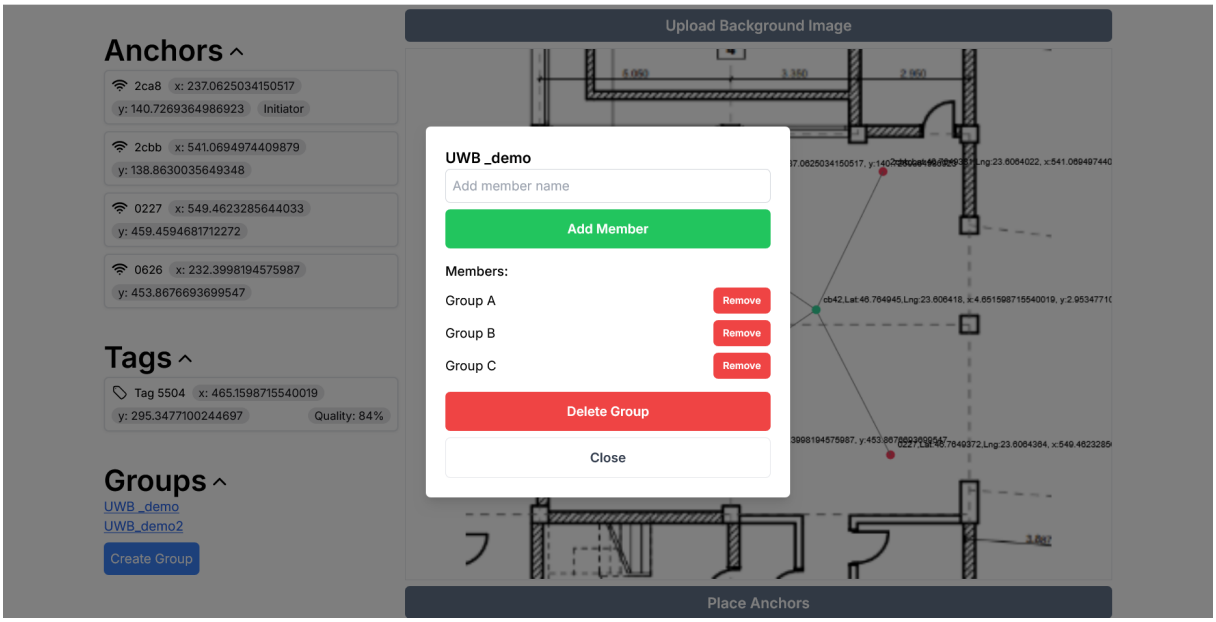
Szerkezetét tekintve az admin felület egy kapcsolatállapot-jelző modullal indul, amely visszajelzést ad a rendszer elérhetőségéről, és képes reagálni a 2.2.2 szakaszban bemutatott hibaeseményekre. Ezt követi egy hálózatválasztó legördülő menü, amely lehetővé teszi, hogy a felhasználó kiválassza a számára releváns hálózatot. A kiválasztás hatására minden, az adott hálózathoz tartozó adat – anchorok, tagek és csoportok – automatikusan megjelenik, és igény szerint szekciónként könnyedén ki- és becsukható.

A csoportok külön kezelőfelületet is kaptak, ahol lehetőség van új csoportok létrehozására, meglévők szerkesztésére vagy akár teljes törlésére is (ld. 10. ábra). Ez különösen hasznos például akkor, ha a felhasználók egy-egy területet (pl. raktárszekció, konferenciaterem stb.) logikai egységként szeretnének kezelni.

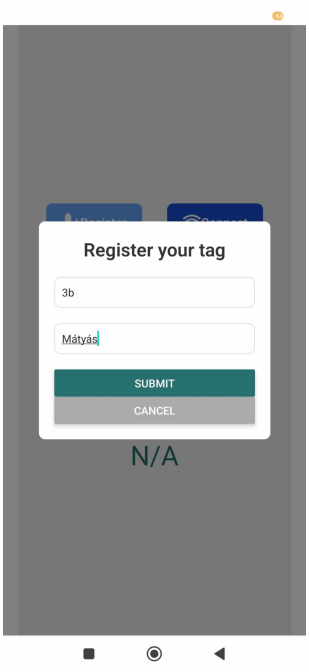
A térképes nézet (ld. 8. ábra) interaktív megjelenítést kínál: a Leaflet térképre kerülnek rá



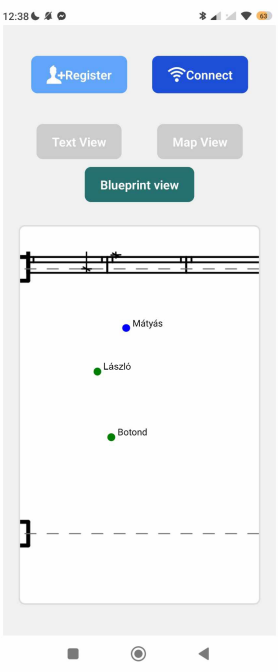
9. ábra. Canvas nézet



10. ábra. Csoport szerkesztése



11. ábra. A kapcsolódási menü



12. ábra. A tervrajz nézet

az anchor- és tag-ikonok, melyek folyamatosan frissülnek a szerverről érkező pozícióüzenetek hatására. További különlegesség a *Canvas*-nézet (ld. 9. ábra), amely lehetővé teszi egyedi alaprajzok feltöltését és az anchorok grafikus pozícionálását. A háttérként feltöltött képen kattintással jelölhetők ki a referenciapontok, majd egy gombnyomással menthető a frissített elrendezés a szerverre. Így egyszerre kapunk egy könnyen testreszabható vizuális szerkesztőt és egy erős, valós idejű monitoringrendszert.

Hasonló működésre alapszik a mobilalkalmazás is. A rendszer használatához első lépésként egy tetszőleges névvel szükséges regisztrálni. Ez azt jelenti, hogy a választott név hozzá lesz rendelve egy konkrét, létező belső azonosítóhoz (internal ID). Ezzel a hozzárendeléssel válik lehetővé, hogy az adott tag a vizuális felületeken megjelenjen és nyomon követhetővé váljon (ld. 11. ábra). Amennyiben a regisztráció megtörtént, és a felhasználó megadta a választott nevet, az alkalmazás azonnal megjeleníti az aktuális pozíciót, valamint a kapcsolódó rendszeradatokat.

Ha a felhasználó egy csoport tagja, az alkalmazás automatikusan megjeleníti az összes olyan tag eszközt, amely ugyanebbe a csoportba tartozik, és amely aktív kapcsolatban áll a szerverrel. A különböző tagek színkódolással kerülnek megkülönböztetésre a könnyebb azonosíthatóság érdekében. Ez különösen hasznos lehet olyan környezetekben, ahol egyszerre több személy vagy eszköz mozgását szükséges figyelemmel kísérni — mint az összejövetelek, telephelyek vagy egészségügyi intézmények.

A rendszer úgy lett kialakítva, hogy az információk háromféleképpen jelenjenek meg az

alkalmazáson belül, a különböző igényekhez igazodva:

1. Szöveges formában, egyszerű módon.
2. Interaktív térképen, amely a Leaflet térképkönyvtárat használja a földrajzi pozíciók megjelenítéséhez.
3. Alaprajz alapú felületen, amely lehetővé teszi az épületen belüli tájékozódást.

Különösen a második és harmadik lehetőség esetén válik szemléletessé az adatok értelmezése, hiszen a felhasználó vizuálisan is nyomon követheti a tagje mozgását a térképen vagy az alaprajzon.

Továbbfejlesztési lehetőségek

A jelenlegi rendszer bár működőképes, több ponton is fejlesztésre szorul, különösen ha célunk a hosszú távú skálázhatóság, pontosság és felhasználóbarát működés biztosítása.

A DWM1001-es modulokra épülő jelenlegi rendszer több szempontból is korlátokat állít a megvalósítók elé. Egyrészt maga a technológia érzékenyen reagál a környezeti viszonyokra: a közvetlen rálátás hiánya az anchor eszközökre a tag részéről jelentősen rontja a pozícióbecslés pontosságát. Ennél is nagyobb problémát jelent azonban az, hogy a DWM1001 által megkövetelt anchor-elrendezési szabályok szigorúak és viszonylag rugalmatlanok (ld. 4.1. fejezet). Ezek az elrendezési kritériumok nehezen alkalmazhatók egyedi építészeti környezetekre vagy komplex beltéri struktúrákra, így a rendszer implementálhatósága helyszíntől függően erősen korlátozott. Ez a tényező komoly kihívást jelent a széles körű gyakorlati alkalmazásban, és hangsúlyozza annak szükségességét, hogy a jövőben rugalmasabb architektúrák irányába induljunk el, például modernebb UWB modulok vagy alternatív pozíciókövetési módszerek bevonásával.

Egy másik kihívás a hardverek energiaigénye. A Raspberry Pi és az UWB modulok jelenlegi konfigurációja nem kedvez a nagyobb léptékű telepítéseknek, mivel a magas fogyasztás jelentősen csökkenti az üzemidőt és növeli az energiaellátás komplexitását. Fontos ugyanakkor kiemelni, hogy projektünk elsődleges célja nem az optimális áramellátás kidolgozása volt, így az energiafogyasztásra vonatkozóan csupán kezdetleges tapasztalatokat szereztünk. A jövőben viszont mindenképp érdemes lenne alacsonyabb fogyasztású, egyedi hardvermegoldások irányába elmozdulni, amelyek akár akkumulátoros működés mellett is hosszabb távú használatot biztosítanak.

A trilaterációs algoritmus finomhangolása szintén kulcsfontosságú, mivel a jelenlegi rendszerben tapasztalható eltérések az absztrakt `(x, y)` koordináták és a valós világban használt földrajzi hosszúsági- és szélességi értékek között pontatlanságokhoz vezetnek. Ezek az eltérések a rendszer vizualizációját is torzíthatják, így pontosabb illesztés, kalibráció és mélyebb algoritmikus optimalizálás szükséges.

A pozíciófrissítések simítására a Kiterjesztett Kalman-szűrő (EKF) alkalmazása egy ígéretes fejlesztési irány. Az EKF segítségével az eszközmozgások becslése folyamatosabbá válik, miközben a mérési zajok hatását is csökkenthetjük. Ez nemcsak a megjelenítés élményét javítja, hanem a rendszer megbízhatóságát is növeli.

A jövő szempontjából rendkívül izgalmas lehetőséget kínál a DWM3001 típusú UWB modulok támogatásának bevezetése. Ezek az újabb generációs eszközök már képesek

kommunikálni a legmodernebb UWB-támogatással rendelkező okostelefonokkal (pl. iPhone 11-től kezdve [2] és újabb Samsung, Google Pixel modellek), megnyitva az utat újfajta interakciós modellek és mobilalkalmazások előtt. Ráadásul a DWM3001 modul képes az eszköz irányának (beesési szög) meghatározására is, ami teljesen új dimenziót adhat a térbeli pozícióérzékelésnek.

Következtetések és tapasztalatok

A projekt során egy stabil és modulárisan felépített rendszert sikerült létrehozni, amely fejlesztői és felhasználói szempontból egyaránt jól működik. Azonban az implementáció és a rendszer valós környezetben történő használata közben számos olyan technikai és gyakorlati kihívással szembesültünk, amelyeket a dokumentáció nem, vagy csak részben tárgyalt. Így a gyakorlatban szerzett tapasztalataink alapján indokolt néhány ponton árnyaltabban értelmezni a dokumentációs anyagok állításait.

A probléma abban állt, hogy egy elméletileg működőképes rendszert valós környezetben kellett működőképessé tenni, úgy, hogy a környezet sajátosságai – például építészeti adottságok, energiaellátási kérdések, telepítési szabályok – jelentős mértékben korlátozták a rendszer szabályos kialakítását. Ez különösen fontos volt, mert csak így nyerhettünk valós képet arról, milyen mértékben alkalmazható a technológia a gyakorlatban. Végső soron arra jutottunk, hogy bár számos kompromisszumra volt szükség, a rendszer alapjai stabilak, és megfelelő fejlesztési irányok kijelölésével ipari szinten is használhatóvá válhat.

A dokumentáció és saját tapasztalataink közötti esetleges ellentmondásokat részben az is indokolja, hogy projektünk fókusza nem a mérési pontosság vagy energiaoptimalizálás volt, hanem egy működő rendszer létrehozása és annak gyakorlati kipróbálása. A mérésekkel kapcsolatos ismereteink ezért jelenleg kezdetlegesek, azonban ezek a jövőbeli fejlesztési munkák fontos alapját képezhetik.

Összességében tehát bizakodásra ad okot, hogy a jelenlegi struktúra megfelelő kiindulópont lehet egy még pontosabb, megbízhatóbb és energiahatékonyabb pozíciókövető rendszer létrehozásához.

Hivatkozások

[1] Szüllő Ádám. „MULTILATERÁCIÓ A GYAKORLATBAN – WAMLAT PILOTRENDSZER”. *Repüléstudományi közlemények* (2016). URL: https://www.repulestudomany.hu/kulonszamok/2013_cikkek/2013-2-53-Szullo_Adam.pdf.

[2] *Apple devices with UWB chip*. URL: https://support.apple.com/en-us/109512 (elérés dátuma: 2025. ápr. 18.)

[3] *Channel Effects on Communications Range and Time Stamp Accuracy*. 2024. URL: https://www.qorvo.com/products/d/da008440 (elérés dátuma: 2025. ápr. 13.)

[4] Yaqin Chen. „Evaluating Off-the-shelf Hardware for Indoor Positioning”. (2017).

[5] *Core concepts*. URL: https://docs.expo.dev/core-concepts/ (elérés dátuma: 2025. ápr. 16.)

[6] *DWM1001 Firmware User Guide*. 2024. URL: https://www.qorvo.com/products/d/da008479 (elérés dátuma: 2025. ápr. 19.)

[7] *DWM1001 Gateway Quick Deployment Guide*. 2024. URL: https://www.qorvo.com/products/d/da008479 (elérés dátuma: 2025. ápr. 13.)

[8] *DWM1001 System Overview And Performance*. 2020. URL: https://www.qorvo.com/products/d/da007974 (elérés dátuma: 2025. márc. 31.)

[9] Bonnie Eisenman. *Learning React Native: building mobile applications with JavaScript*. O'Reilly, 2015.

[10] *Global Positioning System History*. 2012. URL: https://www.nasa.gov/general/global-positioning-system-history/ (elérés dátuma: 2025. ápr. 3.)

[11] *How UWB is Taking Over The World: Exploring the Many Uses of Ultra-Wideband*. 2023. URL: https://ubisense.com/many-uses-ultra-wideband/ (elérés dátuma: 2025. márc. 31.)

[12] Xu Hui. „RFID-Based Positioning Technology”. *Computer Knowledge and Technology* (2013).

[13] Changhui Jiang és mások. „UWB NLOS/LOS Classification Using Deep Learning Method”. *IEEE Communications Letters* 24 (2020), 2226–2230. oldal. DOI: 10.1109/LCOMM.2020.2999904.

[14] *LeafletJS*. URL: https://leafletjs.com/ (elérés dátuma: 2025. ápr. 18.)

[15] *NextJS Framework*. URL: https://nextjs.org/ (elérés dátuma: 2025. ápr. 18.)

[16] Scott Pace. „The global positioning system: policy issues for an information technology”. *Space Policy* 12.4 (1996), 265–275. oldal. ISSN: 0265-9646. DOI: https://doi.org/10.1016/0265-9646(96)00028-8. URL: https://www.sciencedirect.com/science/article/pii/0265964696000288.

[17] *Prisma ORM*. URL: https://www.prisma.io/ (elérés dátuma: 2025. ápr. 18.)

[18] Yu-Chi Pu és P. You. „Indoor positioning system based on BLE location fingerprinting with classification approach”. *Applied Mathematical Modelling* (2018). DOI: 10.1016/J.APM.2018.06.031.

[19] P. Sadhukhan és mások. „An efficient clustering with robust outlier mitigation for Wi-Fi fingerprint based indoor positioning”. *Appl. Soft Comput.* 109 (2021), 107549. oldal. DOI: 10.1016/J.ASOC.2021.107549.

[20] George T. Schmidt. „GPS/INS Technology Trends for Military Systems”. 1997. URL: https://api.semanticscholar.org/CorpusID:110692014.

[21] Shuyu Shi és mások. „Accurate Location Tracking From CSI-Based Passive Device-Free Probabilistic Fingerprinting”. *IEEE Transactions on Vehicular Technology* 67 (2018), 5217–5230. oldal. DOI: 10.1109/TVT.2018.2810307.

[22] *SocketIO*. URL: https://socket.io/ (elérés dátuma: 2025. ápr. 18.)

[23] *TailwindCSS*. URL: https://tailwindcss.com/ (elérés dátuma: 2025. ápr. 18.)

[24] R. Tesoriero és mások. „Improving location awareness in indoor spaces using RFID technology”. *Expert Syst. Appl.* 37 (2010), 894–898. oldal. DOI: 10.1016/j.eswa.2009.05.062.

[25] Cheng Tu és mások. „UWB Indoor Localization Method Based on Neural Network Multi-classification for NLOS Distance Correction”. *Sensors and Actuators A: Physical* (2024). DOI: 10.1016/j.sna.2024.115904.

[26] *Ultra-Wideband (UWB) Technology: What is it and what is it used for?* 2023. URL: https://synzen.com.tw/blog/item/what-is-uwb-technology (elérés dátuma: 2025. ápr. 8.)

[27] *Using TypeScript*. URL: https://reactnative.dev/docs/typescript (elérés dátuma: 2025. ápr. 16.)

[28] *What Are the Limitations of GPS Technology?* 2024. URL: https://beemaps.com/blog/limitations-of-gps/ (elérés dátuma: 2025. ápr. 3.)

[29] *What is MQTT?* URL: https://aws.amazon.com/what-is/mqtt/ (elérés dátuma: 2025. ápr. 15.)

[30] *What Is Ultra-wideband (UWB) Wireless Communication?* 2023. URL: https://article.murata.com/en-global/article/what-is-uwb-wireless-communication#:~:text=UWB%20is%20the%20abbreviation%20for,consumer%20equipment%20in%20recent%20years. (elérés dátuma: 2025. márc. 31.)

[31] *Zod library*. URL: https://zod.dev/ (elérés dátuma: 2025. ápr. 18.)