Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

Beltéri pozícionáló rendszer fejlesztése UWB alapokon

Szakdolgozat

Készítette
Molnár Marcell

Konzulens
dr. Vida Rolland

Külső konzulens
Luspay Tamás Gábor

2018. december 7.

Tartalomjegyzék

Ábrák jegyzéke

HALLGATÓI NYILATKOZAT

Alulírott *Molnár Marcell*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2018. december 7.

_____

Molnár Marcell
hallgató

Kivonat

Napjainkban egyre több helyen jelenik meg az igény egy beltéri pozícionáló rendszerre. Helyfüggő szolgáltatások, személyre szabott kirakatok, intelligens otthoni világítás, önmagától beparkoló autók. Ezek csupán a felhasználási körök egy töredéke, ahol alkalmazni lehet a technológiát.

Az Ipar 4.0 fejlődése is megkívánja, hogy egy gyár rendelkezzen ilyen rendszerrel. Ahhoz, hogy például az önműködő árumozgató robotok fel tudják ismerni a helyzetüket és ki tudják kerülni egymást, szükséges egy pozícionáló rendszer. A logisztikában egyre nagyobb problémákat vet fel a hatékonyság, de egy pozícionáló rendszer segítségével nyomon lehet követni az árumozgatási folyamatokat és ezáltal lehetőség nyílik az optimalizálásra.

Jelen dolgozatban egy olyan beltéri pozícionáló rendszert és annak fejlesztését mutatom be, amely egy teremben elhelyezkedő eszköz helyzetét tudja nyomon követni. A fejlesztés a Magyar Tudományos Akadémia (MTA) Számítástechnikai és Automatizálási Kutatóintézet (SZTAKI) Rendszer és Irányításelméleti Kutatólaboratóriumának (Systems and Control Lab - SCL) készül. A cél egy olyan platform létrehozása, ahol drón-szabályozási algoritmusok tesztelhetők, ennek pedig elengedhetetlen kelléke egy megbízható pozícionáló rendszer.

Abstract

Nowadays the demand for an indoor positioning system appears at more and more places. Location-specific services, personalized shop windows, smart home lighting, self-parking cars, these are just a fraction of the use cases, where this technology can be applied.

The development of Industry 4.0 also requires the factories to have such a system. In order to make the autonomous robots able to avoid crashing into each other, a positioning system is required. In the logistics, an increasing problem appears in the efficiency, but with the aid of a positioning system, we can monitor the product moving processes and thereby be able to optimize them.

In this thesis, I present a positioning system and its development that can track the location of a device in a room. The development is for the Systems and Control Lab of the Institute for Computer Science and Control, a research institute that is part of the Hungaryian Academy of Sciences. The goal is to create a platform where quadcopter controlling algorithms can be tested; a reliable positioning system is essential for this purpose.

Bevezető

A beltéri pozícionáló rendszerek jó pár éve szolgálnak a kutatás-fejlesztési területek egyik meghatározó témájául. Az igény, hogy pontosan ismerjük egy eszköz, tárgy vagy személy helyzetét nem újdonság. A helymeghatározás első nagy mérföldköve a GPS (Global Positioning System) volt. A rendszert az Amerikai Egyesült Államok Haditengerészete kezdte el fejleszteni, amelynek előzménye a Szputnyik-1, a Föld első műholdja volt. A Doppler-effektus vizsgálatával pontosan meg tudták határozni a műhold helyzetét. Az amerikai haditengerészet következő fejlesztése egy olyan rendszer volt, amely segítségével a tengeralattjárók 6 műhold figyelésével meg tudták határozni a helyzetüket. [11] Azóta számos fejlesztésen esett át a rendszer és ma már több, mint 20 műhold segítségével egy 24 órás helymeghatározó rendszer érhető el a Föld bármely pontján.

A GPS-nek több hátránya is van. Az egyik az, hogy sok időbe (akár percekbe is) telhet a kezdeti szinkronizáció. Hátrány az is, hogy a rendszer pontossága méter nagyságrendű, ezért nem használható, ha kis mozgásokat szeretnénk meghatározni, például egy szobán belül. Mindemellett beltéren jelentősen csökken a vett jel erőssége és a rendszer használhatatlanná válik, ezért a beltéren történő helymeghatározásra a GPS helyett másik technológiát kell választanunk.

Beltéri helymeghatározásra számos megoldás született már, azonban nincs egy olyan általános megoldás, amely jóval népszerűbb lenne a többinél. Ezek a rendszerek nagyrészt a rádiós kommunikáció technológiájában térnek el egymástól, így ez alapján csoportosíthatjuk őket. A WiFi alapú technológiák előnye, hogy már kiépített infrastruktúrát használhatunk, hiszen nagyon sok helyen jelen van a technológia. A vett jelerősség (RSSI - Received Signal Strength Indicator) alapján távolságot és háromszögelés segítségével pozíciót tudunk meghatározni. Ennél eggyel kifinomultabb módszer az ún. "fingerprinting", amikor feljegyzéseket készítünk lehetőleg minél több pontban a vett jelerősségekről, valamint a mért pont koordinátáiról és ezeket eltároljuk egy adatbázisban. Működés során az RSSI értékeket összehasonlítjuk az adatbázisban található értékekkel és a legpontosabb egyezést mutató rekordhoz tartozó koordináta lesz a pozícionálás eredménye. A hátránya ennek a megoldásnak az, hogy az RSSI értékek nagymértékben ingadozhatnak a környezetben történő

változás hatására.

Egy másik széles körben elterjedt technológia a Bluetooth. A Bluetooth SIG (Special Interest Group) állítása szerint a Bluetooth kizárólag a közelség érzékelésére használható. Ennek ellenére számos megoldás alkalmazza ezt a technológiát és a Bluetooth hálózati topológiájának köszönhetően egy cellás rendszer alakítható ki, amellyel akár egy 111000 km^2 nagyságú terület is lefedhető [5].

Az UWB (Ultra-wideband) egy olyan rádiós technológia, amely nagyon nagy sávszélességet használ és ezáltal nagy adatátviteli sebességet érhetünk el vele. Mivel nagyon keskeny impulzusokat használ, ezért a vevő pontosan el tudja különíteni az egyes impulzusokat, mert nem jelentkeznek a többutas terjedésből fakadó problémák. A rádiós technológiák közül ez kecsegtet a legprecízebb, közel centiméteres pontosságú távolságméréssel. A keskeny impulzusoknak köszönhetően egy UWB-s eszköznek a fogyasztása is alacsony lehet, így hosszú élettartam érhető el az elemről működtetett készülékeknél.

Dolgozatomban egy UWB technológián alapuló RTLS (Real-Time Location System) rendszer megvalósítását mutatom be. Az RTLS rendszerek ismérvei a következők. "Real-Time", azaz azonnal vagy nagyon kis késleltetéssel tudunk információt kapni egy eszköz helyzetéről. "Location", azaz olyan információt hozunk létre, amely egy tárgy vagy ember helyzetét adja. "System", vagyis az adatok gyűjtése, feldolgozása és megjelenítése hardverek és szoftverek sokaságával, egy rendszerezett és strukturált úton történik.

A dolgozat 1. fejezetében röviden bemutatom az UWB technológiát, a jellemzőit, továbbá néhány beltéri pozícionáló rendszert, amelyek kész termékként megvásárolhatók. A 2. fejezetben a távolságmérés és az ez alapján történő helymeghatározás kérdéskörét tárgyalom, összehasonlítva a lehetséges megoldásokat. A 3. fejezetben a távolságmérés pontosságát meghatározó elemeket és olyan módszereket mutatok be, amelyekkel nagyobb pontosság érhető el. A zajos mérések pontosítására jól használható Kálmán szűrővel a 4. fejezetben foglalkozok. Rövid elméleti bevezetés után a távolságmérés pontosítására való használatát mutatom be. Az 5. fejezet összefoglalva tartalmazza az elkészült pozícionáló rendszert, a fejlesztésének egyes lépéseit és a validálását. Végül az utolsó fejezetben az összefoglalás és továbbfejlesztési lehetőségek találhatók meg.

1. fejezet

Ultra-wideband (UWB) áttekintés

Az UWB egy olyan rádiós technológia, amellyel nagyon kis energiafogyasztás mellett érhető el nagy adatátviteli sebesség. A kis energiafogyasztás annak köszönhető, hogy a kommunikáció során az adóegység időben nagyon rövid impulzusokat bocsát ki. Ezáltal lecsökken a kitöltési tényező, ami az energiafogyasztás csökkenését eredményezi. A rövid impulzusok további következménye, hogy a kibocsátott jel nagyon széles frekvenciatartományt fed le. Az FCC (Federal Communications Commission - Amerikai Szövetségi Távközlési Bizottság) definíciója szerint az UWB olyan rádiós technológia, amely nagyobb sávszélességet használ, mint a sávközépi frekvencia 20%-a vagy nagyobb a sávszélessége, mint 500 MHz.

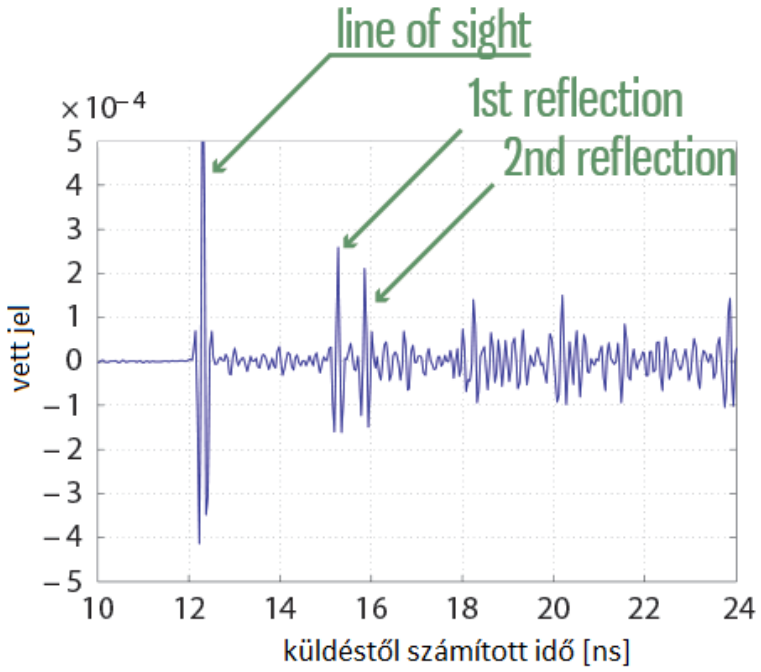
1.1. Heisenberg-féle határozatlansági elv - a sávszélesség megválasztása

A Heisenberg-féle határozatlansági elv kimondja, hogy bizonyos fizikai mennyiségek egy időpillanatban nem figyelhetők meg tetszőleges pontossággal. Ilyen mennyiség például egy szinuszos jel frekvenciája és az időzítése. Ha ismerjük a frekvenciáját, akkor nem tudhatjuk mikor kezdődött. Ha összeadunk egyre nagyobb frekvenciájú szinuszos jeleket, megkaphatjuk a távközlésből ismert emelt koszinusz jelet. Minél több, egyre nagyobb frekvenciájú szinusz jelet adunk hozzá az eredeti jelhez, annál vékonyabb impulzust kapunk. A legnagyobb felhasznált frekvenciát Δf-fel jelöljük és ez lesz a jelünk sávszélessége. A Heisenberg-féle határozatlansági elvből jó becslést kaphatunk az impulzus Δt hosszára:

$$\Delta t \Delta f \geqslant \frac{1}{4\pi} \tag{1.1}$$

A fenti képletből látható, hogy minél keskenyebb impulzust szeretnénk előállítani - ami szükséges a pontos időméréshez -, annál nagyobb sávszélességet kell hasz-

nálnunk. 500 MHz-es sávszélesség esetén 0,16 ns széles impulzust kapunk, vagyis maximum ilyen pontossággal határozhatjuk meg az impulzus vételének idejét. A fénysebességgel számolva ez 4,8 cm pontosságot jelent. [14]



1.1. ábra. *UWB jel időtartományban [14]*

A fenti ábrán látható, hogy ilyen rövid impulzusok esetén a többutas terjedésből adódó problémák is megszűnnek, hiszen egyértelműen elkülönül a reflexiómentes jel (line of sight) a visszaverődöttektől (1st & 2nd reflection).

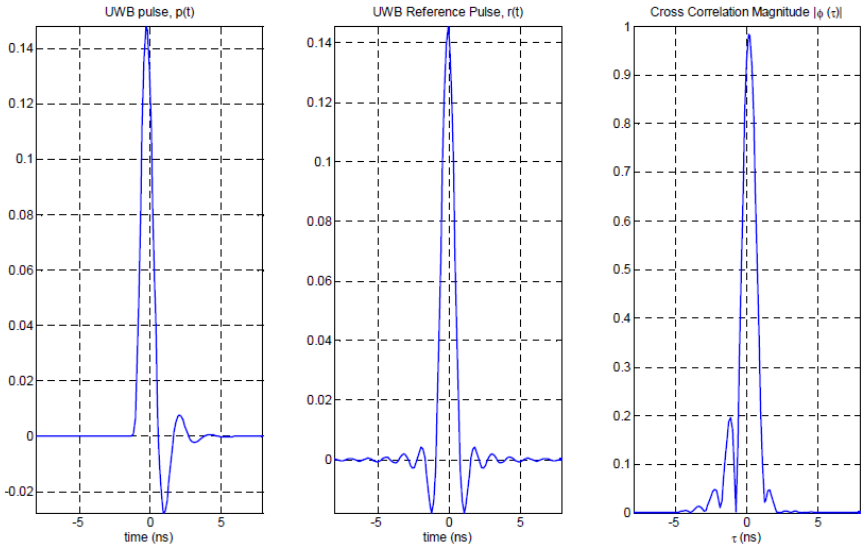
1.2. Az UWB jellemzése

A fizikai jel

Az UWB jelalakja az IEEE 802.15.4a szabványban van meghatározva, amelyet az 1.2. ábra szemléltet. A szabvány [3] előírja, hogy az általunk sugárzott jelnek, milyen mértékű korrelációt kell mutatnia a középen látható emelt koszinuszos referencia jellel. Csak egy bizonyos értéket meghaladó korrelációs szint mellett kompatibilis a kisugárzott jel az IEEE szabvánnyal.

Ahhoz, hogy UWB-vel kommunikálhassunk a helyi szabályozásokat is be kell tartanunk. Az UWB-re szigorú megkötések vonatkoznak az adóteljesítmény és a sugárzási idő tekintetében a nagy sávszélessége miatt. Magyarországon az NMHH (Nemzeti Média- és Hírközlési Hatóság) szabályozza és kezeli a frekvenciasávokat. A részletes szabályozás [15] és az NMHH honlapján olvasható, itt csak a lényeges

elemeket említem meg. Az előírások szerint az UWB alkalmazásnak vagy beltéri használatúnak kell lennie vagy további speciális szabályokat szükséges betartanunk, ha kültéren szeretnénk használni.



1.2. ábra. *Egy UWB kompatibilis jel (balra), a referenciajel (középen) és a kettő közötti korreláció (jobbra) [3]*

Sugárzási idő szempontjából folyamatosan maximum 5 ms ideig adhatunk és a kitöltési tényezőnek[1] másodpercenként 5%, óránként 0,5% alatt kell lennie. Az adóteljesítmény sem lehet tetszőleges nagyságú. Az 1.3. ábrában található táblázat adja meg, hogy adott frekvencia tartományban mekkora maximális átlagos és csúcs EIRP értékkel adhatunk. Az EIRP (Effective Isotropic Radiated Power, azaz izotrópikusan sugárzott egyenértékű teljesítmény) megadja, hogy egy izotróp antennának mekkora adóteljesítménnyel kellene sugároznia ahhoz, hogy ugyanakkora jelszintet érjen el, mint amekkorát elér az adott antenna a fő sugárzási irányában.

A táblázat alapján például a 3,8 és 4,2 GHz-es tartományban maximum -70 dBm/MHz adóteljesítménnyel adhatunk. Az alacsony energiafogyasztás részben ennek az alacsony szintnek is köszönhető. Azonban ennek hátránya, hogy csak kis távolságban tudunk megfelelően kommunikálni, mert a sugárzott jel hamar zajszint alá csillapodik. Annak érdekében, hogy megoldjuk ezt a problémát, több pulzust is elküldünk, amelyek együtt jelentenek 1 bitet. A vevő oldalon ezeket a pulzusokat összeadjuk (legtöbbször átlagolással), majd ahogy egyre több pulzust összeadunk, egy vékony tüske fog kiemelkedni a zajszintből, amely az átküldött bitet reprezentálja.

[1] A kitöltési tényező egy arányszám, amely az adatátvitel időtartamának és a két átvitel között eltelt időnek a hányadosa.

	A	B	C	D
1	**Frekvencia-tartomány**	**Dokumentum**	**Maximális átlagos EIRP-sűrűség [dBm/MHz]**	**Maximális csúcs EIRP-sűrűség [dBm/50 MHz]**
2	<1,6 GHz	2007/131/EK, 2009/343/EK	–90	–50
3	1,6–2,7 GHz	ECC/DEC/(06)04	–85	–45
4	2,7–3,1 GHz	MSZ EN 302 065	–70	–36
5	3,1–3,4 GHz	2007/131/EK, 2009/343/EK	–70	–36
6	3,4–3,8 GHz	ECC/DEC/(06)04	–80	–40
7	3,8–4,2 GHz	MSZ EN 302 065	–70	–30
8	4,2–4,8 GHz		2010. december 31-ig forgalomba hozott berendezések esetén:	
9			–41,3	0
10			2010. december 31. után forgalomba hozott berendezések esetén:	
11			–70	–30
12	4,8–6 GHz	2007/131/EK, 2009/343/EK ECC/DEC/(06)04 MSZ EN 302 065	–70	–30
13	6–8,5 GHz	2007/131/EK, 2009/343/EK	–41,3	0
14	8,5–9 GHz	ECC/DEC/(06)04 MSZ EN 302 065, MSZ EN 302 500-2	–65	–25
15	9–10,6 GHz	2007/131/EK, 2009/343/EK	–65	–25
16	>10,6 GHz	ECC/DEC/(06)04 MSZ EN 302 065	–85	–45

1.3. ábra. *UWB maximális adóteljesítményére vonatkozó táblázat (részlet) [15]*

Az IEEE 802.15.4a szabvány

Az IEEE (Institute of Electrical and Electronics Engineers) 802-es szabványcsaládja, azon belül is a 15-ös munkacsoport (Working Group[2]) foglalkozik a vezeték nélküli személyi hálózatokkal. Az IEEE 802.15.4 szabvány vonatkozik az alacsony sebességű hálózatokra (LR-WAN). A 802.15.4 (2003) megjelenése után 4 évvel jött ki a 802.15.4a (2007) szabvány, amely az eredeti szabvány módosítása és elsősorban a távolságmérés pontosítására, az adatsebességek skálázhatóságára, nagyobb hatótávolságra, valamint alacsonyabb energiafogyasztásra és költségekre fókuszál. A szabvány a MAC (Media Access Control) és a fizikai réteget definiálja. A 802.15.4-es szabvány PSK-ra (Phase-shift keying), ASK-ra (Amplitude-shift keying), frekvencia söpréses modulációra (CSS - Chirp Spread Spectrum), UWB-re, valamint GFSK-ra (Gaussian frequency-shift keying) definiálja a fizikai réteget. Ezek közül minket az UWB érdekel, így csak ezzel foglalkozok ebben a fejezetben. 3 független sáv van megengedve a szabványban. Az egyik egy 1 GHz alatti sáv, amely 249.6 MHz-től 749.6 MHz-ig foglal helyet. A másik két sáv 3.1 GHz-től 4.8 GHz-ig és 6.0 GHz-től 10.6 GHz-ig terjed. Ha megfigyeljük, az első sáv pontosan 500 MHz széles, ezért ebben a sávban csak 1, míg a másik kettőben 4, ill. 11 csatorna található.

[2]Az IEEE szabványok először nagyobb csoportokba vannak rendezve (pl. 802: LAN/MAN), és ezeken belül különböző munkacsoportok (WG - Working Group) különböző technológiákkal foglalkoznak. A 11-es munkacsoport a WLAN-hoz, a 15-ös a WPAN (Wireless Personal Area Network), azaz a vezeték nélküli személyes hálózatokhoz tartozik, de ezeken kívül számos munkacsoport is létezik.

Synchronization header (SHR)	PHY header (PHR)	PHY payload (PSDU)

1.4. ábra. *IEEE 802.15.4 fizikai réteg (PHY) üzeneteinek formátuma [4]*

A fizikai réteg üzeneteinek formátumát az 1.4. ábrán láthatjuk. A "Preamble" és az SFD (Start of Frame Delimiter - Keret Határoló Kezdete) nevű mezővel kezdődik, amelyek együtt alkotják a szinkronizáló fejlécet (SHR - Synchronization Header). Ez a rész szolgál arra, hogy a vevő oldalon érzékelni tudjuk, hogy egy új üzenet fog érkezni. Az üzenet ezen részét egy előre meghatározott sebességgel kell adnunk, amely minden csatornára meg van határozva [3]. Ez azért szükséges, hogy a különböző sebességet használó eszközök is érzékelni tudják, ha már foglalt a csatorna. Az SHR után a fizikai réteg fejléce következik (PHY header), amely többek között meghatározza az adatmező hosszát és hogy mekkora sebességgel fogjuk azt elküldeni.

Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/14	variable	2
Frame Control	Sequence Number	Destination PAN Identifier	Destination Address	Source PAN Identifier	Source Address	Auxiliary Security Header	Frame Payload	FCS
		Addressing fields						
MHR							MAC Payload	MFR

1.5. ábra. *IEEE 802.15.4 MAC réteg üzeneteinek formátuma [4]*

A fizikai réteg üzenetének adatrésze már választható adatsebességgel küldhető. Ez lehet 110 kb/s, 850 kb/s, 6,8 Mb/s vagy 27,2 Mb/s. Ennek az adatmezőnek a felépítése az 1.5. ábrán látható. Az FC (Frame Control) mező tartalmazza az üzenet típusát és a MAC fejléc további mezőinek a leírását. Az SQN (Sequence Number) az üzenet sorszáma. A fejléc többi mezője a hálózat, forrás- és céleszköz azonosító, kiegészítő biztonsági fejléc és az adatmező, amiben az átküldendő hasznos adat van. Az utolsó mező az FCS (Frame Check Sequence), ami egy hibadetektálásra szolgáló 16 bites ellenőrző összeg.

1.3. Jelenlegi UWB technológián alapuló megoldások

Az UWB rendkívül sok előnye miatt számos beltéri pozícionálással foglalkozó cég választja ezt a technológiát. A továbbiakban ezek közül három lényegesebb céget említek meg.

Pozyx Labs

A Pozyx Labs egy 2015-ös alapítású belga cég, akik bel- és kültéri pozícionáló rendszereket készítenek. Leírásuk alapján ők tervezik a rendszerhez a hardvert, ők írnak firmware-t az eszközökre, saját maguk fejlesztenek algoritmusokat és alkalmazást is készítenek a rendszerrel való kommunikáláshoz. A cég egy kickstarter[3] projektből indult és ma már egy több tíz fős cégként üzemelnek. A rendszerük egy Arduino és Raspberry Pi kompatibilis megoldást nyújt, amelyben 4 anchor[4] tudja maximum 4 tag[5] pozícióját meghatározni. A távolságmérésre UWB-t használnak. 1 eszköz helyzetének frissítése 138 Hz-cel történik, ha automatikus a helymeghatározás és 40 Hz, ha PC-ről vezérelve szeretnénk pozíciót meghatározni. Több tag esetén a vezérelt meghatározás frissítési frekvenciája annyival leosztódik, ahány tag van. Egy webes felületen lehet a rendszert vezérelni és szükség esetén kalibrálni, továbbá itt tudjuk nyomon követni a tagek helyzetét. [14] Az UWB kommunikáció a DecaWave által gyártott DWM1000 chippel történik.

Sewio

A Sewio azért egy különleges cég, mert olyan vásárlóik vannak, mint a Volkswagen, a Škoda, a Pirelli, a gumiabroncsokat gyártó Matador és a cseh sörgyártó Budweiser Budvar. A Sewio RTLS rendszere a pozíció meghatározáshoz a TDoA (Time Difference of Arrival) megoldást alkalmazza, azaz egy tag helyzete az alapján lesz meghatározva, hogy a tag által elküldött csomag, mekkora időkülönbséggel érkezik meg az egyes anchorokhoz. Ebben a megoldásban elengedhetetlen, hogy az anchorok kb. 100 ps-os pontossággal legyenek szinkronizálva, ugyanis 100 ps-os hiba az időmérésben már 3 cm-es eltérést jelent a távolságban. [16] A Sewio rendszere szintén a DWM1000 chipet használja.

[3] A kickstarter egy olyan platform, ahol ötleteket lehet közzétenni, hogy közösségi finanszírozásból meg lehessen valósítani.

[4] Az anchorok fix telepítésű eszközök, amelyeknek ismerjük a helyzetét és a segítségükkel pozícionálunk.

[5] A tag az az eszköz, amelynek meg szeretnénk határozni a helyzetét.

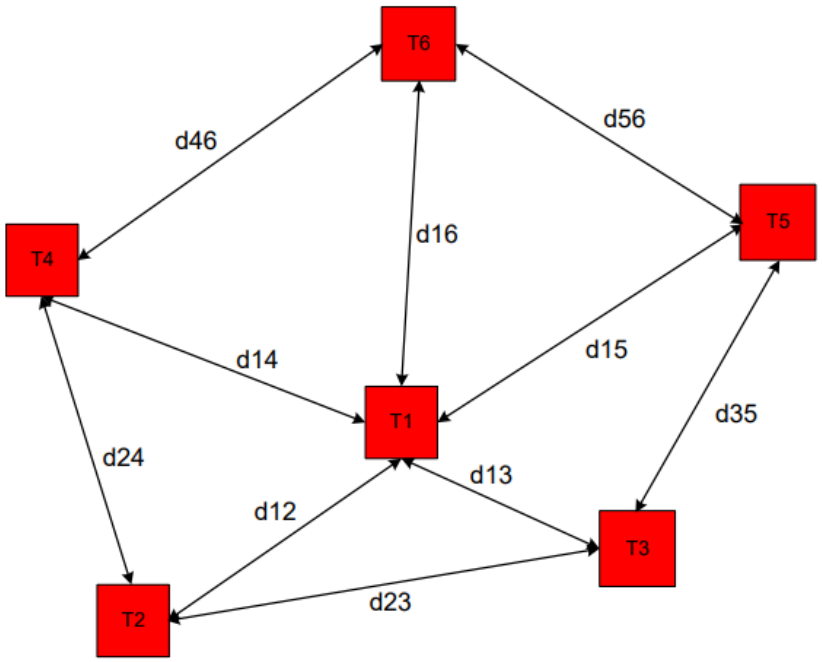
DecaWave

A DecaWave gyártja azt a chipet, amelyet szinte az összes UWB-n alapuló beltéri pozícionáló rendszer használ. Ez a chip a DWM1000 nevet kapta és kifejezetten a beltéri pozícionálásra lett kifejlesztve. Az IEEE 802.15.4-es szabványnak megfelelően működik, több csatornát és adatátviteli sebességet támogat. A termék weboldalán 10 cm pontosságú távolságmérést ígér a gyártó, amelyet azonban jelentősen lehet javítani különböző módszerek segítségével. A DecaWave-től külön megvásárolható mind a rádiós IC, mind a modul, amely az IC-n kívül antennát is tartalmaz. Továbbá különböző ún. "Evaluation Kit"-ek is elérhetők, amelyek egy kész modulon tartalmazzák a DWM1000 chipet, a hozzátartozó antennát és egy vezérlő, vagy "host" mikrokontrollert, amely a chip vezérléséért felelős. A csomagtól függően vagy 1 anchor és 1 tag közti távolságmérést lehet bemutatni, vagy a 3 anchorból és 1 tagből álló, a DecaWave által fejlesztett RTLS rendszert tudjuk kipróbálni. A chip előnye, hogy a gyártó honlapjáról ingyenesen letölthetők hozzá példaprogramok. Ezek a programok az STM32 típusú mikrokontroller családra lettek megírva, így, ha mi is STM32 típusú mikrokontrollert választunk, akkor csupán a lábkiosztást kell megváltoztatnunk, a kód többi része az egységes API (Application Programming Interface) miatt ugyanúgy használható.

2. fejezet

A pozíció meghatározás folyamata

Ebben a fejezetben különböző pozícionálási technikákat mutatok be, rávilágítva az egyes megoldások előnyeire és hátrányaira. A cél viszont mindegyik módszernél közös: egy eszköz pozíciójának meghatározása. Már itt több jelentéssel bírhat ez a kifejezés, ugyanis beszélhetünk relatív vagy abszolút pozícióról. Előbbire akkor lehet szükség, amikor egy helyen nincs kiépítve dedikált infrastruktúra a pozícionáláshoz, mégis szeretnénk bizonyos eszközök helyének megállapítását. Ilyenkor használhatjuk a relatív lokalizációt, amely mozgó eszközök egymáshoz viszonyított helyzetét hivatott meghatározni.



2.1. ábra. *Relatív pozícionálás folyamatosan mozgó eszközökkel [8]*

Erre egy szemléletes példa, amikor tűzoltók bemennek egy házba, ahol nincs ki-

épített helymeghatározó rendszer, azonban szükséges, hogy ismerjük a helyzetüket vészhelyzet esetén. Minden tűzoltónál van egy-egy készülék, amelyek ad-hoc módon hálózatot alakítanak ki és egymás közötti üzenetváltásokkal meg tudják határozni az egymáshoz viszonyított pozíciójukat. Ezzel szemben az abszolút lokalizáció az, amikor van egy fix helyre (például egy épületen belül) kiépített rendszer, és az adott fix helyen belül mozgó eszközöknek képesek vagyunk az abszolút pozícióját meghatározni. A tagek egyenként kommunikálnak az anchorokkal és ezen üzenetváltások segítségével tudjuk meghatározni a tagek helyzetét. A következő fejezetek csupán az abszolút pozíciómeghatározással foglalkoznak.

Tovább csoportosíthatjuk a módszereket aszerint, hogy a vett jel erőssége (RSSI), időbeli mérések vagy a vett jel beesési szöge alapján számolunk pozíciót. Az RSSI alapján történő távolságbecslés azon alapul, hogy a kibocsátott rádiójel erőssége a távolság növekedésével monoton csökken. Ha ismerjük a kibocsátott jel erősségét, az adott közegben a rádiójelek csillapításának karakterisztikáját és mérjük a vett jel erősségét, akkor megbecsülhető az adó és vevő közötti távolság. Bizonyos feltételek mellett megfelelően működik ez a módszer, de nagyon pontos információval kell rendelkeznünk a jelterjedésről, ill. a méréseinknek is megbízhatóknak kell lenniük ahhoz, hogy a szükséges pontosságot elérjük.

Az Angle of Arrival (beesési szög) módszer a vett jel irányából számítja ki az adó eszköz pozícióját. Annak érdekében, hogy mérni tudjuk egy jel beesési szögét, ahhoz antennarendszert szükséges telepítenünk. Az antennarendszer több elemi antennából áll, amelyek szorosan egymás mellett helyezkednek el. Ha mérjük a vett jel beérkezésének idejét minden elemi antennán, akkor meghatározhatjuk, hogy melyik irányból érkezett a jel. Az időkülönbség mérést azonban nem tudjuk elég pontosan elvégezni a túl közeli antennák és a rádiójel túl gyors terjedése miatt. Ugyanakkor a vett jelek közötti fázis különbséget könnyen lehet mérni például analóg fázisdetektorral. A fáziskülönbség és a frekvencia ismeretében meg tudjuk határozni az időkülönbséget a két antenna között és így a beesési szöget is. Ezt több antennarendszernél elvégezve kiszámíthatjuk az irányvonalak metszéspontját, amely megadja az adó eszköz helyzetét.

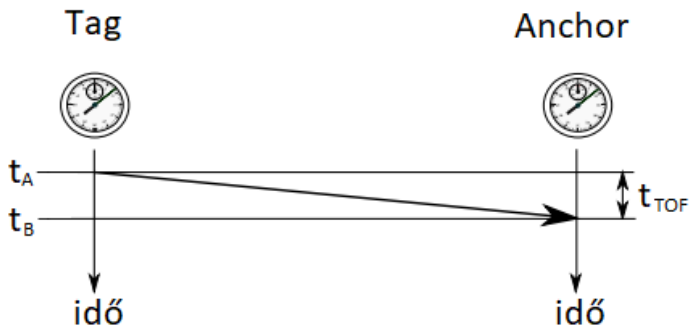
Időmérés alapján is lehetséges távolságot és ezáltal pozíciót becsülni. A továbbiakban ilyen módszerek kerülnek részletes bemutatásra.

2.1. ToF (Time of Flight)

A Time of Flight egy időmérésen alapuló távolságbecslés, amelynél a kisugárzott jel terjedési idejét mérjük két eszköz között. Ismerve a jel terjedési sebességét, meghatározható az adó és a vevő közötti távolság. A ToF mérésekben a tag és az anchorok közötti üzenetek elküldésének és fogadásának idejét kell feljegyeznünk (időbélyegekkel) és ezek segítségével meghatározható az elküldött rádióhullám terjedés ideje. Többféle üzenetváltási séma létezik, amelyek alkalmazhatósága feltételekhez kötött, illetve az előnyök és hátrányok mérlegelésével kell egy megfelelőt választanunk.

One Way Ranging (OWR), azaz egy üzenetváltásos távolságmérés

Az üzenetváltások szempontjából legegyszerűbb séma amikor a tag 1 db üzenetet küld az anchornak. Ha a tag t_A időpontban küldte el az üzenetet és az anchor t_B időpontban vette azt, akkor a terjedési idő $t_{TOF} = t_B - t_A$.

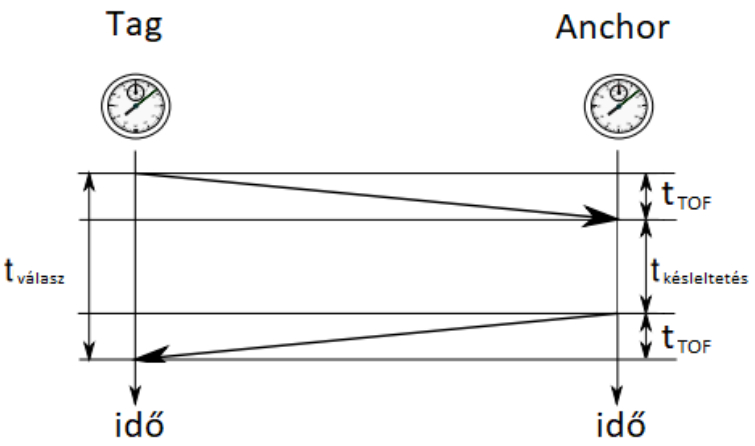


2.2. ábra. *Egy üzenetváltásos távolságmérés*

Ennek a sémának az előnye, hogy nagyon gyorsan juthatunk új méréshez, így a frissítési frekvencia ennél a módszernél a legnagyobb. Ugyanakkor a legnagyobb hátránya az, hogy nagyon pontos szinkronizációt igényel a tag és az anchor között. Erre léteznek megoldások, például az IEEE 1588-2008 szabvány, amellyel akár μs alatti pontosság is elérhető. Az implementálása viszont ezeknek a protokolloknak felesleges energiát emészthet fel, ugyanis - ahogy ezt a következő alfejezetben látni fogjuk - az óraszinkronizációtól való függőség egy plusz üzenettel feloldható.

Two Way Ranging (TWR), azaz két üzenetváltásos távolságmérés

Az eszközök közötti idő szinkronizációt hivatott kiküszöbölni a two way ranging. Az üzenetváltási sémát a 2.3. ábra szemlélteti.



2.3. ábra. *Két üzenetváltásos távolságmérés*

Az eszközök nincsenek szinkronizálva, ezért az üzenetek elküldésének és fogadásának idejét az eszközöknek el kell tárolniuk. A tag tudja, hogy mikor küldte el az első üzenetet és feljegyzi, hogy mikor fogadta a választ. Az anchor az első üzenet fogadásának idejét jegyzi fel, illetve, hogy mikor küldte el a válaszüzenetet. Ha az anchor az első üzenet fogadásától számítva $t_{\text{késleltetés}}$ idő múlva válaszolt és ezt a tag az első üzenet kiküldésétől számítva $t_{\text{válasz}}$ idő alatt kapta meg, akkor a terjedési idő a

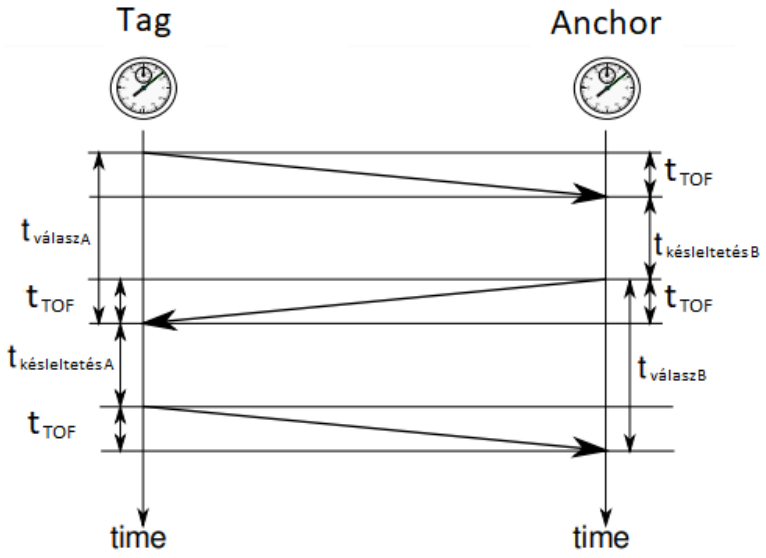
$$t_{TOF} = \frac{t_{\text{válasz}} - t_{\text{késleltetés}}}{2} \tag{2.1}$$

képlettel számolható. A módszer előnye, hogy nem szükséges szinkronizálni az eszközöket. Hátránya azonban, hogy ha az eszközök időmérése pontatlan[1], akkor a $t_{\text{késleltetés}}$ idő növekedésével lineárisan nő a távolságmérés hibája. Ez a jelenség az eszközök közötti frekvencia különbségből adódik. Részletesebben a 3. fejezetben mutatom majd be ezt a fajta hibát.

[1] Az időmérésben jelentkező hiba a chipben található kvarc kristály pontatlanságából adódhat.

Symmetrical Double-Sided Two Way Ranging (SDS-TWR)

Az SDS-TWR egy három üzenetből álló megoldás, amely a hagyományos TWR módszer frekvencia különbségből adódó hibáját csökkenti. Fontos megjegyezni, hogy ez a módszer nem küszöböli ki teljesen ezt a hibát, de jelentős mértékben csökkenti azt.



2.4. ábra. *Három üzenetváltásos távolságmérés*

A három üzenet tulajdonképpen 2 db TWR üzenetváltás összeolvadása, ahol az első TWR második üzenete ugyanaz, mint a második TWR első üzenete. A frekvencia különbségből adódó hibát azáltal csökkenti, hogy amíg a hagyományos TWR-nél csak az egyik, addig az SDS-TWR-nél mindkét eszköz hibája befolyásolja a távolságmérés hibáját. Ezáltal egymás hibáját fogják kompenzálni. A jelenség magyarázatától most eltekintek, részletes leírás található [6]-ben. A terjedési idő a következő képlettel számolható:

$$t_{TOF} = \frac{t_{\text{válaszA}} \times t_{\text{válaszB}} - t_{\text{késleltetésA}} \times t_{\text{késleltetésB}}}{t_{\text{válaszA}} + t_{\text{válaszB}} + t_{\text{késleltetésA}} + t_{\text{késleltetésB}}} \tag{2.2}$$

Az eljárás egyik hátránya, hogy sokkal több időt vesz igénybe, mint az egy üzenetváltásos módszer, viszont sokkal pontosabb eredményt kapunk bármelyik eddigi módszerhez képest.

Pozíció kiszámítása

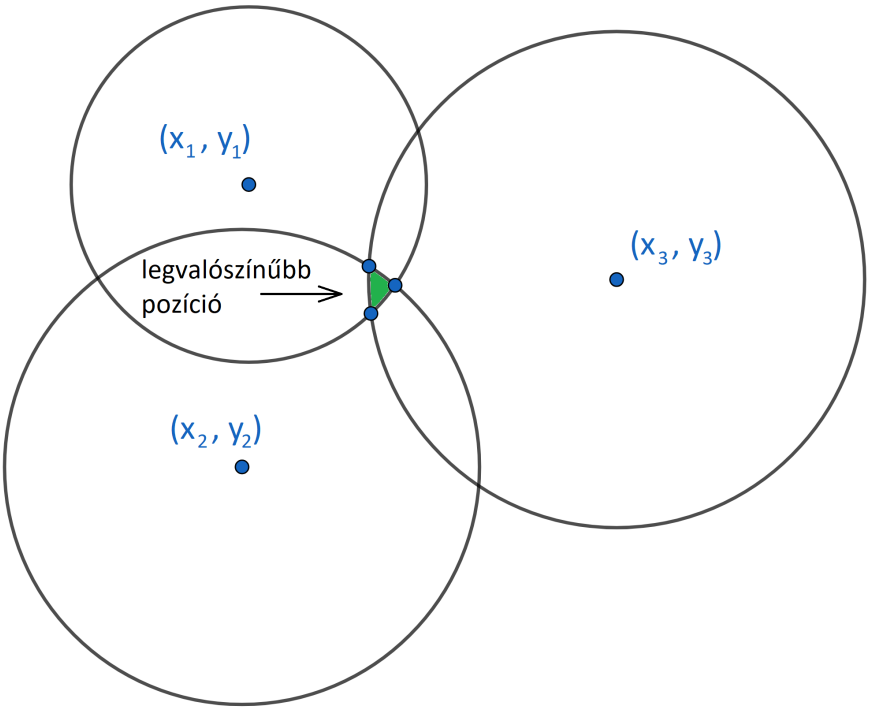
A pozíció számítását először 2 dimenzióban mutatom be, mert sokkal egyszerűbb, mint 3 dimenzióban. Tudjuk, hogy a térben 3 pont feszít ki egy síkot, így minimum három eszközre van szükségünk a pozícionálásra (valamint egy negyedikre, amelynek helyzetét akarjuk kiszámolni). Egy anchortól adott távolságra elhelyezkedő tag az anchor körül egy körön helyezkedhet el, melynek sugara a kettő közötti távolság. Ez végtelen sok pontot jelent. Ha még egy anchortól mérjük a távolságot, akkor ezen anchor körül is egy körön helyezkednek el a tag lehetséges pozíciói. Ennek a két körnek legfeljebb két metszéspontja lehet. Ha nincsen metszéspont, az azt jelenti, hogy a távolságmérésünk nem volt pontos, javítani kell a pontosságon, így ezt az esetet figyelmen kívül hagyhatjuk. Egy metszéspont akkor fordul elő, ha az egyik anchortól a, a másiktól b távolságra vagyunk és a két anchor közötti távolság $a + b$. Ez nagyon ritkán fordul elő a méréseket terhelő zaj és a lebegőpontos számábrázolás miatt. Az esetek túlnyomó többségében két metszéspontot kapunk a két körből. Ez még mindig nem ad egyértelmű megoldást, így szükségünk van egy harmadik anchorra is, amely egyértelműsíti, hogy a két metszéspont közül melyik a tag valódi helyzete. Ezzel beláttuk, hogy 3 anchor szükséges és elégséges is a síkban történő pozíció kiszámolásához.

A fentiekből sejthető, hogy térben történő pozícionáláshoz minimum 4 anchor kell. Háromdimenziós térben egy ponttól adott távolságra levő pontok egy gömbön helyezkednek el. Ha két ilyen gömböt veszünk fel két anchor köré, a tőlük a tagig mért távolságnak megfelelő sugárral, akkor azok metszete legtöbb esetben egy kör lesz. Ha viszont három ilyen gömböt veszünk fel, azaz három anchorhoz képest mérjük a tag távolságát, akkor a három gömb két pontban metszi egymást. Ebből láthatjuk, hogy legalább 4 anchorra van szükségünk, hogy egyértelmű megoldást kapjunk. Most, hogy ismerjük hány anchor szükséges az egyértelmű megoldhatósághoz, néhány lehetséges eljárást mutatok be a pozíció kiszámítására.

Statisztikai módszeren alapuló modellek Az ilyen fajta modellek bizonyos statisztikai feltételezéssel rendelkeznek a mérésekről és a tag helyzetéről, valamint egy valószínűségi modellel, amely összekapcsolja a kettőt. A legtöbb ilyen modell a legnagyobb valószínűség (ML - Maximum Likelihood) elvét alkalmazza az aszimptotikus viselkedése és az ML becslők (MLE - ML Estimators) magas hatékonysága miatt. A mérési hiba eloszlását általában Gauss-eloszlásúnak feltételezik. Ahhoz, hogy megoldjuk az ilyen erősen nemlineáris modelleket, lineáris közelítéseket és iteratív numerikus módszereket kell alkalmaznunk. Ennek következtében szükséges, hogy legyen egyfajta *a priori* becslésünk a megoldásra, hogy gyorsítani tudjuk a

megoldás folyamatát. A statisztikai megközelítést alkalmazó algoritmusokat gyakran osztályozzák a nyitott formájú algoritmusokhoz, mert nem írhatók fel zárt alakban. Ugyanakkor, ha megfelelő modellünk van a mérésekről, akkor ez a módszer optimális becslést ad a pozícióról. [10]

Algebrai módszeren alapuló modellek Algebrai módszeren olyan megoldást értünk, ahol vagy a távolságokra felírt (Pitagoraszi-) egyenleteket manipulálva, vagy különböző geometriai tulajdonságokat kihasználva határozzuk meg a kérdéses pozíciót. Ebbe a kategóriába sorolható a sokak által ismert háromszögelési eljárás is [7]. A legtöbb esetben a pozíció kiszámítása a körök metszéspontjainak megkeresésével kezdődik [13]. Ezek után előállíthatjuk a keresett pozíciót például úgy, hogy a három legközelebb eső metszéspont koordinátáit kiátlagoljuk.



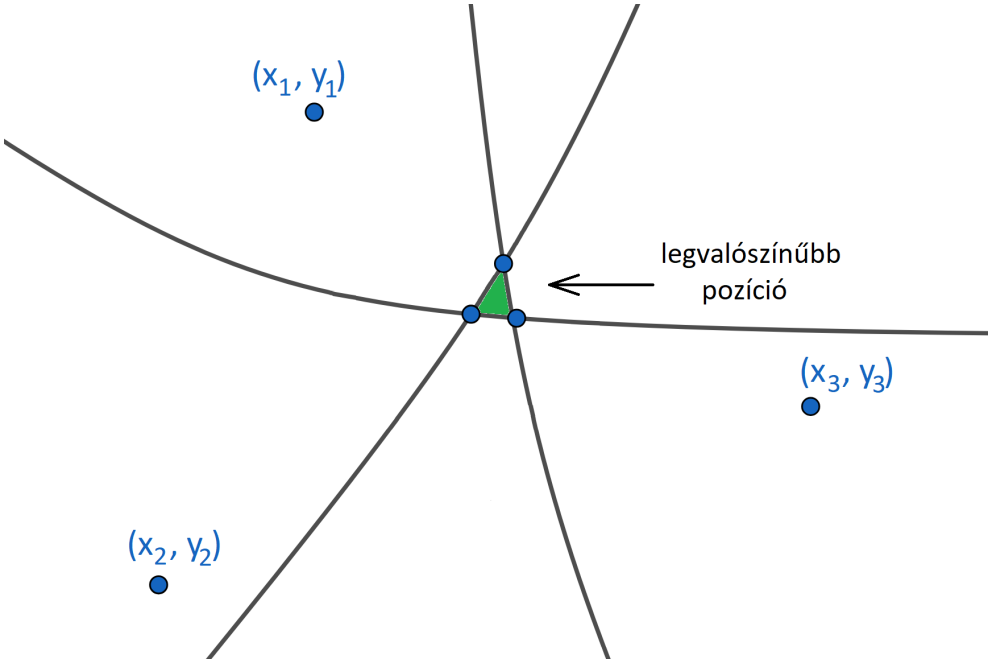
2.5. ábra. *Háromszögelés síkban*

Numerikus módszeren alapuló modellek Ezek a modellek közvetlen keresik a megoldást a tag helyzetére egy nemlineáris optimalizálás segítségével. Háromdimenziós esetben három változó (x, y, z) az ismeretlen, amelyek a tag pozícióját jelentik, és 12+4 ismert paraméterünk van, amelyek a 4 anchor helyzetét adják meg és a tőlük a tagig mért távolságot. Egy $c(x, y, z)$ költségfüggvényt állítunk fel, amelyben szerepelnek az anchorok és a mért távolságok is. A költségfüggvény annál kisebb értéket ad, minél kisebb az eltérés a tag (x, y, z) és a valódi pozíciója között. A módszer lényege, hogy olyan (x, y, z) értékeket keressünk, amelyek mellett ez a költségfüggvény minimális. Amikor megtaláltuk ezt az értéket, akkor a legkisebb a különbség a becsült tag helyzetének az anchoroktól vett távolsága és a mért távolságok között.

A költségfüggvényt célszerű úgy összeállítani, hogy minden esetben pozitív számot kapjunk. Gyakran ezt úgy érik el, hogy négyzetre emelik az egész kifejezést. Emiatt a legtöbb esetben a változók között a legnagyobb előforduló hatvány az valamely páros szám. Ennek következtében a költségfüggvénynek lesz valódi globális minimuma, azonban, ha a fokszáma nagyobb vagy egyenlő 4-nél, akkor több lokális minimum értéke is lehet. Az optimalizáló algoritmusnak ezért érdemes megfelelő határokat szabni a változók értékeire vonatkozóan, hogy biztosan jó megoldást kapjunk a tag helyzetére. Egy ilyen értelmes határ például az előző pozíció köré írt gömb, amelynek sugara akkora, hogy a tag a maximális sebességgel se érje el a gömb határát. Másik megfelelő határt kaphatunk az algebrai megoldás felhasználásával. Mivel a háromszögelést könnyen el tudjuk végezni síkban, ezért három azonos magasságban levő anchort kiválasztva megkeressük a 2.5. ábrán látható zöld területet, amelyet a három metszéspont határoz meg. Ezekből az x és y koordinátákra megfelelő minimum és maximum értékeket kaphatunk, a z koordinátára pedig a legalacsonyabban és legmagasabban található anchor helyzete alapján adhatunk határokat. A megoldás halmazának szűkítése nem csak a valódi megoldás megtalálásában segít nekünk, hanem a futási időt is jelentősen csökkenti. [10]

2.2. TDoA (Time Difference of Arrival)

A TDoA egy olyan módszer, amelyben a pozícionáláshoz szükséges három anchort szintén ismert helyekre telepítjük és ezeket időben nagyon pontosan szinkronizáljuk. A távolságmérés úgy történik, hogy a tag elküld egy üzenetet és az anchorok feljegyzik a vétel időpontját. A tag helyzetétől függően az anchorok ezt más időpillanatban érzékelik. Ha kiszámoljuk, hogy két anchor mekkora időkülönbséggel kapta meg az üzenetet, akkor a tag pozíciója egy, a két anchor közt áthaladó hiperbolán lesz. Ebben az esetben is szükséges ezért még egy anchor, amelynek segítségével további két hiperbolát tudunk felvenni. A három hiperbola közös metszéspontja fogja meghatározni az eszköz helyzetét.

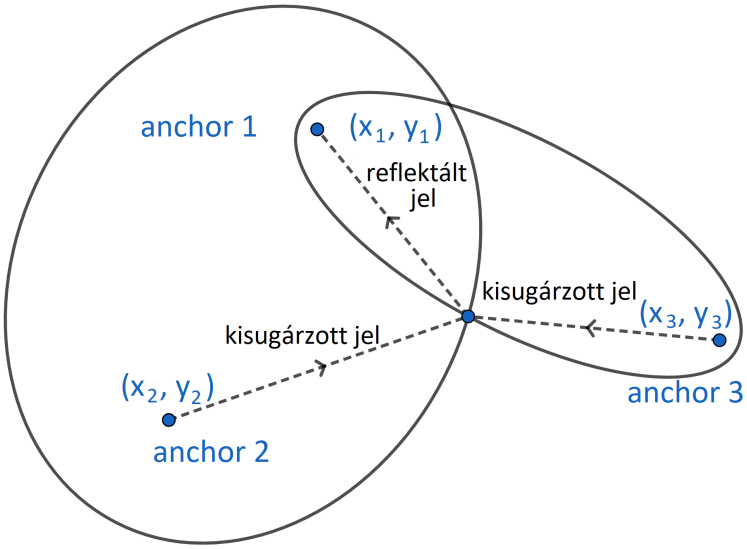


2.6. ábra. *Időkülönbségen alapuló pozícionálás*

Ezzel a módszerrel is nagy pontosság érhető el, viszont a használhatóságának feltétele, hogy az anchorok legalább 100 ps-os pontossággal legyenek szinkronizálva, ugyanis a maximális hiba ekkor is már kb. 3 cm.

2.3. TSoA (Time Sum of Arrival)

A TSoA egy olyan megközelítést alkalmaz, amelyben egy anchortól elküldött és a tagről visszavert jelet érzékeljük egy harmadik eszköz segítségével. Ha mérjük a jel elküldésétől a harmadik eszköz érzékeléséig eltelt időt, akkor ebből azt a távolságot tudjuk kiszámolni, amely a tag és a két anchor közötti távolságok összege. A tag tehát az ismert helyzetű anchorok körüli ellipszisen helyezkedhet el. A módszer legfőbb előnye, hogy a tag eszközök passzívak lehetnek és így akár az energiafelvételük is megszüntethető. [12]



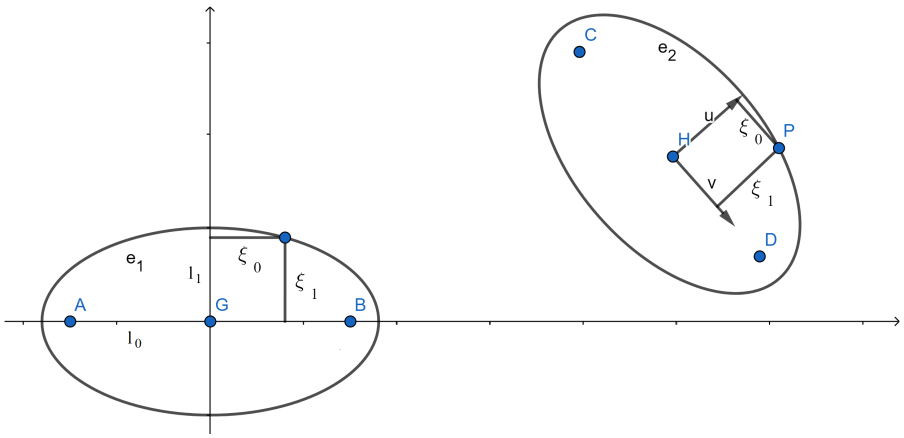
2.7. ábra. *Reflektáláson alapuló pozícionálás*

Ennél a módszernél mutatkozik meg az UWB-nek az az előnye, hogy képesek vagyunk elkülöníteni a reflexiómentes jelet a reflektáltaktól. A 2.7. ábrán látjuk a távolságmérés folyamatát. A 2-es számú anchor kisugároz egy jelet, amely egyenes úton is megérkezik az 1-es anchorhoz, azonban a tagről visszaverődve is elér hozzá. Ugyanígy a 3-as anchor is kisugároz egy jelet, amely a tagről visszaverődve elér az 1-es anchorhoz. Ha ismerjük a jel küldésének és fogadásának idejét, akkor meghatározható az ellipszis fókuszpontjaitól (anchorok) mért távolságok összege, amely ellipszis esetén egy állandó érték és megegyezik a fél nagytengely kétszeresével. Ismerjük továbbá a két fókuszpont közötti távolságot, így fel tudjuk írni a 2 ellipszis egyenletét, amely x és y változókban is egy másodfokú egyenlet:

$$1 = M_{00}(x - x_0)^2 + (M_{01} + M_{10})(x - x_0)(y - y_0) + M_{11}(y - y_0)^2, \tag{2.3}$$

ahol M_{00}, M_{01}, M_{10} és M_{11} az ellipszis orientációját és tengelyeinek nagyságát meghatározó konstansok, míg x_0 és y_0 az ellipszis középpontjának koordinátái. A kons-

tansok meghatározásához tekintsük a következő elrendezést, e_1 ellipszist transzformáljuk e_2-be.



2.8. ábra. *Ellipszis transzformálása tetszőleges helyzetbe*

Legyen (ξ_0, ξ_1) egy tetszőleges pont az e_1-n ellipszisen. Ekkor

$$1 = \left(\frac{\xi_0}{l_0}\right)^2 + \left(\frac{\xi_1}{l_1}\right)^2 \tag{2.4}$$

egyenlőség áll fent, ahol l_0 és l_1 rendre az ellipszis kis és nagy féltengelye. e_2 ellipszisen is meghatározható a P pontra ez az egyenlet:

$$1 = \left(\frac{u \cdot (P - H)}{l_0}\right)^2 + \left(\frac{v \cdot (P - H)}{l_1}\right)^2 \tag{2.5}$$

Rövid rendezés után azt kapjuk, hogy

$$1 = (P - H)^T M (P - H), \tag{2.6}$$

ahol

$$M = \begin{bmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{bmatrix} = \begin{bmatrix} \frac{u_0^2}{l_0^2} + \frac{v_0^2}{l_1^2} & \frac{u_0 u_1}{l_0^2} + \frac{v_0 v_1}{l_1^2} \\ \frac{u_0 u_1}{l_0^2} + \frac{v_0 v_1}{l_1^2} & \frac{u_1^2}{l_0^2} + \frac{v_1^2}{l_1^2} \end{bmatrix} \tag{2.7}$$

Ezen konstansok segítségével felírható két 2.3 alakú egyenlet a 2.7. ábrán látható két ellipszisre. Az ebből kapott egyenletrendszer megoldását a 2.1. fejezetben ismertetett numerikus módszerrel kereshetjük meg legkönnyebben. A 2.7. ábrán látható, hogy ekkor még mindig két megoldás lehetséges, így szükségünk van egy harmadik ellipszisre is. A megoldást az a metszéspont adja, amely mind a három ellipszisen rajta van. [1]

3. fejezet

Pontosságot befolyásoló tényezők

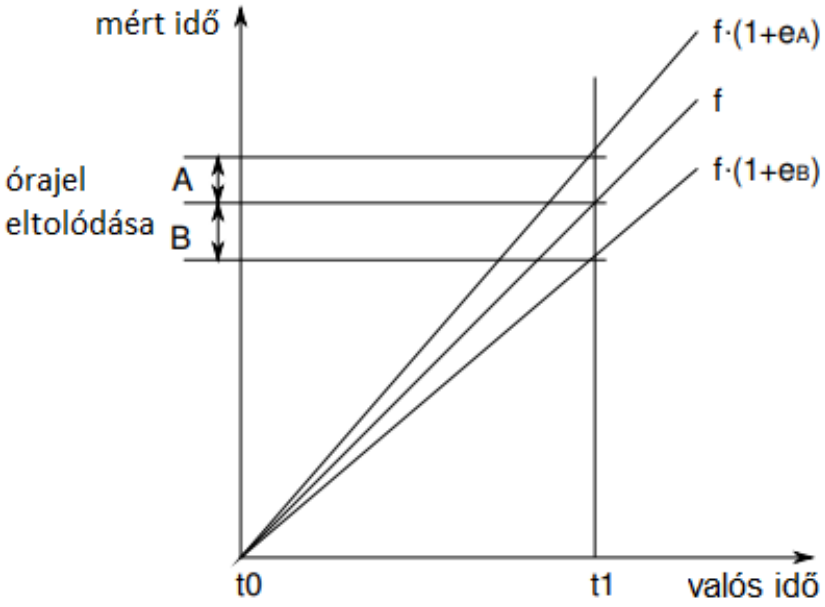
Ebben a fejezetben olyan jelenségeket mutatok be, amelyek nagy hatással vannak a távolságmérés pontosságára. Emiatt az ilyen típusú hibákat vagy ki kell küszöbölni, hogy ne lépjenek fel, vagy kompenzálni kell valamilyen hardveres vagy szoftveres megoldással. Látni fogjuk, hogy ezek a hibák nem szüntethetők meg teljes mértékben, így a kész rendszer hibával terhelt marad. A maradék hiba egy Gauss-eloszlású zaj lesz 0 várhatóértékkel és kb. 3-4 cm szórással. Ilyen nagy mérési hiba ellenére is egy folytonos távolságbecslést kaphatunk egyszerű szűrő algoritmusokkal. Ennek bemutatására a 4. fejezetben kerül sor, jelen fejezetben csak a hibák forrásával és azok minél nagyobb mértékben történő kiküszöbölésével foglalkozok.

3.1. Órajel frekvencia eltolódás hatása (crystal drift)

A továbbiakban feltételezzük, hogy nem OWR, azaz egy üzenetváltásos távolságmérést végzünk, hanem az óra szinkronizáció szükségességét megszüntetve TWR vagy SDS-TWR módszert használunk. Ezekben a megoldásokban két külön eszközön függetlenül mérjük az időt például egy kristály oszcillátorral és egy mikrokontrollerrel. A két eszköz között frekvencia ofszet, egy eszközön pedig frekvencia csúszás léphet fel.
Ha a nominális f frekvenciához képest az A eszköz e_A-val gyorsabban és B eszköz e_B-vel lassabban számol, akkor t_0 idő elmúlásával

$$\Delta t = t_0 \left(\frac{1}{1+e_A} - \frac{1}{1+e_B} \right) \tag{3.1}$$

időkülönbséget mutat a két eszköz órája.



3.1. ábra. *Az időmérés eltolódása két eszköz között frekvencia különbség hatására*

TWR frekvencia különbséggel

Ha a két eszköz között TWR kommunikációval határozzuk meg a terjedési időt, akkor a

$$t_{TOF} = \frac{t_{\text{válasz}} - t_{\text{késleltetés}}}{2} \tag{3.2}$$

képlettel számolhatunk, ahol $t_{\text{késleltetés}}$ az idő, amíg az anchor válaszol. Ha az eszközök frekvenciahibája továbbra is e_A és e_B, akkor a helyes t_{TOF} helyett a

$$\hat{t}_{TOF} = \frac{1}{2}(t_{\text{válasz}}(1 + e_A) - t_{\text{késleltetés}}(1 + e_B)) \tag{3.3}$$

értéket kapjuk. Ebből kivonva a helyes t_{TOF} értéket megkaphatjuk az időmérés hibáját:

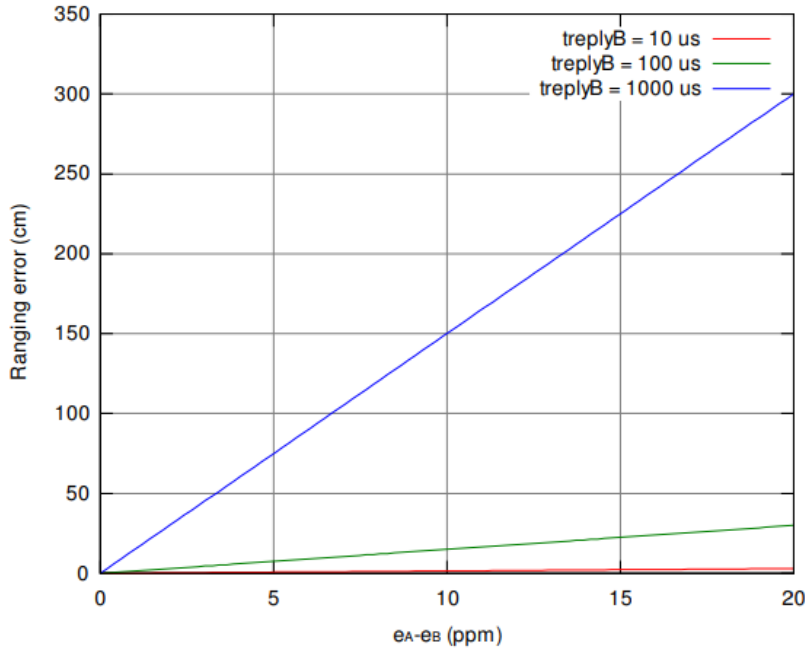
$$\hat{t}_{TOF} - t_{TOF} = \frac{1}{2}(t_{\text{válasz}} e_A - t_{\text{késleltetés}} e_B) \tag{3.4}$$

$$= \frac{1}{2}((t_{\text{késleltetés}} + 2t_{TOF})e_A - t_{\text{késleltetés}} e_B) \tag{3.5}$$

$$\cong \frac{1}{2} t_{\text{késleltetés}}(e_A - e_B) \tag{3.6}$$

Vagyis, az időmérés hibája a következő képlettel számítható:

$$t_{hiba} = \frac{1}{2} t_{\text{késleltetés}}(e_A - e_B) \tag{3.7}$$



3.2. ábra. *TWR időmérés hibája a frekvencia különbség hatására távolságra átszámolva ($t_{replyB} = t_{késleltetés}$) [9]*

Látható, hogy már 100 μs-os válaszidőnél és 10 ppm-es frekvencia különbség hatására is kb. 15 cm-es hibát kapunk a távolságmérésben. Ez az érték jóval meghaladja azt az értéket, amilyen mértékű pontosságot el szeretnénk érni.

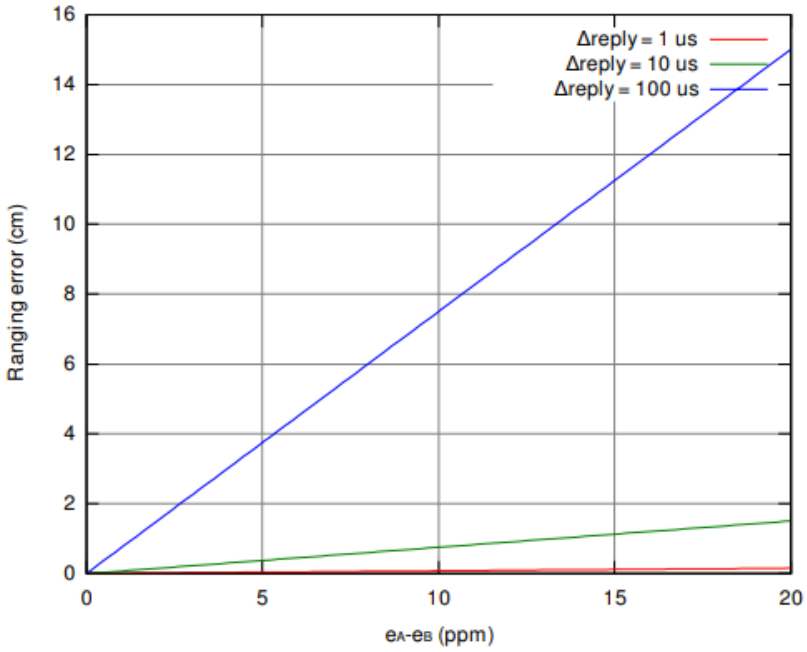
SDS-TWR frekvencia különbséggel

Az előző fejezetben már tettem rá utalást, hogy az SDS-TWR a hagyományos TWR módszer nagy frekvencia különbségre való érzékenységét hivatott kiküszöbölni. A 3.3. ábra jól szemlélteti, hogy ez tényleg így van.
Jiang és Leung [6] részletesen levezette, hogy ebben az esetben hogyan számítható a terjedési idő mérésének hibája. Itt csupán a végeredményt adom meg:

$$t_{\text{hiba}} = \frac{1}{2}\Delta t_{\text{válasz}}(e_A - e_B), \tag{3.8}$$

ahol $\Delta t_{\text{válasz}}$ jelöli a két eszköz válaszküldési ideje közötti különbséget. TWR esetében 100 μs-os válaszidő és 10 ppm frekvencia különbség hatására 15 cm-es hibát kaptunk a távolságra.

SDS-TWR esetében a válaszidők közti különbséggel lesz arányos a hiba, vagyis, ha



3.3. ábra. *SDS-TWR időmérés hibája a frekvencia különbség hatására távolságra átszámolva ($t_{replyB} = t_{késleltetés}$) [9]*

a 100 μs-os válaszidők egymáshoz képest 1 μs-mal térnek el és 10 ppm frekvencia különbség van a két eszköz között, akkor a távolságmérés hibája kb. 0.75 cm. Ez pedig jelentős javulás a TWR esethez képest.

A frekvencia különbség kompenzálása

A frekvencia különbség két eszköz között a bennük található kristály oszcillátor pontatlanságából adódik. Minden kristály oszcillátornak megvan adva a pontossági osztálya, amelyen belül fix frekvenciával működnek. Ha ez a két frekvencia különbözik, akkor az a rendszer pontosságában fogja kifejteni a hatását. Ezért szükséges csökkenteni a frekvencia különbséget a két eszköz között. Ennek az egyik módja a TCXO-k használata. A TCXO (Temperature Compensated Crystal Oscillator - hőmérséklet kompenzált kristály oszcillátor) általában tartalmaz egy hagyományos oszcillátort és egy kisebb áramkört, amely szabályozza a kimeneti jel frekvenciáját. TCXO használatával jelentős pontosság növekedés érhető el, azonban jelentősen megnövelik a rendszer költségét és az eszközök energiafogyasztását.

Egy másik lehetőség a kristály oszcillátor trimmelése. A trimmelés során a kristállyal párhuzamosan kapcsolunk kondenzátorokat. Ha a kristályt egy rezgőkörnek tekintjük, akkor egy vele párhuzamosan kapcsolt reaktancia - legyen az kapacitás vagy induktivitás - a rezgőkör rezonancia frekvenciáját fogja megváltoztatni. A változás mértéke a reaktancia nagyságával van összefüggésben. Sok esetben azonban

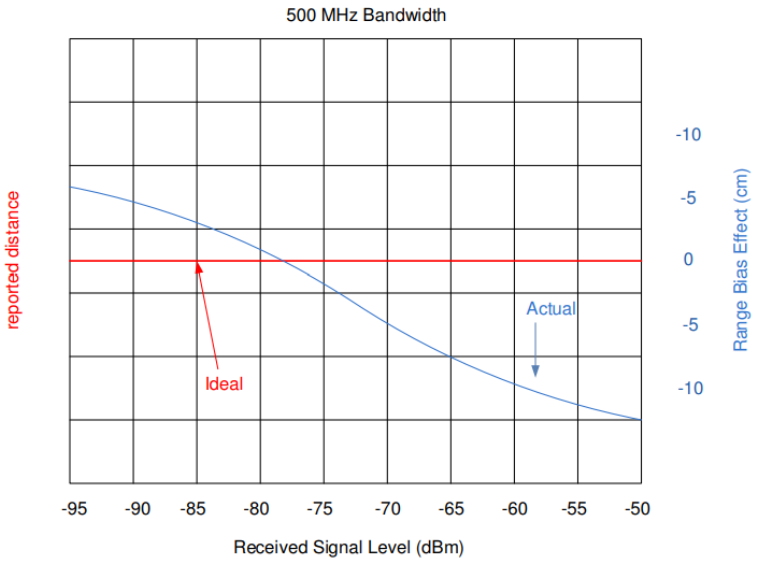
nem férünk hozzá közvetlenül egy adó-vevő chip belső kristályához. Ilyenkor vagy nem oldható meg a trimmelés, vagy a gyártó programozhatóan teszi elérhetővé a kristály trimmelését. Ez annyit jelent, hogy a gyártó által megadott módon felprogramozott chip, általában az egyik belső regiszterébe írt érték alapján, belső kondenzátorokat kapcsol a kristállyal párhuzamosan. A regiszterbe írt érték alapján különböző nagyságú kapacitások kerülnek az áramkörbe, így szabályozva a kristály frekvenciáját.

3.2. Range bias

Ideális esetben nincs kapcsolat a vett jel erőssége (Received Signal Level - RSL) és a mért távolság között, vagyis a távolságmérés hibája konstans 0 a jelerősség függvényében.

A jelenség leírása és magyarázata

A 3.4. ábrán a piros vonal jelzi az ideális esetet. A gyakorlatban azonban a jelerősség változásával a mért távolság is változik, ez a kék görbével van ábrázolva. A görbe jellege alapján is sejthető ennek a magyarázata: ha gyengébb a jel (<-80 dBm), akkor csak később érzékeljük az érkező üzenetet, tehát nagyobb értéket kapunk a terjedési időre és emiatt a mért távolság is nagyobb lesz (negatív *range bias*). Erősebb jelnél előbb érzékeljük a jelet és ezért kisebb távolságot kapunk, ami pozitív *range biast* jelent.



3.4. ábra. *A vett jel erősségének hatása a mért távolság hibájára [9]*

Range bias jelenség kiküszöbölése

A legegyszerűbb módja, hogy megszüntessük a vett jel erőssége miatt fellépő hibát az, ha szoftveresen kompenzáljuk a mért távolságot. Közvetlen rálátás[1] esetén fix távolsághoz fix vett jelerősség tartozik. Így megfelelő módon tudunk kompenzálni, ha ismerjük a saját eszközünk *range bias* görbéjét. Gyakori megoldás az ún. *look-up table* használata, amely az egyszerű implementáláson kívül gyors futási időt is eredményez.

3.1. táblázat. *Különböző jelerősségekhez tartozó range bias értékek*

RSL (dBm)	PRF 64 MHz 500 MHz (cm)
-61	-11,0
-65	-10,0
-69	-8,2
-73	-5,1
-77	0,0
-81	3,5
-85	4,9
-89	7,1
-93	8,1

A vett jel erősségét össze kell hasonlítani az RSL értékekkel és ahol a legnagyobb egyezést találjuk, ahhoz a sorhoz tartozó korrekciós értéket választjuk. A táblázatot természetesen tetszőleges tartományra kibővíthetjük, illetve az RSL értékek közti felbontást is szabadon megválaszthatjuk. Ha nincs kellően nagy felbontásunk, akkor két szomszédos RSL közötti értékhez tartozó korrekciót lineáris interpolációval is előállíthatjuk.

[1] Olyan esetben, amikor az adó és vevő között meghúzható egy egyenes, ami nem ütközik tömör anyagba, azaz a két eszköz között csak levegő van.

3.3. Közvetlen rálátás, illetve annak hiánya

Közvetlen rálátás (Line Of Sight - LOS) esetében a jel levegőben, ismert sebességgel terjed. Mivel ez a sebesség konstans, így megbízható eredményt kapunk a távolságra az időmérés alapján. Előfordulhat azonban olyan eset, amikor valamilyen nagy kiterjedésű tömör tárgy kerül az adó és vevő közé (Non Line Of Sight - NLOS). Ilyenkor az egyenes úton haladó jel lelassul a tömör anyagban, több időbe telik mire elér az adótól a vevőig, és ezáltal nagyobb távolságot mérünk. Az ilyen típusú hiba sajnos kivédhetetlen a környezet változásainak ismerete nélkül. Ha pedig kompenzálni szeretnénk az NLOS okozta hibát, akkor ismernünk kell a tömör anyagban a terjedési sebességet, és a távolságot, amelyet ebben az anyagban megtett. Ezek ismeretében a következő egyenlet írható fel:

$$t_{\text{terjedés}} = \frac{d-w}{c} + \frac{w}{c'}, \tag{3.9}$$

ahol d a két eszköz közötti távolság, w az árnyékoló anyag vastagsága, c a fénysebesség, c' pedig a tárgy anyagában a terjedési sebesség. A fenti egyenletben $t_{\text{terjedés}}$-t mérjük. Ismerve w-t és c'-t, meghatározható a két eszköz közötti távolság.

3.4. További pontosságot befolyásoló tényezők

Fontos paraméter az antenna késleltetés is. Ez az érték megadja, hogy a jel antennához való megérkezésétől mennyi idő telik el az időbélyeg generálásáig. Ez néhány 100 ps nagyságú érték, amely az analóg jelterjedési időt és a szoftveres feldolgozási időt foglalja magában. Ezt elegendő kb. 10 ps-os pontossággal beállítani, ugyanis ekkor a hiba már csupán néhány mm.

A legtöbb esetben a környezet hőmérséklete és a tápfeszültség is befolyásolja a mérési eredményeinket. A DecaWave által gyártott DWM1000 chipnek 2,15 mm/°C-os hőmérséklet és 5,35 cm/$V_{\text{táp}}$-os tápfeszültség függése van, amelyet szoftveres úton tudunk kompenzálni.

4. fejezet

Kálmán-szűrő

A Kálmán-szűrő egy diszkrét idejű szűrő, amely rendszerek állapotbecslését teszi lehetővé. A szűrőt Kálmán Rudolf Emil egy 1960-ban publikált cikkében mutatta be. Azóta számos területen kezdték el alkalmazni és használják a mai napig is, ugyanis kis memória- és számítási kapacitás igénye van, így akár egy beágyazott rendszeren is könnyedén futtatható. A gyors elterjedésének oka többek között az, hogy zajos mérésekből is nagyon pontos becslést tud adni egy rendszer állapotára. Ebben a fejezetben bemutatom röviden az elméletét, működési elvét és hogy hogyan alkalmaztam a távolságmérés pontosítására.

4.1. Elmélet

A szűrő egy dinamikus rendszer állapotára ad optimális becslést zajjal terhelt mérésekből és a rendszer leírásából kiindulva. Lényeges megkötés, hogy az eredetileg publikált algoritmus, csak lineáris rendszerekre és normális eloszlású zajjal terhelt mérésekre használható jól. Azóta több algoritmus is született ezeknek a korlátoknak a leküzdésére, mint például a kierjesztett Kálmán-szűrő (EKF - Extended Kalman-filter), amely nemlineáris rendszereken is használható. Most azonban csak a lineáris rendszerek állapotbecslésével foglalkozunk, amelyek leírását egy lineáris differencia egyenlettel adhatjuk meg:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}. \tag{4.1}$$

Az állapotokra a

$$z_k = Hx_k + v_{k-1} \tag{4.2}$$

egyenlet alapján kapunk méréseket, ahol z_k a mért mennyiségek vektorba rendezve. Gyakran egy állapotot nem tudunk közvetlen mérni, ezért szükséges, hogy feltételezzünk egy H mátrixot, ami kapcsolatot teremt a mért értékek és a rendszer állapotai

között. w_k és v_k a rendszer és a mérési zaj, amelyek 0 várható értékű normális eloszlású, egymástól függetlennek tekinthető zajok, rendre Q és R kovariancia mátrixokkal. A Kálmán-szűrő első lépésként kiszámolja a differencia egyenlet szerinti $\hat{x}_k^-$ *a priori* becslését, majd a

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{4.3}$$

egyenlet alapján adja meg a rendszer állapotára vonatkozó végső becslését. A szűrő az $\hat{x}_k$, ún. *a posteriori* becslést úgy határozza meg, hogy közben az $e_k = x_k - \hat{x}_k$ hiba kovariancia mátrixát minimalizálja. Ez a minimalizálás egy egyszerű matematikai levezetéssel elvégezhető és ebből megkaphatjuk az optimális K_k értékét:

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R} \tag{4.4}$$

ahol P_k^- az *a priori* becslés hibájának kovariancia mátrixát jelöli.
Látható a 4.3 és 4.4 egyenleteken, hogy ha mérési zaj közelít a nullához, azaz

$$\lim_{R \to 0} K_k = \frac{1}{H}, \tag{4.5}$$

akkor a 4.3 egyenlet $\hat{x}_k = z_k$ egyenlőséggé redukálódik. Azt is mondhatjuk, hogy a szűrő algoritmus egyre jobban fog "bízni" a mérésekben és nem fogja figyelembe venni a differencia egyenlet becslését, mert nagyon pontos méréseink vannak. Továbbá, ha az *a priori* becslés hibájának kovariancia mátrixa közelíti meg a nullát, azaz

$$\lim_{P_k^- \to 0} K_k = 0, \tag{4.6}$$

akkor a 4.3-ból $\hat{x}_k = \hat{x}_k^-$ egyenlet lesz, vagyis a szűrő a méréseket fogja figyelmen kívül hagyni, ugyanis a modellünk ad nagyon pontos becslést a rendszer állapotáról. [17]

4.2. A szűrő működése

A Kálmán-szűrő algoritmusát két fő lépésre oszthatjuk fel. Először kiszámoljuk a rendszer modellje által adott becslést, majd egyfajta visszacsatolásként a méréseket használjuk fel a becslés pontosítására. Az első lépésnek, mivel az előző állapotból becsül a mostanira, vagyis az előző időlépésből következtet a jelenlegire, ezért idő frissítés a neve (*time update*), a második lépésben pedig a mérésekkel pontosítjuk a becslésünket, így ezt a részét az algoritmusnak mérési frissítésnek (*measurement update*) nevezzük. Összességében az algoritmus 5 egyenletből áll:

Kálmán-szűrő *time update* fázisa:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{4.7}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{4.8}$$

Kálmán-szűrő *measurement update* fázisa:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{4.9}$$

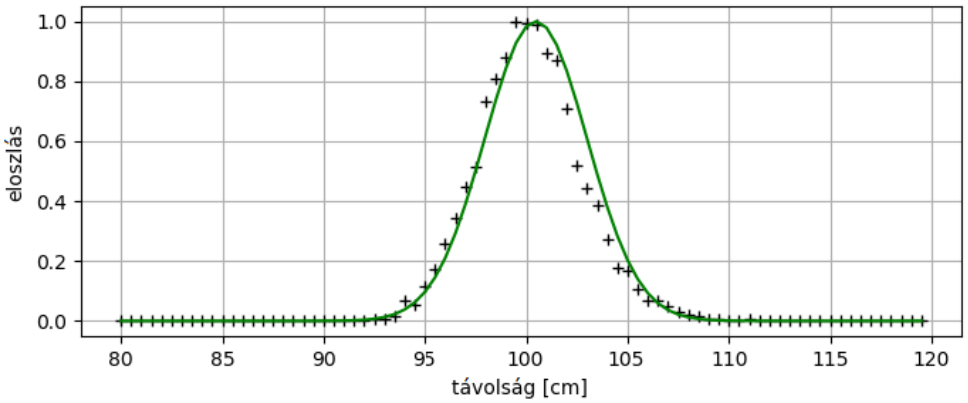
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{4.10}$$

$$P_k = (I - K_k H)P_k^- \tag{4.11}$$

Az egyenletekből látható, hogy nem szerepel a változók között $k-1$-nél kisebb indexű változó, így az algoritmusnak csupán $\hat{x}_k$, P_k és u_k változókat kell eltárolnia a következő iterációig. A műveletek között csupán összeadás, mátrix szorzás és egy inverz mátrix számolás szerepel, amelyek kis méretű mátrixoknál nem igényelnek nagy számítási kapacitást. Ezen előnyök miatt az algoritmus akár egy beágyazott mikrokontrolleren is futtatható.

4.3. A Kálmán-szűrő alkalmazása a távolságbecslés pontosítására

A mérnöki gyakorlatban sok sztochasztikus folyamatra mondjuk, hogy normális az eloszlása, ugyanis az ilyen eloszlással rendelkező folyamatokat könnyen tudjuk megfigyelni és szabályozni. Az esetek többségében ez a feltételezés valóban megállja a helyét, azonban sok esetben nem feltétlenül jogos a közelítés. Az UWB alapú távolságmérésnél az előbbi helyzet áll fenn, ugyanis ahogy a 4.1. ábrán látjuk a mérések jó közelítéssel normális eloszlásúak. A zöld görbe egy valódi Gauss-eloszlás, amely μ várható értéke és σ szórása a mérésekből lett kiszámolva.



4.1. ábra. *Konstans távolság méréseinek eloszlása (10000 mérésből számolva) és a ráillesztett Gauss-eloszlás sűrűségfüggvénye*

Mindegyik anchortól mért távolságot egy-egy Kálmán-szűrővel pontosítunk. Az állapotváltozó természetesen a távolság lesz, és esetünkben az x vektor egy skalárrá redukálódik. Mivel nincs ismeretünk a tag mozgásáról, ezért a modellben azt feltételezzük, hogy konstans távolságot becslünk, vagyis az A mátrix egy skalár lesz 1-es értékkel, míg a B vektor is egy skalár 0-ás értékkel. A továbbiakban látni fogjuk, hogy ez nem fog minket meggátolni abban, hogy változó távolságot is megfelelően tudjunk meghatározni. Továbbá mivel közvetlen mérjük az állapotváltozó, azaz a távolság értékét, ezért a H mátrixra is egy skalárt kapunk 1-es értékkel. Így a Kálmán-szűrőnk egyenletei a következők:

$$\hat{x}_k^- = \hat{x}_{k-1}$$

$$P_k^- = P_{k-1} + Q$$

$$K_k = P_k^-(P_k^- + R)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{x}_k^-)$$

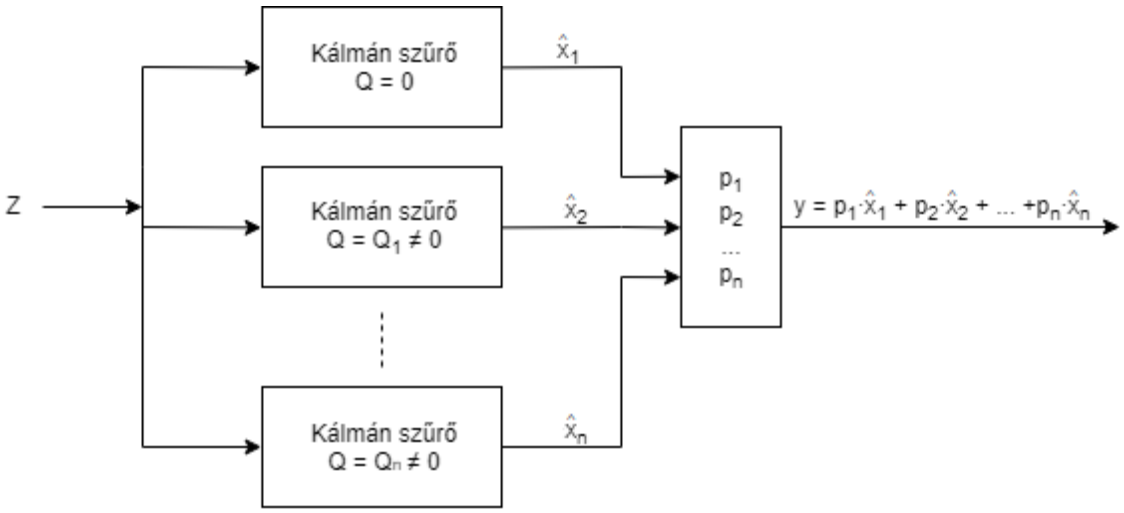
$$P_k = (1 - K_k)P_k^-$$

Ha valóban konstans távolságot mérünk, R-hez egy nagy értéket és Q-nak 0-át választva a szűrő teljesen rossz kezdeti értékről is hamar beáll a helyes értékre és nem változik. Ha megfigyeljük azonban, hogy mi történik változó távolság esetén, akkor azt tapasztaljuk, hogy a szűrő kimenete nagyon lassan fog elmozdulni az új távolság irányába. Különböző értékeket vizsgálva azt figyelhetjük meg, hogy minél nagyobb a Q értéke, a kimenet annál pontosabban tudja lekövetni a változó távolságot. Konstans távolságnál $Q \neq 0$ esetén a becslés várható értéke továbbra is a helyes érték lesz, azonban egy normális eloszlású zajt figyelhetünk meg rajta. Minél nagyobb Q

értéke, annál nagyobb ennek a zajnak a szórása. Összefoglalva egy szűrő állandó távolságot tud jól becsülni vagy a változásokat követi le nagyon pontosan. Mivel egy szűrő az előbbiek közül egyszerre csak egyik tulajdonsággal rendelkezhet, ezért egy olyan megoldásra van szükségünk, amivel több szűrő eredménye közül tudunk választani az alapján, hogy mozog-e a tag vagy sem. Két szűrő esetén, ha az egyiknek $Q = 0$ értéket állítunk be, a másiknak pedig 0-tól eltérőt, akkor, nem mozgó tag esetén az első szűrő eredményét vesszük helyes értéknek, ha pedig mozog a tag, akkor a másodikét, mivel az tudja lekövetni a tag mozgását.

4.4. Több modell alapú távolságbecslés

A több modell alapú állapotbecslés egymással párhuzamosan futtatott Kálmán-szűrőt jelent, amelyek kimeneteit valamilyen módon súlyozva kapjuk meg a végső eredményt. A több szűrő futtatása nem okozhat problémát, ugyanis ezek egyenleteit már ismerjük, a paramétereit be tudjuk állítani. Ennek a résznek a célja, hogy meghatározzuk a súlyozáshoz használt értékeket.



4.2. ábra. *Több Kálmán-szűrő együttes használata*

A Kálmán-szűrő futása során fontos paraméter a reziduál, amely megadja, hogy mekkora az eltérés a modell által becsült érték és a valódi mért érték között:

$$r_k = z_k - H\hat{x}_k^- . \tag{4.12}$$

A másik fontos paraméter az A_k, amely a reziduál kovariancia mátrix:

$$A_k = H_k P_k^- H^T + R, \tag{4.13}$$

amely fordítottan arányos a mérés megbízhatóságával. Ezen paraméterek segítségével és [2]-ben leírtak alapján, kiszámítható a j-edik Kálmán-szűrőhöz a mérések sűrűségfüggvénye a t_k-adik időpontban:

$$f_j(t_k) = \beta_j \cdot e^{\alpha_j}, \tag{4.14}$$

ahol

$$\beta_j = \frac{1}{(2\pi)^{m/2} \cdot |A_k|^{1/2}} \text{ és } \alpha_k = -\frac{1}{2} r_k^T A_k^{-1} r_k. \tag{4.15}$$

Ezek után egy normalizálást hajtunk végre, amely eredményeképpen előállnak a p_j értékek. A normalizálás során az előző időlépésben kiszámolt értékeket frissítjük a következő szerint:

$$p_j(t_k) = \frac{f_j(t_k) p_j(k-1)}{\sum_{i=1}^{K} f_i(t_k) p_i(k-1)}, \tag{4.16}$$

ahol K az anchorok számát jelöli. A normalizálás miatt p_j értékek összege 1. Ezek az értékek ezért egy teljes eseménytér független eseményeinek a valószínűségeiként is felfoghatók. Ebből a gondolatmenetből intuitívan következtethetünk arra, hogy a végső becslést a várható érték kiszámításának szabályával határozzuk meg:

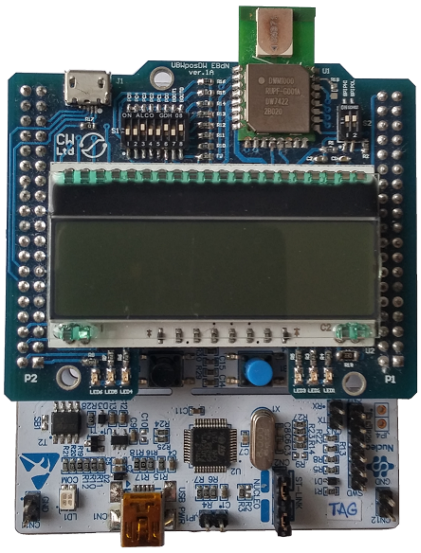
$$\hat{x} = p_1 \hat{x}_1 + p_2 \hat{x}_2 + ... + p_n \hat{x}_n \tag{4.17}$$

5. fejezet

Az elkészített rendszer

5.1. Felhasznált hardver elemek

Az UWB kommunikációhoz a DecaWave által gyártott DWM1000 chipet használtam. Az iparban ez nagyon gyakori választás, ha UWB kommunikációról van szó, ráadásul kifejezetten pozícionálási célokra lett tervezve. A chippet a STM32F401RET6U típusú mikrokontroller vezérli, amely a Nucle-64 fejlesztői panelen helyezkedik el. A két eszköz egy, a SZTAKI-ban tervezett nyomtatott áramkörrel van összekapcsolva. Ezen a NYÁK-on található még 6 db LED a különböző állapotok, események jelzésére, DIP kapcsolók a chip módjának és az UWB kommunikáció paramétereinek kiválasztásához, illetve 1 db két vagy három soros LCD kijelző.



5.1. ábra. *STM32 Nucleo-64 panel és ráhelyezett DWM1000 chip az összekötő NYÁK-kal*

5.2. A fejlesztés menete

A DWM1000 chiphez kaptam egy példaprogramot (firmware-t), amely a Nucleo kártyára volt megírva és két eszköz közötti távolságmérést tette lehetővé a 2. fejezetben ismertetett SDS-TWR technikával. Ez a példakód C nyelven van megírva, és tartalmazza mind az UWB kommunikációt megvalósító állapotgép működését, mind a chip vezérlését lehetővé tevő függvényeket. A program azonban hiányos volt, mivel bekapcsolás után csak 1 db távolságmérés történt meg, majd a program leállt. Első lépésként ezt a hibát javítottam ki, hogy folyamatos távolságmérés álljon rendelkezésre. A kijavított firmware-rel kb. 80 Hz-es frekvenciával tudunk távolságot mérni.

A mérési tulajdonságok megismerése

A folyamatos távolságmérés lehetővé tette, hogy meg tudjam vizsgálni a mérések paramétereit. A két legfontosabb tulajdonság a mérések átlaga és szórása. Ezek megállapításához két eszköz közötti üzenetváltásokat egy harmadik eszköz segítségével hallgattam le. Az üzenetekből kiolvasott távolságokat soros porton keresztül egy PC-nek továbbítottam, hogy ott további feldolgozást hajthassak végre az adatokon. Ennek az eredményét állandó távolság esetére már láthattuk a 4.1. ábrán. A mérések szórása kb. 3-4 cm távolságtól függően. A fontosabb tulajdonság azonban az, hogy mekkora ezeknek az értékeknek az átlaga. Az előző fejezetben ismertetett *range bias* jelenség miatt az átlag el fog tolódni a valós értékhez képest. Ennek megoldásaként a szoftveres kiküszöbölés lehet a legegyszerűbb, ugyanis csak egy összeadást vagy kivonást kell elvégezni a kapott távolságon. A *range bias* valódi forrása a vett jelerősség változása, tehát jogosan érvelhetnénk amellett, hogy a mért jelerősség alapján kompenzáljuk a jelenség hatását. Ennél azonban létezik egy egyszerűbb, és a DecaWave által kiadott példaprogramban is használt megoldás. Az alkalmazott eljárásban a nem kívánt *range bias* hatásával terhelt mérésekre egy skálázást hajtunk végre, amely visszaadja a valódi távolság értéket. Egy adott távolsághoz egyértelműen hozzárendelhető a vett jelerősség, és minden vett jelerősséghez 1 db *range bias* érték tartozik (mindkét pár között függvényszerű kapcsolat van[1]), ezért minden távolsághoz egyértelműen meg tudjuk határozni a neki megfelelő korrekciós értéket.

A skálázás valójában meghatározott értékkel történő eltolást jelent. A DecaWave példakódjában megtalálható minden csatornához és adatsebességhez tartozó táblázat, amely a skálázást reprezentálja. Ennek első oszlopából ki kell olvasnunk a

[1]Az egyértelműséget a szabadtéri csillapításra vonatkozó Friis-képlet és a 3.4. ábrán látható függvénykapcsolat biztosítja.

mért távolságot és a második oszlopban megtaláljuk a szükséges korrekciós értéket. A DecaWave által megadott értékek azonban nem bizonyultak helyesnek. Ez a példaprogram átírása és az összekötő NYÁK hatása miatt lehetséges. A megfelelő működés érdekében kimértem helyes korrekciós értékeket. Ezekhez adott távolságban elhelyeztem két eszközt és megnéztem, hogy mennyi a kettő között mért távolságok átlaga. A korrekciós értéket az átlag és a valódi távolság különbségéből kaptam meg. A mért értékeket az 5.1. táblázat tartalmazza.

5.1. táblázat. *Különböző távolságokhoz tartozó range bias értékek*

mért távolság (cm)	range bias érték (cm)
30	-11,8
40	-11,3
50	-11,1
60	-11,0
70	-10,7
80	-10,5
90	-10,2
100	-9,5
110	-8,1
120	-7,2
130	-6,3
140	-4,9
150	-4,2

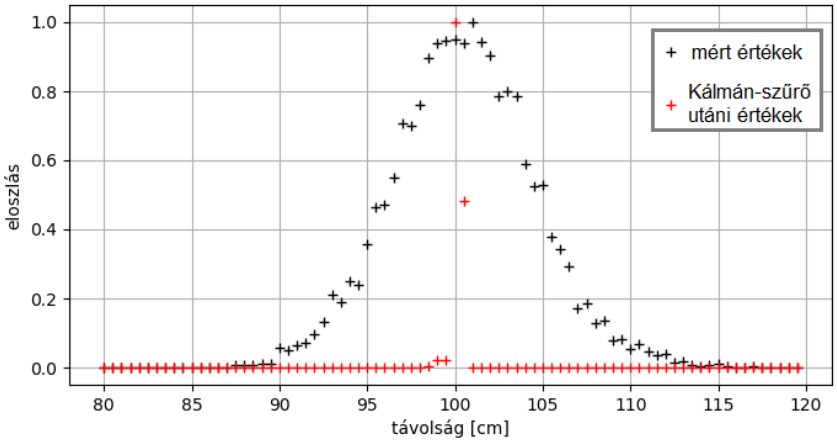
A mérési eredményeket ezekkel az értékekkel pontosítva a helyes távolságot kapjuk a mérések átlagára. Az értékek azonban továbbra is zajjal terheltek, így további feldolgozás szükséges.

A mérési bizonytalanság csökkentése

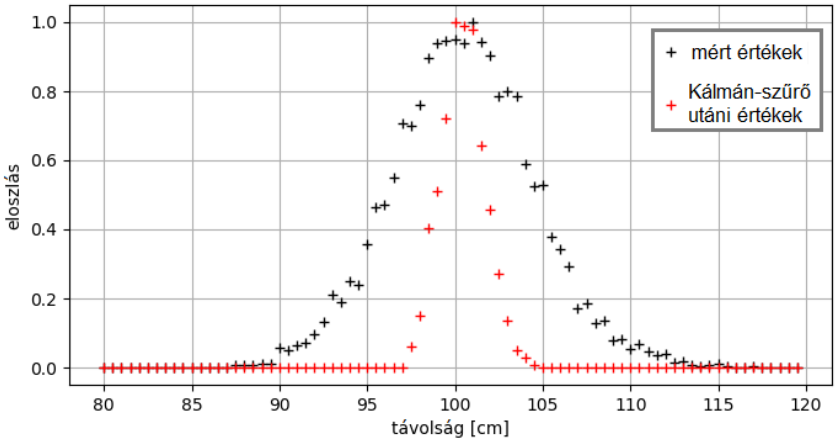
A következő lépésben a Kálmán-szűrést próbáltam ki konstans távolságmérés pontosítására. A szűrő megfelelő működéséhez tudnunk kell a mérések szórását. Ezt könnyen ki lehet számolni, csupán nagyon sok (pár ezres nagyságrendű) mérést kell végeznünk és ezekből már jó közelítést adhatunk a szórásra. A rendszerzajt (Q) azonban szabadon megválaszthatjuk. A következő ábrákon kb. 5000 mérésből számolt eloszlásfüggvényt, valamint különböző Q értékekkel futtatott Kálmán-szűrők kimenetét láthatjuk. Az ábrákat vizsgálva azt a következtetést vonhatjuk le, hogy konstans távolság esetén minél kisebb Q értéke, annál jobban csökkenti a mérések szórását a szűrő.

A Q értékét azonban nem állíthatjuk tetszőlegesen kis értékre, mert akkor nem tudná lekövetni a változó távolságokat. Ezért a több modell alapú becslést alkal-

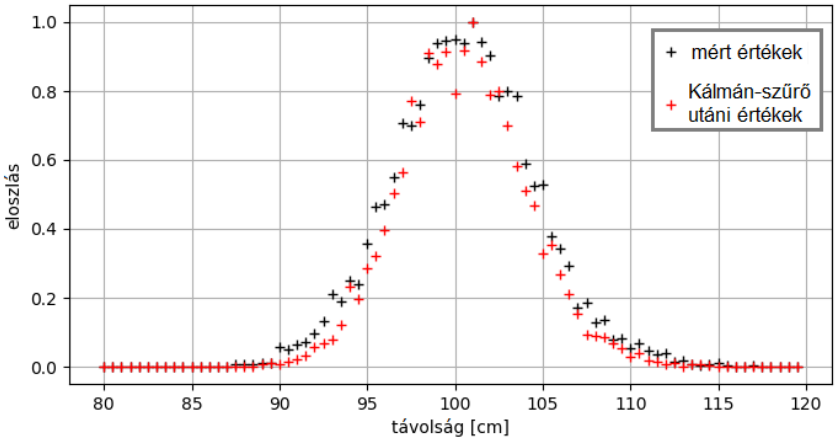
maztam két Kálmán-szűrővel. Ezeknek $Q = 0,01\ cm$ és $Q = 0,2\ cm$ paramétereket állítottam be, $R = 16\ cm$ mérési bizonytalansággal. Ezzel a két modellel álló és mozgó eszköz esetén is pontos becslést kapunk a távolságra.



5.2. ábra. *Kálmán-szűrő hatása,* $R = 16\ cm$ *és* $Q = 0\ cm$



5.3. ábra. *Kálmán-szűrő hatása,* $R = 16\ cm$ *és* $Q = 0,1\ cm$

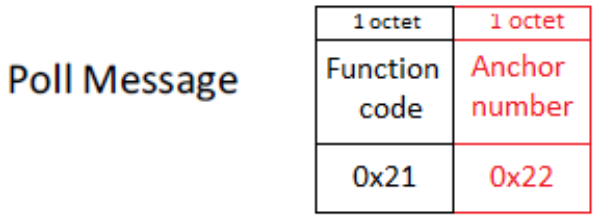


5.4. ábra. *Kálmán-szűrő hatása,* $R = 16\ cm$ *és* $Q = 100\ cm$

Annak ellenére, hogy több Python nyelven megírt Kálmán-szűrőt találhatunk, egy saját implementálását választottam. Ez a kód megtalálható F.2 függelékben. Erre azért volt szükség, mert így könnyebben tudtam befolyásolni a szűrő működését, illetve így olyan értékekhez is hozzá tudtam férni, mint a reziduál vagy a becslés hibájának kovariancia mátrixa. A több modell alapú távolságbecsléshez ezek az értékek feltétlen szükségesek.

Az üzenetek formátuma

A mérési bizonytalanság minimalizálása után a firmware-t kellett átírnom, hogy egy eszköz egyszerre több másikkal tudjon kommunikálni. Mindegyik eszközt meg kell tudnunk különböztetni, ezért 0-tól sorszámozva, minden eszköznek adtam egy egyedi számot, a 0, 1 és 2 közül. Ezután minden eszközbe raktam egy számlálót, amely akkor számol 1-et, ha hallott egy "Final"[2] üzenetet. Amikor ez a számláló elér a saját azonosítójához, akkor a következő "Poll"[3] üzenetre ő fog válaszolni. Mivel mindegyik eszköznek egyedi az azonosítója, ezért mindegyik külön-külön fog válaszolni. A módszer azonban nem működik abban az esetben, ha az egyik eszköz nem hall meg egy "Final" üzenetet, mert így a számláló nem ér el az azonosítójáig. Ilyen eset pedig gyakran előfordulhat, ezért másik módszert kellett használnom.



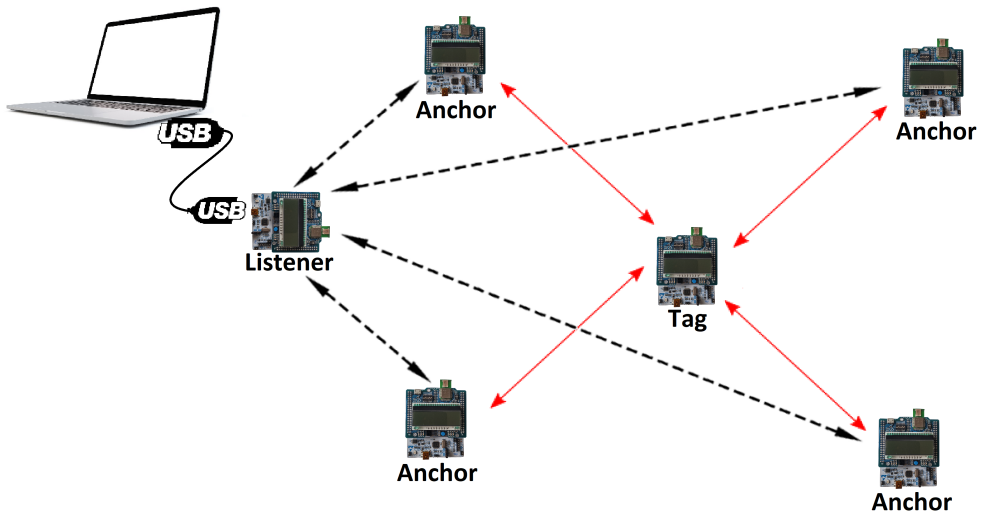
5.5. ábra. *A poll üzenet kiegészítése*

Az 5.5. ábrán a DecaWave által kiadott példaprogram eredeti "Poll" üzenete (fekete) és az általam kiegészített mező (piros) látható. A "Function Code" mező az üzenet típusát határozza meg, vagyis azt, hogy az SDS-TWR üzenetek közül ez hányadik. Az "Anchor number" nevű mezőbe a tag azt a számot írja, amelyik sorszámú anchorral szeretne kommunikálni. Így az elosztott számlálók helyett egy központi számláló van, amelyet a tag irányít és minden "Poll"-ban elküld az anchoroknak. Ez a módszer leállás nélküli és robusztus távolságmérést biztosít az eszközök között.

[2] A "Final" üzenet az SDS-TWR harmadik üzenete.
[3] A "Poll" üzenet az SDS-TWR első üzenete.

5.3. A pozíció kiszámítása

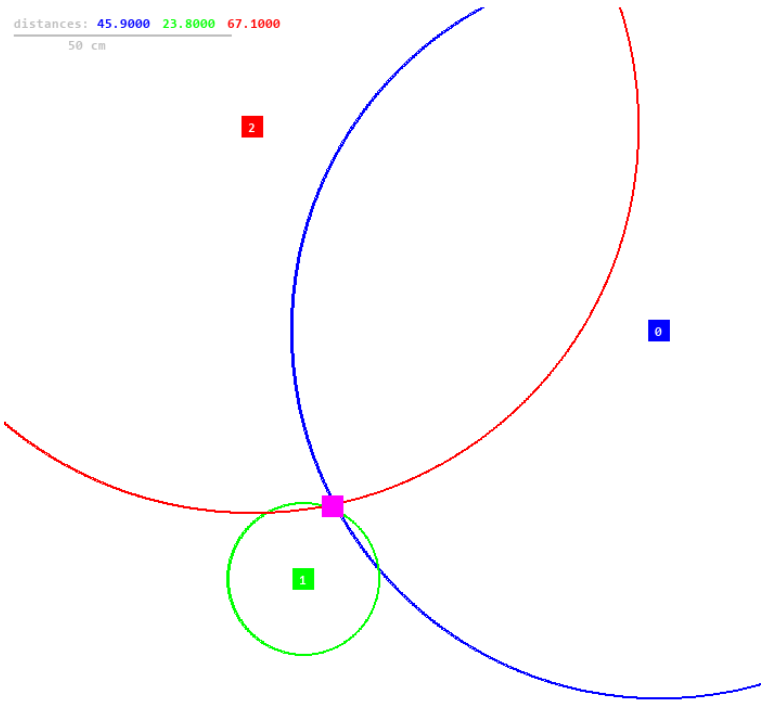
Miután sikerült megoldani több eszköz együttes kommunikációját és a távolságmérés bizonytalanságát is minimálisra csökkentettem, a pozíció kiszámítása következett. Ezt egy PC-n futó alkalmazásban tettem meg. Ehhez azonban szükséges, hogy a távolság adatokat továbbítani tudjuk a PC-nek. Ezt egy kitüntetett eszköz végzi, amely a távolságmérésben nem vesz részt, azonban hallgatja az üzeneteket és USB kapcsolaton keresztül elküldi a távolságokat a PC-nek. A hálózati elrendezést mutatja az 5.6. ábra.



5.6. ábra. *A pozícionáló rendszer hálózati topológiája*

A *range bias* miatti korrekció még az anchorokon történik, a Kálmán-szűrés és a pozíciószámolás pedig már a PC-n. Első lépésként a síkban történő pozícionálást próbáltam ki, amelyhez egy algebrai módszert implementáltam. Ennek a Python-ban megírt kódja megtalálható az F.1 függelékben. Az algebrai módszer azt jelenti, hogy kizárólag geometriai ismeretek alapján számoljuk ki a tag pozícióját. Az ehhez készült alkalmazás felületét mutatja az 5.7. ábra.

Az algoritmus legalább három kör esetén működik. Minden lehetséges módon párba állítja a köröket, és megkeresi ezeknek a metszéspontját. Ha nincsen metszéspontjuk, akkor a körök sugarait felhasználva, a körök középpontjainak koordinátáit súlyozva határoz meg egy pontot. Egy metszéspont esetén egyértelmű a megoldás. Ha két metszéspont van, akkor közülük azt választja, amelyik közelebb van a harmadik körívhez. Ezt mindegyik párra elvégzi a program és az eredményül kapott pontok koordinátáit átlagolva kapjuk meg a három (vagy több) kör közös metszéspontját.



5.7. ábra. *Két dimenzióban történő pozícionálás három anchorral (piros, zöld, kék) és egy taggel (rózsaszín)*

A fejlesztés következő lépése a három dimenzióra való áttérés volt. Az elrendezésben ez egy plusz eszközt jelent, a pozícionálásban pedig egy teljesen új számítási módszert. Három dimenzióban az algebrai módszerrel gömbök metszetét kellene számolnunk, de ez jelentősen bonyolítaná a megoldást. Ehelyett áttértem a numerikus módszerre és a következő egyenlettel határoztam meg a költségfüggvényt:

$$c(x, y, z) = \sum_{i=1}^{N}((x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 - d_i^2)^2, \tag{5.1}$$

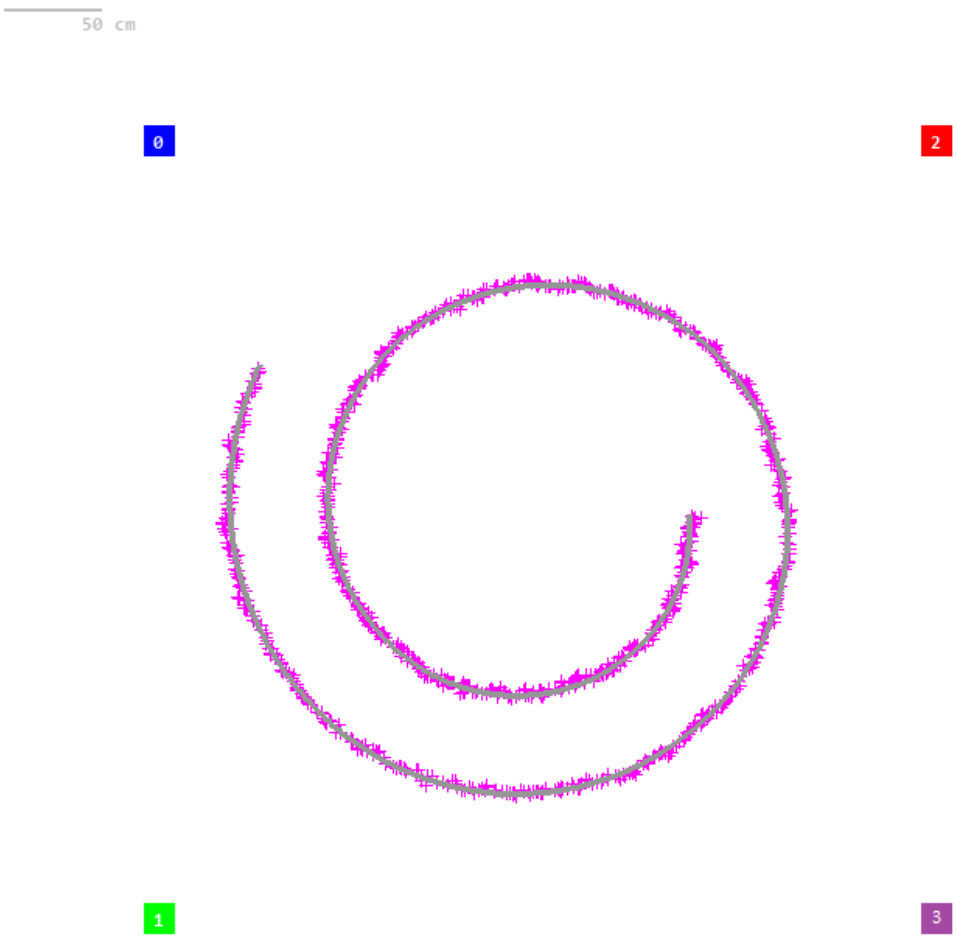
ahol N jelöli az anchorok számát, (x, y, z) a tag, (x_i, y_i, z_i) az i. anchor pozícióját, d_i pedig a tag távolságát az i. anchortól. A tag helyzetének kiszámítása annyit jelent, hogy megkeressük azt az (x, y, z) számhármast, ahol a költségfüggvény értéke minimális. Ez egy nemlineáris optimalizálási feladat, amelyet a Python nyelven elérhető Scipy könyvtárban található
$minimize(cost\_function,\ x0,\ method = 'TNC',\ bounds = bnds,\ jac = cost\_jac)$
függvény segítségével oldottam meg. Ebben többféle metódus is használható, a pontosság és a gyorsaság alapján az "L-BFGS-B" és "TNC" algoritmusokat találtam a legmegfelelőbbnek. Ezek kb. 8 ms-os futási idővel rendelkeznek és a megoldásra vonatkozó feltételeket is figyelembe tudják venni. A gyorsabb számolás érdekében a költségfüggvény Jacobi-mátrixát is meg lehet adni.

5.4. A rendszer validálása

Olyan rendszereknél, amelyeknél mérések alapján jutunk valamilyen információhoz, nagyon fontos a validáció. A validálás arra szolgál, hogy meghatározzuk az adott rendszer pontosságát.

Kétdimenziós szimuláció

Először offline módon végeztem el a validálást. Ez annyit jelent, hogy egy előre meghatározott görbe mentén végighaladva generáltam a távolságmérési adatokat. Ezen értékekre a valós mérésekből számolt paraméterű Gauss-eloszlású zajt ültettem, és ezeket a távolságokat adtam bemenetként a pozíciót számoló algoritmusnak, amely az F.1 függelékben található. Egy 4x4 m-es négyzet sarkaiban helyeztem el az anchrokat és egy spirál pályán mozgattam a taget. A szimuláció eredménye az 5.8. ábrán látható. A bejárt pálya egy szürke vonallal van jelölve, a kiszámolt pozíciókat pedig lila "+" jelek mutatják.

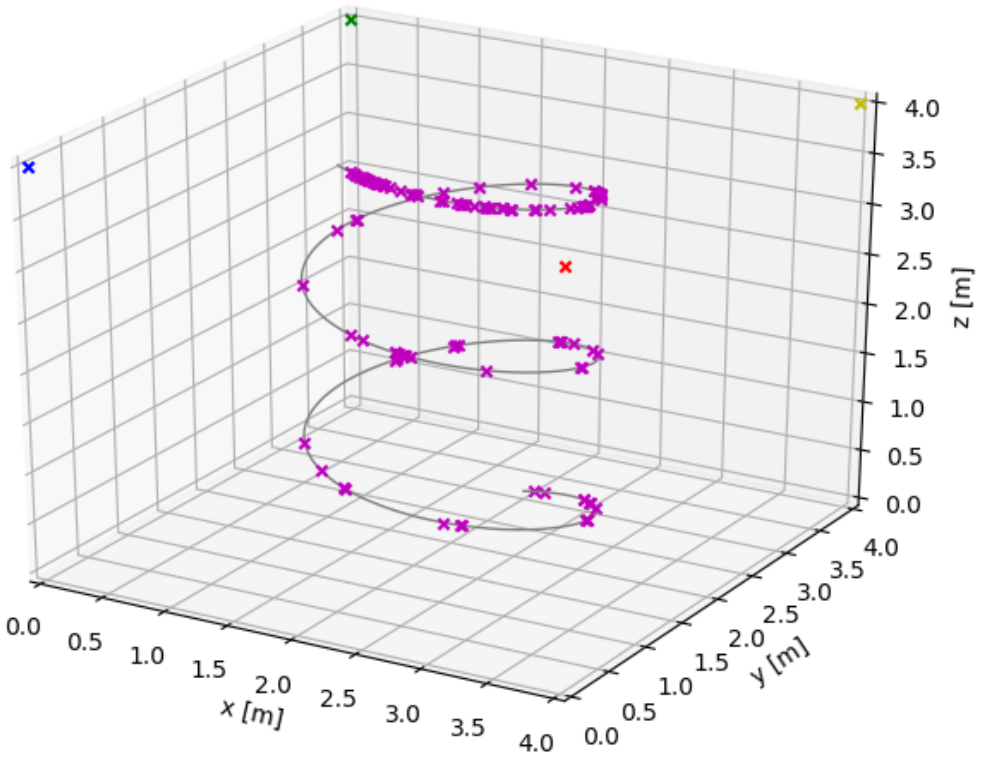


5.8. ábra. *A validálás eredménye két dimenzióban*

Mindegyik anchorhoz egy darab Kálmán-szűrőt hoztam létre $R = 16\,cm$ és $Q = 4\,cm$ paraméterekkel, mert a tag a szimulációban folyamatosan mozog, így nincs szükség kisebb Q-val rendelkező Kálmán-szűrőre. A hibák átlaga 2,5 cm, míg a szórásuk 1,7 cm volt.

Háromdimenziós szimuláció

Ezután három dimenzióban is szimulációval végeztem el a pontosság kiszámítását. Most is négy anchor pontot vettem fel, ugyanabban az elrendezésben, mint az előző, kétdimenziós esetben. Most viszont 3 m-es magasságba helyeztem őket, és a taget 0 és 3 m-es magasság között mozgattam. Ennek az eredménye látható az 5.9. ábrán.



5.9. ábra. *A validálás eredménye három dimenzióban*

Az ábrán csak azokat a helyeket jelöltem meg lila x-szel, ahol nagyobb volt a hiba, mint 5 cm. Látható, hogy a görbe alján ezek kisebb számban vannak jelen, míg a négy anchor síkjához közeledve egyre sűrűbbek. Ez amiatt van, mert ebben a síkban adott toleranciaszinthez több jó megoldás is tartozik.

Ebben az esetben is minden anchorhoz egy darab Kálmán-szűrőt hoztam létre $R = 16\,cm$ és $Q = 4\,cm$ paraméterekkel. Kisebb Q-val rendelkező Kálmán-szűrőre most sincs szükség, mert az anchor folyamatosan mozog. A hibák átlaga 3,45 cm, míg a szórásuk 2,22 cm volt.

Valós környezetben történő validálás

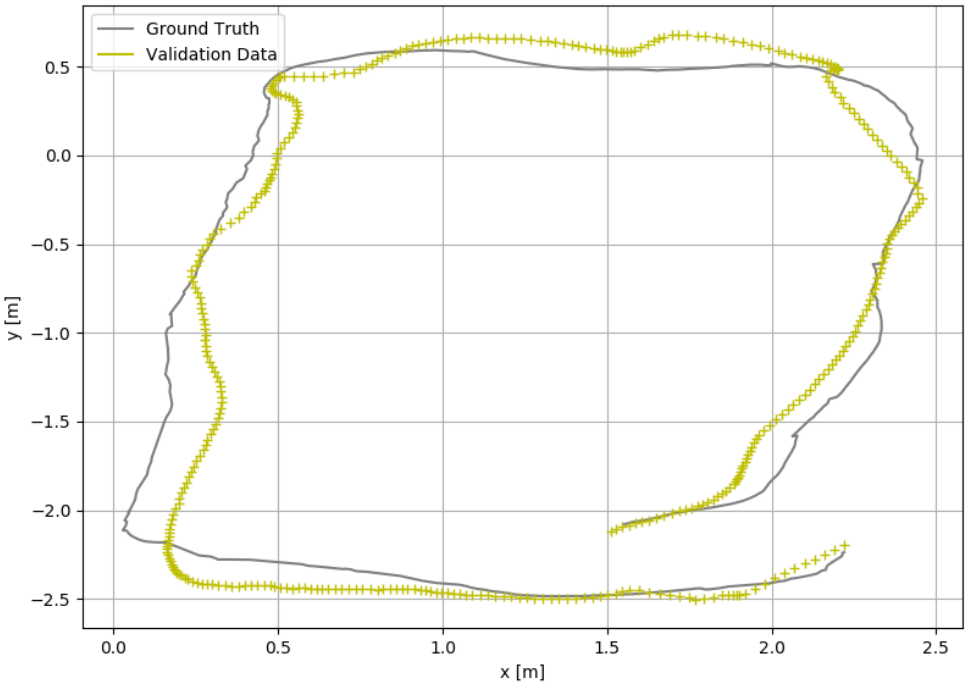
Szimulált környezetben könnyű volt a hibákat kiszámolni, ugyanis ismertük a pontos pozíciókat. Valós környezetben azonban nem áll rendelkezésünkre ilyen információ, mivel kézzel mozgatjuk a taget. Ezért egy referencia rendszerre van szükségünk, amely sokkal pontosabb mérést tesz lehetővé, mint azt a saját rendszerünktől várjuk. A referencia rendszer által szolgáltatott pozíciót *Ground Truth*-nak szokás hívni.

A validálás során egy optikai rendszer volt a referencia. Ez 12 infrakamera segítségével mm pontossággal képes meghatározni a pozíciót ún. markerek segítségével, amelyeket a tagen kell elhelyezni. A rendszer neve MTA SZTAKI MIMO (MIcro aerial vehicle and MOtion capture) arena, azaz MTA SZTAKI mikró repülőgép és gépi mozgáskövető aréna. Ez a rendszer egy kb. 5,5 m x 10 m-es helyiségbe lett telepítve, amelynek csak a felét használtuk a mérés során.

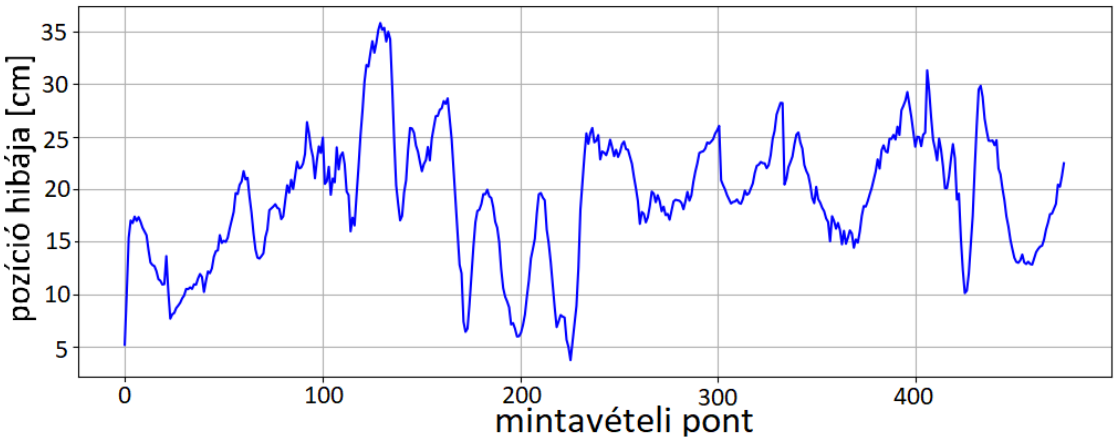
Négy anchor pontot helyeztem el a szobában 3 m-es magasságban. Optimális elhelyezésnél mindegyik anchor a vízszintessel kb. 45 °-ot bezárva, lefelé, a szoba közepe felé néz, annak érdekében, hogy minimalizálva legyenek a nem közvetlen rálátásból adódó hibák. Ezt azonban nem sikerült teljes mértékben kivitelezni az elhelyezés során. A taget egy körpálya mentén vittem végig a szobában, ügyelve arra, hogy ne menjek ki a referencia rendszer látózónájából. A két rendszer méréseit két külön naplófájlba mentettem el, majd offline módon végeztem el az összehasonlítást.

A két mérési sorozatot több szempont szerint is szinkronizálni kellett. Az első szempont az volt, hogy időben szinkronban legyenek az adatok. Ez egyrészt azt jelenti, hogy a két listából 1-1 elem kiolvasása ugyanakkora időlépést jelentsen, másrészt, hogy ugyanabban az időpontban kezdődjön a két adatsor. A második szempont az volt, hogy azonos koordináta rendszer szerint kapjuk meg a pozíciókat. Ezt úgy érhetjük el legkönnyebben, hogy a referencia rendszer koordináta rendszerét használjuk, és az anchorok ebben meghatározott helyzetét használjuk fel a validálandó rendszerben.

A szinkronizálás után az összehasonlítás következett. Ezt úgy végeztem el, hogy végigmentem mindkét adatsoron, és kiszámoltam minden időpillanatban a két rendszer által meghatározott pozíciók közötti távolságot. Ennek eredménye látható az 5.11. ábrán. A legnagyobb előforduló hiba 35 cm volt, amely az $x < 0.5\ m$ tartományban keletkezett, ahol már a négy anchor által lefedett terület határán mozgott a tag. Az adatsoron a hibák átlaga 19,2 cm és a szórásuk 6 cm volt. A nagy hiba oka többek között az anchorok rossz orientációja és a közelben levő fémcsövek lehetnek. Továbbá a mennyezet közelsége (4-5 cm) is nem kívánt problémákat okozhat.



5.10. ábra. *A validálás eredménye két dimenzióra vetítve*



5.11. ábra. *A pozíció számítás hibája a mért adatsoron*

6. fejezet

Összefoglalás, továbbfejlesztési lehetőségek

6.1. Összefoglalás

A szakdolgozat célja egy működő beltéri pozícionáló rendszer megépítése volt, amelyet maradéktalanul elvégeztem. Összehasonlítottam a rádiós kommunikációval megvalósítható pozícionálási technikákat, amelyek közül egyet kiválasztva meg is valósítottam azt. A kiválasztást nagyban befolyásolta az a tény, hogy egy kezdetleges példaprogram már rendelkezésemre állt. Ez a program viszont rosszul működött és csak két eszköz közötti kommunikációt tett lehetővé. Az én feladatom volt, hogy ezt a programot kijavítsam és továbbfejlesszem úgy, hogy egy tag négy anchorral tudjon egyszerre kommunikálni. Ezt sikerült megoldanom és kb. 20 Hz-es frissítési frekvenciát értem el a távolságokra nézve.

Körbejártam a pontosság kérdéskörét és javaslatokat tettem, hogy hogyan növelhető a távolságmérés megbízhatósága. A mérések szórása azonban nem csökkenthető 0-ra csupán a mérési módszer finomhangolásával, ezért Kálmán-szűrést alkalmaztam, hogy pontosabb becslést kapjak a távolságokra.

Az elkészült rendszerben alkalmazott három dimenzióban történő pozíció kiszámítását az 5. fejezetben ismertettem. Az ehhez készült algoritmust egy PC-n futó alkalmazásban implementáltam, amely a későbbiekben is hasznos lehet, ha irányítani szeretnénk PC-ről egy eszközt (például drónt).

A feladat meglehetősen komplex volt, így a rendszer elkészítése során számos területen sikerült tapasztalatot szereznem. Többek között megismerkedtem az UWB technológiával, a pozíciómeghatározás folyamatával, a Kálmán-szűrővel, továbbá jártasságot szereztem a beágyazott szoftverfejlesztésben. Sok hasznos ismeretre tettem szert, amelyeket a jövőben is biztosan kamatoztatni tudok.

6.2. Továbbfejlesztési lehetőségek

A rendszert sokféle irányban lehet továbbfejleszteni. A távolságmérés és háromszögeléses technika helyett a TDoA módszert is ki lehetne próbálni és összevetni, hogy a két közül melyikkel kapunk pontosabb eredményt.

Ha maradunk a távolságméréses eljárásnál, akkor a kommunikáció gyorsaságát lehet kiemelni, mint javítható tulajdonságot. Jelenleg négy anchorral kb. 50 ms-onként tudjuk frissíteni a pozíciót. Ez az idő azonban csökkenthető, ha az üzenetváltási sémát lecseréljük egy hatékonyabbra. Ez azt jelenti, hogy az SDS-TWR három üzenetes módszert a tag nem egyenként mindegyik anchorral hatja végre. A három üzenet közül az első és az utolsó mindegyik anchornak ugyanaz, a középső üzenet helyett pedig az anchorok számának megfelelő üzenet van jelen.

A legfontosabb továbbfejlesztési lépés azonban a pontosság növelése. Elsősorban a Kálmán-szűrést kell átírnunk, hogy ne konstans távolságot becsüljön. Várhatóan a drón dinamikai modelljén alapuló, több mérést is felhasználó Kálmán-szűrő nagyobb pontossággal képes a távolságok meghatározására. Egy inerciális szenzorral (Inertial Measurement Unit - IMU) is felszerelhetjük a taget, amelyet a pozícióval használva szenzorfúziót hajthatunk végre, ezzel is még pontosabb eredményt elérve.

Természetesen a rendszer további fejlesztése az MTA SZTAKI igényeinek függvényében történhet.

Irodalomjegyzék

[1] Eberly David. Intersection of ellipses. *Geometric Tools*, 2000.

[2] P. D. Hanlon and P. S. Maybeck. Multiple-model adaptive estimation using a residual correlation Kalman filter bank. *IEEE Transactions on Aerospace and Electronic Systems*, April 2000.

[3] IEEE. IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirement Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). *IEEE Std 802.15.4a-2007 (Amendment to IEEE Std 802.15.4-2006)*, 2007.

[4] IEEE. IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, Sept 2011.

[5] Tennessean Jamie McGee. Music City Center app guides visitors, Hozzáférés dátuma: 2018. november. `https://eu.tennessean.com/story/money/tech/2014/11/12/music-city-center-app-guides-visitors/18945489/`.

[6] Y. Jiang and V. C. M. Leung. An Asymmetric Double Sided Two-Way Ranging for Crystal Offset. In *2007 International Symposium on Signals, Systems and Electronics*, July 2007.

[7] Jiahong Li, Xianghu Yue, Jie Chen, and Fang Deng. A Novel Robust Trilateration Method Applied to Ultra-Wide Bandwidth Location Systems. In *Sensors*, 2017.

[8] Decawave Ltd. *Real Time Localization Systems - An Introduction(Application Note No. APS003)*. Decawave, 2014.

[9] Decawave Ltd. *Sources of Error in DW1000 Based Two-Way Ranging (TWR) Schemes (Application Note No. APS011)*. Decawave, 2014.

[10] Ivan A. Mantilla-Gaviria, Mauro Leonardi, Gaspare Galati, and Juan Vicente Balbastre-Tejedor. Localization algorithms for multilateration (MLAT) systems in airport surface surveillance. *Signal, Image and Video Processing*, 2015.

[11] NASA. GPS története (angol), Hozzáférés dátuma: 2018. november. `https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html`.

[12] O. Onalaja, M. Adjrad, and M. Ghavami. Ultra-wideband-based multilateration technique for indoor localisation. *IET Communications*, 8(10):1800–1809, July 2014.

[13] Bourke Paul. Circles and spheres. 1992.

[14] Pozyx. Pozyx cég honlapja, Hozzáférés dátuma: 2018. november. `https://www.pozyx.io/`.

[15] Nemzeti Média és Hírközlési Hatóság. Sávhasználati feltételek és frekvenciagazdálkodási követelmények, Hozzáférés dátuma: 2018. november. `http://nmhh.hu/dokumentum/165870/NFFF_03_mell_TE.pdf`.

[16] Sewio. Sewio cég honlapja, Hozzáférés dátuma: 2018. november. `https://www.sewio.net/`.

[17] Greg Welch and Gary Bishop. An Introduction to the Kalman Filter. Department of Computer Science, University of North Carolina. *ed: Chapel Hill, NC, unpublished manuscript*, 2006.

Függelék

F.1. Háromszögeléses eljárás 2 dimenzióban Python nyelven

```
# This function calculates the intersection of 3 circles.
def calcTagPos_fromThree(d, anchors, indices):
    # We add every result to this variables, then
    #    we will divide by 3 so
    #    we will get the average of the 3 intersections.
    xest = 0.0
    yest = 0.0

    # run position calculation for every pair of anchor
    for currI,i in  enumerate(indices):
        for j in indices[currI+1:]:

            # finding the 3rd anchor's index
            k = indices[0]
            if i == indices[0]: k = indices[1]
            if j == indices[1]: k = indices[2]

            # step 1: Prework
            di, dj, dk = d[i],d[j],d[k]
            xi, yi = anchors[i].x, anchors[i].y
            xj, yj = anchors[j].x, anchors[j].y
            xk, yk = anchors[k].x, anchors[k].y

            dist = math.sqrt((xi-xj)**2 + (yi-yj)**2)

            # step 2: Intersection judgement criterion
            if di+dj < dist or dist+di < dj or dist+dj < di:
                # step 3: "Distance compensation".
                # Take a weighted average of the centers.
                xest +=  float(xi*dj+xj*di)/(di+dj)
                yest +=  float(yi*dj+yj*di)/(di+dj)
            else:
                # step 4: Determination of intersections
                a = (di*di - dj*dj + dist*dist)/(2*dist)
                h = math.sqrt(di*di - a*a)
                xij  = xi+(xj-xi)*a/dist + h*(yj-yi)/dist
                yij  = yi+(yj-yi)*a/dist - h*(xj-xi)/dist
```

```
                xij2 = xi+(xj-xi)*a/dist - h*(yj-yi)/dist
                yij2 = yi+(yj-yi)*a/dist + h*(xj-xi)/dist

                # choose the one that's distance from the third
                #     anchor is more closer to the actual distance
                if (dk-math.sqrt((xij-xk)**2 + (yij-yk)**2))**2 <
                    (dk-math.sqrt((xij2-xk)**2 + (yij2-yk)**2))**2:
                    xest += xij
                    yest += yij
                else:
                    xest += xij2
                    yest += yij2

    # step 5: Take average
    xest /= 3
    yest /= 3

    return [xest, yest]
    pass

# This function calculates the tag's position from the anchors'
#    positions and the distances from them.
def calcTagPos(d, anchors):
    indices = [[0,1,2],
               [0,1,3],
               [0,2,3],
               [1,2,3] ]
    x_real = 0.0
    y_real = 0.0

    for ind in indices:
        [x,y] = calcTagPos_fromThree(d,ind)
        x_real += x
        y_real += y

    x_real /=  len(indices)
    y_real /=  len(indices)
```

F.2. Kálmán-szűrő implementálása Python nyelven

```
import numpy as np

class Kalmanfilter:
    def createVariable(self, a):
        if self.type == 0:
            return a
        else:
            return np.array(a)
    def __init__(self, _A, _B, _H, _Q, _R, _x=0, _P=0, _u=0):
        if np.isscalar(_A):
            self.type = 0   # we have constants
        else:
            self.type = 1   # we have vectors
        self.A = self.createVariable(_A)
        self.B = self.createVariable(_B)
        # transfer function between measurements and states
        self.H = self.createVariable(_H)
        # process noise
        self.Q = self.createVariable(_Q)
        # measurement noise
        self.R = self.createVariable(_R)

        self.xhatkminus1 = self.createVariable(_x)
        self.Pkminus1 = self.createVariable(_P)
        self.ukminus1 = self.createVariable(_u)

    def getInverse(self,x):
        # if we have scalars, return 1/x
        if self.type == 0:
            return 1/x
        else:
            try:
                return np.linalg.inv(x)
            except np.linalg.LinAlgError as err:
                try:
                    return 1/x
                except ValueError:
                    return 0

    def getIdentityWithSize(self, z):
        # if we have scalars, return 1
        if self.type == 0:
            return 1
        else:
            try:
                # np.shape returns "('n'L, 'm'L)" where 'n' &
                #     'm' are the apropriate values
                [n,m] = [ int(x) for x in  str(np.shape(self.A)).replace(
```

```
                ",", "").replace("(", "").replace(")", "").replace("L",
                        "").split()]
        except ValueError:
            # if z is scalar, np.shape() throw ValueError
            return 1
        # return with an n x n identity matrix
        return np.eye(n)

    def multiplyItems(self,a,b,c=None):
        if self.type == 0:
            if c == None:
                return a*b
            else:
                return a*b*c
        else:
            if c is None:
                return np.matmul(a,b)
            else:
                return np.matmul(np.matmul(a,b),c)

    def run(self, z, uk=0, getP=False, getRes = False):
        # time update
        xhatminusk = self.multiplyItems(self.A, self.xhatkminus1) +
                             self.multiplyItems(self.B, self.ukminus1)
        Pminusk = self.multiplyItems(self.A, self.Pkminus1, np.transpose(
                                    self.A)) + self.Q
        # measurement update
        K = self.multiplyItems(Pminusk, np.transpose(self.H),
              self.getInverse(self.multiplyItems(self.H, Pminusk,
                                np.transpose(self.H)) + self.R))
        xhatk = xhatminusk + self.multiplyItems(K, (z -
                        self.multiplyItems(self.H, self.xhatkminus1)))
        Pk = self.multiplyItems((self.getIdentityWithSize(self.A) -
                 self.multiplyItems(K, self.H)), Pminusk)

        residual = z - self.multiplyItems(self.H, self.xhatkminus1)
        Ak = self.multiplyItems(self.H, Pminusk,
                      np.transpose(self.H)) + self.R

        # save variables
        self.ukminus1 = uk
        self.xhatkminus1 = xhatk
        self.Pkminus1 = Pk

        ret = [xhatk]
        if getP == True:
            ret = [ret, Pk]
        if getRes == True:
            ret = [ret, residual, Ak]
        return ret
```