Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

Drónaréna fejlesztése irányítási algoritmusok teszteléséhez

Tudományos Diákköri Konferencia dolgozat

Készítette
Molnár Marcell

Konzulens
dr. Kiss Bálint

Külső konzulens
Luspay Tamás Gábor

2019. október 28.

Tartalomjegyzék

Kivonat

Az autonóm járművek fokozatosan egyre nagyobb szerepet kapnak mind az iparban, mind a polgári felhasználásban. Utóbbira gondolva a legtöbb embernek az önvezető autók jutnak eszébe, azonban számos más polgári alkalmazásban is fellelhetők az ilyen technológiák. Egy példa lehet az Ázsiában már számos étterem által alkalmazott robotpincér. Nem kell azonban ilyen távolra mennünk, hiszen már Magyarországon is van olyan hely, ahol robot szállítják ki az ételt a vendégeknek.

Az ipari alkalmazásokban is számos helyen fel lehet lelni az önmaguktól speciális feladatot ellátó robotokat. A legtöbb autógyár rendelkezik önállóan dolgozó robotokkal, mivel hosszútávon olcsóbbak és megbízhatóbbak, mint az ember. Automatizált raktározási feladatokban is alkalmazhatunk önműködő robotokat.

A földi járművökön kívül a légijárművekben is egyre több önjáró funkció jelenik meg. Az utasszállító repülőgépek már maguktól képesek felszállni, leszállni, valamint adott pályát követni. A hagyományos repülőgépeken kívül egyre nagyobb teret nyernek a kvadkopterek és drónok. A két kifejezést a köznyelv szinonimaként használja, azonban jelentős különbség van a kettő között. Míg előbbi négy rotoros légijárműveket jelent, addig utóbbi a pilóta nélküli járművekre vonatkozik.

Jelenleg is számos kutatás folyik, valamint sok cikk jelenik meg a kvadkopterek irányításának témájában. Ahhoz, hogy ezeket az irányítási algoritmusokat valós rendszereken is ki lehessen próbálni, olyan platformra van szükségünk, amellyel valós időben monitorozni lehet a kvadkopter állapotát. Az olyan platformokat, amelyek kifejezetten drónok tesztelésére szolgálnak drónarénának is nevezhetünk. Ilyen rendszerrel a gép pozícióját és orientációját szeretnénk mérni. Amennyiben erre képesek vagyunk, akkor össze tudjuk hasonlítani a referencia pályát a gép által ténylegesen bejárttal, ami alapján minősíteni tudjuk az irányítási algoritmust.

Eddigi munkám során egy UWB rádiós technológián alapuló beltéri pozícionáló rendszert készítettem, amely egy bizonyos környezetben már validálásra is került. Egy állapotbecslőt is terveztem, amely a kvadkopter dinamikai modelljét és a pozícionáló rendszert használja a gép monitorozására. Jelen dolgozatban a pozícionáló rendszert, annak telepítését és kalibrálását, az állapotbecslőt, magát a kvadkoptert mutatom be és a tervezett jövőbeli munkát ismertetem.

Abstract

Self-driving vehicles are gaining importance in both industry and civil applications. Thinking of the latter, self-driving cars come to most people's mind, but there are many other civil applications of these technologies. An example is the robot waiters employed by many restaurants in Asia. But we don't have to go that far, because in Hungary also exists a place where robots deliver the food to guests.

In industrial applications, robots that perform special tasks on their own can be found in many places. Most automotive companies have autonomous robots because they are cheaper and more reliable than humans in long run. We can also use automated robots for automated storage tasks.

In addition to ground vehicles, more and more self-driving functions appear in aircrafts. Passenger aircrafts are capable of taking off, landing and following a course autonomously. In addition to traditional aircrafts, quadcopters and drones are getting more attention. The two terms are used interchangeably, but there is a significant difference between them. While the former refers to four-rotor aircrafts, the latter refers to unmanned vehicles.

Numerous studies are ongoing and many articles are published on the subject of quadcopter control. In order to test these control algorithms on real systems, we need a platform that can monitor the status of the quadcopter in real time. Systems that are made especially for drones can be called drone arena. With such a system we want to measure the position and orientation of the machine. If we are able to do this, we can compare the reference path with the machine's actual path, which we can use to qualify the control algorithm.

During my work so far, I have developed an UWB radio technology based Indoor Positioning System (IPS), which has already been validated in a certain environment. I also designed a state estimator that uses the dynamic model of the quadcopter and the positioning system to monitor the machine. In this paper I present the positioning system, its installation and calibration, the state estimator, the quadcopter itself and the planned future work.

Bevezető

Egy drónaréna meglehetősen komplex abból a szempontból, hogy hány alrendszer együttes működése szükséges ahhoz, hogy egy drónt - pontosabban kvadkoptert - önműködően lássunk repülni benne. Először is szükségünk van egy pozícionáló rendszerre, amely képes meghatározni a drón helyzetét egy szobán belül. Erre a feladatra a GPS nem használható, ugyanis pontossága nem elegendő (néhány 10 cm-től 1 m-ig is), beltéren gyakran használhatatlan a vett jel erősség csökkenése miatt, valamint nem elég gyors a frissítési frekvencia. Ezen szempontok alapján egyértelmű, hogy valamilyen másik technológiát kell alkalmaznunk. Ilyen megoldásokat a dolgozat 1. fejezetében mutatok be, leginkább kiemelve az UWB technológiát.

Az UWB-nél alkalmazott távolságmérés után azonban még számos módon eljárhatunk, hogy megkapjuk az általunk érdekelt pozíciót, így ennek a lehetséges módjait a 2. fejezetben mutatom be. A fejezetben néhány nem időmérés alapú pozícionálási megközelítést is bemutatok.

Maga a drón az egyik legfontosabb része a rendszernek, így erről is beszélnünk kell. Ugyanis nem triviális, hogy hogyan lesz képes magától mozogni és önmagát vezérelni a drón. A gép megépítését és konfigurálását, a drón dinamikus modelljét, valamint paraméter identifikációjának lehetőségeit a 4. fejezetben tárgyalom.

Kihasználva az információt, hogy egy drón helyzetét szeretnénk meghatározni, állapotbecslőt is használhatunk. Ez az állapotbecslő a drón dinamikai modelljét, az UWB-s távolság adatokat és a drónra adott bemeneteket fogja felhasználni, hogy minél pontosabban tudjuk megfigyelni a drón állapotát. A dinamikai modellben azonban számos olyan paraméter van, amely egy adott drónra jellemző szám, és csak ezek pontos ismeretében használható a modell. Ezen paraméterek meghatározására identifikációt kell végeznünk a drónon. A dinamikus modell, az UWB távolságadatai, valamint IMU szenzor adatainak fúzionálása a 5. fejezetben kerül bemutatásra. A fúzionálást egy kiterjesztett Kálmán-szűrő fogja végezni.

Végül az összefoglalás és a továbbfejlesztési lehetőségek a 6. fejezetben olvasható.

1. fejezet

Beltéri pozícionáló technológiák

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.

1.1 WiFi, Bluetooth

Elsőként a WiFi-t érdemes megemlíteni. Ennek a technológiának az előnye, hogy már kiépített infrastruktúrát használhatunk, hiszen nagyon sok helyen jelen van. A pozíció számolásra többféle megközelítés is lehetséges. Az első megoldás, hogy a vett jelerősség (RSSI - Received Signal Strength Indicator) alapján távolságot és háromszögelés segítségével pozíciót számolunk. Az RSSI értékek azonban a környezeti változásokra könnyen megváltozhatnak, és így a háromszögelés eredményre megbízhatatlan lesz. Ennek a módszernek egy továbbfejlesztett változata 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 "fingerprinting" módszernek is az a hátránya, hogy az RSSI értékek nagymértékben ingadozhatnak a környezetben történő változások hatására.

Egy másik széles körben elterjedt technológia a Bluetooth. Míg régebben a Bluetooth SIG (Special Interest Group) állítása szerint a Bluetooth kizárólag a közelség érzékelésére volt használható, az újabb verziók ma már relatíve pontos helymeghatározásra is használhatók. 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 [8]. Ezek a rendszerek

persze arra alkalmasak, hogy navigáljanak bennünket nagyobb épületeken belül, a pontosságuk így is több 10 cm feletti. A legújabb 5.1-es verzió [1] azonban már pontosabb helymeghatározást tesz lehetővé és akár cm-es pontosságot is elérhetünk vele [5].

MapsPeople

A MapsPeople egy dán gyökerű cég, akik több, mint 20 év tapasztalattal rendelkeznek beltéri pozícionálás és feltérképezés területén. Ezt a céget azért érdemes kiemelni, mert olyan partnereik vannak, mint például a Google, Ericsson, Scania, Siemens és GLS [16]. A MapsPeople szoros együttműködésben van a Google-lel, hiszen a telefonos applikációjuk is a Google Maps-re épül. A WiFi és Bluetooth alapú rendszert is ki tudnak építeni, választást hagyja, hogy a legmegfelelőbb technológiát lehessen alkalmazni.

1.2 UWB

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.

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.

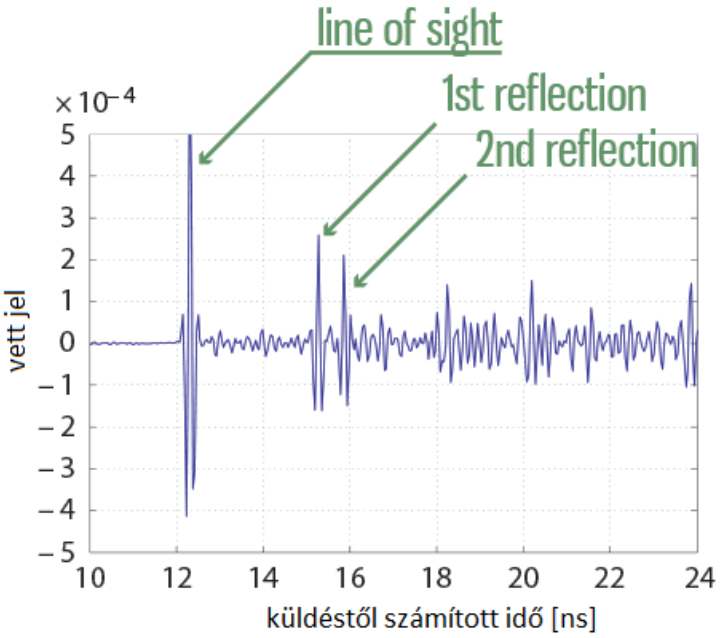
Heisenberg-féle határozatlansági elv - a sávszélesség megválasztása

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 használnunk. 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. [21]



1.1. ábra. *UWB jel időtartományban [21]*

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).

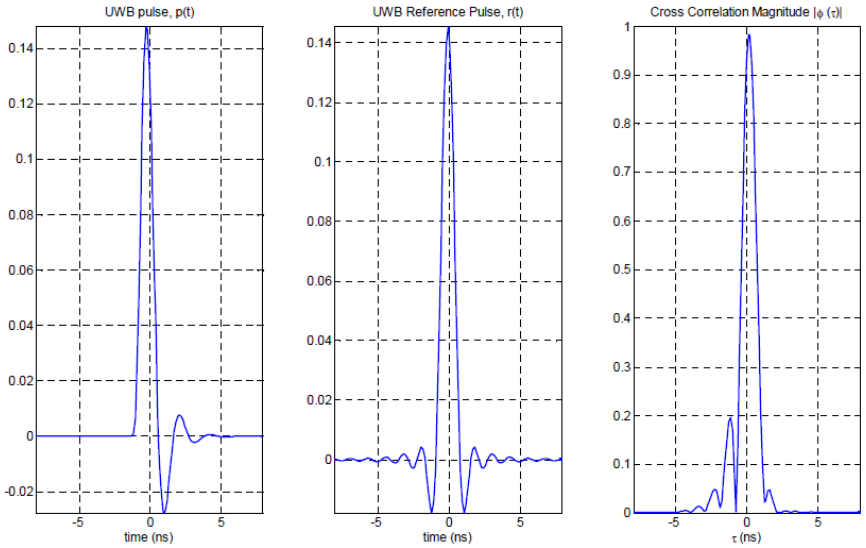
Az UWB jellemzése

A fizikai jel

Az UWB jelalakja az IEEE 802.15.4a szabványban van meghatározva, amelyet az 1.2. ábra szemléltet. A szabvány [6] 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 [22] 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) [6]*

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

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

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.

	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 ECC/DEC/(06)04 MSZ EN 302 065	–90	–50
3	1,6–2,7 GHz		–85	–45
4	2,7–3,1 GHz		–70	–36
5	3,1–3,4 GHz	2007/131/EK, 2009/343/EK ECC/DEC/(06)04 MSZ EN 302 065	–70	–36
6	3,4–3,8 GHz		–80	–40
7	3,8–4,2 GHz		–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 ECC/DEC/(06)04 MSZ EN 302 065, MSZ EN 302 500-2	–41,3	0
14	8,5–9 GHz		–65	–25
15	9–10,6 GHz	2007/131/EK, 2009/343/EK ECC/DEC/(06)04 MSZ EN 302 065	–65	–25
16	>10,6 GHz		–85	–45

1.3. ábra. *UWB maximális adóteljesítményére vonatkozó táblázat (részlet) [22]*

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

[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.

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.

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

1.4. ábra. *IEEE 802.15.4 fizikai réteg (PHY) üzeneteinek formátuma [7]*

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 [6]. 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.

<table>
<tr><td>Octets: 2</td><td>1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td><td>0/5/6/10/14</td><td>variable</td><td>2</td></tr>
<tr><td rowspan="2">Frame Control</td><td rowspan="2">Sequence Number</td><td>Destination PAN Identifier</td><td>Destination Address</td><td>Source PAN Identifier</td><td>Source Address</td><td rowspan="2">Auxiliary Security Header</td><td rowspan="2">Frame Payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="7">MHR</td><td>MAC Payload</td><td>MFR</td></tr>
</table>

1.5. ábra. *IEEE 802.15.4 MAC réteg üzeneteinek formátuma [7]*

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.

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. [21] 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. [24] 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.

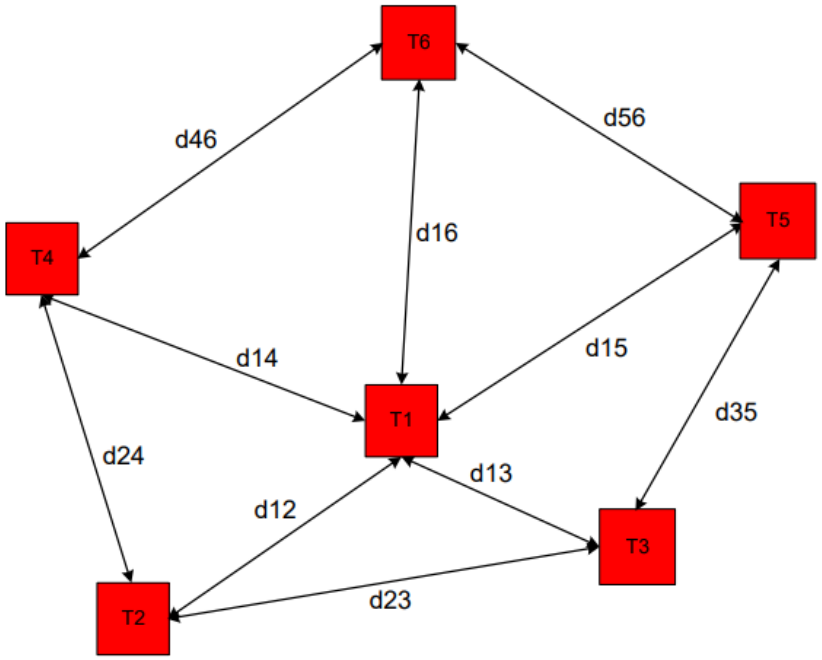
Saját rendszer fejlesztésének motivációja

Annak ellenére, hogy számos beltéri pozícionáló megoldás létezik az iparban, mégis egy saját rendszer fejlesztése a cél. Ezt többek között az indokolja, hogy így teljes hozzáférésünk van a rendszerhez, mi építjük fel az egészet. Az egész rendszert alaposan ismerni fogjuk, ami például nagy előnyt jelent hibakereséskor. További előny, hogy a rendszer minden belső jeléhez hozzáférünk, fel tudjuk használni bármelyik belső információját. Ezen kívül a kész rendszerektől eltérően ezt teljes mértékben a saját igényeinkre tudjuk szabni, olyan rendszert építhetünk, amely minden követelményünket kielégíti.

2. fejezet

A pozícionálás 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 [14]*

Erre egy szemléletes példa, amikor tűzoltók bemennek egy házba, ahol nincs kiépített 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. Ezt a módszert gyakran alkalmazzák WiFi és Bluetooth alapú technológiáknál.

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.

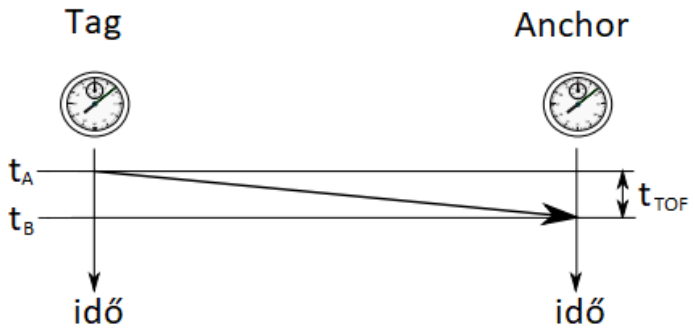
Időmérés alapján is lehetséges távolságot és ezáltal pozíciót becsülni. A továbbiakban ilyen módszerek, valamint a pozíció meghatározásának lehetséges megközelítései kerülnek részletes bemutatásra.

2.1 ToF (Time of Flight), azaz időmérés alapú távolságmérés

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 $t_{TOF} = t_B - t_A$.

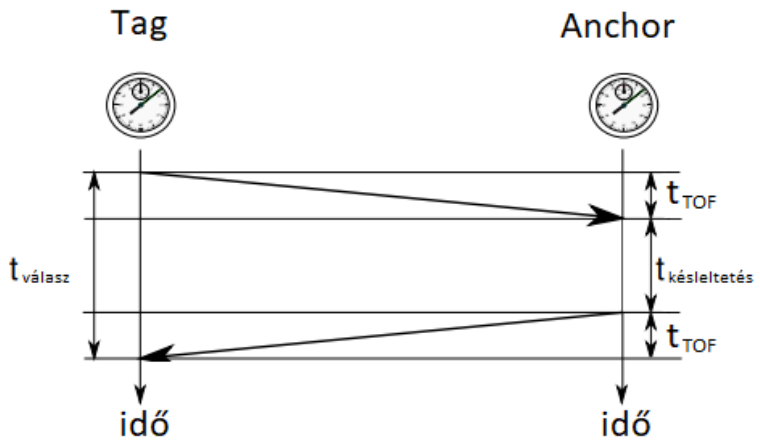


2.2. ábra. *Egy üzenetváltásos távolságmérés*

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

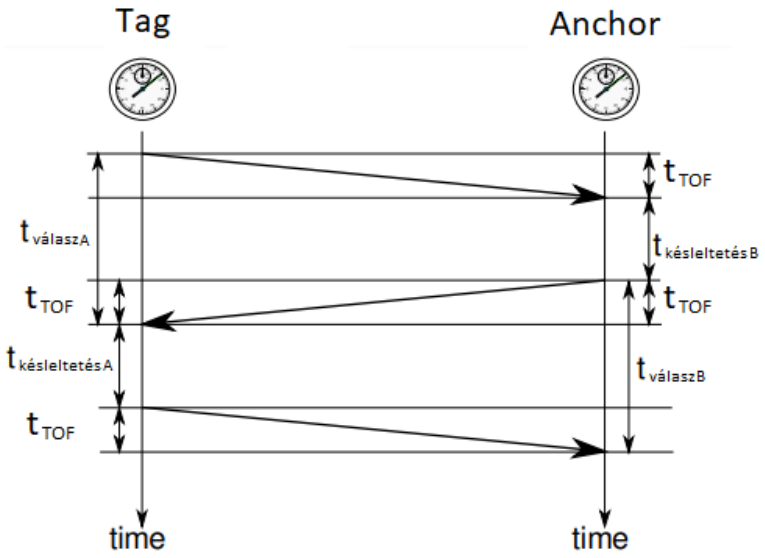
$$t_{TOF} = \frac{t_{\text{válasz}} - t_{\text{késleltetés}}}{2} \tag{2.1}$$

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.

[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 [9]-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.

2.2 Pozíció kiszámítása távolságadatok alapján

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.

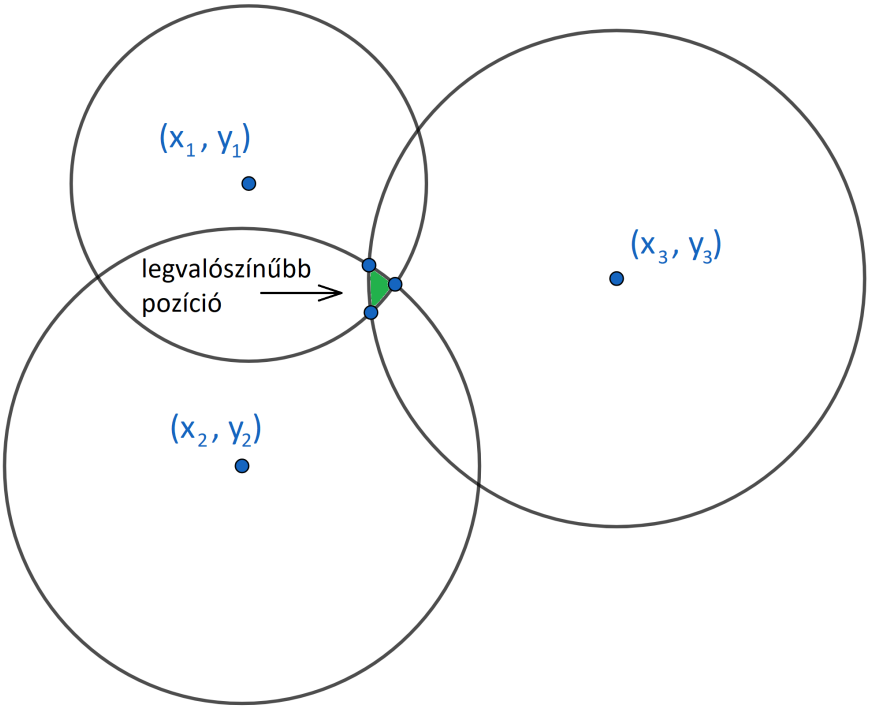
Statisztikai módszeren alapuló modellek

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. [15]

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 [11]. A legtöbb esetben a pozíció kiszámítása a körök metszéspontjainak megkeresésével kezdődik [20]. 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.

Egy lehetséges költségfüggvény a következő:

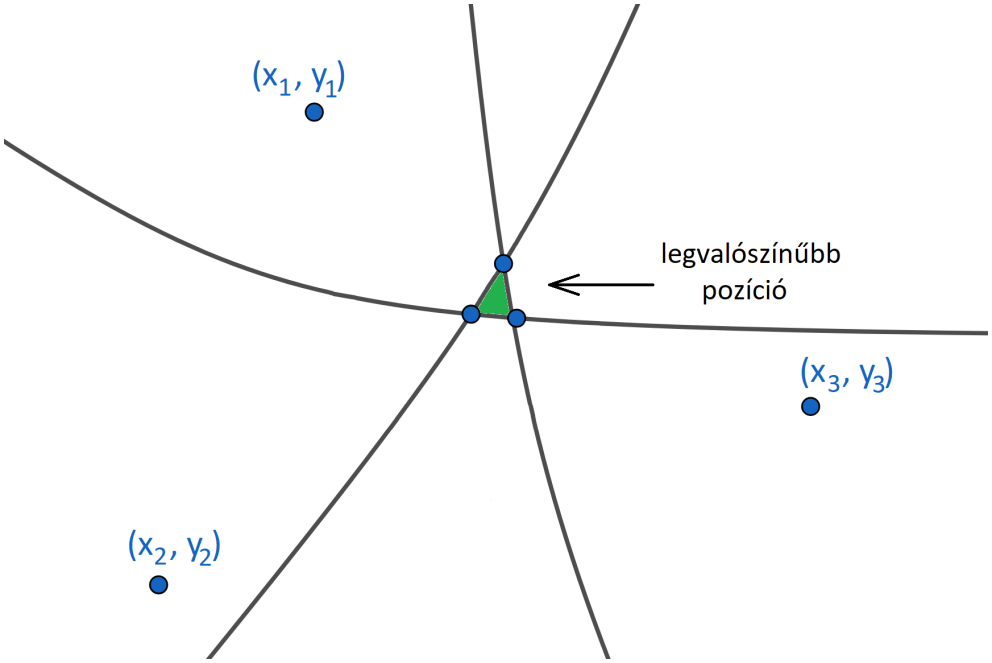
$$c(x, y, z) = \sum_{i=1}^{N}((x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 - d_i^2)^2, \tag{2.3}$$

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. [15]

2.3 Egyéb módszerek

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.

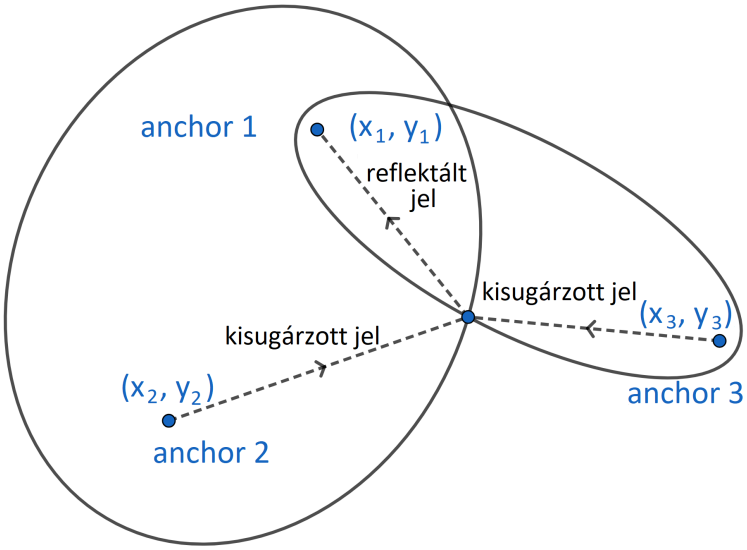


2.6. ábra. *Időkülönbségen alapuló pozícionálás*

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.

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 meg is szüntethető. [19]



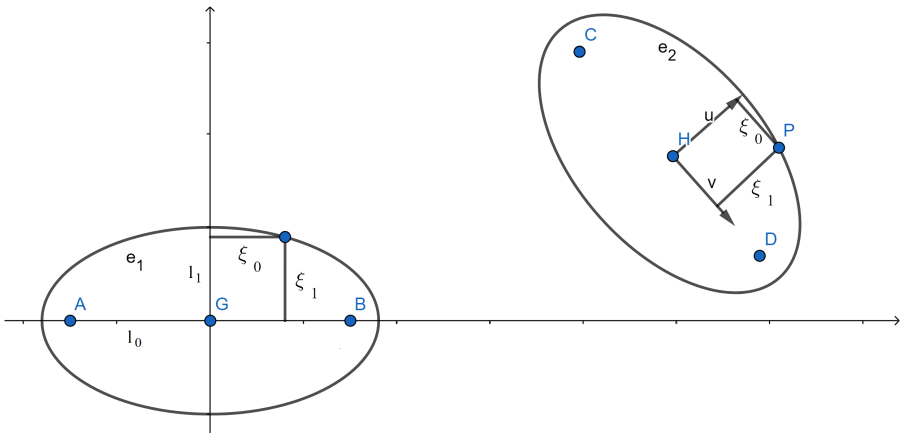
2.7. ábra. *Reflektáláson alapuló pozícionálás*

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:

$$1 = M_{00}(x - x_0)^2 + (M_{01} + M_{10})(x - x_0)(y - y_0) + M_{11}(y - y_0)^2, \tag{2.4}$$

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 konstansok meghatározásához tekintsük a következő elrendezést, e_1 ellipszist transzfor-

máljuk e_2-be.



2.8. ábra. *Ellipszis transzformálása tetszőleges helyzetbe*

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

$$1 = \left(\frac{\xi_0}{l_0}\right)^2 + \left(\frac{\xi_1}{l_1}\right)^2 \tag{2.5}$$

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:

$$1 = \left(\frac{u \cdot (P-H)}{l_0}\right)^2 + \left(\frac{v \cdot (P-H)}{l_1}\right)^2 \tag{2.6}$$

Rövid rendezés után azt kapjuk, hogy

$$1 = (P-H)^T M (P-H), \tag{2.7}$$

ahol

$$M = \begin{bmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{bmatrix} = \begin{bmatrix} \frac{u_0^2}{l_0^2} + \frac{v_0^2}{l_1^2} & \frac{u_0 u_1}{l_0^2} + \frac{v_0 v_1}{l_1^2} \\ \frac{u_0 u_1}{l_0^2} + \frac{v_0 v_1}{l_1^2} & \frac{u_1^2}{l_0^2} + \frac{v_1^2}{l_1^2} \end{bmatrix} \tag{2.8}$$

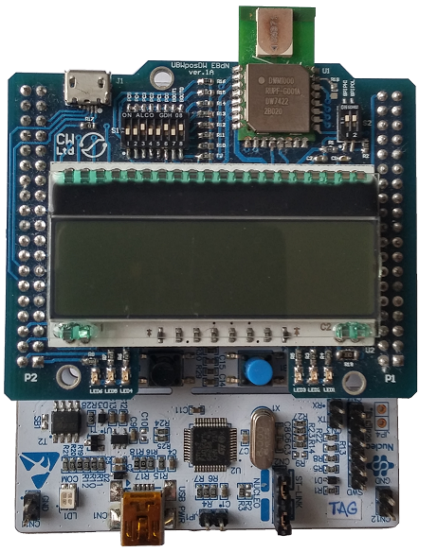
Ezen konstansok segítségével felírható két 2.4 alakú egyenlet a 2.7. ábrán látható két ellipszisre. Az ebből kapott egyenletrendszer megoldását a 2.2. fejezetben ismertetett numerikus módszerrel kereshetjük meg legkönnyebben. A 2.7. ábrán látható, hogy ekkor még mindig két megoldás lehetséges, így szükségünk van egy harmadik ellipszisre is. A megoldást az a metszéspont adja, amely mind a három ellipszisen rajta van. [4]

3. fejezet

A pozícionáló rendszer megépítése és kalibrációja

3.1 Felhasznált hardver elemek

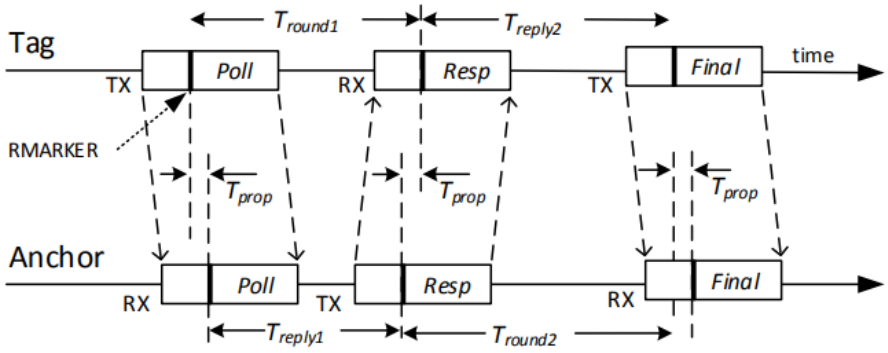
Az UWB kommunikációhoz a DecaWave által gyártott DWM1000 chipet használtam. Az iparban ez nagyon gyakori választás, ha UWB kommunikációról van szó, ráadásul kifejezetten pozícionálási célokra lett tervezve. A chipet a STM32F401RET6U típusú mikrokontroller vezérli, amely a Nucle-64 fejlesztői panelen 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.



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

3.2 A fejlesztés menete

A DWM1000 chiphez kaptam egy példaprogramot (firmware-t), amely a Nucleo kártyára volt megírva és 1 db tag 4 db anchorral történő 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 programmal kb. 80 Hz-es frekvenciával tudunk távolságot mérni 1 db tag és 1 db anchor között. A kommunikációs séma viszont nem volt túl hatékony, mivel a tag mindegy anchorral külön-külön végezte el az SDS-TWR üzenetváltást. Ennek eredményeképpen a pozícionáláshoz szükséges 4 db távolságadat kb. 20 Hz-es frekvenciával frissült.

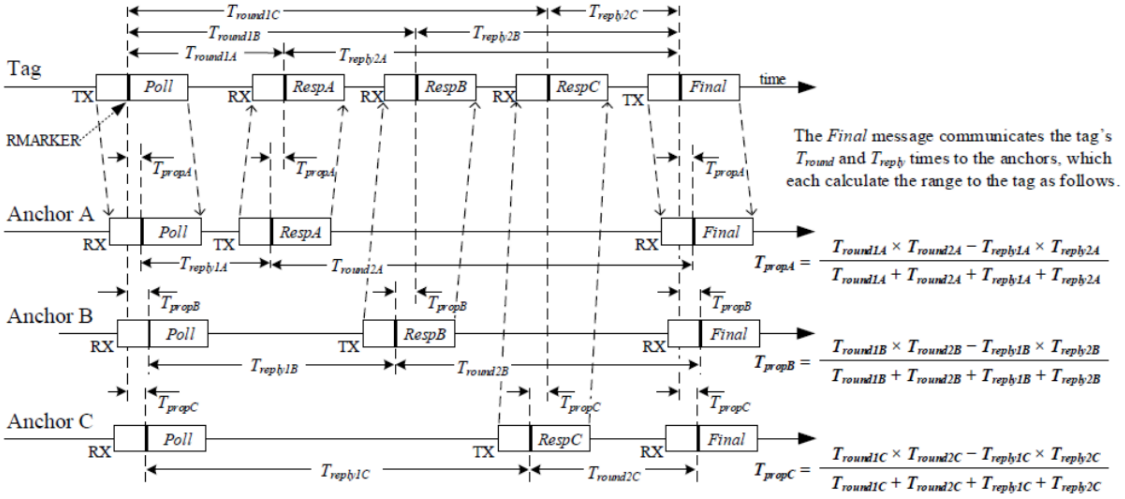


The *Final* message communicates the tag's T_{round} and T_{reply} times to the anchor, which calculates the range to the tag as follows:

$$T_{prop} = \frac{T_{round1} \times T_{round2} - T_{reply1} \times T_{reply2}}{T_{round1} + T_{round2} + T_{reply1} + T_{reply2}}$$

3.2. ábra. *Az SDS-TWR üzenetváltási séma [13]*

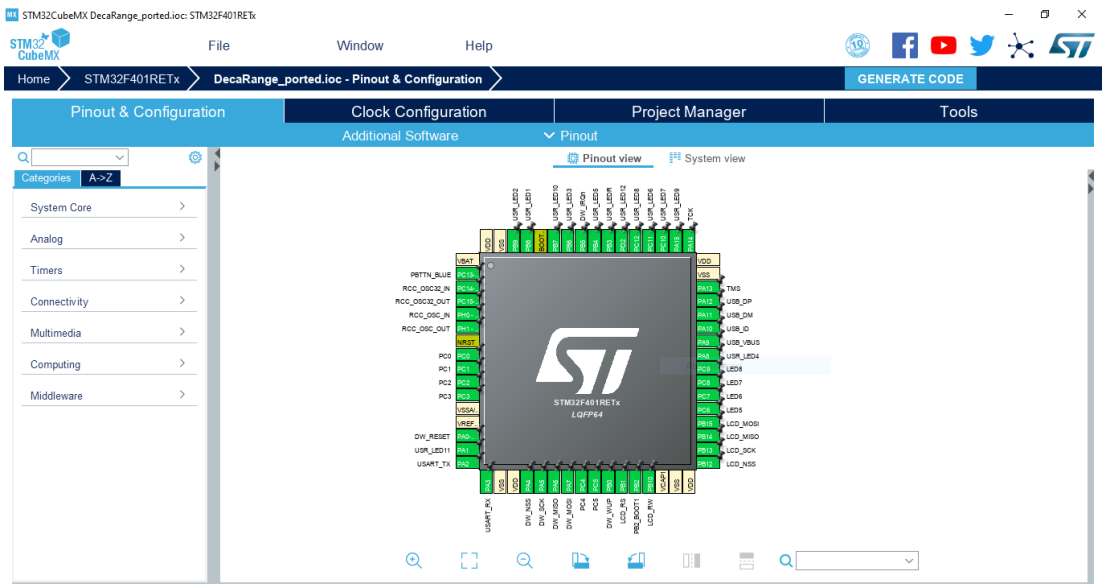
Ennél a megközelítésnél létezik egy kifinomultabb módszer is. Mivel pazarlás minden egyes anchornak külön poll és final üzenetet küldeni, ezért ezeket elég, ha csak egyszer küldjük el. A poll elküldése után, mindegyik anchort megvárjuk, hogy válaszoljon, és csak akkor küldjük el a final üzenetet, amikor már mindegyik válaszolt. Az anchoroknak van egy sorszáma, amely egyértelműen meghatározza, hogy hányadikként fog válaszolni. Mivel nem egyszerre érzékelik a poll üzenetet, ezért a választ is kicsit előbb vagy kicsit később küldik. Ez azonban sokat nem rontja el a kommunikációt, mivel elég nagy késleltetéssel küldik az anchorok a válaszokat. A javított üzenetváltási sémát láthatjuk az 3.3. ábrán.



3.3. ábra. *Az SDS-TWR üzenetváltási séma javított változata (3 anchorral) [13]*

A DecaWave honlapján [12] megtalálhatók példaprogramok, amelyek 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.

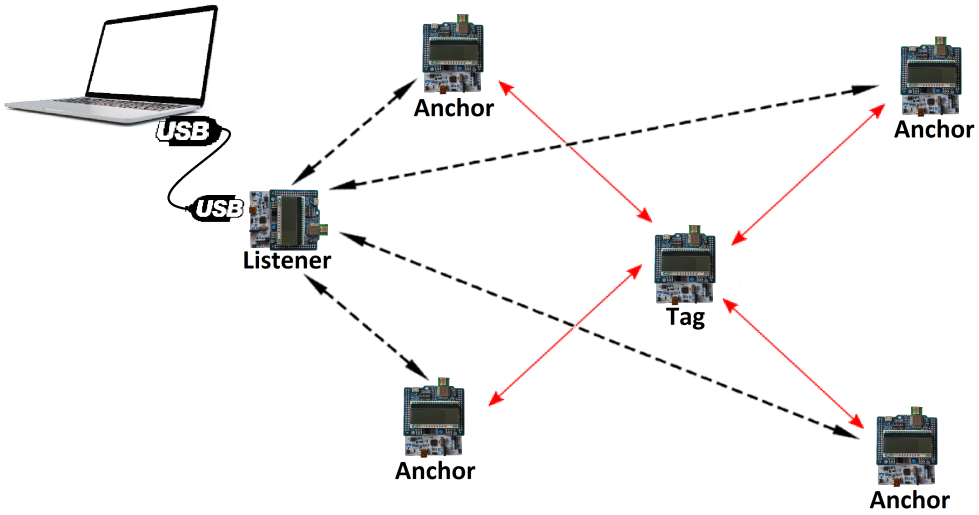
A honlapról letöltött forrás állományban egy STM32CubeMX projekt, valamint a firmware-t alkotó forrásfájlok találhatók. A lábkiosztást STM32CubeMX programban írhatjuk át a nekünk megfelelőre.



3.4. ábra. *A lábkiosztás megváltoztatása (STM32CubeMX)*

Ezután az STM32CubeMX-ben generálhatunk egy új projektet Eclipse alapú fejlesztői környezetekhez, amelyben már megváltoztathatjuk a programot, és fordíthatunk kódot a célhardverre.

Az elkészített rendszerben 4 db anchort szükséges telepítenünk fix helyre. Egy ötödik eszközzel (3.5. ábrán "Listener") - amely teljesen független, nem vesz részt a távolságmérésben - hallgathatjuk az üzenetek. Ezekből kiolvashatjuk a távolság adatokat, és továbbíthatjuk a PC felé, amelyen további feldolgozást végezhetünk.



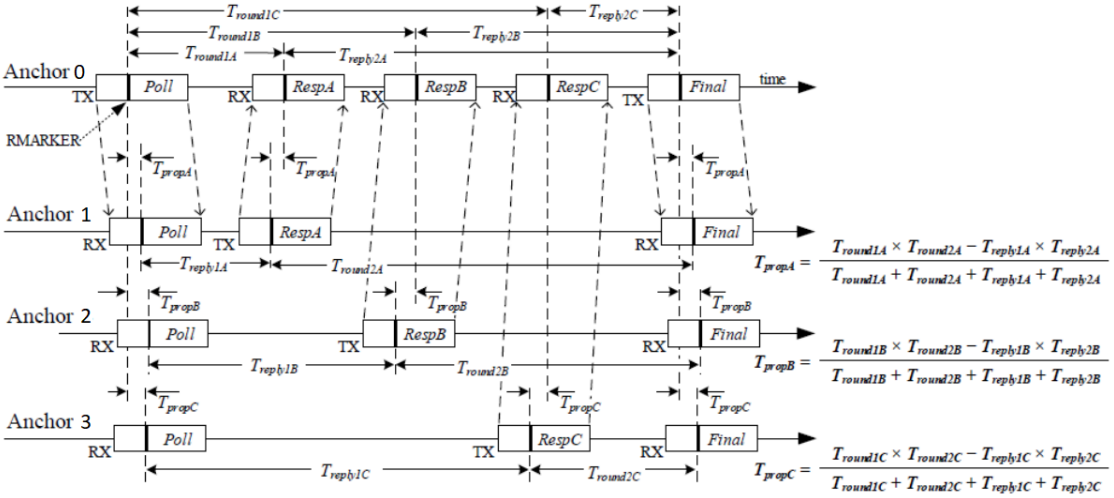
3.5. ábra. *A pozícionáló rendszer hálózati topológiája*

Az új firmware-rel kb. 150 Hz-es frissítési frekcenciát sikerült elérni, amelyben már mind a négy távolságadat egyszerre frissül.

3.3 Kezdeti kalibráció

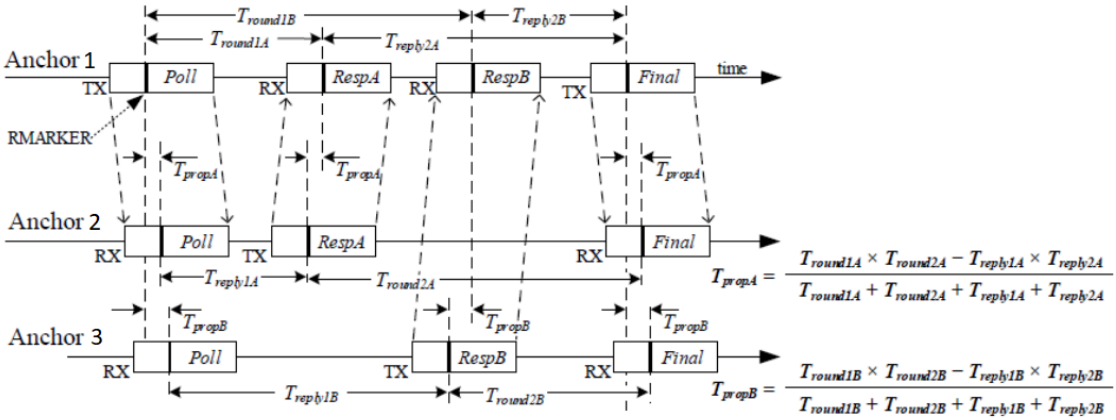
Annak érdekében, hogy pontosan tudjuk pozícionálni, szükséges, hogy az anchorok helyzete minél pontosabban ismert legyen. Hogy ne kelljen telepítéskor kézzel lemérni az anchorok helyzetét, ezért átalakíthatjuk úgy a firmware-t, hogy a rendszer bekapcsolásakor a 4 anchor közötti távolságokat meghatározza.

Első lépésként a nullás sorszámú anchor küld egy poll üzenetet, amire az anchor 1, anchor 2 és anchor 3 sorban válaszol. Ezután az anchor 0 küld egy final üzenetet, és így már meghatározható az anchor 0 távolsága a többi három anchortól.



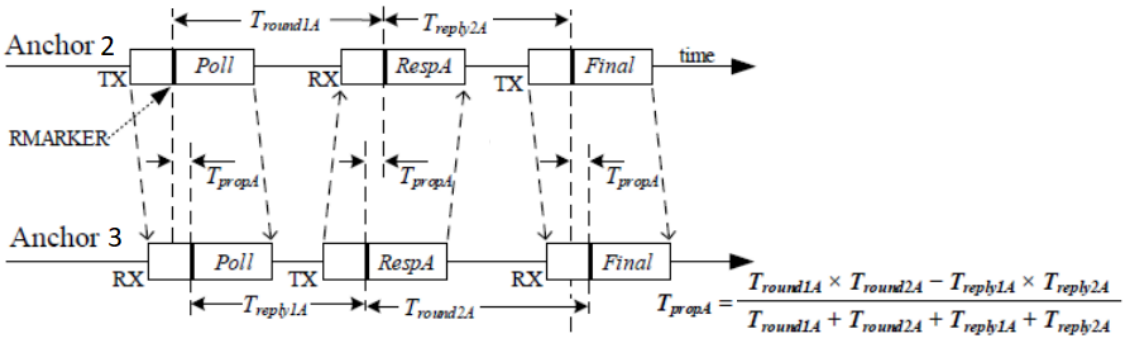
3.6. ábra. *A kezdeti kalibráció első lépés (anchor 0 mér távolságot)*

Ezután az anchor 1 küld poll üzenetet, amelyre már csak az anchor 2 és anchor 3 válaszol. Mivel az anchor 0 távolsága az anchor 1-től ugyanannyi, mint anchor 1 távolsága anchor 0-tól, ezért a kettő között nem mérünk még egyszer távolságot. Kettő válasz üzenet után az anchor 1 elküldi a final üzenetét.



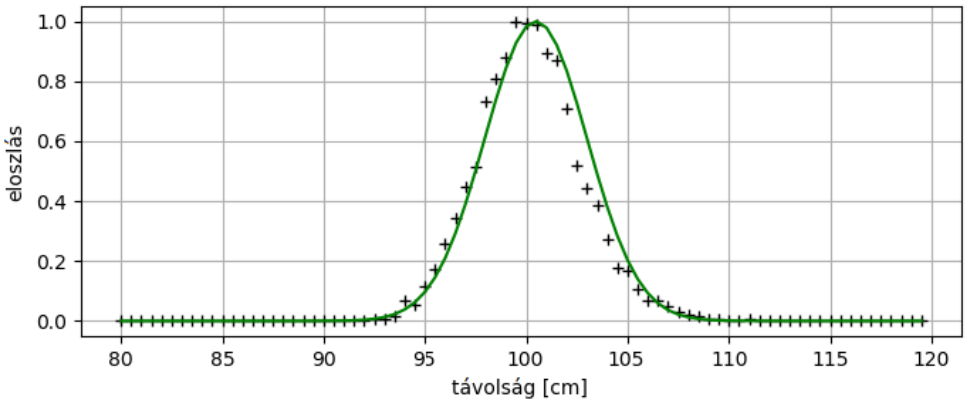
3.7. ábra. *A kezdeti kalibráció második lépés (anchor 1 mér távolságot)*

Végül az anchor 2 küld poll üzenetet, amelyre már csak az anchor 3 válaszol. Ezután a folyamat vagy leáll, vagy kezdődik elölről, hiszen az anchor 3-nak már mindegyik másik anchortól vett távolsága ekkor már ismert.



3.8. ábra. *A kezdeti kalibráció harmadik lépés (anchor 2 mér távolságot)*

A fenti módszer végrehajtása után bármely két anchor közötti távolságot meg tudjuk már határozni. Természetesen a fenti szekvenciát többször is megismételhetjük egymás után. Ez a lépés célszerű is, hiszen, ha sok mérést elvégzünk egy állandó távolságra akkor nem állandó mért értékeket fogunk kapni. Ha felosztunk egy tartományt diszkrét távolság értékekre, és számoljuk, hogy az egyes értékekből hány darabot mértünk, akkor egy normál eloszlást fogunk kapni. A 3.9. ábrán a zöld görbe egy valódi Gauss-eloszlás, amely μ várható értéke és σ szórása a mérésekből lett kiszámolva. μ értéke a valós távolság, σ-é pedig 4-5 cm.



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

Ha elegendő mérést elvégzünk, akkor a valós távolság egy egyszerű átlagolással elvégezhető.

A firmware-t úgy módosítottam, hogy bekapcsoláskor a fenti kalibrációs séma paraméterezhető darabszámszor végigjátszódjon. Implementálva volt egy hasonló eljárás a hivatalos firmware-ben, azonban nem a nekünk megfelelő módon. A tag és az anchorok között kommunikáció keretekre volt felosztva. Egy szuperkeret 10 db keretből állt. Ezek közül az első 8 a tagé volt, amelyben az anchoroktól mért távolságot (8 alkalommal). Az utolsó 2 keretben pedig az A0 mért távolságot az A1-gyel és A2-vel, valamint az A1 mért távolságot az A2-vel. Ezt kellett úgy módosítanom, hogy az A3 is részt vegyen a kalibrációban, valamint ne a tag távolságmérései között, hanem a rendszer indítása után közvetlen történjen meg ez a folyamat egy előre meghatározott darabszámszor.

4. fejezet

Drón megépítése, konfigurálása, dinamikai modellje és identifikációja

A drónaréna egyik legfontosabb része a drón. Ennek összeszerelése, felkészítése repülésre és az identifikációja is fontos lépése a rendszer megépítésének. Az ehhez kapcsolódó feladatokat ebben a fejezetben mutatom be.

4.1. ábra. *Az összeszerelt drón (propellerek nélkül)*

4.1 A fedélzeti számítógép konfigurálása

A kvadkopteren két fedélzeti számítógép is található, melyeknek más-más feladatuk van. Az egyik egység egy ún. "Flight Controller", azaz egy kisebb teljesítményű számítógép, amely az RC jelek feldolgozását és a motorok vezérlését végzi. Konkrétan a Pixhawk 1 nevű egységről van szó, amiben 180 MHz-es órajelű, 32 bites STM32F427 mikroprocesszor található. Ez az egység SBUS-on keresztül kommunikál az RC vevővel, és PWM jelekkel vezérli az ESC modulokat.

4.2. ábra. *Pixhawk 1 [27]*

A másik számítógép egy Raspberry Pi 3 Model B+. Ez már egy 64 bites 1,4 GHz-es órajelű beágyazott számítógép, amelyen a Raspbian operációs rendszer fut. Ezzel az eszközzel már számításigényesebb feladatokat is eltudunk látni, mint például autopilóta, bonyolult irányítási algoritmusok futtatását vagy akár kvadkopterre rögzíthető kamera valós idejű képfeldolgozását.

ArduPilot

Az ArduPilot egy open source autópilóta szoftver számos hasznos funkcióval, amelyet több, mint 5 éve fejlesztenek. Több különféle járművön lehet használni, például földijárműveken, kvadkoptereken, helikoptereken és tengeralattjárókban. Ha megfelelően van csatlakoztatva minden, akkor a szoftver telepítése és konfigurálása után automatikusan elvégez számos feladatot, az RC jelek feldolgozásától kezdve, a motorok vezérléséig mindent.

Az ArduPilot többféle hardveren futtatható. A teljes lista az ArduPilot [25] honlapján olvasható. Az ArduPilotot többek között a Mission Planner nevű szoftverrel

tudjuk konfigurálni.

Pixhawk konfigurálása Mission Planner szoftverrel

A Mission Planner egy olyan szoftver, amellyel könnyen tudunk konfigurálni ArduPilotot futtató eszközöket. Első lépésként a megfelelő firmware-t kell telepítenünk a Pixhawkra.

Ezután már a megfelelő szoftver fut a fedélzeti egységen, így elkezdhetjük a rendszer kalibrálását, amely két részre osztható. Az egyik a kötelező elemek, mint például a gyorsulásmérők kalibrálása, a másik pedig az opcionális elemek, mint például kamerastabilizátor (gimbal) konfigurálása. Itt most csak a kötelező elemek kalibrálását mutatom be röviden kvadkopter specifikusan, azonban egyéb típusú járműveknél is hasonló módon tudunk eljárni.

Először a váz elrendezést kell beállítanunk a gépen. Ezzel azt tudjuk megadni, hogy a gépen hány motor van, illetve, hogy a gép előrefele nézve "+" jelet vagy egy "x"-et formáljon a szárnyaival.

4.3. ábra. *A váz elrendezés kiválasztása Mission Plannerben*

Gyorsulásmérők kalibrálása

A következő lépés a gyorsulásmérők kalibrálása. Itt 6 féle orientációban kell helyezni a kvadkoptert, annak érdekében, hogy a szenzorok ofszeteit a szoftver kompenzálni tudja. A 6 orientáció eléggé magától értetődő: vízszintben, bal oldalán, jobb oldalán, orral lefelé, orral felfelé, valamint fejjel lefelé vízszintben. A kalibrációt akkor célszerű elvégezni, amikor már rögzítve van a kvadkopterhez a fedélzeti egység.

Magnetométer kalibrálása

Ezután az iránytűt (magnetométert) is hasonló módon tudjuk kalibrálni. A kalibrálás elindítása után a kvadkoptert folyamatosan forgatni kell úgy, hogy lehetőleg minden orientációt bejárjon, amivel a későbbiekben találkozhat.

ESC-k kalibrálása

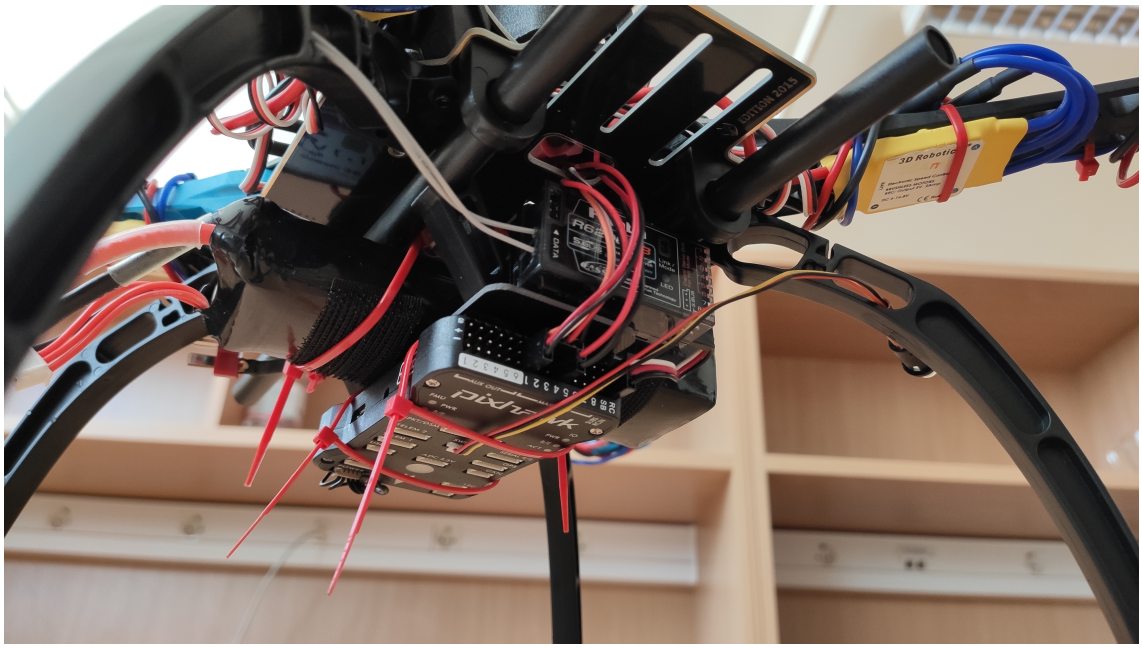
A legfontosabb az ESC-k kalibrálása, ugyanis ezek jelentős szerepet játszanak a gép mozgatásában. Ezek kalibrálása annyiból áll, hogy az ESC megtudja, hogy mekkora minimum és maximum PWM értékeket kapnak repülés közben a "Flight Controller"-től. Ennek ellenére típustól függően vannak olyan ESC-k, amelyek nem támogatják a kalibrálást és csak előírt (például 1000 és 2000 közötti) PWM értékeket fogadnak el bemenetként. Célszerű az ESC-ket a rádió kalibrálása után elvégezni. Fontos, hogy ezt a kalibrációt PROPELLEREK NÉLKÜL végezzük el az esetleges balesetek elkerülése végett.

Rádió kalibrálása

Ehhez a kalibrációhoz szükséges, hogy már előre konfiguráljuk a távirányítót (lásd 4.2 pontban). Ha ezzel megvagyunk, akkor a rádió kalibrálás során a "Flight Controller" megismerni, hogy a különböző csatornákon mekkora minimum és maximum értékeket fog kapni. Ezt úgy tudjuk elérni, hogy elindítjuk a rádió kalibrálás módot, majd a távirányítón a bemeneteket többször végkitérítésbe állítjuk.

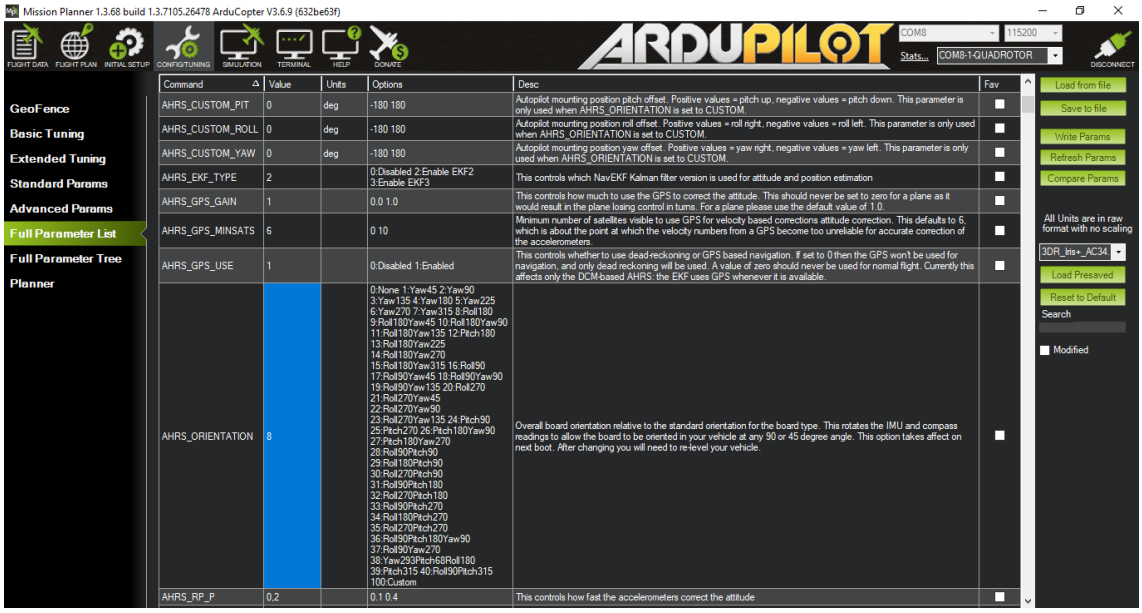
Egyéb paraméterek

Az ArduPilot firmware-ben kb. 800 paraméter van, amely közül mindegyikhez szabad hozzáférésünk van, így ezeket csak óvatosan szabad állítgatni. A legtöbb esetben, azonban az alapbeállítások meg fognak felelni mindenkinek, ugyanis a kalibráció során a paraméterek automatikusan beállítódnak. Jelen esetben 1 paraméter volt, amelyet kézzel át kellett állítani, ez pedig a "Flight Controller" orientációja. Alap esetben ugyanis vízszintesen kell elhelyezni fejjel felfelé, azonban csak fejjel lefelé sikerült elhelyezni.



4.4. ábra. *A "Flight Controller" elhelyezése fejjel lefelé*

Annak érdekében, hogy a fedélzeti egység normál helyzetben ne "érezze" úgy, hogy felborult, ezért az "AHRS orientation" paramétert ennek megfelelően kell beállítani. Ahogy a 4.5. képen látható, nagyon sok orientáció közül választhatunk. Jelen esetben a vízszintes helyzethez képest a "Flight Controller" az x tengely körül van elforgatva 180°-kal, így az ennek megfelelő 8-as (Roll180) értéket állítottam be az "AHRS orientation" paraméternek.



4.5. ábra. *A "Flight Controller" orientációjának konfigurálása*

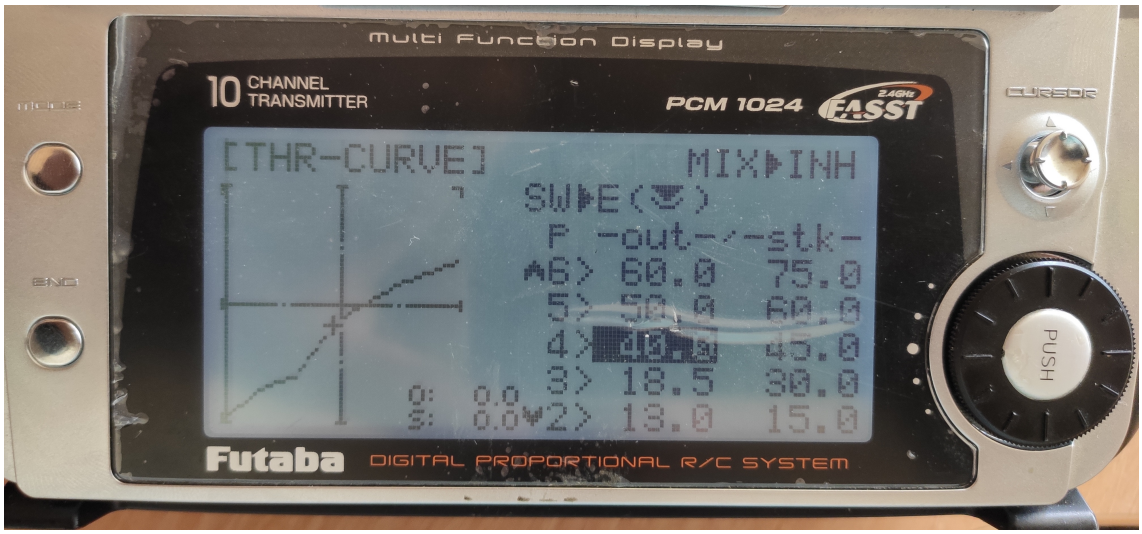
4.2 Távirányító konfigurálása

Távirányítóként a Futaba T10CP nevű eszközt használtam. Ebben a készülékben két előre definiált mód közül választhatunk, amiket utána szabadon konfigurálhatunk. Ez a két mód a repülőgép és a helikopter mód. Én az előbbit választottam, ugyanis helikopter módban a 6-os csatorna hozzá van kötve a 3-ashoz, azonban a "Flight Controller" megfelelő konfigurálása után, a 6-os csatornánk a gép érzékenységét tudjuk hangolni, ehhez viszont az kell, hogy ez a csatorna szabad legyen.

Ezen megfontolás után szabadon konfigurálhatjuk a távirányítót. A legfontosabb, hogy a 4 fő beavatkozó jel - vagyis a gázkar, csűrő-, oldal- és magassági kormány - megfelelő irányba mozogjon. Ezt Mission Plannerben könnyedén tudjuk ellenőrizni, ugyanis itt valós időben kapunk visszajelzést a bemenetekről. Fontos, hogy a gázkar, csűrő- és oldalkormánynál a joystick mozgatásával egyező, míg a magassági kormány esetében azzal ellentétes irányú elmozdulást kell látnunk.

Ezen kívül számos beállításra van lehetőségünk. Egy ilyen beállítás például a gázkar görbéje. Ezzel be tudjuk állítani, hogy a joystick lineáris mozgatása esetén milyen legyen a kimenet karakterisztikája.

A távirányítón ugyan meg van kötve, hogy melyik fizikai beavatkozó jelét, melyik csatornánk küldi ki a távirányító, azonban Mission Plannerben szabadon átkonfigurálhatjuk a "Flight Controller" működését, hogy melyik csatornát melyik beavatkozó jelként vegye figyelembe. Ehhez az "RCMAP_PITCH", "RCMAP_ROLL", "RCMAP_THROTTLE" és "RCMAP_YAW" paramétereket kell beállítanunk.



4.6. ábra. *A gázkar görbéjének beállítása*

4.3 Raspberry Pi konfigurálása

A Pixhawk és a Raspberry soros kommunikációs csatornán keresztül tudnak kommunikálni egymással. Ehhez használhatjuk a Pixhawk 2-es telemetria portját, vagy micro USB-n keresztül is hozzáköthetjük a Raspberry-hez. Az ArduPilot honlapján egy részletes útmutató [26] található, hogy hogyan tudjuk a két eszközt konfigurálni, hogy egymással tudjanak kommunikálni. A főbb lépések:

- a soros kommunikáció paramétereinek beállítása a Pixhawkon,
- Raspbian telepítése SD kártyára a Raspberry-nek,
- a szükséges csomagok telepítése Raspberry-re (Python, Pymavlink, Mavproxy, stb.),
- az operációs rendszer irányításának letiltása a soros port felett a Raspberry-n.

Ezek után a telepített csomagokkal egy szkriptet is találhatunk a fájlok között "mavproxy.py" néven, amellyel a megfelelő kapcsolók[1] használata esetén ellenőrizhetjük, hogy helyesen működik-e a kommunikáció.

Hasznos lehet még telepíteni a DroneKit nevű csomagot. Ezzel a könyvtárral Python nyelven írhatunk egyszerű programokat, amellyel a Pixhawk állapotait tudjuk lekérdezni, illetve megfelelő beállítás után irányítani is.

4.4 A kvadkopter dinamikai modellje

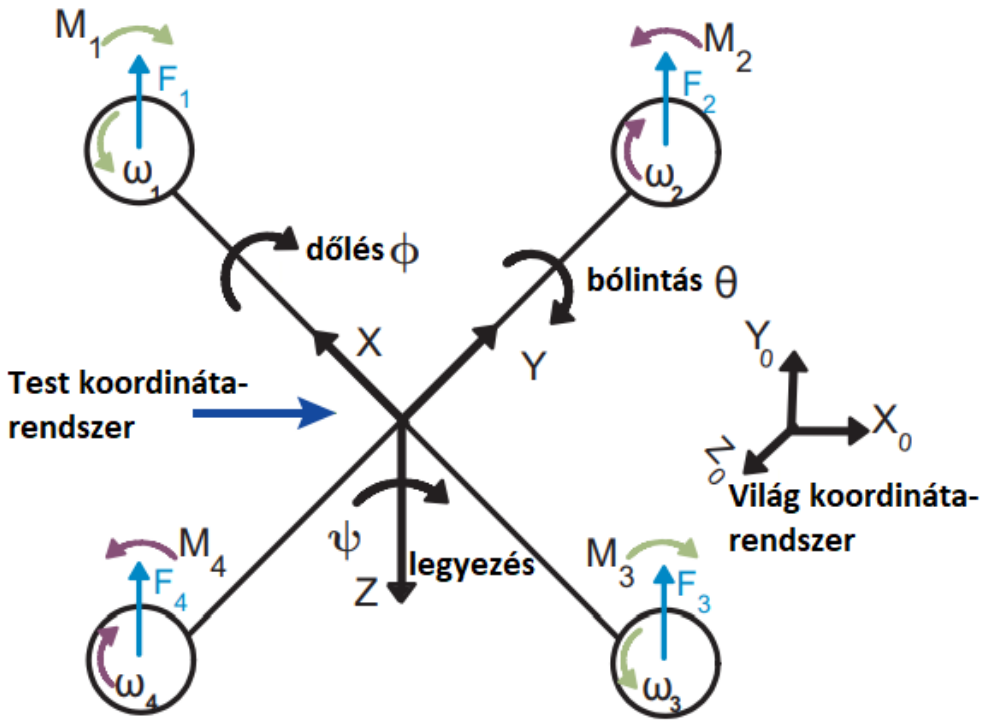
A drón modellezéséhez ismernünk kell az állapotváltozós leírását. Ehhez 12 állapotváltozó szükséges. Ebből p_x, p_y és p_z ([m]) a pozíciót, ϕ, θ és ψ ([rad]) az orientációt, v_x, v_y és v_z ([m/s]) a sebességet és p, q és r ([rad/s]) a szögsebességet adja meg.

A pozíció, a sebességek és a szögsebességek a világ koordináta-rendszerben vannak felírva. Az orientáció szögei a test koordináta-rendszer és a világ (földi) koordináta-rendszer közötti transzformációt írja le.

A pozíció változását a közvetlenül a sebességek adják meg:

$$\begin{pmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{p_z} \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \tag{4.1}$$

[1] A kapcsolók olyan parancssori elemek, amelyekkel extra paramétereket adhatunk a futtatandó programnak.



4.7. ábra. *A kvadkopter [23]*

A gyorsulást a drónra ható erő és az orientáció befolyásolja:

$$\begin{pmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{pmatrix} = \begin{pmatrix} \frac{F}{m} \cdot (cos(\phi) \cdot sin(\theta) \cdot cos(\psi) + sin(\phi) \cdot sin(\psi)) \\ \frac{F}{m} \cdot (cos(\phi) \cdot sin(\theta) \cdot sin(\psi) - sin(\phi) \cdot cos(\psi)) \\ \frac{F}{m} \cdot cos(\phi) \cdot cos(\theta) - g \end{pmatrix}, \tag{4.2}$$

ahol F a rotorok által keltett függőleges irányú erő, m pedig a kvadkopter tömege.

Az orientáció változás a szögsebességekkel van kapcsolatban

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & sin(\phi) \cdot tan(\theta) & cos(\phi) \cdot tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) \cdot sec(\theta) & cos(\phi) \cdot sec(\theta) \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}, \tag{4.3}$$

míg a szöggyorsulások a szögsebességektől és a forgatónyomatékoktól függenek:

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \frac{J_y - J_z}{J_x} \cdot q \cdot r \\ \frac{J_z - J_x}{J_y} \cdot p \cdot r \\ \frac{J_x - J_y}{J_z} \cdot p \cdot q \end{pmatrix} + \begin{pmatrix} \frac{L}{J_x} \\ \frac{M}{J_y} \\ \frac{N}{J_z} \end{pmatrix}, \tag{4.4}$$

ahol J_x, J_y és J_z az inerciamátrix főátlóbeli elemei, J_{act} a propeller tehetetlenségi nyomatéka, Ω a propeller szögsebességeinek előjeles összege $(-\omega_1 + \omega_2 - \omega_3 + \omega_4)$, L, M és N pedig a kvadkopterre ható forgatónyomatékok.

$$\begin{pmatrix} F \\ L \\ M \\ N \end{pmatrix} = \begin{pmatrix} b \cdot (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ b \cdot l \cdot (-\omega_2^2 + \omega_4^2) \\ b \cdot l \cdot (-\omega_1^2 + \omega_3^2) \\ d \cdot (-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{pmatrix}, \tag{4.5}$$

egyenletekkel számolhatók ki, ahol l a kvadkopter középpontja és a propellerek középpontja közötti távolság, b a tolóerő-együttható (thrust force coefficient) és d az ellenálláserő-együttható (drag force coefficient). [23] [2]

4.5 Az identifikáció menete

Identifikációt többféle módon végezhetünk. Egy inkább gépészmérnökibb megközelítés, hogy minden paramétert megfelelő méréssel számítunk ki. A [17] cikkben olvasható több módszer is a különböző paraméterek kimérésére. Az inercia paraméterek meghatározásához bifilárisan felfüggesztve a kvadkoptert, a lengés periódus idejéből tudjuk kiszámolni az egyes paramétereket. A motor paraméterek számításához tachométert (fordulatszámmérőt) használhatunk, amivel ki lehet mérni, hogy mi az összefüggés a motorra adott feszültég és a kialakuló fordulatszám között.

Egy másik megközelítés a szürkedobozos modell identifikálás. Ennél a módszernél a kvadkopter mozgásegyenleteit írjuk fel, amelyeket az előző fejezetben ismertettem. Az egyenletekben szereplő paraméterek határozzák meg, hogy adott bemenetekre hogyan viselkedik a modell. A szürkedobozos identifikálás lényege, hogy valós mérések alapján próbáljuk úgy hangolni a paramétereket, hogy minél jobban hasonlítson a modell kimenete (állapotváltozói) a valós mérésekre.

A manőverek és az identifikálás menete

A mozgásegyenletekből látható, hogy 7 paraméter van, amelyeket meg kell határozni. Ezek az m tömeg, J_x, J_y J_z inerciamátrix elemei, b tolóerő-együttható, d ellenálláserő-együttható, valamint l, amely a kvadkopter középpontja és a propellerek középpontja közötti távolság. Ezek közül a tömeg és az l paraméterek kézi méréssel is meghatározhatók.

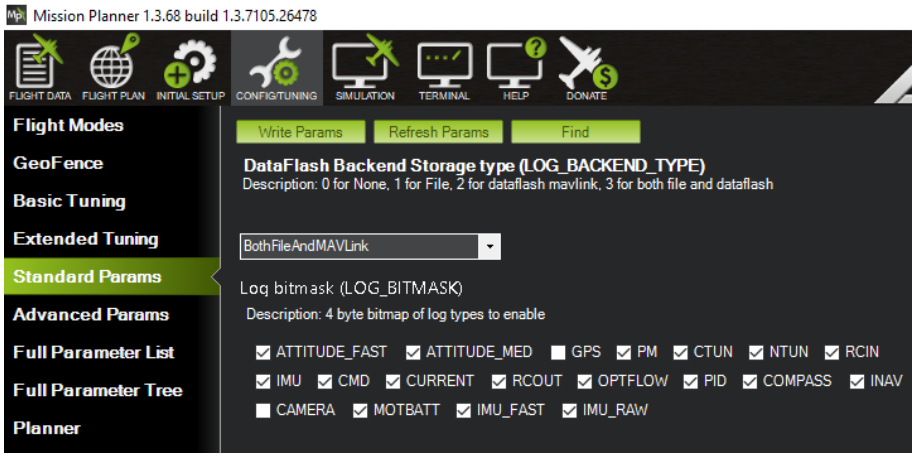
A 4.4 fejezetben található egyenletekből látszik, hogy lebegés esetén $\dot{v}_z = 0$, azaz a tömeg és a lebegés közbeni propeller szögsebeségek ismeretében a b tolóerő-együttható kiszámítható.

A többi paraméter meghatározásához valamilyen szöghelyzetbe kell kitéríteni a gépet, hogy az adott bemenetnek legyen érzékelhető hatása valamelyik állapotváltozón. Például, ha csupán x irányban mozgatjuk a gépet, akkor a *pitch* szög (θ) fog változni, amit ebben a mozgásban csak q befolyásol. q változása pedig a bemenettől és J_y paramétertől függ:

$$\dot{q} = \frac{M}{J_y} = \frac{b \cdot l \cdot (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)}{J_y} \tag{4.6}$$

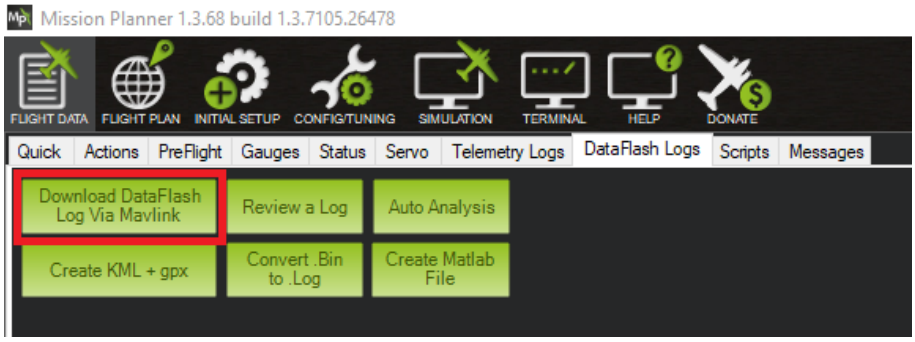
Szenzor adatok begyűjtése

Ahhoz, hogy identifikálni tudjunk, szükségünk van repülés közben loggolni a szenzor adatokat. Ezt megfelelő konfigurálás után az ArduPilot elvégzi helyettünk.



4.8. ábra. *Loggolás beállítása Mission Plannerben*

A 4.8. ábrán látható módon Mission Plannerben könnyen konfigurálhatjuk, hogy milyen adatokat loggoljon a szoftver. Ezután, ha repültünk a géppel, a log fájlok a Pixhawkon találhatók meg, amelyeket Mission Plannerben egyszerűen le tudunk tölteni, és Matlab fájlt is készíthetünk, amely remek lehetőség, hogy gyorsítsuk az adatkonverziót a megfelelő formátumra.



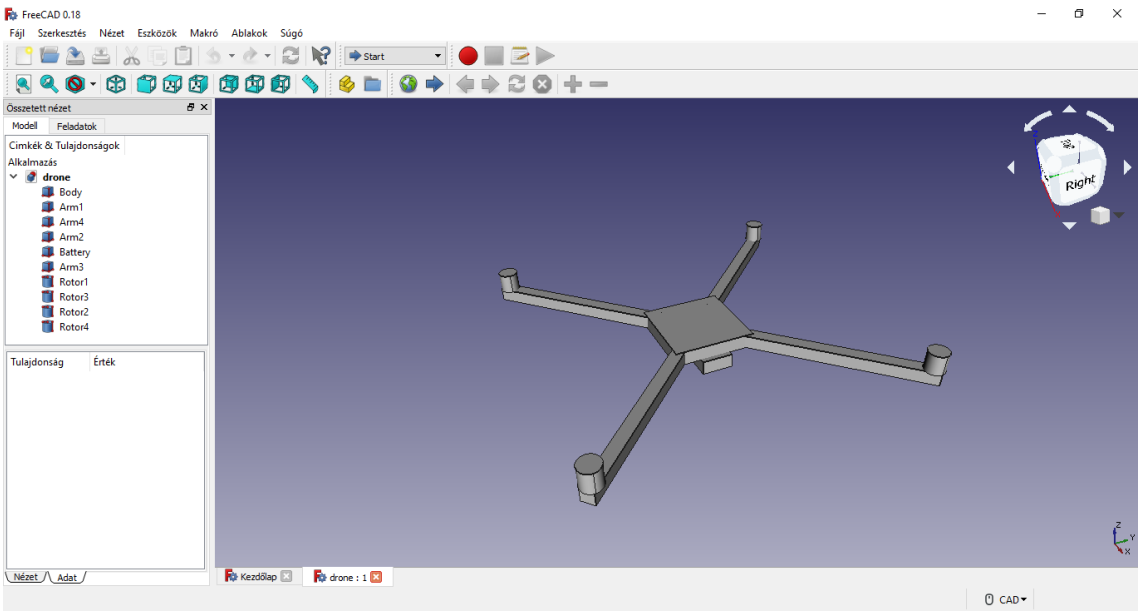
4.9. ábra. *A log fájlok letöltése Mission Plannerben*

Az elkészült fájlt már könnyedén tudjuk Matlabban importálni, és további feldolgozást végezni vele, valamint felhasználni a szürkedobozos identifikáció során.

A kezdő paraméterek megválasztása

Az identifikáció végeredménye azonban sok esetben jelentősen függ a kiinduló paraméterektől. Ha véletlenszerűen inicalizáljuk a kezdeti paramétereket, akkor valamennyire az identifikált paraméterek is véltlenszerűek lesznek. Ennek elkerülése végett számíthatunk nagyságrendileg helyes paramétereket.

Ehhez használhatunk olyan CAD szoftvert, amely tud inerciát számolni.



4.10. ábra. *A kvadkopter modellezése CAD szoftverrel*

Egy vázlatos modellt használva a 4.10. ábrán látható eredményhez juthatunk. Modellezni külön érdemes mind a lábakat, a szárnyakat és magát a testet. Az egyes részeket durva közelítéssel hengernek és téglatestnek lehet venni. A különálló részeket homogénnek feltételezhetjük, és grammos mérleggel meghatározhatjuk a valóságban

a tömegét. Ennyi információval el is készíthetjük a CAD modellt, majd a szoftver automatikusan kiszámolja nekünk az inerciamátrix elemeit.

Ha nem áll rendelkezésünkre ilyen szoftver, akkor kézzel is kiszámíthatjuk az inerciamátrixbeli értékeket. Ehhez a Steiner-tételt (angolszász szakirodalomban Huygens-Steiner-tétel) használhatjuk.

A meghatározni kívánt mátrix a következő:

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz}, \end{bmatrix}$$

ahol

$$I_{xx} = \sum_{i=1}^{N} m_i(y_i^2 + z_i^2)$$

$$I_{yy} = \sum_{i=1}^{N} m_i(x_i^2 + z_i^2)$$

$$I_{zz} = \sum_{i=1}^{N} m_i(x_i^2 + y_i^2)$$

$$I_{xy} = I_{yx} = -\sum_{i=1}^{N} m_i x_i y_i$$

$$I_{xz} = I_{zx} = -\sum_{i=1}^{N} m_i x_i z_i$$

$$I_{yz} = I_{zy} = -\sum_{i=1}^{N} m_i y_i z_i$$

m_i az egyes tömegpontokat jelöli, x_i, y_i és z_i pedig a tömegpont koordinátáit.

Az xy síkra forgásszimmetrikusnak tekinthetjük a gépet, így $I_{xy} = I_{yx} = 0$ értékeket feltételezhetünk, illetve a többi nem főátlóbeli elem is elég kicsit ahhoz, hogy 0-nak feltételezzük őket. Mivel most még csak kiinduló paramétereket keresünk, ezek a feltételezések nem jelentenek problémát, hiszen majd az identifikáció során megkapjuk majd a helyes értékeket.

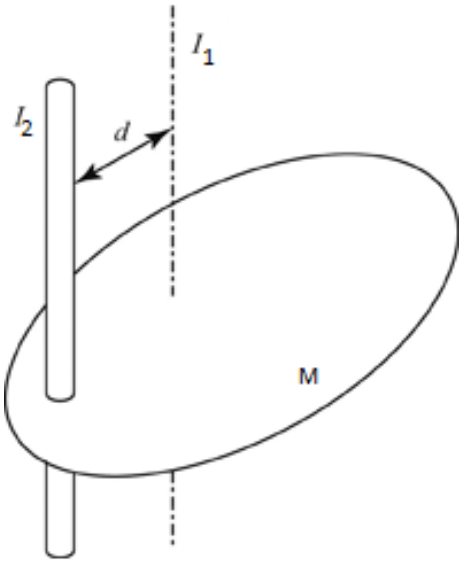
A többi paraméter a Steiner-tétellel számítható.

$$I_2 = I_1 + Md^2,$$

ahol I_1 a merev test tömegközéppontján áthaladó tengelyre vett tehetetlenségi nyomatéka, I_2 a test tehetetlenségi nyomatéka egy az I_1-hez tartozó tengellyel párhuzamos tengelyre, M a merev test tömege, d pedig a két tengely távolsága.

A Steiner-tétel használatához mindegyik kvadkopter részéhez ki kell számolni a

tömegközéppontján áthaladó 3 tengelyre vett tehetetlenségi nyomatékát. Ezeket a tengelyeket pedig úgy kell felvenni, hogy párhuzamosak legyenek azzal a 3 tengellyel, amelyre meg szeretnénk határozni az egész gép tehetetlenségi nyomatékát.



4.11. ábra. *A Steiner-tétel szemléltetése [3]*

Ezt elvégezve már használhatjuk a Steiner-tételt, és az egyes részelemek tehetetlenségi nyomatékát áthelyezhetjük a megfelelő tengelyre. A kiszámolt nyomatékokat pedig tengelyenként külön kell összegezni, és így megkaphatjuk I_{xx}, I_{yy} és I_{zz} értékeket.

További ismeretlen paraméter még a b tolóerő-együttható és a d ellenálláserő-együttható. Ezekhez nagyságrendileg megfelelő paramétereket a szakirodalomban [18] található eredményekből kereshetünk.

A meghatározott kezdő paraméterek:

Paraméter	Érték
m	1,271 [kg]
I_{xx}	0,0282 [kgm^2]
I_{yy}	0,0285 [kgm^2]
I_{zz}	0,0561 [kgm^2]
b	$1{,}14{\cdot}10^{-6}$ [Ns^2]
d	$2.98{\cdot}10^{-5}$ [Nms^2]

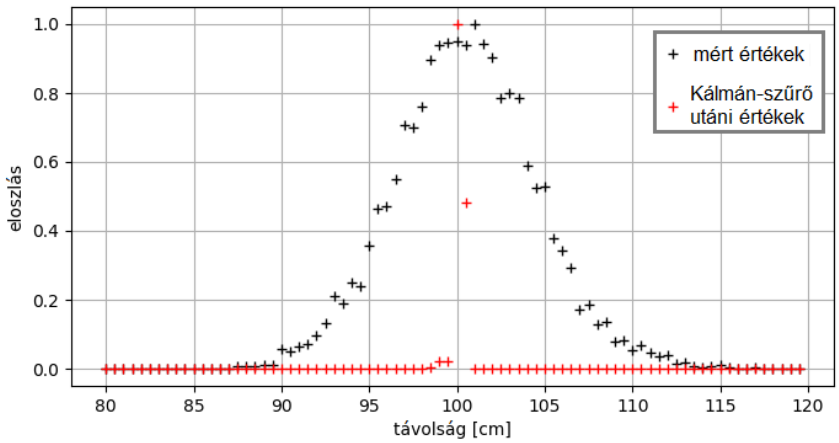
5. fejezet

Az állapotbecslő

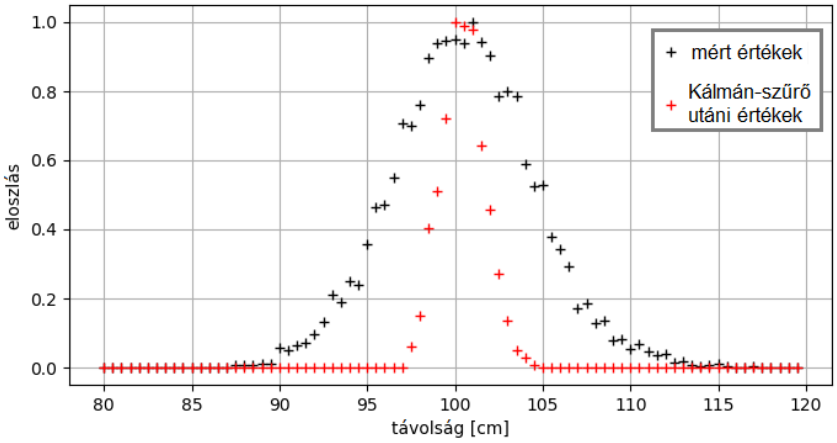
A 2. fejezetben megismert pozícionáló rendszert és a 4.4. fejezetben bemutatott dinamikai modellt együttesen felhasználva egy pontosabb becslést kaphatunk a kvadkopter állapotáról. Míg a pozícionáló rendszerrel csak pozíciót mérhetünk, addig a dinamikai modell segítségével egyéb állapotokat is mérhetünk. Utóbbi viszont érzékeny a modell pontosságára, valamint az ofszett típusú hibákra, ugyanis ezek a numerikus integrálás során összeadódnak és egyre növekvő hibát okoznak. Ennek elkerülése végett használhatjuk a pozícionáló rendszer méréseit, amellyel korrigálni tudjuk a dinamikai modell kimenetét.

A dinamikai modell és a mérések együttes használatára kézenfekvő megoldás a Kálmán-szűrő. Ezzel a szűrővel optimális becslést kaphatunk egy rendszer állapotára a mért adatokból.

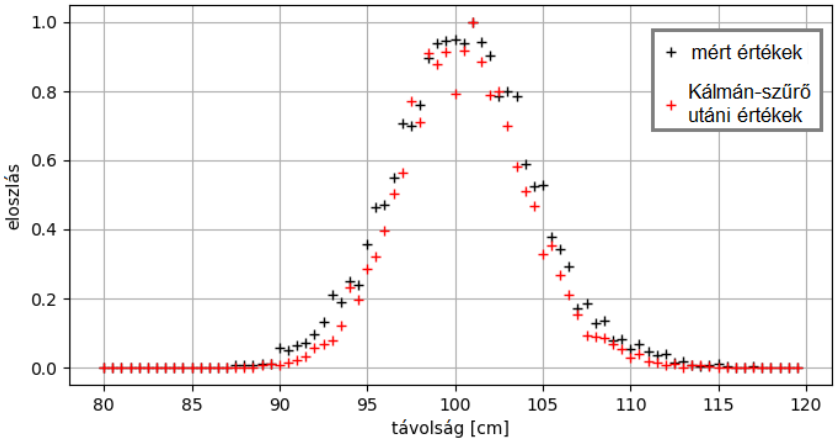
Először vizsgáljuk meg, hogy egy mért adat javítására hogyan használhatjuk. A példában egy konstans távolságot szeretnénk becsülni. A Kálmán-szűrőben, ahogy azt majd látni fogjuk, van két olyan paraméter (Q és R), amelyek megválasztása kiemelten fontos. Ezen paraméterek a szűrőre gyakorolt hatását a következő ábrák szemléltetik.



5.1. ábra. *Kálmán-szűrő hatása,* $R = 16\ cm^2$ *és* $Q = 0\ cm^2$



5.2. ábra. *Kálmán-szűrő hatása,* $R = 16\ cm^2$ *és* $Q = 0,1\ cm^2$



5.3. ábra. *Kálmán-szűrő hatása,* $R = 16\ cm^2$ *és* $Q = 100\ cm^2$

Látható, hogy megfelelő paraméter megválasztása után a Kálmán-szűrő jelentős mértékben lecsökkenti a mérés szórását. Ennek azonban hátránya is van, ugyanis minél jobban bekorlátozzuk a kimenet szórását, annál kevésbé tudja lekövetni a szűrő a gyorsan változó jeleket.

A becsülni kívánt mennyiségek azonban egy drón állapotai. Ezekre azonban dinamikai modellt is felállíthatunk, amellyel még pontosabbá tehetjük a becslést. Ez a dinamikai modell viszont nemlineáris, ezért a kiterjesztett Kálmán-szűrőt (Extended Kalman Filter - EKF) kell alkalmaznunk. Ennek a megoldásnak a validálását szimulációs környezetben végeztem el.

Jelen fejezetben az kiterjesztett Kálmán-szűrőt és a szimuláció felépítését mutatom be, illetve mindezek előtt a pozícionáló rendszer validálását mutatom be valós környezetben.

5.1 A pozícionáló rendszer valós környezetben történő validálása

Mielőtt felhasználnánk a dinamikai modellt és az állapotbecslőt, a nyers távolságadatokat felhasználva a 2.2 fejezetben megismert numerikus módszerrel valós környezetben is kipróbálhatjuk a rendszert. Szimulációban pontosan ismernénk a tag helyzetét, ugyanis mi mondjuk meg, hogy hol legyen. Valós esetben 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.

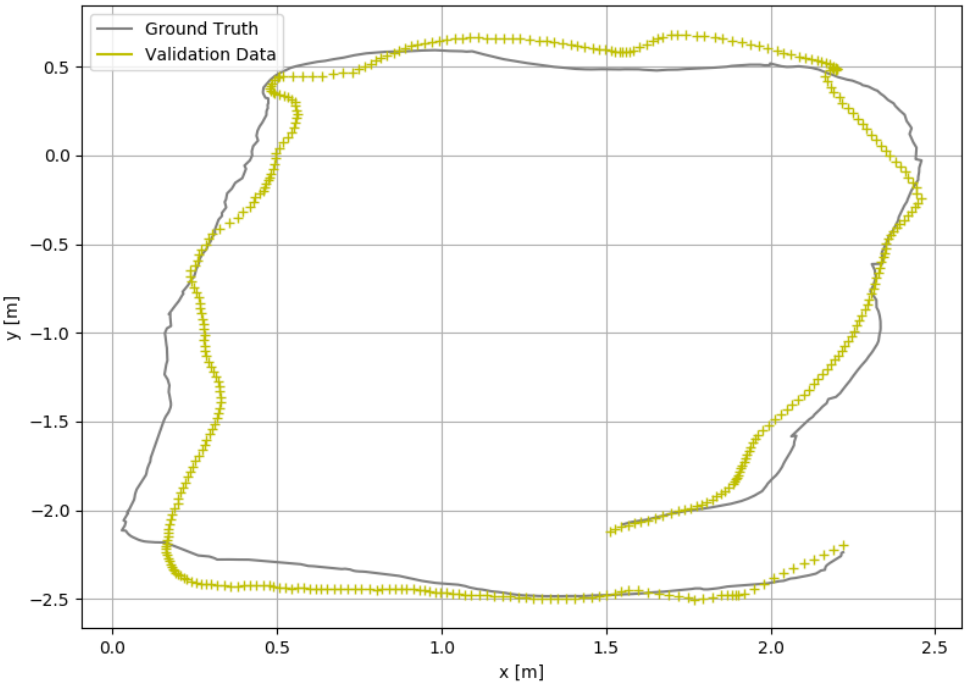
A validálás során egy optikai rendszer volt a referencia. Ez 12 infrakamera segítségével mm pontossággal képes meghatározni a pozíciót ún. markerek segítségével, amelyeket a tagen kell elhelyezni. A rendszer neve MTA SZTAKI MIMO (MIcro aerial vehicle and MOtion capture) arena, azaz MTA SZTAKI mikró repülőgép 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.

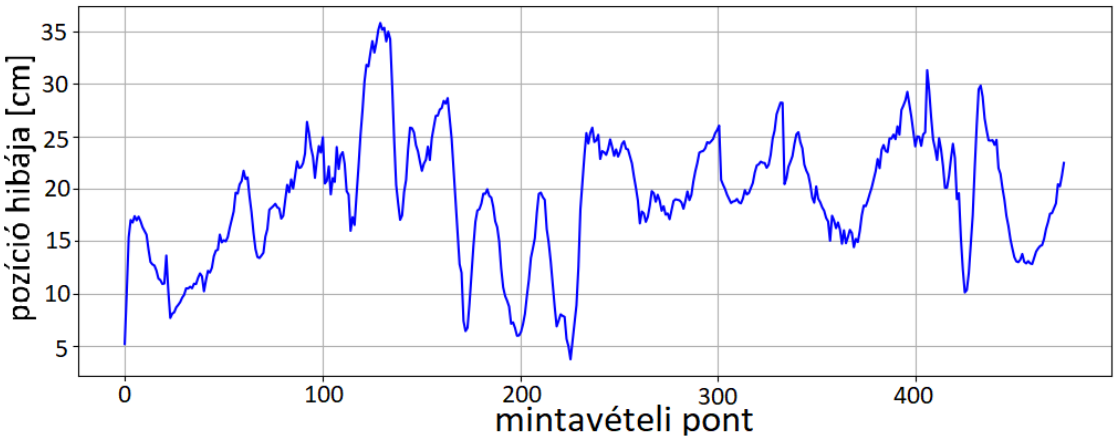
A két mérési sorozatot több szempont szerint is szinkronizálni kellett. Az első szempont az volt, hogy időben szinkronban legyenek az adatok. Ez egyrészt azt jelenti, hogy a két listából 1-1 elem kiolvasása ugyanakkora időlépést jelentsen, másrészt, hogy ugyanabban az időpontban kezdődjön a két adatsor. 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 5.5. 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.



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

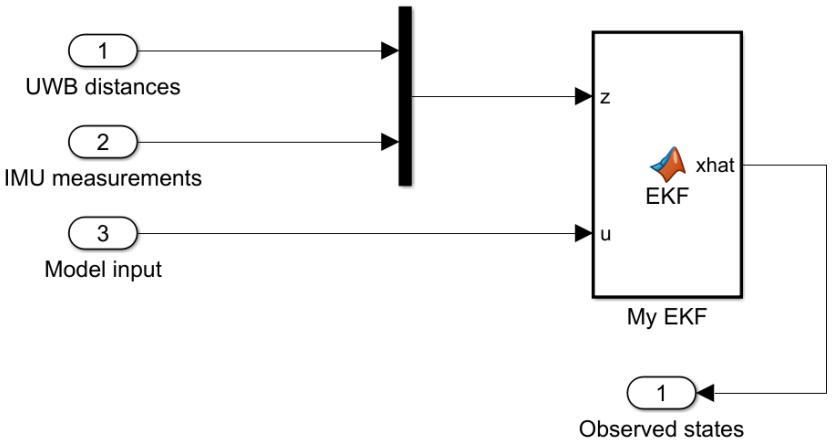


5.5. ábra. *A pozíció számítás hibája a mért adatsoron*

Az állapotbecslő feladata, hogy ezt a pontosságot javítsa.

5.2 A kiterjesztett Kálmán-szűrő működése

Az állapotbecslőt egy EKF valósítja meg. Ebben megtalálható a kvadkopter nemlineáris modellje, amely segítségével és a drónra adott bemenetek felhasználásával frissíthetjük a drón állapotait. Ezután az UWB-s pozícionáló rendszer távolságadatait és az IMU méréseit felhasználva pontosítjuk azokat.



5.6. ábra. *Az állapotbecslő*

Az EKF azért szükséges, mert ezzel egy dinamikai rendszert is megadhatunk, amellyel becsülni lehet a drón mozgását. Ez azért célszerű, mert sok állapotváltozót közvetlenül nem tudunk mérni, egy dinamikai modellel azonban képesek vagyunk kiszámolni azokat. A drón modellje pedig nemlineáris (ld. 4.4. fejezet), ezért van szükségünk a kiterjesztett Kálmán-szűrőre. Egy másik előnye az EKF-nek, hogy több szenzorból származó mérést is képes egyszerre felhasználni az állapotbecsléshez.

A kiterjesztett Kálmán-szűrő elmélete

A megfigyelni kívánt szakasz nemlineáris és a következő differenciaegyenlettel adható meg:

$$x_k = f_k(x_{k-1}, u_{k-1}, w_{k-1}), \tag{5.1}$$

valamint a méréseket és az állapotváltozókat a

$$z_k = h(x_k, v_{k-1}) \tag{5.2}$$

egyenlet köti össze. Az EKF első lépésként kiszámolja ezeknek a nemlineáris differenciaegyenleteknek az elsőfokú (lineáris) közelítéséhez szükséges mátrixokat

$$x_k \approx \widetilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + Ww_{k-1} \tag{5.3}$$

$$z_k \approx \widetilde{z}_k + H(x_k - \widetilde{x}_k) + Vv_k, \tag{5.4}$$

ahol

$$\widetilde{x}_k = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{5.5}$$

$$\widetilde{z}_k = h(\widetilde{x}_k, 0) \tag{5.6}$$

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_{k-1}, 0) \tag{5.7}$$

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_{k-1}, u_{k-1}, 0) \tag{5.8}$$

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\widetilde{x}_{k-1}, 0) \tag{5.9}$$

$$V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\widetilde{x}_{k-1}, 0) \tag{5.10}$$

Ezután megadható az EKF működése, amely két részre bontható. Az első fázis az ún. *time update*:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{5.11}$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_k W_k^T. \tag{5.12}$$

A második fázis az ún. *measurement update*:

$$K_k = P_k^- H^T (H P_k^- H^T + V_k R V_k^T)^{-1} \tag{5.13}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-)) \tag{5.14}$$

$$P_k = (I - K_k H) P_k^-, \tag{5.15}$$

ahol P_k^- az *a priori* becslés hibájának kovariancia mátrixa, P_k pedig az *a posteriori* becslés hibájának kovariancia mátrixa. Két fontos paramétere a Kálmán-szűrőnek a Q és az R. A Q a folyamat zaj kovariancia mátrixa, ami azt adja meg, hogy mennyire pontos a modellünk. Az R a mérések hibájának kovariancia mátrixa. A K súlyozó mátrix - amelyet gyakran szokás Kálmán-erősítésnek hívni - a következő képlettel számolható:

$$K_k = \frac{P_k^- H^T}{H P_k^- H^T + V_k R V_k^T.} \tag{5.16}$$

Látható tehát, hogy az R mátrix közvetlen hatással van a Kálmán-erősítésre. Ha a mérési zaj közelít a nullához, azaz

$$\lim_{R \to 0} K_k = \frac{1}{H}, \tag{5.17}$$

akkor szűrő egyre jobban fog "bízni" a mérésekben és nem fogja figyelembe venni a modellt, mert nagyon pontos méréseink vannak. Továbbá, ha az *a priori* becslés hibájának kovariancia mátrixa ($Pkminus$) közelíti meg a nullát, azaz

$$\lim_{P_k^- \to 0} K_k = 0, \tag{5.18}$$

akkor a szűrő a méréseket fogja figyelmen kívül hagyni, ugyanis a modellünk ad nagyon pontos becslést a rendszer állapotáról.

A fentiek alapján Q és R paraméter meghatározására nagy hangsúlyt kell fektetnünk, hogy megfelelő eredményt kapjunk. Szimulációs környezetben, mivel a folyamat zajt és a mérési zajt is mi állítjuk elő, ezért a Q az R mátrix számítható. A valóságban azonban R mátrixot a valós méréseink alapján kell meghatározni, a Q mátrixot pedig a modellünk pontossága alapján. [28] [10]

A folytonos idejű modell használata a Kálmán-szűrőben

A dinamikai modellünk folytonos időben van felírva, a Kálmán-szűrő azonban diszkrét időben végez számításokat. Ezért diszkretizálni kell a modellt, amit én az előrelépő Euler-módszert oldottam meg. Tehát az állapotok frissítése az

$$x_{k+1} \approx x_k + T_s \cdot f(x_k, u_k) \tag{5.19}$$

egyenlet alapján történik, ahol T_s a mintavételi idő, melynek értékét $0,025$ s-nak választottam.

A meghatározott egyenletek

Láttuk, hogy a $h()$ függvény a mérések és az állapotváltozók között teremt kapcsolatot. Jelen esetben ezt a következőképpen írhatjuk fel:

$$\begin{pmatrix} h_1(p_x,p_y,p_z) \\ h_2(p_x,p_y,p_z) \\ h_3(p_x,p_y,p_z) \\ h_4(p_x,p_y,p_z) \\ h_5(p) \\ h_6(q) \\ h_7(r) \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} = \begin{pmatrix} \sqrt{(p_x-x_1)^2+(p_y-y_1)^2+(p_z-z_1)^2} \\ \sqrt{(p_x-x_2)^2+(p_y-y_2)^2+(p_z-z_2)^2} \\ \sqrt{(p_x-x_3)^2+(p_y-y_3)^2+(p_z-z_3)^2} \\ \sqrt{(p_x-x_4)^2+(p_y-y_4)^2+(p_z-z_4)^2} \\ p \\ q \\ r \end{pmatrix}, \tag{5.20}$$

ahol (x_i, y_i, z_i) az i. anchor pozíciója. Ezek alapján a $h()$ függvény Jacobi-mátrixa:

$$H = \begin{pmatrix} \frac{\partial h_1}{\partial p_x} & \frac{\partial h_1}{\partial p_y} & \frac{\partial h_1}{\partial p_z} & 0 & \dots & 0 & 0 & 0 & 0 \\ \frac{\partial h_2}{\partial p_x} & \frac{\partial h_2}{\partial p_y} & \frac{\partial h_2}{\partial p_z} & 0 & \dots & 0 & 0 & 0 & 0 \\ \frac{\partial h_3}{\partial p_x} & \frac{\partial h_3}{\partial p_y} & \frac{\partial h_3}{\partial p_z} & 0 & \dots & 0 & 0 & 0 & 0 \\ \frac{\partial h_4}{\partial p_x} & \frac{\partial h_4}{\partial p_y} & \frac{\partial h_4}{\partial p_z} & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & \frac{\partial h_5}{\partial \omega_1} & \frac{\partial h_5}{\partial \omega_2} & \frac{\partial h_5}{\partial \omega_3} \\ 0 & 0 & 0 & 0 & \dots & 0 & \frac{\partial h_6}{\partial \omega_1} & \frac{\partial h_6}{\partial \omega_2} & \frac{\partial h_5}{\partial \omega_3} \\ 0 & 0 & 0 & 0 & \dots & 0 & \frac{\partial h_7}{\partial \omega_1} & \frac{\partial h_7}{\partial \omega_2} & \frac{\partial h_7}{\partial \omega_3} \end{pmatrix} \tag{5.21}$$

alakban írható fel, mivel a h_1, h_2, h_3 és h_4 függvények csak x, y és z értékeitől, h_5, h_6 és h_7 függvények pedig csak ω_1, ω_2 és ω_3 értékeitől függenek. A deriváltakat

kiszámolva a következő eredményt kapjuk:

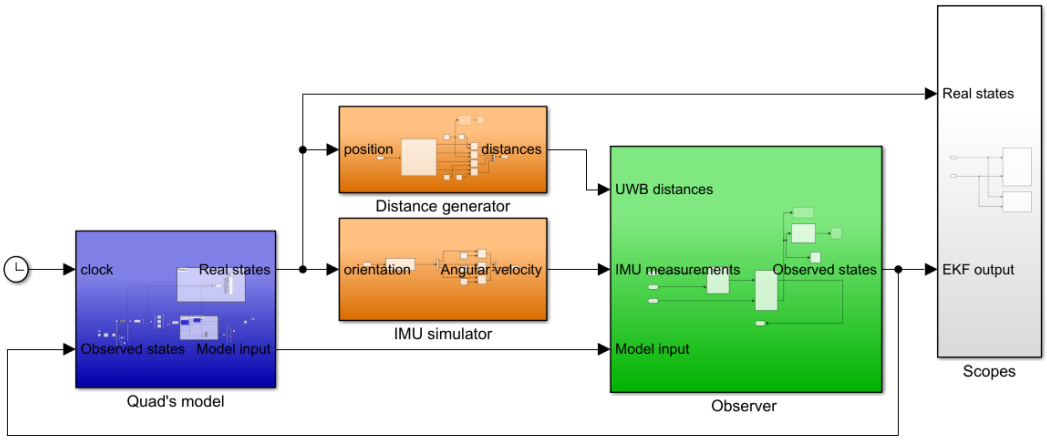
$$\begin{pmatrix} \frac{x-x_1}{d_1} & \frac{y-y_1}{d_1} & \frac{z-z_1}{d_1} & 0 & \dots & 0 & 0 & 0 & 0 \\ \frac{x-x_2}{d_2} & \frac{y-y_2}{d_2} & \frac{z-z_2}{d_2} & 0 & \dots & 0 & 0 & 0 & 0 \\ \frac{x-x_3}{d_3} & \frac{y-y_3}{d_3} & \frac{z-z_3}{d_3} & 0 & \dots & 0 & 0 & 0 & 0 \\ \frac{x-x_4}{d_4} & \frac{y-y_4}{d_4} & \frac{z-z_4}{d_4} & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.22}$$

A H mátrix meghatározása után a Kálmán-szűrő egy K súlyozó mátrixot számol. Ennek az egyes elemei azt határozzák meg, hogy az adott sorhoz tartozó állapotváltozónál milyen súllyal vegyük figyelembe a modell alapján becsült értéket és az adott oszlophoz tartozó mérés értékét.

Az $f()$ függvény alapján hasonló módon számolhatjuk ki az A mátrixot, amely az F.2 függelékben tekinthető meg.

5.3 A szimuláció felépítése

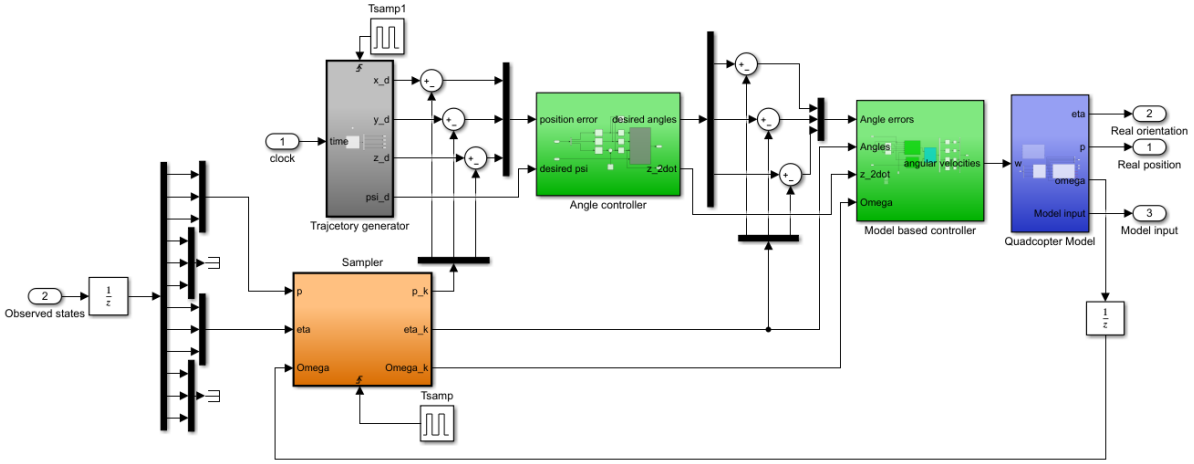
Szimulációs környezetnek a MATLAB Simulinket választottam. Ezen a platformon könnyen lehet hierarchikus felépítésű rendszereket tervezni, ami nagyban megkönnyíti a szimuláció átláthatóságát.



5.7. ábra. *Szimuláció legfelsőbb szintű elemei*

A 5.7. ábrán a kék színnel jelölt alrendszer a drón modelljét és a hozzá tartozó szabályozót tartalmazza. Ez a blokk reprezentálja a valódi drónt, melynek szimulációja két okból szükséges. Egyrészt az állapotbecslőnek szüksége van a drón bemeneteire, amelyeket így lehet előállítani. Másrészt így tudjuk összehasonlítani az állapotbecslő kimenetét a valódi állapotokkal. Narancssárga színnel vannak jelölve a méréseket előállító blokkok. Ezek a jelenleg működő pozícionáló rendszer távolság adatait, valamint az inerciális szenzor (IMU) méréseit állítják elő a valódi állapotokból, megfelelő minőségű zajjal terhelve. Az állapotbecslő pedig zöld színnel került jelölésre. Ebben egy kiterjesztett Kálmán-szűrő található (EKF), amely a méréseket, a modell bemeneteit és a nemlineáris modellt felhasználva állítja elő a megfigyelt állapotokat. A továbbiakban ezek a blokkok kerülnek részletes bemutatásra.

A drónt egy meghatározott pálya mentén szeretnénk mozgatni, azonban a rendszer erősen nemlineáris, ezért a hagyományos PID szabályozás nem tud megfelelően működni. Helyette egy ún. modell inverziós szabályozást alkalmaztam, amely kiszámolja a szükséges gyorsulásokat, majd ezekből a modell ismeretében visszaszámolja a szükséges erőt és nyomatékokat. Ezekből pedig a szükséges rotor szögsebességek, ill. rotor nyomatékok egyszerűen számíthatók. A szabályozási kör MATLAB Simulinkben készített implementációja az alábbi képen látható.



5.8. ábra. *Szabályozási kör*

A szakasz és a modell nem idealitásának figyelembevétele

A szakasz (a 5.8. ábrán kék színű "Quadcopter Model") egy nemlineáris modellen alapszik, melynek az állapotai és a rájuk vonatkozó differenciálegyenletek 4.4. fejezetben olvashatók.

Mivel a dinamikus modellünk nem írja le pontosan a kvadkopter valódi mozgását, ezért ezt a hatást modelleznünk kell. Jelen esetben egyféle leírásunk van, amit a

Kálmán-szűrőben és a szabályozott szakaszban is használunk. A valóságban azonban a szabályozott szakasznak más a dinamikája, mint amivel azt modellezzük, ezért bizonyos nagyságú zajt kell kevernünk a szabályozott szakasz kimenetére. A továbbiakban ezt a zajjal terhelt kimenetet tekintjük a kvadkopter valós pozíciójának és orientációjának.

A szabályozás

A szabályozási kör 3 fő elemből áll: a szakasz, maga a szabályozó és a trajektória generáló. Az utóbbi a követni kívánt (x, y, z) koordinátákat írja elő (a 5.8. ábrán "Trajectory generator").

A szabályozást egy kéthurkos kaszkád-szabályozás valósítja meg. A külső hurokban pozícióra szabályozunk, míg a belsőben szöghelyzetre. A pályamenti eltérésből kiszámolhatjuk a szükséges gyorsulásvektort. Ezután már meg tudjuk határozni a pályakövetéshez szükséges szögeket (a 5.8. ábrán "Angle controller"). Ezeket a szögeket az x, y és z irányú gyorsulásokból, valamint az előírt ψ szögből a következő egyenletekkel határozhatjuk meg:

$$\phi_d = sin^{-1}\left(\frac{\ddot{x} \cdot sin(\psi_d) - \ddot{y} \cdot cos(\psi_d)}{d}\right) \tag{5.23}$$

$$\theta_d = tan^{-1}\left(\frac{\ddot{x} \cdot cos(\psi_d) + \ddot{y} \cdot sin(\psi_d)}{\ddot{z}}\right) \tag{5.24}$$

A szabályozó (a 5.8. ábrán "Model based controller") a modell, a kvadkopter jelenlegi helyzete, a pozíció eltérés és a szöghibákat felhasználva tengelyenként külön PID szabályozókkal számolja ki a szükséges erőket és nyomatékokat. Ezután a modell egyenleteiből 4.5.-öt felhasználva generál bemenetet a modellhez. [29]

A pozíció és orientációs hibákat az előírt trajektória egy pontja és a kiterjesztett Kálmán-szűrő által becsült állapotokból számoljuk.

A mérések generálása

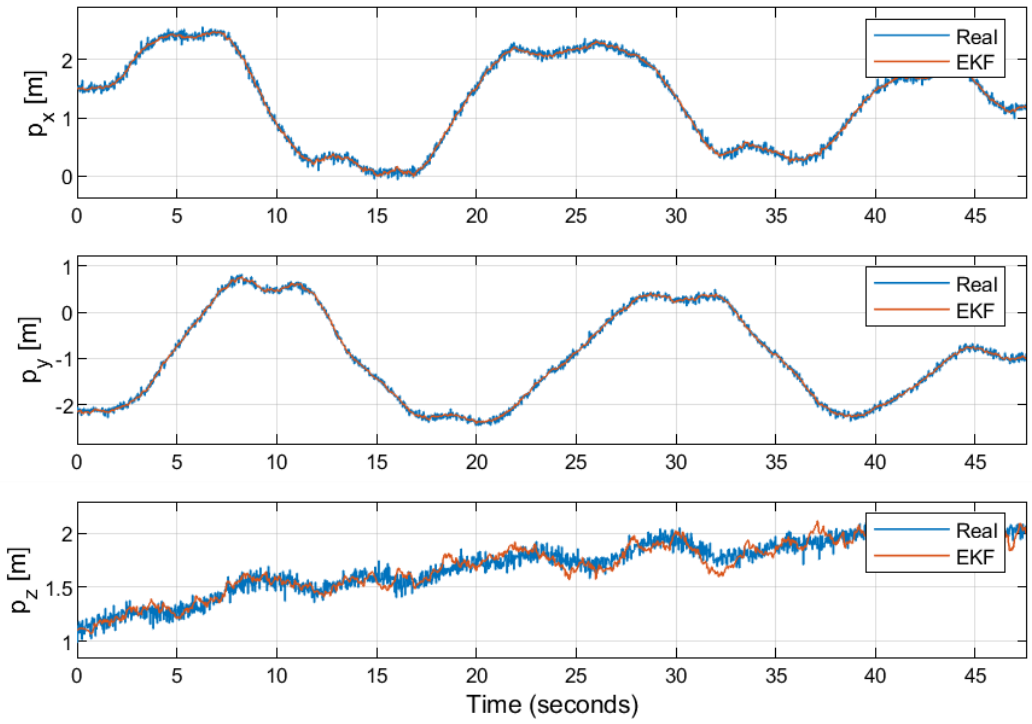
A távolságokat a 5.7. ábrán látható "Distance generator" blokk generálja a valós pozícióból és az anchorok helyzetéből. A szimulált mérésekre független normál eloszlású zajt keverünk, melynek szórását a valódi mérésekből határoztam meg. IMU szenzor méréseinek szimulálását is elvégeztem. Ennek a pontos működése F.1 függelékben olvasható.

Az EKF-ben felhasznált jelek

A modell kimenetét fogjuk felhasználni, mint ténylegesen bejárt trajektória ("Reference position"). Az EKF-ben is megtalálható a kvadkopter modell, melynek működéséhez ismernünk kell a bemeneteket. Következésképpen a modell bemenetét is fel kell használnunk ("Model input").

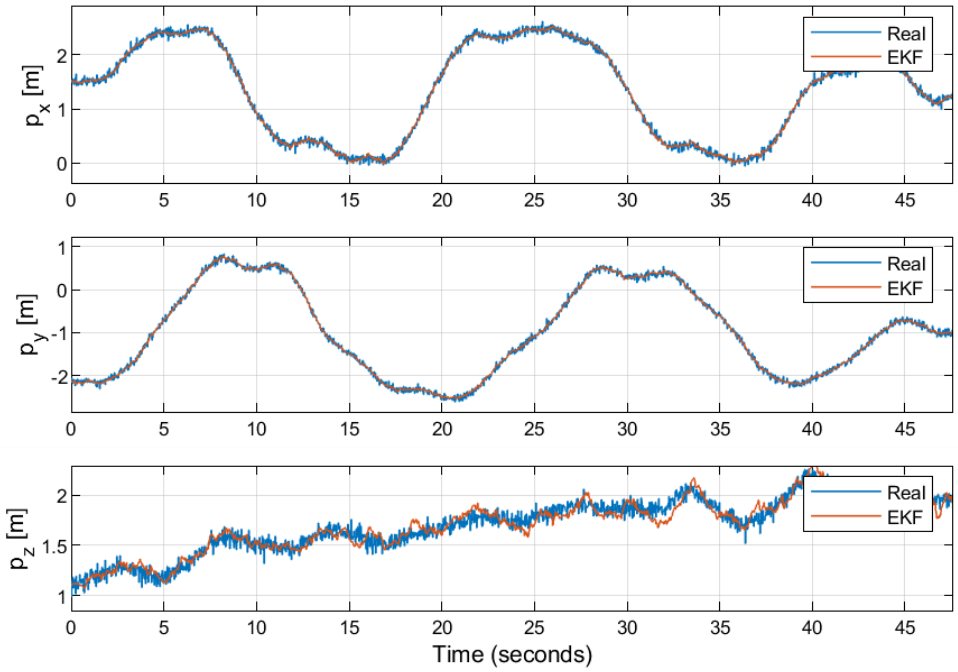
5.4 A szimuláció eredményei

Az elkészített állapotbecslőt először nyitott körben (open-loop) vizsgáltam meg, hogy képes-e követni a drón helyzetét. Ennek eredménye látható a 5.9. ábrán.



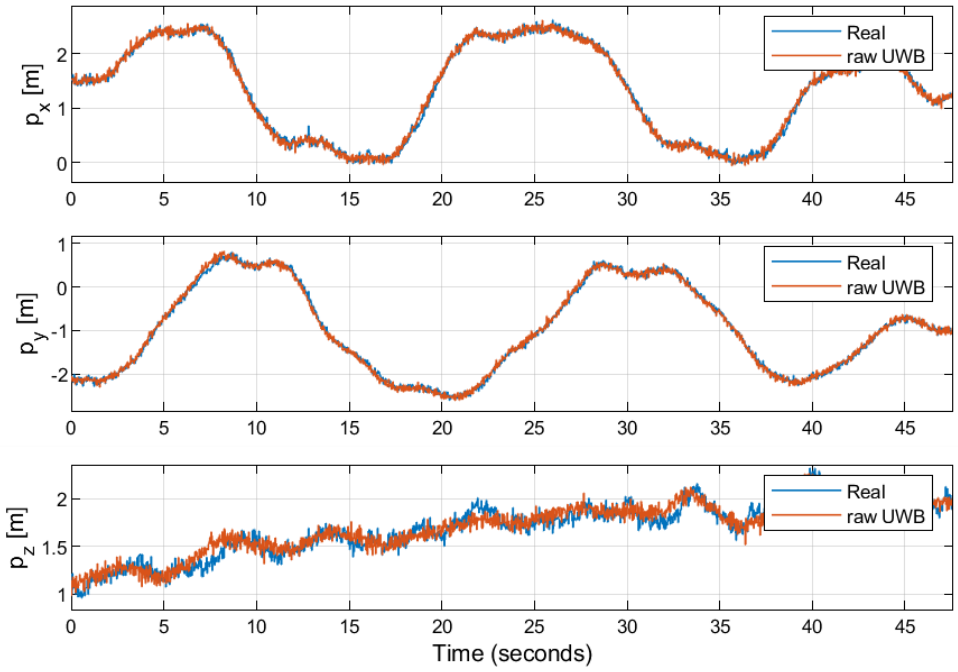
5.9. ábra. *A valódi és az EKF által becsült pozíciók koordinátái (nyitott körben)*

Nyitott körben jól működött az állapotbecslő, ezért zárt körben is teszteltem. Vagyis ebben az esetben az állapotbecslő kimenete adta a szabályozónak a drón állapotváltozóinak az értékét. Az alábbi képen látható a zárt körben való viselkedése a rendszernek.



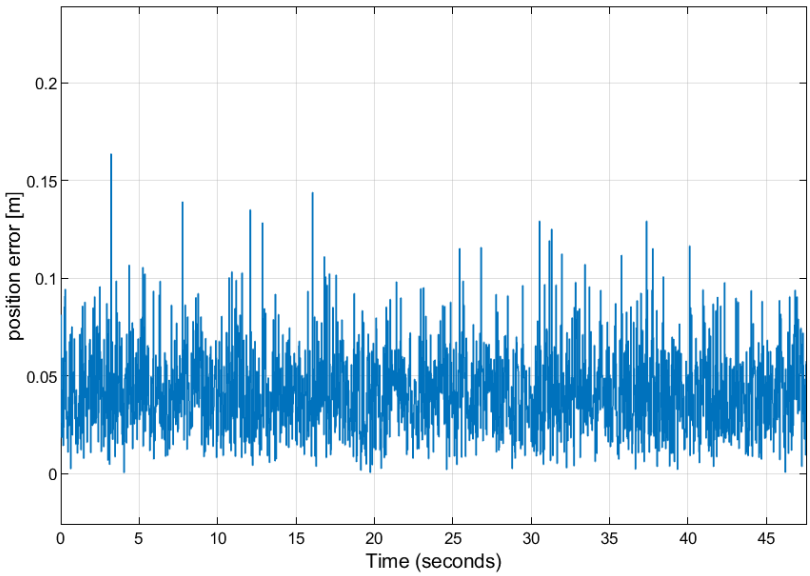
5.10. ábra. *A valódi és az EKF által becsült pozíciók koordinátái (zárt körben)*

Mivel jól működik az állapotbecslő, ezért mindegy, hogy a valós vagy a becsült pozíciót csatoljuk vissza, ezért a két ábra is közel azonos. Az eredményt összehasonlíthatjuk a pozícionáló rendszer által (csak a távolságok alapján) becsült pozíciókkal.



5.11. ábra. *A valódi és az UWB távolságok alapján becsült pozíciók*

Látható, hogy csupán a nyers távolság adatokat felhasználva jelentős zajjal terhelt pozíció kimenetet kapunk. Ez sok esetben a szabályozónak sem túl kedvező. Az EKF esetében azonban elég pontosan sikerül lekövetni a drón helyzetét. Ezt a pontosságot minden időpillanatban ki lehet számolni. Ezt láthatjuk a 5.12. ábrán.



5.12. ábra. *Az pozícióbecslés hibája minden egyes időpillanatban*

A pozíció becslés leginkább a z koordinátában nem pontos. A kiugró értékek is ebből származnak. Az átlag hiba azonban 4,26 cm. Ez az eredmény elég jónak mondható, összevetve a csupán UWB-s távolságokkal elért kb. 20 cm-es pontossággal.

6. fejezet

Összefoglalás, további elvégzendő feladatok

6.1 Összefoglalás

A dolgozat célja egy drónaréna és annak működésének bemutatása volt. Mivel egy ilyen rendszer sok, már magukban is bonyolult alrendszerből áll össze, ezért ezeket külön fejezetekben tárgyaltam.

Az 1. fejezetben bemutatásra került az UWB, mint rádiós technológia, annak a használata, valamint már működő UWB-s pozícionáló rendszerek.

A 2. fejezetben az idő alapú távolságmérés különböző módjait és a távolságadatok alapján a pozíció számítás lehetőségeit ismertettem. Megvizsgáltam a lehetséges üzenetváltási sémát előnyeit, hátrányait, valamint javaslatot tettem a numerikus pozíciószámítás költségfüggvényére.

Az elkészített pozícionáló rendszert a 3. fejezetben mutattam be, kitérve a felhasznált hardverekre. Szintén ebben a fejezetben került bemutatásra egy új üzenetváltási séma, amellyel gyorsabban tudunk távolságot mérni. Ezzel időegység alatt több információhoz jutunk, ami pontosabb pozícióbecslést tesz lehetővé. A rendszert úgy módosítottam, hogy a kezdeti kalibrációt automatikusan elvégezze. Ez megkönnyíti a dolgunkat, hiszen így nem kell kézzel méréseket végeznünk.

A drón megépítését és konfigurálását is elvégeztem, melynek menetét a 4. fejezetben tárgyaltam. A drón dinamikus modellje, valamint az identifikáció lehetséges megközelítéseit is itt ismertettem.

Végül az 5. fejezetben az állapotbecslő került bemutatásra. Az állapotbecslőt egy kiterjesztett Kálmá-szűrő valósítja meg, amely a pozícionáló rendszer távolságadatait, a drónra adott bemeneteket és az IMU szenzor szögsebességmérőjét használva ad egy pontosabb becslést a drón állapotaira. Szimulációval validáltam is az állapotbecslőt.

6.2 További feladatok

A következő lépésben meg kell tervezni az irányítási architektúrát. Az állapotbecslőnek vagy egy külső PC-n kell futnia vagy magán a drónon. Előbbi esetben további két lehetőségünk van. A drónt szintén a PC-n futó irányítási algoritmus szabályozza vagy egy drónon futó alkalmazás. Azonban mindkét esetben ki kell építeni valamilyen valós idejű kommunikációt a drón és a PC között. Míg előbbi esetben csak a beavatkozó jelet kell elküldenünk a drónnak, addig a második esetben magának a drónnak az állapotait.

A második lehetőség, hogy az állapotbecslő és az irányítási algoritmus is a drónon fut. Ehhez felhasználhatjuk a drónon található kiegészítő számítógépet, vagyis a Raspberry Pi-t. Ez ugyanis rendelkezik akkora számítási teljesítménnyel, hogy mindkét feladatot képes legyen ellátni. További előnye, hogy számos valós idejű operációs rendszer létezik rá.

Ezután a kiválasztott platformon implementálnunk kell az állapotbecslőt. További feladatunk megoldani, hogy az állapotbecslőnek minden szükséges bemenő adata rendelkezésre álljon egy hardveren.

Egy irányítási algoritmust is tervezni kell, amely megfelelően tudja majd irányítani a drónt. Ezt a 5.3. fejezetben bemutatott szimulációs környezetben könnyedén meg is tehetünk.

Végül az elkészült rendszert valós környezetben is validálni kell. Meg kell határoznunk, hogy az állapotbecslő mennyire pontosan tudja meghatározni a drón helyzetét, mert csak ennek ismeretében tudjuk elvégezni a különböző irányítási algoritmusok minősítését.

A felsorolt feladatokkal a továbbiakban foglalkozni is fogok és ezt lehetőség szerint Önálló laboratórium és Diplomatervezés tárgyak keretein belül szeretném megvalósítani.

Irodalomjegyzék

[1] Inc Bluetooth SIG. Location Services, Hozzáférés dátuma: 2019. október. https://www.bluetooth.com/solutions/location-services/.

[2] Tammaso Bresciani. Modelling, identification and control of a quadrotor helicopter, 2008. Student Paper.

[3] Chegg. Steiner tétel, Hozzáférés dátuma: 2019. október. https://www.chegg.com/homework-help/definitions/parallel-axis-theorem-5.

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

[5] Jon Fingas. Bluetooth direction finding will locate your keys, Hozzáférés dátuma: 2019. október. https://www.engadget.com/2019/01/28/bluetooth-direction-finding/.

[6] 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.

[7] 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.

[8] Tennessean Jamie McGee. Music City Center app guides visitors, Hozzáférés dátuma: 2019. október. https://eu.tennessean.com/story/money/tech/2014/11/12/music-city-center-app-guides-visitors/18945489/.

[9] 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.

[10] B Lantos. Theory and design of control systems i-ii. *Akademia Press, Budapest*, 2003.

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

[12] Decawave Ltd. Decawave honlapja, Hozzáférés dátuma: 2019. október. `https://www.decawave.com/`.

[13] Decawave Ltd. *DecaRangeRTLS ARM source code guide.* Decawave, 2014.

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

[15] 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.

[16] MapsPeople. Mapspeople honlapja, Hozzáférés dátuma: 2019. október. `https://www.mapspeople.com/`.

[17] Zaki Mustapa, Shakir Saat, S H. Husin, and Thoriq Zaid. Quadcopter physical parameter identification and altitude system analysis. pages 130–135, 09 2014.

[18] Zaki Mustapa, Shakir Saat, S. Husin, and Thoriq Zaid. Quadcopter physical parameter identification and altitude system analysis. pages 130–135, 09 2014.

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

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

[21] Pozyx. Pozyx cég honlapja, Hozzáférés dátuma: 2019. október. `https://www.pozyx.io/`.

[22] Nemzeti Média és Hírközlési Hatóság. Sávhasználati feltételek és frekvenciagazdálkodási követelmények, Hozzáférés dátuma: 2019. október. `http://nmhh.hu/dokumentum/165870/NFFF_03_mell_TE.pdf`.

[23] Ahmed Samir, Abdallah Hammad, Ashraf Hafez, and Hala Mansour. Quadcopter trajectory tracking control using state-feedback control with integral action. 2017.

[24] Sewio. Sewio cég honlapja, Hozzáférés dátuma: 2019. október. https://www.sewio.net/.

[25] ArduPilot Dev Team. Ardupilot által támogatott hardverek listája. http://ardupilot.org/copter/docs/common-autopilots.html.

[26] ArduPilot Dev Team. Pixhawk és raspberry konfigurálása. http://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html.

[27] PX4 Dev Team. Pixhawk 1 fedélzeti számítógép honlapja. https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk.html.

[28] 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.

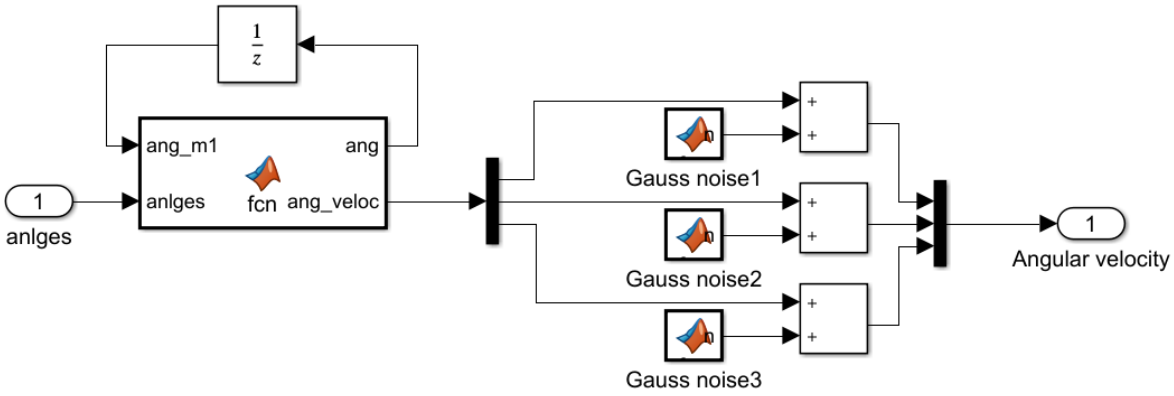
[29] Z. Zuo. Trajectory tracking control design with command-filtered compensation for a quadrotor. *IET Control Theory Applications*, 4(11):2343–2355, November 2010.

Függelék

F.1 IMU szenzor szimulációja

Az IMU szenzor a legtöbb esetben 3 mennyiséget tud mérni: szögsebességet (giroszkóp), gyorsulást (gyorsulásmérő) és mágneses teret (magnetométer). Jelenleg csak a szögsebesség mérésének szimulációját végeztem el, de a magnetométert is fel lehet használni az orientáció becslésének pontosítására.

A szögsebességek előállításához az aktuális és az egy mintavételi idővel előbbi orientációs szögek különbségét használjuk. Ezt az értéket elosztva a mintavételi idővel, megkaphatjuk a szögsebességeket a világ koordináta-rendszerben. Ezután egy egyszerű mátrix szorzással áttérhetünk a test koordináta-rendszerre, amelyet a dinamikai modell is használ a szögsebességek esetén. A mérésekre egy-egy normál eloszlású zajt keverünk a mérési pontatlanság figyelembevételére.



F.1.1. ábra. *IMU szimulációja MATLAB Simulink környezetben*

A F.1.1. ábrán látható "Matlab Function" blokk kódja végzi el a fent részletezett számítást a következő függvénnyel:

```
1 function [ang,ang_veloc] = fcn(ang_m1,anlges, ts)
2
```

```
phi = anlges(1);
tht = anlges(2);
psi = anlges(3);

ang_derivs = [(anlges(1)-ang_m1(1))/ts, (anlges(2)-ang_m1(2))/ts, (
    anlges(3)-ang_m1(3))/ts]';

A = [1, sin(phi)*tan(tht), cos(phi)*tan(tht);
     0, cos(phi),                  -sin(phi);
     0, sin(phi)*sec(tht), cos(phi)*sec(tht)];

ang_veloc = A\ang_derivs;

ang = anlges;
```

F.2 EKF-ben használt A mátrix

Az A mátrix egy 12x12 méretű kvadratikus mátrix

$$A = \begin{pmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & ... & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & ... & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & ... & 0 \\
0 & ... & 0 & v_x^\phi & v_x^\theta & v_x^\psi & 0 & 0 & 0 \\
0 & ... & 0 & v_y^\phi & v_y^\theta & v_y^\psi & 0 & 0 & 0 \\
0 & ... & 0 & v_z^\phi & v_z^\theta & v_z^\psi & 0 & 0 & 0 \\
0 & ... & 0 & \phi^\phi & \phi^\theta & 0 & 1 & \phi^q & \phi^r \\
0 & ... & 0 & \theta^\phi & 0 & 0 & 0 & cos(\phi) & -sin(\phi) \\
0 & ... & 0 & \psi^\phi & \psi^\theta & 0 & 0 & \psi^q & \psi^r \\
0 & ... & 0 & 0 & 0 & 0 & 0 & c_1 \cdot r & c_1 \cdot q \\
0 & ... & 0 & 0 & 0 & 0 & c_2 \cdot r & 0 & c_2 \cdot p \\
0 & ... & 0 & 0 & 0 & 0 & c_3 \cdot q & c_3 \cdot p & 0
\end{pmatrix}$$

ahol minden sorban a 0...0 helyén 6 nulla szerepel. A többi paraméter az alábbiak szerint vannak definiálva:

$$c_1 = \frac{J_y - J_z}{J_x}$$

$$c_2 = \frac{J_z - J_x}{J_y}$$

$$
\begin{aligned}
c_3 &= \frac{J_x - J_y}{J_z} \\
v_x^{\phi} &= \frac{F}{m} \cdot (-sin(\phi) \cdot sin(\theta) \cdot cos(\psi) + cos(\phi) \cdot sin(\psi)) \\
v_x^{\theta} &= \frac{F}{m} \cdot (cos(\phi) \cdot cos(\theta) \cdot cos(\psi)) \\
v_x^{\psi} &= \frac{F}{m} \cdot (-cos(\phi) \cdot sin(\theta) \cdot sin(\psi) + sin(\phi) \cdot cos(\psi)) \\
v_y^{\phi} &= \frac{F}{m} \cdot (-sin(\phi) \cdot sin(\theta) \cdot sin(\psi) - cos(\phi) \cdot cos(\psi)) \\
v_y^{\theta} &= \frac{F}{m} \cdot (cos(\phi) \cdot cos(\theta) \cdot sin(\psi)) \\
v_y^{\psi} &= \frac{F}{m} \cdot (cos(\phi) \cdot sin(\theta) \cdot cos(\psi) + sin(\phi) \cdot sin(\psi)) \\
v_z^{\phi} &= \frac{F}{m} \cdot (-sin(\phi) \cdot cos(\theta)) \\
v_z^{\theta} &= \frac{F}{m} \cdot (-cos(\phi) \cdot sin(\theta)) \\
v_z^{\psi} &= 0 \\
\phi^{\phi} &= q \cdot cos(\phi) \cdot tan(\theta) - r \cdot sin(\phi) \cdot tan(\theta) \\
\phi^{\theta} &= q \cdot sin(\phi) \cdot sec(\theta) \cdot sec(\theta) + r \cdot cos(\phi) \cdot sec(\theta) \cdot sec(\theta) \\
\phi^{q} &= sin(\phi) \cdot tan(\theta) \\
\phi^{r} &= cos(\phi) \cdot tan(\theta) \\
\theta^{\phi} &= -q \cdot sin(\phi) - r \cdot cos(\phi) \\
\psi^{\phi} &= q \cdot cos(\phi) \cdot sec(\theta) - r \cdot sin(\phi) \cdot sec(\theta) \\
\psi^{\theta} &= q \cdot sin(\phi) \cdot sec(\theta) \cdot tan(\theta) + r \cdot cos(\phi) \cdot sec(\theta) \cdot tan(\theta) \\
\psi^{q} &= sin(\phi) \cdot sec(\theta) \\
\psi^{r} &= cos(\phi) \cdot sec(\theta)
\end{aligned}
$$