

SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR
AUTOMATIKA ÉS ALKALMAZOTT INFORMATIKA SZAK

**FPGA ALAPÚ HARDVERREL
TÁMOGATOTT UAV SZABÁLYOZÁSI
FELADATOK MEGVALÓSÍTÁSA**

DIPLOMADOLGOZAT

Témavezető:
Dr. Brassai Sándor Tihamér
egyetemi előadótanár

Végzős hallgató:
Székely István-Zsolt

2019

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÂRGU MUREŞ
SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

**SOLUȚII DE CONTROL CU
IMPLEMENTARE HARDWARE
PENTRU SISTEME UAV**

PROIECT DE DIPLOMĂ

**Coordonator științific:
Conf. Dr. Ing.
Brassai Sándor Tihamér**

**Absolvent:
Székely István-Zsolt**

2019

Specializarea: **Automatică și informatică aplicată**

LUCRARE DE DIPLOMĂ

Coordonator științific:

Conf. dr. ing. Brassai Sándor Tihamér

Candidat: Székely István-Zsolt

Anul absolvirii: **2019**

a) Tema lucrării de licență:

Soluții de control cu implementare hardware pentru sisteme UAV

b) Problemele principale tratate:

Probleme teoretice:

- Utilizarea circuitelor FPGA pentru implementarea algoritmilor de reglare cu funcționare în timp real
- Soluții de reprezentare a orientării sistemelor UAV: unghiuri Euler, unghiuri RPY
- Senzori inerțiali. Determinarea unghiurilor de înclinare a sistemelor UAV pe baza senzorilor inerțiale prin filtru complementar
- Structuri și algoritmi de reglare. Regulatoare PID. Soluții de acordare a regulatoarelor PID.

- Implementare în FPGA:

- a interfeței de achiziție a semnalelor senzorului inerțial, semnalelor receptorului RF,
- implementarea hardware a algoritmului de reglare PID
- implementarea hardware a filtrului complementar pentru determinarea orientării sistemului UAV în timp real
- implementarea hardware a filtrului complementar
- studiul posibilităților de optimizare a modulelor implementate în hardware cu scopul de a reduce resursele hardware utilizate

c) Desene obligatorii:

- arhitectura sistemului implementate
- schemele System Generator Simulink pentru achiziție semnale IMU, filtru complementar, achiziția semnalelor de la receptorul RF, regulatoarele PID implementate
- reprezentarea rezultatelor măsurătorilor privind testarea și punerea în funcțiune a submodulelor sistemului respectiv acordarea regulatoarelor PID

d) Softuri obligatorii:

-Implementarea in VHDL a submodulelor sistemului: automate de stări, pentru sincronizarea și controlul submodulelor

-Integrarea submodulelor sistemului în System Generator

e) Bibliografia recomandată:

[1] Zimmerman, N. M. (2016). Flight control and hardware design of multi-rotor systems

[2] Brassai Sándor Tihamér, Újrakonfigurálható digitális áramkörök tervezési és tesztelési módszerei, Editura Scientia , Cluj-Napoca, 2018, ISBN 978 606 975 020-9

f) Termene obligatorii de consultații: săptămânal

g) Locul și durata practiciei: Universitatea Sapientia, Facultatea de Științe Tehnice și Umaniste din Târgu Mureș, laborator 309

Primit tema la data de: 27.04.2018

Termen de predare: 01.07.2019

Semnătura Director Departament

Semnătura responsabilului programului de studiu

Semnătura coordonatorului

Semnătura candidatului

Declarație

Subsemnata/ul *Székely István-Zsolt*, absolvent(ă) al/a specializării *Automatică si informatică aplicată*, promoția 2019 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea,

Data:

Absolvent

Semnătura.....

Model tip b.

Declarație

Subsemnata/Subsemnatul *Dr. Brassai Sándor Tihamér*, funcția , titlul științific declar pe propria răspundere că *Székely István-Zsolt*, absolvent al specializării *Automatică și informatică aplicată* a întocmit prezenta lucrare sub îndrumarea mea.

În urma verificării formei finale constat că lucrarea de licență/proiectul de diplomă/disertația corespunde cerințelor de formă și conținut aprobate de Consiliul Facultății de în baza reglementărilor Universității „Sapientia”. Luând în considerare și Raportul generat din aplicația antiplagiat „Turnitin” consider că sunt îndeplinite cerințele referitoare la originalitatea lucrării impuse de Legea educației naționale nr. 1/2011 și de Codul de etică și deontologie profesională a Universității Sapientia, și ca atare sunt de acord cu prezentarea și susținerea lucrării în fața comisiei de examen de licență/diplomă/disertație.

Localitatea,

Data:

Semnătura îndrumătorului

Nyilatkozat

Alulírott *Székely István-Zsolt* a Sapientia EMTE Marosvásárhelyi Karának *Automatika* és *alkalmazott informatika* alapszakos/mesterszakos végzős hallgatója, ismerve a 2011/1-es Tanügyi törvény valamint a Sapientia EMTE Etikai kódexének a szakmai dolgozatok eredetiségére vonatkozó rendelkezéseit és a hamis nyilatkozat következményeit, kijelentem, hogy jelen szakdolgozat/diplomamunka/disszertáció saját munkám eredménye, a kutatómunkát/tervezést én végeztem, a szakirodalomból átvett információkat és adatokat megfelelő módon hivatkoztam, megjelölve a forrást.

....., 2019.....

Végzős hallgató aláírása

.....

Nyilatkozat

Alulírott *Dr. Brassai Sándor Tihamér*, oktatói fokozat *előadótanár tudományos cím docens* saját felelősségemre kijelentem, hogy *Székely István-Zsolt, Automatika és alkalmazott informatika* alapszakos/mesterszakos hallgató ezt a szakdolgozatot az én irányításommal készítette.

A szakdolgozat/terv/disszertáció végső változatát ellenőriztem és megállapítottam, hogy a dolgozat megfelel a *Marosvásárhelyi Kar* által megfogalmazott formai és tartalmi elvárásoknak, a Sapientia EMTE vonatkozó szabályzatával összhangban. Figyelembe véve a *Turnitin* plágiumellenőrző alkalmazásból generált jelentés eredményeit is, megállapítom, hogy a dolgozat eleget tesz a 2011/1-es számú Tanügyi törvény, valamint a Sapientia EMTE Etikai kódexének a szakmai dolgozatok eredetiségrére vonatkozó követelményeinek, és támogatom a szakdolgozat bemutatását és védését a záróvizsga bizottság előtt.

....., 2019.....

Témavezető aláírása

Soluții de control cu implementare hardware pentru sisteme UAV

Extras

Scopul acestei lucrări de licență este stabilizarea quadcopterului în plan orizontal și urmărirea unui semnal de referință de poziție unghiulară. Sistemul este implementat într-un FPGA. La Universitatea Sapientia era deja un proiect creat cu acest scop, cu singura diferență că majoritatea sistemului a fost implementat în software. La nivel hardware, au fost implementate doar protocoalele de comunicare. Scopul acestui proiect a fost de a examina comportamentul unui sistem implementat în hardware și de a vedea ce fel de rezultate pot fi obținute cu acesta.

Am implementat și testat un PID modificat și un controler PID în cascadă. Pentru orientarea sistemului, a fost utilizat un senzor IMU. Există o opțiune implementată în proiect, care permite utilizatorului să aleagă semnalul de referință, care poate fi de la telecomandă sau de la calculator. Quadcopterul a fost montat pe un suport. Scopul acestui stand a fost de a asigura un mediu sigur pentru măsurare și calibrare. La dezvoltarea sistemului am luat în considerare gestionarea resurselor și am încercat să o optimizăm pentru cea mai mică cantitate posibilă, fiindcă dacă vom adăuga module diferite în viitor, nu va trebui să rezolvăm această problemă. Avem implementat un controler PID, iar acesta calculează toate semnalele de control. Deoarece ciclul de ceas al unui FPGA este considerabil mai mare decât unui microcontroler, în acest fel nu trebuie să ne îngrijorăm de timpul lent de calcul.

În cadrul proiectului, am reușit să implementăm și să aplicăm un controler PID cu un canal D modificat, astfel încât quadcopterul a putut să urmeze un semnal de referință variat. Sistemul, care a fost implementat în hardware, a fost, de asemenea, stabil pentru o perioadă mai îndelungată.

Kivonat

Az államvizsga dolgozat célja egy quadkopter szabályozása vízszintes síkban, illetve egy előírt szögpozíció alapjel követése. A rendszer megvalósítása hardveresen történik, FPGA-n. A Sapientia egyetem keretén belül már volt egy ilyen megvalósítású projekt, ami viszont nem hardveresen, hanem szoftveresen volt megvalósítva. Hardveres szinten csak a kommunikációs egységek voltak implementálva. A cél az volt, hogy hardveres megközelítésből megfigyeljük, hogy milyen eredményeket lehet elérni a szoftveres megoldáshoz képest.

A quadkopter szabályozójaként PID és kaszkád PID algoritmusokat implementáltunk és teszteltünk. A rendszer orientációjának meghatározására egy IMU szenzort alkalmaztunk. A szabályozó az előírt alapjelet távirányítótól vagy számítógéptől kapta. A quadkopter egy standra volt felszerelve, aminek a célja az volt, hogy biztonságosan lehessen tesztelni és hangolni a rendszert. A quadkopter fejlesztése közben az is figyelembe volt véve, hogy a megtervezett modulok erőforrásigénye alacsony legyen, a jövőbeli bővíthetőség szempontjából. A rendszerben egy PID szabályozó van beépítve, és ezzel az eggyel számolódik ki az összes szabályozójel. Erőforrásigény szempontjából a szabályozó kevesebb szorzó áramkört igényel, viszont több időt, amíg kiszámolódik az eredmény. Az FPGA-nak köszönhetően a számításokhoz szükséges idővel nem szükséges foglalkozni, mivel sokkal gyorsabbak egy mikrokontrollerhez képest.

A projekt keretén belül sikerült egy módosított D csatornás PID algoritmust implementálni és behangolni úgy, hogy a quadkopter képes volt egy előírt, változó alapjelet követni. Az elkészült, hardveresen megvalósított rendszer stabilan működött huzamosabb ideig.

Kulcsszavak: FPGA, quadkopter, szabályozó, PID, stabilizálás

Abstract

The purpose of this state examination paper is the stabilization of the quadcopter in horizontal plane, and the tracking of a reference signal, which stays for angular position. The system is implemented in a FPGA. At the Sapientia University there already was a project created with this purpose, with the only difference that it was implemented in software. On hardware level only the communication protocols were implemented. The purpose of this project was to examine the behavior of a hardware implemented system, and to see what kind of results can be achieved with it.

We implemented and tested a modified PID and a cascade PID controller. For the orientation of the system, an IMU sensor was used. There is an option implemented in the design, which allows the user to choose the reference signal, which may be from the remote controller, or the computer. The quadcopter was mounted on a stand. The purpose of this stand was to ensure a safe environment when measuring and calibrating. At the development of the system, we considered resource management, and we tried to optimize it for the least needed resource amount, so if we are going to add different modules to it, we won't have to deal with this problem. We have one PID controller implemented, and this calculates all the control signals. Since the clock cycle of an FPGA is considerably higher than a microcontroller, this way we don't have to worry about the slow computing time.

Within the project, we managed to implement and tune a PID controller with a modified D channel, so the quadcopter was able to follow a varying reference signal. The system, which was implemented in hardware was also stable for a longer period of time.

Keywords: FPGA, quadcopter, controller, PID, stability

Tartalomjegyzék

1.	Bevezető.....	15
2.	Projekt célja.....	15
3.	Elméleti megalapozás	15
3.1.	Quadkopter általánosságok.....	15
3.2.	Szögek	16
3.2.1.	RPY szögek.....	16
3.2.2.	Euler szögek.....	16
3.2.3.	Gimbal zár.....	17
3.2.4.	Kvaterniók	17
3.2.5.	Alkalmazott koordináta rendszerek	17
3.3.	IMU szenzorok	18
3.3.1.	Giroszkóp.....	18
3.3.2.	Gyorsulásmérő	19
3.3.3.	Magnetométer	19
3.3.4.	Komplementer szűrő	20
3.4.	A rendszermodell becslése	21
3.5.	Szabályozók	21
3.5.1.	PID szabályozó.....	22
3.5.2.	Módosított PID algoritmusok.....	23
3.5.3.	PID hangolása	24
3.5.4.	Ziegler Nichols hangolás	24
3.5.5.	Adaptív szabályozás	25
3.5.6.	Referencia modellen alapuló adaptív szabályozás (MRAC).....	25
3.5.7.	RST szabályozó	26
3.6.	FPGA általánosságok	27
4.	Gyakorlati megvalósítás.....	32
4.1.	Rendszer architektúrája	32
4.2.	Pmod NAV IMU.....	34
4.3.	Impulzust generáló modul	40
4.4.	Átlagoló IIR szűrő.....	40
4.5.	Komplementer szűrő	41
4.6.	Távirányító	43

4.7.	Referencia kiválasztó	45
4.8.	Quadkopter szabályozási módszerek.....	46
4.8.1.	Módosított D csatornás PID algoritmus.....	46
4.8.2.	Kaszkád szabályozó.....	49
4.9.	Szabályozó jelek összesítése.....	51
4.10.	PWM jelgenerálás	51
5.	Mérési eredmények.....	53
5.1.	Mérési eredmények mentése hardver ko-szimulációt alkalmazva.....	53
5.2.	Pmod NAV IMU	55
5.3.	Módosított D csatornás PID.....	65
5.4.	RST szabályozó	70
5.5.	Adaptív MIT szabályozó	72
5.6.	Vezérlőjelek összegzése	75
6.	Fejlesztés során felmerült problémák	76
7.	Következtetések.....	77
8.	Jövőbeli tervezek.....	78
9.	Könyvészeti	80
10.	Függelék	82

Ábrajegyzék

3-1 ábra Euler szögek meghatározása, a koordináta tengelyek forgatása[18]	17
3-2 ábra Mágneses elhajlás[19].....	20
3-3 ábra Szabályozó tömbvázlata.....	22
3-4 ábra Referencia modellen alapuló szabályozó tömbvázlata.....	26
3-5 ábra FPGA chip[20]	27
3-6 ábra FPGA belső felépítése.....	29
3-7 ábra FPGA összekapcsolása más eszközökkel[21]	31
4-1 ábra A rendszer felépítése modulokra lebontva	32
4-2 ábra A rendszer architektúrája	33
4-3 ábra A rendszer belső felépítése modulokra leosztva az FPGA-ban	34
4-4 ábra FPGA és IMU kapcsolásának tömbvázlata	35
4-5 ábra IMU vezérlő működésének állapotdiagramja	35
4-6 ábra IMU szenzor inicializálásának állapotdiagramja	36
4-7 ábra SPI modul állapotdiagramja	37
4-8 ábra Gyorsulásmérő adatai olvasásának állapotdiagramja.....	39
4-9 ábra Távirányító, és a karoknak megfelelő funkciók[22]	44
4-10 ábra Előírt érték kiválasztásának tömbvázlata	45
4-11 ábra PID szabályozó és a vezérlőegység tömbvázlata.....	48
4-12 ábra Kaszkád PID szabályozó tömbvázlata	49
4-13 ábra PI/PD szabályozó és a vezérlőegység tömbvázlata.....	50
4-14 ábra Szabályozójeleket összegző egység tömbvázlata	51
4-15 ábra PWM modul állapotdiagramja.....	52
5-1 ábra A hardver ko-szimuláció tömbvázlata	54
5-2 ábra Távirányítónak leküldendő paraméterek, és az általa visszaküldött eredmények mentése és ábrázolása Simulink környezetben hardver ko-szimulációt alkalmazva	55
5-3 ábra Giroszkóp mérési adatai, stacionárius eszköz esetén.....	55
5-4 ábra Giroszkóp mérési adatainak statisztikája	56
5-5 ábra Gyorsulásmérő mérési adatai, stacionárius eszköz esetén	57
5-6 ábra Barométer mérési eredményei	58
5-7 ábra Magnetométer forgatása, mágneses tér érzékelése	59
5-8 ábra Rendszer billenésének mérése	60
5-9 ábra Giroszkóp segítségével mért mérési eredmények a szűrő jelenléte nélkül.....	61
5-10 ábra Giroszkóp segítségével mért mérési eredmények a szűrő jelenlétében.....	62
5-11 ábra Gyorsulásmérő segítségével mért mérési eredmények a szűrő jelenléte nélkül	63
5-12 ábra Gyorsulásmérő segítségével mért mérési eredmények a szűrő jelenlétében	64
5-13 ábra Magnetométer segítségével mért adatok forgó motorok jelenlétében.....	65
5-14 ábra PID szabályozó azonnal változó előírt értékkel.....	66
5-15 ábra PID szabályozó, fokozatosan változó előírt értékkel	67
5-16 ábra Szögpozíció változása, és szabályozójel előírt szinusz jel alapján	68
5-17 ábra Quadkopter szabályozása előírt négyzetjel esetén változó D paraméter értékekkel	69
5-18 ábra Referencia megváltoztatása, és a rendszer válasza	70
5-19 ábra RST szabályozó, jól megbecsült paraméterekkel	71
5-20 ábra RST szabályozó, rosszul megbecsült paraméterekkel.....	72

5-21 ábra Adaptív MIT szabályozó, zaj nélkül	73
5-22 ábra Adaptív MIT szabályozó, zajjal.....	74
5-23 ábra Adaptív MIT szabályozó, minimális zajjal	75
5-24 ábra Motorokra adott vezérlőjeleknek a szimmetriája két szemközti motor esetén	76
8-1 ábra Motorokra adott vezérlőjeleknek a szimmetriája két szemközti motor esetén	79
10-1 ábra A rendszer belső kapcsolási rajza	82
10-2 ábra IMU szenzor, átlagoló IIR szűrő, és a komplementer szűrő kapcsolási rajza	83
10-3 ábra IMU szenzor, átlagoló IIR szűrő, és az erőforrásbarát komplementer szűrő kapcsolási rajza ...	84
10-4 ábra Átlagoló IIR szűrő kapcsolási rajza	85
10-5 ábra Távirányító értékek beolvasásának és skálázásának kapcsolási rajza.....	85
10-6 ábra Módosított D csatornás PID szabályozó, vezérlő, és tároló kapcsolási rajza	86
10-7 ábra Kaszkád PID szabályozó kapcsolási rajza.....	86
10-8 ábra PI szögpozíció szabályozó, vezérlő, és tároló egységének kapcsolási rajza	87
10-9 ábra Értékeket tároló egység kapcsolási rajza.....	87
10-10 ábra PI szabályozó kapcsolási rajza.....	88
10-11 ábra Szabályozójelek összegzőjének kapcsolási rajza	88
10-12 ábra PWM jelgenerátor kapcsolási rajza.....	89
10-13 ábra Hardver ko-szimuláció kapcsolási rajza	90

Táblázatok

Táblázat 1 Ziegler-Nichols által előírt paraméterezés	25
Táblázat 2 Módosított Ziegler-Nichols paraméterezés túllövés ellen	25

1. Bevezető

Napjainkban egyre elterjedtebb a repülő járművek használata. Jelenleg az álláspont ott tart, hogy különböző feladatokat próbálnak megvalósítani kisebb nagyobb drónok. Egyik legelterjedtebb drón típus a négypropelleres távirányítóval reptethető drónok, melyeket quadkopternek szoktak hívni. Ezen rendszerek már elérhetőek egyszerű felhasználó számára is, bizonyos megkötésekkel. Ezen rendszerek tervezése és szabályozása kihívás a mérnökök számára. Egy quadkopter tervezése során a kihívás főleg a szabályozási algoritmus, mérőegységek és egyéb rendszeren használandó eszközök kiválasztása. Nem egyszerű feladatnak bizonyul ezen rendszerek tesztelése sem. A vízszintes síkban történő stabilizálásra alkalmazott szabályozási hurokban, vagy akár a kritikusnak számító IMU érzékelőről jelek beolvasása során fellépő hibák a rendszer károsodásához vezetnek. Általában a rendszerek tesztelésére különböző standokat terveznek, amelyek lehetővé teszik a rendszer laborban történő tesztelését.

2. Projekt célja

Az államvizsga dolgozat célja egy quadkopter szabályozása, valamint egy előírt szögpozíció követése. A rendszer megvalósítása hardveresen történik, FPGA-n, mivel az egyetemen már volt építve egy quadkopter ugyanerre a cérra, viszont más megközelítésből, úgy döntöttünk, hogy érdemes kipróbálni egy másik szemszögből is megvalósítani, továbbfejleszteni a rendszert. A megépített rendszerben, az érzékelők, a kommunikációs interfészek, IMU, ultrahangos távolságmérő, távirányító vevő, PWM vezérlők hardveresen viszont az érzékelőkről a jelek olvasása a szabályozási hurkok szoftveresen voltak megvalósítva Linux operációs rendszert használva. [12] A dolgozat célja az, hogy a rendszer szempontjából kritikusnak tekinthető szabályozási hurkok, amelyek biztosítják a rendszer stabilitását, és a szabályozási hurkokhoz kapcsolódó érzékelők jeleinek beolvasása, kommunikációs interfészek, hibaszámítás, rendszer orientációjának a meghatározása, zajszűrés, PID szabályozók hardveresen legyenek megvalósítva.

3. Elméleti megalapozás

3.1. Quadkopter általánosságok

A quadkopterek, amint azt nevük is mondja, négy propellerrel ellátott repülő rendszerek. Két egymással szemben, egy tengelyen elhelyezkedő motorok ugyanabba az irányba forognak, míg az ellenkező tengelyen ellenkező irányba. Ha mind a négy motor ugyanolyan erővel hajtja meg a

gépet, akkor a rendszer rendelt Z tengely mentén mozdul el, és nem fordul a Z tengely körül. . Abban az esetben, ha az eszközt az X illetve Z tengelyek körül szeretnénk megdölteni, akkor egyszerre két motor szögsebességét szükséges változtatni, mivel így egy motorpár által kiváltott forgatónyomaték nem befolyásolja a rendszer Z tengely körüli elfordulását. Ha viszont a rendszert a Z tengely körül szeretnénk forgatni, akkor két szemben levő motor erejét kell lecsökkenteni, és a másik kettőt megnövelni annyira, hogy a quadkopter ne kezdjen el zuhanni vagy emelkedni.

3.2. Szögek

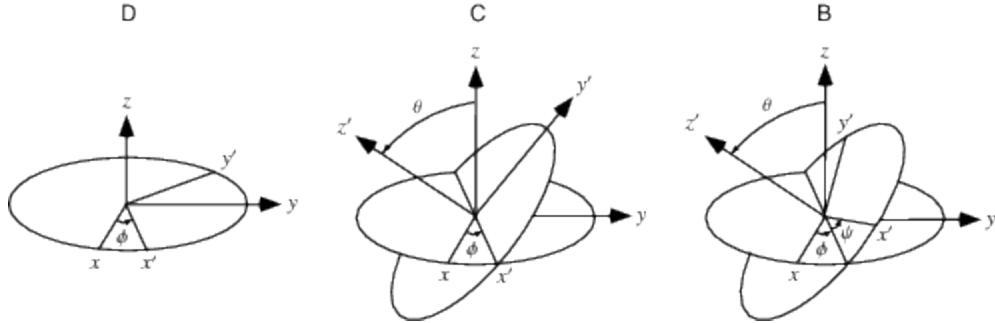
3.2.1. RPY szögek

Az RPY szögek a három tengelyhez képest mért szögfordulásokat jelölik. Az R, P, valamint Y megfelelője az angol *roll*, *pitch* és *yaw* szavakból erednek. Ezeknek a magyar megfelelője rendre, billenés, bólintás, és elfordulás. Ahhoz hogy meg lehessen állapítani, hogy a rendszer jelenleg milyen irányba van beállva, szükség van egy világkoordináta rendszerre, amely viszonyítási pont jelleggel rendelkezik. Ha ismert a világkoordináta rendszer tengelyeihez mért szögeknek az értéke, abban az esetben meg lehet mondani pontosan, hogy milyen az eszköz orientációja. Ez érvényes fordítva is. Ha a repülő szerkezetet tekintsük konstansnak, és érdekelt a világkoordináta rendszer iránya, akkor ebben az esetben is szögekre van szükség, viszont itt a szögek értéke ellentétes előjelűek kell, hogy legyenek, ha az eszközhöz rendelt koordináta tengelyeihez viszonyítunk. Ahhoz, hogy egy adott orientációt le lehessen írni, három forgatást kell elvégezni, ezért van szükség a még két keretre. Az első forgatás a z tengely mentén történik, amely során az x és y tengelyek elforgatjuk, viszont ugyanabban a síkban maradnak. Ezt a szöget ψ -vel szokták jelölni. Ez az újonnan kialakult keret az első jármű keret. A második járműkeret eléréséhez most már nem az inerciális keretet kell felhasználni, hanem az első jármű keretet. Jelen esetben a forgatást az y tengely mentén kell elvégezni. Ez adja meg, hogy a rendszer orra milyen magasan van. Ezt a szöget Θ -val jelölik. Az utolsó forgatásnál pedig a második jármű keretet kell, hogy felhasználni a test keretnek a megszerzséshoz. Mivel az z és az y tengely mentén volt már forgatás, egyértelmű, hogy most az x tengelyt kell elforgatni. Ez a szög megadja, hogy milyen mértékben van eldőlve a rendszer valamelyik oldalra. Ezt a szöget pedig Φ -vel jelölik.

3.2.2. Euler szögek

Az RPY szögekhez képest jelen esetben a különbség annyi, hogy a megkötés, hogy két egymásután következő forgatást nem ugyanazon a szögeken végzi el. Ez azt jelenti, hogy míg az RPY szögek

esetében rendre, az X, Y és Z tengelyek mentén történt a forgatás, Euler szögek esetében az X, majd Y, és végül ismét az X tengely mentén is lehetséges a forgatás. A 3-1 ábra szemlélteti, hogyan is fordulnak el a szögek rendje. Az Euler szögek egyik nagy problémáját gimbal zárnak nevezik. Az Euler szögek segítségével könnyen át lehet térti az inerciális keretből a test keretbe és fordítva.



3-1 ábra Euler szögek meghatározása, a koordináta tengelyek forgatása[18]

3.2.3. Gimbal zár

A Gimbal zár abban áll, hogy a három dimenziójú rendszer elveszti az egyik szabadságfokát. Ez akkor történik meg, amikor a rendszer orrának 90 fokot fordul, és így két tengely egymásra kerül. Ebben az esetben két forgatás közül bármelyik elvégezhető, mivel az eredmény ugyanaz lesz minden esetben. Ilyenkor a rendszer nem tud megbízható becsléssel szolgálni, ezért más módszerek ajánlottak, mint például a kvaterniók.

3.2.4. Kvaterniók

Az Euler szögek a kvaterniókkal szemben egyszerűek, viszont problémába ütköznek, a fennebb említett gimbal zár miatt. A kvaterniók négy részből állnak. Az első része valós értékeket, a másik három pedig imaginárius elemeket tartalmaz. Ebben az esetben már nincs szükség sem négy keretre, mint az Euler szögeknél, hanem csak kettőre, az inerciálisra, és a test keretre.

3.2.5. Alkalmazott koordináta rendszerek

Az Euler szögek segítenek egy eszköz, orientációjának a leírásában a Földhöz viszonyítva. Ebben az esetben több koordináta rendszert, keretet használnak fel. A négy keret, amelyről bővebben fogok tárgyalni: az inerciális keret, az első, és második jármű keret, valamint a test keret. Az inerciális keret a Földhöz van kötve, nem mozgó koordináta rendszer határolja be. Két különböző koordináta rendszert használnak az inerciális keret esetében. Az egyik a NED, a másik a SWU. A

NED keret esetében az x tengely északra, az y tengely keletre, míg a z tengely lefele mutat. Az SWU keretrendszer esetében az x délre, az y nyugatra, a z tengely pedig felfelé mutat.

3.3. IMU szenzorok

Az IMU szenzor rövidítése *Inertial Measurement Unit*-ot jelenti. Ezekben általában egy giroszkóp, és egy gyorsulásmérőt lehet megtalálni, újabbakban már magnetométert is szoktak beiktatni. Az IMU-k elsődleges célja egy eszköz orientációjának meghatározása. A szenzor által visszatérített értékek segítségével már lehet egy pozíciót számítani, abban az esetben, ha ismert egy kezdeti pont. Repülő eszközök esetében a viszonyítási pontként a Földet szokták választani, főleg akkor, amikor a rendszer nagyobb távolságokat tesz meg. A mért értékek folyamatos integrálásából a hiba folyamatosan nő, hosszú távú, folyamatos használatra nem alkalmasak. Az IMU szenzorok esetében több hibalehetőség áll fent. Az egyik az, ha egy állandó, invariáns térben van elhelyezve a szenzor, akkor sodródik-e idővel az átlagolt érték az elvárttól, vagy egy másik eset, amikor kétszer végezzük el ugyanazt a tesztet ugyanazon környezeti behatásokkal, de a két mérés között a rendszeren más hatások jelentkeznek. E mellett egy másik befolyásoló tényező a *zaj*. Egy harmadik tényező, amely befolyásolja az eredményt az a *nemlinearitás*. Mivel integrálunk, és nem folytonos rendszerként számolunk, hanem diszkrétként, éppen ezért nem lehet pontosan tudni, hogy két mintavételi periódus között milyen változások jelentkeztek.

3.3.1. Giroszkóp

A giroszkóp olyan eszköz, amelynek segítségével szögsebességet lehet mérni *fok/s* vagy *radián/s*. A giroszkóp esetében megemlíthető a sodródás, valamint az érzékelő nem-linearitásából adódó hibák. Ezért csak önmagában ezt a szenzort nem lehet alkalmazni hosszabb ideig tartó mérések alapján a szögpozíció meghatározásra. A meghatározott szöghiba csökkentése illetve kiküszöbölése miatt különböző becslő algoritmusokat, illetve egyszerre több típusú szenzort alkalmaznak.

Ezen eszközök kalibrációja program szintjén úgy történik, hogy egy olyan környezetben kell, hogy méréseket végezzünk, amelyben a giroszkóp nem mozdul el, így az elvárt szögsebesség átlaga nulla. A mérésekből átlagértéket számolva megkapható, hogy mekkora az adott tengely mentén a giroszkóp alapeltolása.

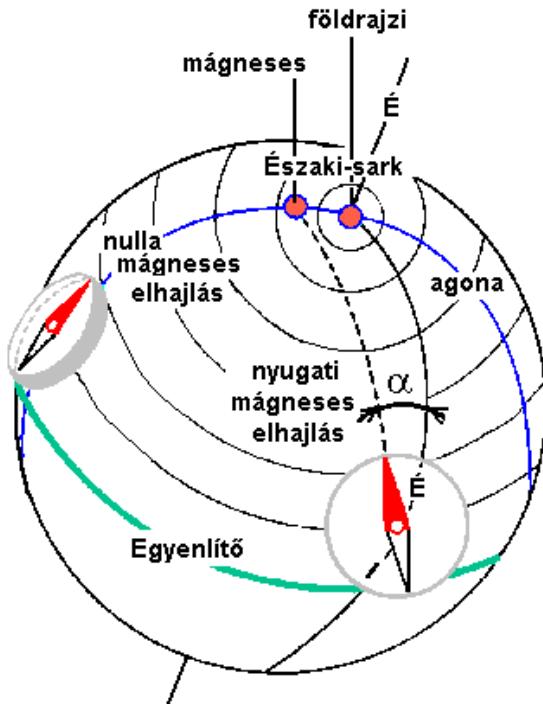
3.3.2. Gyorsulásmérő

A gyorsulásmérők a lineáris gyorsulást érzékelik. Abban az esetben, amikor az eszköz nem mozdul, egy helyben áll, akkor ez egy 1 g erőnek, 9.81 m/s^2 megfelelő értéket kellene, hogy mutasson. A gyorsulásmérőknek egy nagy előnyük a giroszkóppal szemben az, hogy nem sodródik hosszú távon. A problémája viszont, amint az előbb is említésre került az, hogy az általa visszatérített értékek zajosak. Ennek eredményeképpen alkalmaznak valamilyen szűrő algoritmust, ami lehet komplementer szűrő, Kálmán szűrő, vagy valami egyéb. A két szenzor (giroszkóp és gyorsulásmérő) segítségével két paramétert lehet meghatározni. Az egyik az, hogy mennyire dölt oldalra, valamint a repülő eszköznek az orra mennyire van fent vagy lent (billenés és bólintás). Azt, hogy a rendszer mennyire van elfordulva a rendszerhez rendelt Z (függőleges) tengely körül, azt ennek a két eszköznek a segítségével nem lehet meghatározni, ebben segít a magnetométer.

A gyorsulásmérő kalibrációja esetén a Föld gravitációs vonzóereje szerint lehet kalibrálni a szenzort. A szenzort először úgy kell elhelyezni, hogy a gyorsulásmérő egyik tengelye merőleges legyen a Föld felületére, így azon a tengelyen a legnagyobb értékeket lehet mérni, míg a másik két tengelyen a mért értékek megfelelnek az eltolásoknak. A mérések alapján egy egyenletrendszer írható fel, amely alapján meghatározzatok a gyorsulásmérő kalibrációs paramétereit. [13]

3.3.3. Magnetométer

A magnetométer segítségével a Föld mágneses terét lehet mérni. A kapott értékeket Gauss-ban vagy μT -ban kapjuk. A magnetométer segítségével most már meg lehet határozni, hogy a repülő eszköz milyen irányba van elfordulva a Föld mágnesé teréhez képest. Ennek egyik nagy hátránya az, hogy az érzékelő környezetében levő fémek, valamint olyan eszközök, amelyek mágneses, elektromágneses teret hoznak létre, torzítják a mágneses teret. Ezek a különböző hatások teljesen megváltoztathatják a szenzor által érzékelt mágneses tér irányát, ezáltal a quadkopter orientációját is. A mágneses térré hatással van egy úgynévezett mágneses elhajlás nevezetű jelenség, vagy más néven mágneses deklináció. Ennek az a lényege, hogy ha a Föld különböző pontjain mérjük egy rendszer orientációját, akkor nem biztos, hogy az északi sarkpont, amerre a mérés mutat arra is van. A mágneses elhajlás azt mutatja meg, hogy hány fokkal van elhajolva az északi sarkponthoz képest a mért orientáció. [2]



3-2 ábra Mágneses elhajlás[19]

3.3.4. Komplementer szűrő

A komplementer szűrő a szögek kiszámítására alkalmas algoritmus. Egy másik, ennél hatékonyabb megoldás a Kálmán szűrő, melynek erőforrásigénye nagyobb, viszont pontosabb eredményeket lehet ezzel elérni. Lényege abban áll, hogy veszi a gyorsulásmérő és a giroszkóp adatait, és az eredményt összegzi. Ezzel elérhető az, hogy minden érzékelőnek a jó tulajdonságaik legyenek összegezve, míg a negatívak eltörpülnek. A gyorsulásmérő segítségével kiszámított adatok nem annyira zajosak, így pontosabb az eredmény. A gyorsulásmérő segítségével kiszámolt szögpozíció idővel nem sodródik, és ez a kompenzáció jelleggel tud hozzájárulni a szög pontos értékének kiszámításához. Mivel a gyorsulásmérőn megjelenő zaj miatt a mérési eredmény pontatlannak tűnik, ezért az értékek összegzésénél nem lehet minden érzékelőnek a felét felhasználni. Ez alatt azt értem, hogy a két értéket súlyozni kell bizonyos mértéken, úgy, hogy a súlyzó paraméterek összege 1 legyen. A komplementer szűrő a (3.1) képlettel írható le. [1]

$$szög_{\text{új}} = \alpha * (szög_{\text{régi}} + szög_{\text{giroszkóp}}) + (1 - \alpha) * szög_{\text{gyorsulásmérő}} \quad (3.1)$$

3.4. A rendszermodell becslése

Egyes esetekben a rendszer “viselkedését” a rendszer modelljét, paramétereit nem ismerjük. Nem tudjuk, például, hogy melyek a pontos paraméterei, vagy esetlegesen a dinamikáját sem ismerjük. Egyes szabályozó algoritmusok, vagy a szabályozó paramétereinek a hangolása igénylik a rendszer modelljének/paramétereinek az ismeretét.

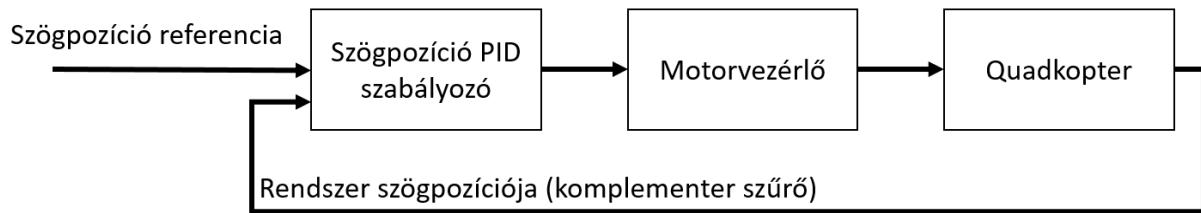
A rendszer paramétereit akkor lehet becsülni, ha a rendszer lineáris. Nem lineáris esetekben a rendszert csak egy linearizált pontban, ennek kis környezetében lehet az adott rendszer működését leírni. E mellett bizonyos szabályozó algoritmusoknak szüksége van a rendszer megközelített paramétereinek értékeire a helyes működésre. Egy rendszert becslő algoritmus alkalmazásakor több különböző módot lehet elkölníteni, hogy hogyan és mit akarunk becsülni egy rendszer esetében. Lehet szó grafikus, parametrikus, vagy egyéb becslésről. A parametrikus becslés esetén elkülöníthető on-line és off-line becslés, ami abban áll, hogy a rendszer paramétereit működése közben becsüljük, vagy elmentett mérési eredmények alapján határozzuk meg a paramétereket.

A projekt keretén belül az on-line parametrikus becslés van alkalmazva, mivel így elérhető, hogy változás jelenlétében a rendszer autonóm módon meghatározza a rendszer új paramétereit. Ahhoz, hogy a rendszer paramétereit becsüljük, szükségünk van bemeneti és kimeneti értékpárokra. Ezen paraméterek segítségével visszafejthető a rendszer dinamikája. E mellett szükséges megadni, hogy hányad fokú rendszerről van szó. Egy rendszer paramétereit csak akkor lehet helyesen megbecsülni, ha a becsülendő rendszer fokszáma megegyezik, vagy nagyobb, mint a becsült rendszeré. Ha tudjuk, hogy a becsülendő rendszer válasza milyen bizonyos bemenetekre, akkor el lehet dönteni, hogy hányad fokú rendszert kell becsülni. Abban az esetben, ha nem tudjuk, hogy a rendszer hogyan reagál bizonyos bemenetekre, akkor a rendszer fokszámának ajánlott legalább akkora fokszámot adni, amellyel biztosan meg lehet becsülni a rendszert. [11]

3.5. Szabályozók

A szabályozók olyan különálló egységek, amelyek arra szolgálnak, hogy a rendszernek olyan bemeneti értékeket adjanak, amelyeknek hatására elérhető egy kívánt működés. Az a kérdés, hogy milyen megoldások vannak, és melyik a legjobb szabályozó, amit bárhol, bármikor lehet használni, erre nincs egyszerű válasz. Ennek több oka is van. Az egyik az, hogy mit várunk el a szabályozótól és a rendszertől, gyors, vagy lassú választ? Abban az esetben, ha tudjuk, hogy egy lassú rendszerről

van szó, akkor értelmetlen egy gyors szabályozót használni. A 3-3 ábrán látható egy szabályozó általános tömbvázlata. [7]



3-3 ábra Szabályozó tömbvázlata

3.5.1. PID szabályozó

A PID szabályozó a legelterjedtebb szabályozó típus. Ez a szabályozó a mérnökök között közkedvelt az egyszerűsége, és könnyű használhatósága miatt. Ez a szabályozó bemenetként kap egy referencia értéket, és a rendszer jelenlegi kimenetét, és ez alapján számít egy kimeneti értéket. Ez így nagyon egyszerűnek hangzik, de mégsem ilyen egyszerű, mint azt a gyakorlat is mutatja. Habár a szabályozót legtöbb egy óra alatt implementálni lehet, mégsem lehet azonnal használni. Ennek három paraméterét jól be kell állítani annak érdekében, hogy a szabályozott rendszer az elvárásoknak megfelelően működjön. Valójában ez emészt fel a legtöbb időt az egész folyamatban. Hogy hogyan lehet ezt a folyamatot mégis valamilyen szinten felgyorsítani, erről majd később lesz szó.

A proporcionális tag hatással van hibára, vagyis az előírt érték és a jelenlegi érték közötti különbségre, a derivatív tag a hiba változását, tendenciáját veszi figyelembe, míg az integráló tag összegzi a múltbeli hibát.

A proporcionális tag befolyásolja a szabályozás sebességét. Ha nagynak választjuk meg a P tagot, akkor a rendszer gyorsan próbálja szabályozni a rendszer. Alacsony érték mellett pedig lassú lesz, viszont biztosabb, hogy az előírt értékre fog beállni. Túl nagy P tag mellett nagy valószínűséggel a rendszer instabil állapotba kerül, és irányíthatatlanná válik. Az optimális P tag esetében nem jelentkezik túllövés, éppen ezért rezgések sem jelennek meg, valamint a rendszer gyorsan beáll az előírt értékekre. A P típusú szabályozó kimenete a (3.2) képlettel írható le.

$$u(t) = K_p * e(t) \quad (3.2)$$

Az integráló tag segítségével korrigálni lehet a különböző hatásokat. Egy alacsony I tag mellett a szabályozó próbálja korrigálni az előírt és aktuális értékek közötti különbséget, viszont ez a különbség lassan csökken. Túl nagy I tag mellett a szabályozó azonnal korrigálni próbálja a szabályozott rendszer jelenlegi hibáját, így az eszköz irányítása nehezebbé válik. Ugyanakkor a nagy I tag mellett a rendszer hasonlóképpen működik, mint amikor a proporcionális tagja nagy. A rendszer oszcillálni kezd, viszont nagyobb frekvenciával. Egy optimális PID szabályozó esetében a quadkoptert nézve, egy jól megválasztott integráló tag mellett a rendszer nem oszcillál, és rövid időn belül az állandósult állapotbeli hibát 0-ra viszi le. Az integráló taggal rendelkező PI szabályozó kimenetét leíró képlet a (3.3) ábrán látható.

$$u(t) = K_p * \left(e(t) + \frac{1}{T_i} * \int_0^t e(\tau) d\tau \right) \quad (3.3)$$

A deriváló tag kitűnő megoldás abban az esetben, ha a proporcionális tag nagy, szabályozáskor a rendszernek van túllövése, illetve oszcillál, amíg stabilizálódik. Kis D tag mellett a rendszeren nem érzékelhetőek kisebb-nagyobb változások. Nagy D tag mellett a rendszer rezegni kezd, folyamatosan próbál előre szabályozni, aminek eredményeképpen a szabályozó folyamatosan rántja a quadkoptert, valamint így a motorok is felhevülnek. Mindhárom taggal rendelkező PID algoritmus kimenetét a (3.4) képlet írja le. [5]

$$u(t) = K_p * \left(e(t) + T_d * \frac{de}{dt} + \frac{1}{T_i} * \int_0^t e(\tau) d\tau \right) \quad (3.4)$$

3.5.2. Módosított PID algoritmusok

Az idők során a PID algoritmuson különböző változásokat is elvégeztek annak érdekében, hogy a szabályozó jobban megfeleljön az adott alkalmazásnak. Ilyen például a módosított D csatornával rendelkező PID algoritmus is, amelynek az a tulajdonsága, hogy a D tag a rendszer kimenetét veszi figyelembe. A PID algoritmus kimenetének a számítása a (3.5) képlettel írható le.

$$u(t) = K_p * \left(e(t) + T_d * \frac{dy}{dt} + \frac{1}{T_i} * \int_0^t e(\tau) d\tau \right) \quad (3.5)$$

3.5.3. PID hangolása

PID hangolása esetében ajánlott a rendszert szögsebesség alapú vezérlési üzemmódba helyezni. A második fontos tényező az ilyen rendszerek esetében az, hogy hol van a súlypontjuk. Egy középen elhelyezett súlypontú rendszer esetében ez könnyen irányítható, valamint a PID beállítása is könnyebb. A PID hangolása esetében több módszer is elterjedt, hogy melyik a legjobb, az már preferencia kérdése. Az egyik elterjedt megoldás az, ha már vannak értékek, amelyekkel már lehet vezérelni a rendszert, akkor megfigyelni, hogy hogyan működik, és ez szerint változtatni a paramétereket. Egy másik gyakran használt megoldás az, hogy minden paramétert olyan kicsinek választják meg, amennyire lehet, és ebből kiindulva növelik az értékeket rendre, és figyelik, hogy a rendszer jobban vagy rosszabbul reagál a változtatásokra. Kezdetnek ajánlott az első két kart külön-külön kalibrálni, majd ezt követően a helyben való forgást. A másik dolog, mit követni szoktak a hangoláskor, az az, hogy milyen sorrendben hangolják a paramétereket. Először a proporcionális tagot állítják be, ezt követően a derivatívat, és végül az integratívat.

3.5.4. Ziegler Nichols hangolás

Ezt a hangolási módszer abban az esetekben lehet alkalmazni, ha a rendszert a stabilitás határán is lehet tesztelni. A szabályozó paramétereinek hangolása azon alapszik, hogy a bemenő jel egy egyszerű P szabályozónak felel meg, amely esetén a K_p értékét növelik. Ezt az értékét nulláról növelik addig, amíg a rendszer elkezd szinuszosan lengeni állandó amplitúdóval. Ezt a K_p értéket K_{p_krit} kritikusnak szokták elnevezni. Egy másik paraméter, amelyet ekkor le kell olvasni, az a lengések periódusa. Ezt T_{krit} kritikusnak nevezik. Ziegler-Nichols előállott egy olyan táblázattal, amely alapján hozzávetőlegesen az optimális paramétereket lehet meghatározni bármilyen rendszerre, annak függvényében, hogy a PID szabályzó melyik tagokat tartalmazza. Az általuk biztosított paraméterek választását a (1) táblázat írja le.

	K_p	T_i	T_d
P	$0.45 * K_{p_krit}$	-	-
PI	$0.45 * K_{p_krit}$	$0.85 * T_{krit}$	-

PID	$0.6 * K_p_krit$	$0.5 * T_krit$	$0.12 * T_krit$
-----	-------------------	-----------------	------------------

Táblázat 1 Ziegler-Nichols által előírt paraméterezés

Ennek a módszernek az egyik hátránya az, hogy túllövéssel rendelkezik. Ezért a későbbiekben mások ezt tovább elemezték, és kidolgoztak ennek mintájára egyéb szorzótényezőket a PID paramétereinek, annak érdekében, hogy ne legyen túllövés. Az újonnan kapott értékek a (2) táblázatban szerepelnek.

	K_p	T_i	T_d
PID	$0.2 * K_p_krit$	$0.5 * T_krit$	$0.33 * T_krit$

Táblázat 2 Módosított Ziegler-Nichols paraméterezés túllövés ellen

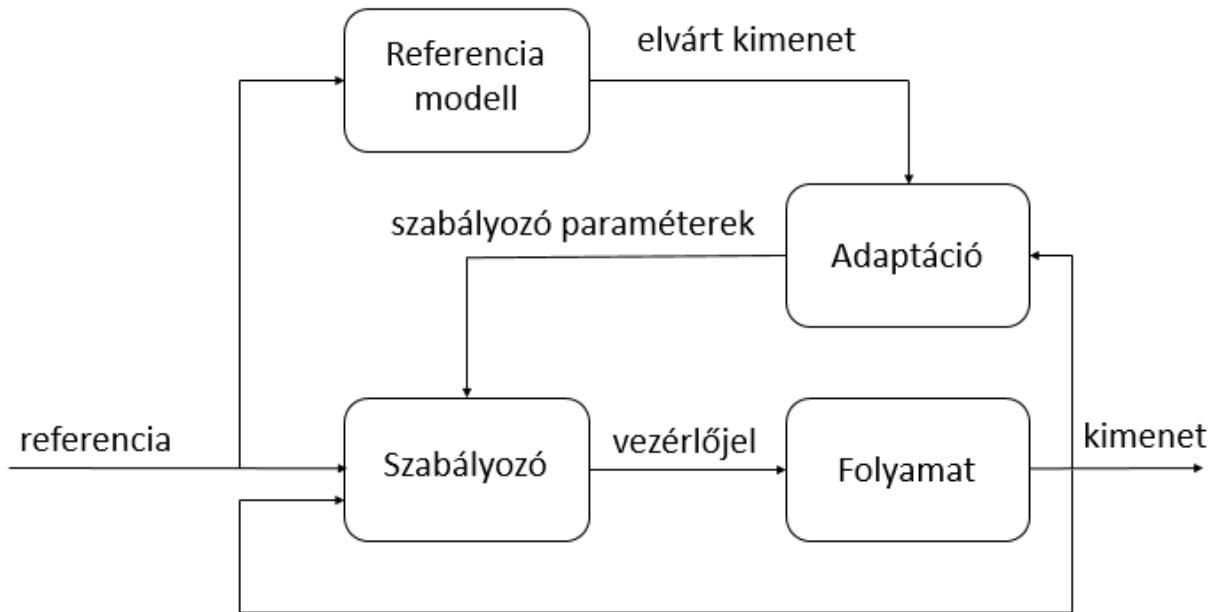
3.5.5. Adaptív szabályozás

Az adaptív szabályozók olyan szabályozók, amelyek a rendszer paramétereit próbálják valós időben megbecsülni, és ez alapján módosítják a szabályozó paramétereit is. Ez által, ha a rendszer valamely paramétere megváltozik, mint például a tömege, akkor a szabályozott folyamat még mindig megfelelően, az előírt modell alapján képes működik, egy bizonyos szintig. Az ilyen jellegű szabályozókat akkor is használják, amikor egy rendszernek bizonyos paramétereit nem ismerjük. [8]

3.5.6. Referencia modellen alapuló adaptív szabályozás (MRAC)

A referencia modellen alapuló szabályozás azt tüzi ki célul, hogy a szabályozandó folyamatunkat úgy próbáljuk meg szabályozni, hogy egy megadott rendszer működését kövesse. Ezzel azt lehet megmutatni úgymond a rendszernek, hogy milyen ideális választ várunk el tőle. A szabályozónak a lényege az, hogy a szabályozó és a folyamat rendszerének az összekapcsolása a referencia rendszert eredményezze. Ezen rendszerek esetén minden tudjuk, hogy adott referenciaértékre milyen választ fog adni a szabályozott folyamat. Többnyire az ilyen szabályozott folyamatok mellé elhelyeznek egy adaptációs algoritmust is. Ezzel biztosítják, hogy a rendszer paramétereinek változása esetén is a szabályozott folyamat kimenete kövesse a referencia modell kimenetét. A rendszerről általában már tudni lehet, hogy lineáris vagy sem, valamint, hogy milyen ennek a struktúrája, mármint tudjuk azt, hogy hány paramétert kell meghatározni. Ezt a rendszert úgy lehet legegyszerűbben leírni, hogy van egy rendszerünk, ami feltehetően másodfokú lengő rendszer. Ennek ellenére az a célunk, hogy olyan szabályozót készítsünk, amelyre a rendszerünk úgy

működik, mint egy jól meghatározott elsőfokú, melynek ismerjük az erősítését és az időállandóját. Ilyen szabályzót terveznek általában a lengéscsillapítók esetében is. A cél az, hogy egy impulzusra, lóketre, a rendszer ne lengjen, ha lehet egyáltalán. A referencia modellen alapuló szabályozók tömbvázlata a 3-4 ábrán látható. [9]



3-4 ábra Referencia modellen alapuló szabályozó tömbvázlata

3.5.7. RST szabályozó

Az RST rövidítés az angol *Reference Signal Tracking*-ből ered. Ennek lényege az, hogy a rendszer a referencia értékből, és a szabályozó aktuális paramétereiből számít egy vezérlő jelet, ami a folyamat bemenete lesz. Ezt követően egy becslő algoritmus segítségével megbecsüljük, hogy a rendszer milyen paraméterekkel rendelkezik, majd pedig ezek alapján meghatározzuk a szabályozó paramétereit. E szabályozó mellé beékelhető egy referencia rendszer is, amely segítségével el lehet érni, hogy a folyamat egy adott rendszer szerint működjön. Ennek a hátránya viszont az, hogy ha elvárjuk, hogy úgy működjön a folyamat, mint a referencia rendszer, akkor ennek a szabályozónak nincs információja arról, hogy a referencia rendszer kimenete milyen értékkel rendelkezik.

3.6. FPGA általánosságok

Az FPGA megnevezés a *Field Programmable Gate-Array*-ból ered, melynek magyar megfelelője az újrakonfigurálható digitális áramköröknek, röviden UKDÁ. A 3-5 ábrán egy FPGA chip látható.

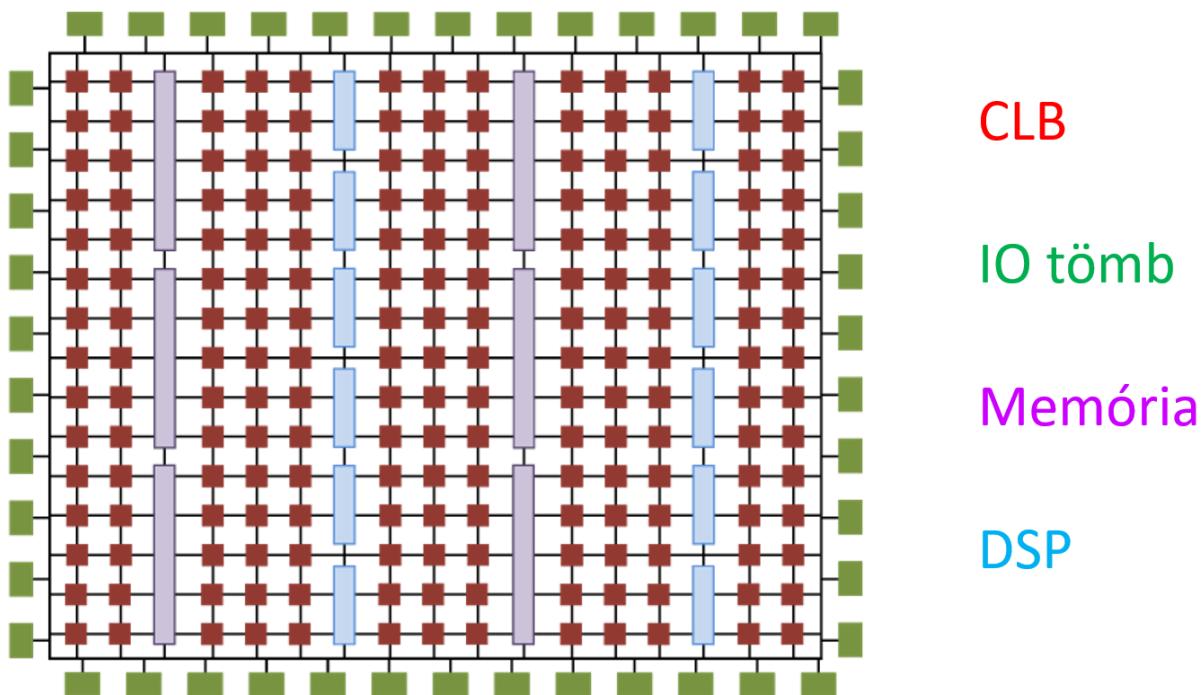


3-5 ábra FPGA chip[20]

Az FPGA-t, ha valamilyen rendszerhez kellene hasonlítani, akkor hozzá legközelebb az ASIC típusú áramkörök állnak. Az ASIC rendszerek alkalmazás specifikus integrált áramkörök, melynek lényege az, hogy az adott egység egyetlen feladatot lát el, és azt a legnagyobb sebességgel képes végrehajtani. Az ASIC áramköröket nem lehet újrakonfigurálni, tehát ha valamilyen okból kifolyólag újra kellene tervezni a rendszert, akkor újra le kell gyártani az adott egységet. Ezzel ellentétben az FPGA áramkörök olyan egységek, amelyek bármikor újraprogramozhatóak, és jól megközelítik egy ASIC típusú áramkör sebességét. A mikrokontrollerek is újraprogramozhatóak, mégis miért lenne előnyösebb egy FPGA-t használni helyettük. A mikrokontrollereket manapság már úgy tekintik, mint egy kis chip-be épített számítógép, melynek angol megfelelője a System on Chip (SoC). A mikrokontrollerek tartalmaznak egy mikroprocesszort, tárolóegységet, ki- és bemeneti egységeket. Ezeknek a célja általában egy feladatnak a végrehajtása, oly módon, hogy a lehető legalacsonyabb legyen a vezérlő egység fogyasztása. Az FPGA felépítéséből kiderül az, hogy miben különbözik a mikrokontrollerektől. Az FPGA rendszereket konfigurálható logikai, ki-

és bemeneti, valamint egyéb funkciókat betöltő blokkok alkotják. A konfigurálható logikai blokkokat angolul *Configurable Logic Block*-nak nevezik (CLB). Ezen blokkok tartalmaznak logikai kapukat és regisztereket, adatok tárolására. Ezen CLB-k összeköttetése mátrix-szerűen van megoldva, lényegében véve minden szomszédos CLB között egy huzal sor van, és ezeknek a megfelelő összekapcsolása a CLB-kel eredményez egy áramkört. A ki- és bemeneti blokkok lényege az, hogy a tömbön áthaladó jelet a megfelelő logikai szintre alakítják át, annak függvényében, hogy milyen szabványt választunk ki az eszköz konfigurációjakor. Ezen szabványok között megemlíthető a CMOS, TTL, stb. Az egyéb funkciókat betöltő blokkok között megemlíthető a *Block RAM* memória (BRAM), DSP modulok, valamint szorzó áramkörök. Ezek is hasonló módon kapcsolódnak egymáshoz és a CLB-khez, mint a CLB-k. Az FPGA-k segítségével alacsony szintű áramköröket lehet megvalósítani, amelyek Boole algebrára is visszavezethetők. E mellett nem feltétlenül függ egyik áramkör egy másiktól. Ezzel azt szeretném jelezni, hogy egy FPGA-n belül jelen lehet két áramkör is, amelyek párhuzamosan futnak, és nem függenek egymástól. Ugyanúgy ugyanilyen egyszerűsséggel egyszerre több áramkör is futhat probléma nélkül. Egy FPGA segítségével, ki lehet alakítani egy olyan architektúrát a rendszeren, amely megegyezik a mikrokontrolleren található architektúrával. E mellett ez több mint valószínű, hogy nem is használja ki teljesen az FPGA erőforrásait. Tehát, miben is különbözik akkor egy mikrokontroller egy FPGA-tól? Miért lenne szükség arra, hogy egy, már meglévő architektúrát újra elkészíteni az FPGA-ra, ha már úgy is kész van? Nem lenne egyszerűbb egy mikrokontrollert megvenni, mivel ezek amúgy is olcsóbbak? Esetenként de. Ha nem egy olyan alkalmazáson dolgozunk, amely időhöz kötött, mármint valós idejű rendszer, akkor elég, ha egy mikrokontrollert használunk, mert tudjuk azt, hogy az adatokat előbb vagy utóbb megkapjuk, feldolgozzuk és továbbküldjük. Ha késik, akkor sem történik probléma, legfeljebb a felhasználót váratja meg egy kicsit alkalmazástól függően. Mi van akkor, ha a rendszer, amelyre tervezük a hardvert időkritikus? Ebben az esetben, ha egy mikrokontrollert választunk céleszközként, akkor az első dolog, amivel számolni kell, az a soros végrehajtás. A mikrokontrollerek esetében nem lehet párhuzamosan utasításokat végrehajtani. Ez azt eredményezi valós idejű rendszerek esetében, hogy ha a mikrokontroller túllépi a neki megengedett időkorlátot, akkor ez súlyos problémákkal járhat. A bonyolult, de még elfogadható megoldás a programkód optimalizálása, viszont bizonyos feladatoknál adott pontnál a rendszer már nem képes gyorsabban befejezni a feladatot, mint az időkorlát, még akkor se, ha teljes mértékben sebességre van optimalizálva a rendszer. Itt jönnek a

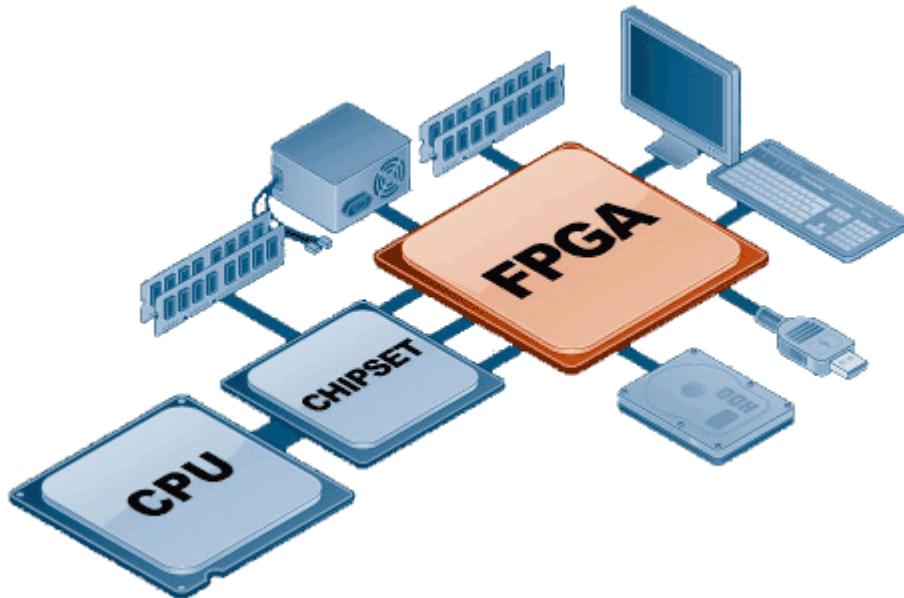
képbe az FPGA-k. Az FPGA-k képesek több műveletet egyszerre elvégezni, legyen az írás, olvasás, vagy valamilyen komplexebb művelet számítása. Például szüksége van a felhasználónak négy szenzor adatára, melyeket párból szükséges egyszer összegezni, valamelyen szenzorfúziós módszert alkalmazva. Tegyük fel, hogy az egyik módszer egy olyan módszer, amely mátrixműveleteket igényel. Ha ezeket a kapott adatokat még egy másik algoritmus is fel kell, hogy használja, akkor nagy a valószínűsége annak, hogy az adat valahol késni fog. A párhuzamosítás segítségével elérhető, hogy míg például a szabályozó számítja az aktuális szabályozó jel értékét, ez idő alatt a szenzoroktól kapott adatok már a fúziós algoritmuson haladnak keresztül. Az FPGA belső felépítése a 3-6 ábrán látható.



3-6 ábra FPGA belső felépítése

Amikor egy ehhez hasonló rendszert teszteltek minden lehetőségre, akkor az elkészített kapcsolási áramkörökből gyakran gyártanak nem módosítható, ASIC típusú áramköröket, mivel így még nagyobb sebességet is lehet biztosítani. Egy másik példa a neurális hálók tanítása. Abban az esetben, ha az egészet számítógép használatának segítségével szeretnénk megközelíteni, és a tanítandó neuronháló mérete nagyobbacska, akkor előfordulhat, hogy a számítógépnek napokra van szüksége, míg betanítja a neuronhálót a rendelkezésre álló adat segítségével. Az FPGA-k

segítségével a neuronháló tanítását úgy lehet megoldani, hogy a háló neuronjait egyszerre tanítja párhuzamosan, így jelentősen lecsökkentve a szükséges számítási időt, valamint ez esetben a rendszernek nem szükséges menedzselnie sem az erőforrásokat. Ez nem teljesen van így, mivel az is meglehet, hogy méretesebb neuronhálók esetében nem lehet a háló összes neuronjának a tanítását párhuzamosan elvégezni, mivel nincs elég erőforrás erre, ezért valamilyen módon mégis szükséges elkülöníteni, hogy mikor melyik neuronokat tanítjuk. Hogyan történik az FPGA-n futó alkalmazások tervezése? Első lépésben meg kell határozni, hogy mit várunk el magától a rendszertől, mit kell, hogy ez csináljon. Második lépésben meghatározunk néhány kritériumot, mint például azt, hogy mennyi erőforrás áll rendelkezésre, mennyi idő alatt kell, hogy lefusson a program, valamint melyek a rendszernek a ki- és bemenetei. Ez követően megtervezzük a rendszer architektúráját, és ha valamilyen problémát észlelünk már a tervezési fázisban, akkor ezt már is lehet javítani. Ha már tudjuk, hogy a rendszer milyen feltételekkel rendelkezik, és hogyan néz ki, akkor neki lehet fogni az implementálásnak. Az implementálás után ellenőrzük a rendszert, hogy megfelelően működik-e, úgy fizikailag, mint szimulációban. Ha bárhol valamely fázisok között probléma észlelhető, akkor vissza kell térti egy előbbi tervezési fázishoz, vagy akár a legelejére, kritériumok meghatározásához, ha nincs megoldás a problémára az adott kritériumok mellett. Manapság már szinte minden hardver szintű réteg fölé kiépítenek egy szoftveres réteget is. Miért jó ez? Különböző paramétereket lehet megosztani a hardveres részekkel, szoftveren keresztül, adatokat lehet gyorsabb módon szerezni, felhasználni. Ezt mutassa be a 3-7 ábra is.

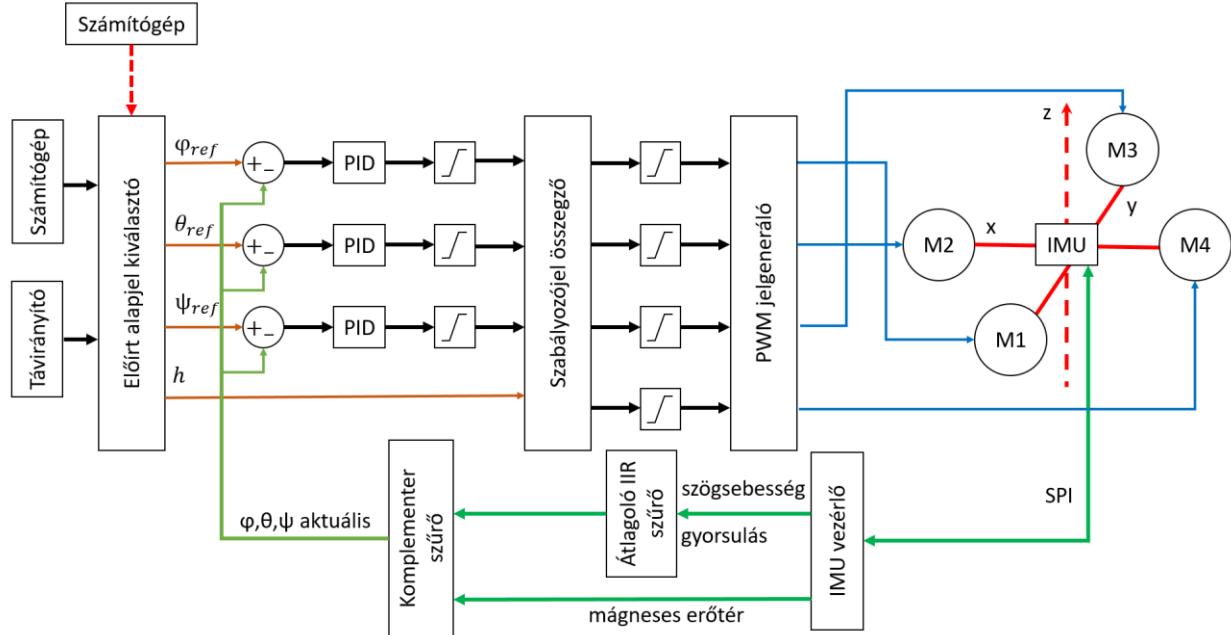


A szoftveres és hardveres rétegek közé beszúrnak egy harmadik, köztes réteget is, amely a szoftveres réteg elől elrejti azokat a műveleteket, amelyek a hardverrel való kommunikációt intézik. Egyszerűbben leírva, a köztes réteg egy szolgáltatást biztosít a szoftveres rétegnek. A köztes rétegen találhatóak az írás, olvasás műveletek, esetlegesen néhány paraméterezési függvény. A rendszer tervezése során felmerülnek bizonyos kérdések, amelyek az implementálás során folyton előjöhetnek. Az első kérdés, amely mindig jelen van az, hogy működik-e? A második kérdés, amely az első előtt van, hogy mit lehet újra felhasználni előző, vagy más projektekből, gondolok most IP magokra, már megírt modulokra. Például nem szükséges egy SPI vezérlőt megírni többször, ha már van egy készen megírva, melyet csak használni kell. Ha nekünk nincs ilyen modul elérhetőségünkben, akkor utána lehet nézni, hogy létezik-e ilyen a hálón, ha igen, akkor ennek mi a beszerzési ára, és megéri-e vagy sem. Egyszerűbben szólva, nem kell újra feltalálni a kereket, ha nem vagyunk rászorulva. Ugyanitt a tervezési fázis legelején, ha hardveres és szoftveres részben is gondolkodunk, akkor érdemes meghatározni, hogy a rendszer mely részei fognak hardveren futni, és melyek szoftveren. Lehet, hogy egy modul részt meg lehet oldani úgy hardveresen, mint szoftveresen, de el kell dönteni, hogy ezt hol éri meg inkább implementálni. Milyen programozási nyelveket lehet használni az FPGA-k programozására? Előbb azt kell letisztázni, hogy melyik tervezési szinten szeretnénk implementálni az adott modult. Alacsony, regiszter és logikai kapu szinten, vagy pedig magasabb, algoritmus szinten? Alacsonyabb szinten jobban lehet optimalizálni a rendszert, viszont itt a tervezés sebessége jóval lassabb. Ezen a szinten valamilyen hardver leíró nyelvre van szükség, mint például a *Verilog*, vagy *VHDL*. Magasabb szinten gyorsabban megy a tervezés, itt már C-ben is lehet programozni FPGA-keket. Ez előnyös azon programozók számára, akik már jártasak a C, vagy valamely hasonló programozási nyelvben, viszont még nem jártasak a hardver leíró nyelvek világában.

4. Gyakorlati megvalósítás

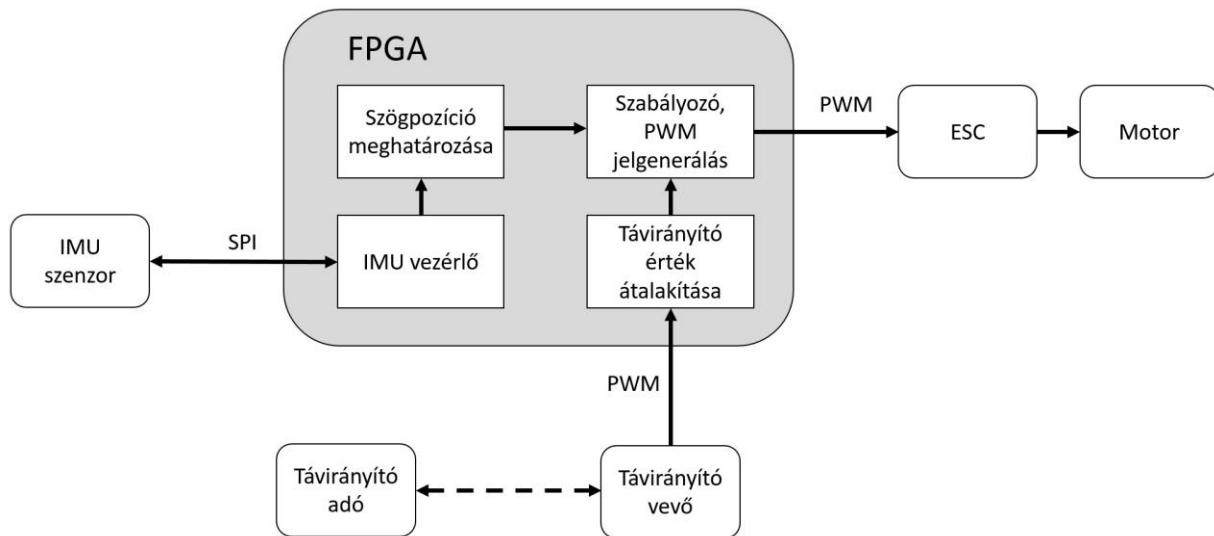
4.1. Rendszer architektúrája

A 4-1 ábrán látható a rendszer architektúrája.



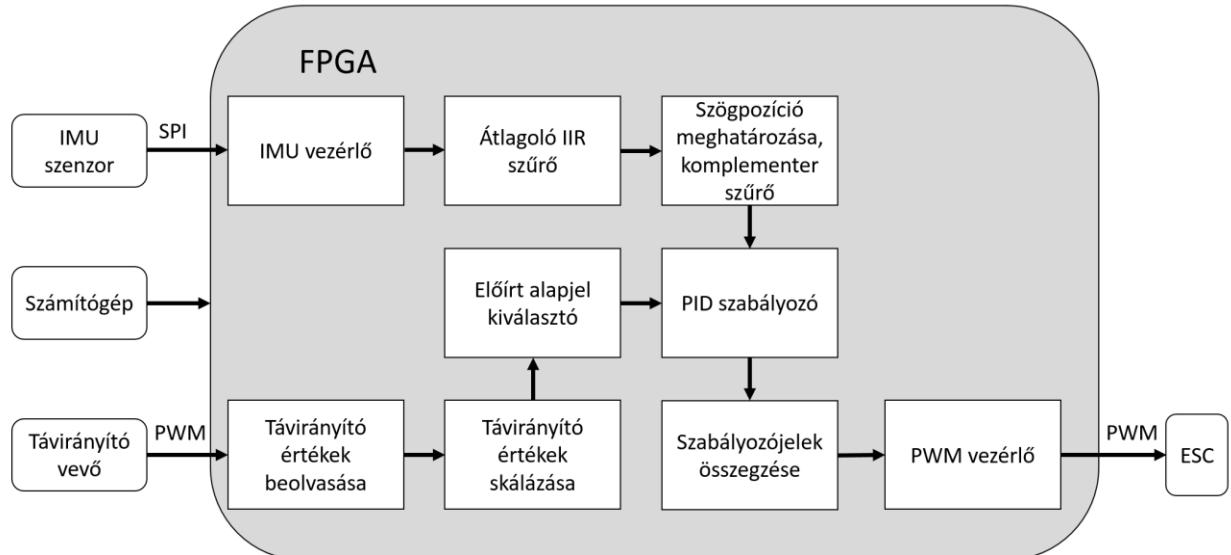
4-1 ábra A rendszer felépítése modulokra lebontva

A 4-2 ábrán látható tömbvázlat megegyezik az előbb bemutatott tömbvázlattal, csak egy másik szempont szerint van kialakítva. A projekt keretén belül használva van egy Pmod NAV IMU szenzor, amellyel SPI interfész segítségével kommunikálok, egy Art-Tech E-Fly 100C analóg távirányító, amely modulált PWM jelet továbbít. A távirányítótól és az IMU szenzortól kapott adatok alapján történik a rendszer szabályozása. Az FPGA a sebességvezérlőknek (ESC) pedig PWM jelet továbbít.



4-2 ábra A rendszer architektúrája

A 4-3 ábrán látható tömbvázlat egy részletesebb leírás arról, hogy konkrétan milyen modulok vannak beépítve rendszerbe. Az IMU szenzortól az adatokat egy vezérlő egység olvassa be, és ez küldi le a konfigurációs paramétereket is. A szenzortól kapott gyorsulásmérő és giroszkóp adatok egy átlagoló IIR szűrőn vannak keresztül vive, a zajok csökkentése érdekében. A szögpozíció meghatározására egy komplementer szűrő van beépítve, ennek segítségével van meghatározva billenés és bólintás szögek. A szögpozíciók a PID szabályozókhöz kerülnek át. Csak a szögpozíciók ismeretével nem lehet kezdeni sok minden, kell egy referencia érték is a szabályozónak. A referencia megadására két lehetőség van kidolgozva. Az egyik az, hogy az előírt érték egy távirányítótól érkezik, a másik pedig az, hogy számítógéptől kapja az adatokat. Az analóg távirányító vevője PWM jeleket ad ki, amit egy beolvasó egység kezel. Mivel a jeleket skálázni kell a megfelelő érékekre, ezért a jelek skálázásával egy külön egység foglalkozik. Innen az adatok két felé ágaznak. Az egyik a referencia kiválasztó egységbe, amely segítségével meg lehet adni a rendszernek azt, hogy honnan vegye az előírt értéket. A másik út egyenesen a szabályozójelek összegző moduljába megy. Ezen az úton egy jel megy, melynek értéke összegeződik a szabályozójelekkel. A kapott előírt referencia és a rendszertől kapott aktuális értékek alapján a PID szabályozó kiszámítja a motorokra adandó szabályozójelet. A szabályozójelek összegzésével egy másik modul foglalkozik, így abban az esetben, ha PID szabályozót le kellene cserélni, az összegző egységet nem kell újra implementálni. Az összegző modul által kiszámolt értékek egy PWM vezérlőbe kerülnek, amely meghajtja a kapott paraméterek alapján a motorokat.

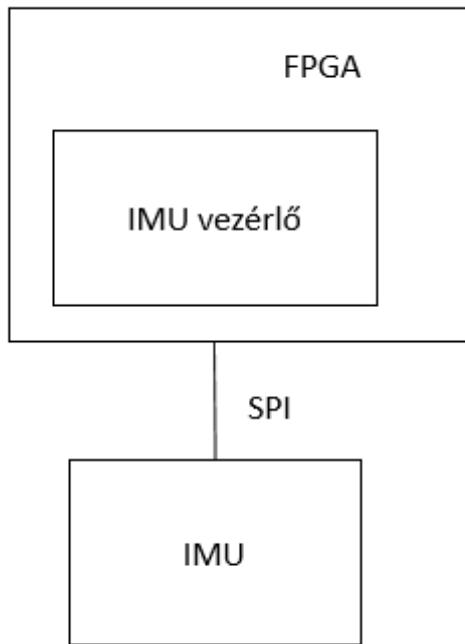


4-3 ábra A rendszer belső felépítése modulokra leosztva az FPGA-ban

A rendszer belső kapcsolási rajza a 4-1 függelékben található.

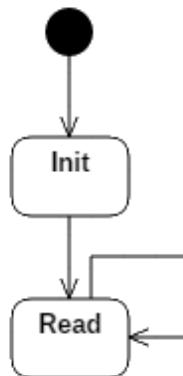
4.2. Pmod NAV IMU

A Pmod NAV szenzor két alegységből tevődik össze, az LSM9DS1, amely tartalmaz gyorsulásmérőt, giroszkópot, és magnetométert, valamint az LPS25HB modulból, amely egy barometrikus nyomásmérőt tartalmaz. Ezekkel az alegységekkel SPI és I2C interfészken keresztül lehet kommunikálni, viszont az eszköz úgy van kialakítva, hogy csak SPI interfész van jelen, egy meghatározott lágkiosztással. Az eszköz áramfogyasztása normál üzemmódban mindenkorának közelében 5 mA, amely hatása elenyészik, a négy motor fogyasztása mellett, tehát e felett el lehet tekinteni. A szenzor segítségével a következő maximális mérési tartományokban lehet mérni. A giroszkóppal $\pm 2000 \text{ fok/s-t}$, a gyorsulásmérővel $\pm 16 \text{ g-t}$, a magnetométerrel $\pm 16 \text{ gauss-t}$, a barometrikus nyomásmérővel pedig 260 és 1260 hPa között lehet mérni. A mérőrendszer által leküldött adatok 16 biten vannak ábrázolva kettes komplemens módon, ami lehetővé teszi a pozitív és negatív értékek használatát. Egyedüli kivétel a barometrikus nyomásmérő, amely az adatokat 24 biten ábrázolja. Az általam használt beállítások miatt, a giroszkóptól és a gyorsulásmérőtől 900 Hz, a magnetométertől 80 Hz, a barometrikus nyomásmérőtől pedig 25 Hz-es frekvenciával kapom az adatokat. [25] A rendszer tömbvázlata a 4-4 ábrán látható.



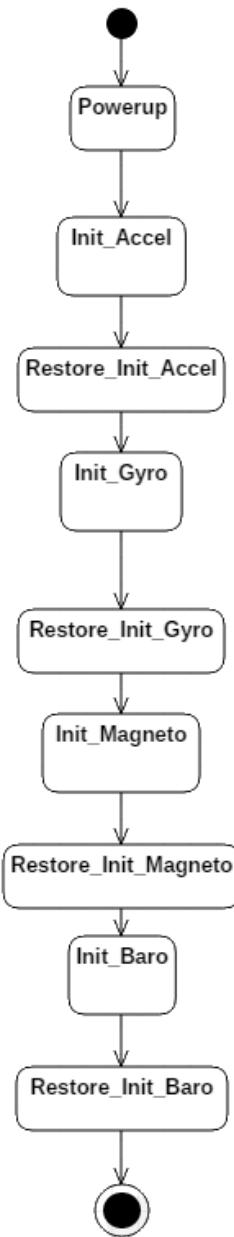
4-4 ábra FPGA és IMU kapcsolásának tömbvázlata

A rendszer tartalmaz egy órajel-osztót, mivel az IMU szenzor maximálisan 10 MHz-es frekvenciával tud adatot küldeni, illetve fogadni. E mellett található egy SPI interfész, egy ROM memória, valamint egy vezérlő, amely ennek az ütemezéséért felel. Az általam megvalósított automata két részből tevődik össze. Egy inicializáló részből, melyben paraméterezzi a szenzort, és egy második fázisból, amely egy végtelen ciklusban folyamatosan olvassa a szenzorjeleket. Az automatának két főbb része a 4-5 képen látható.



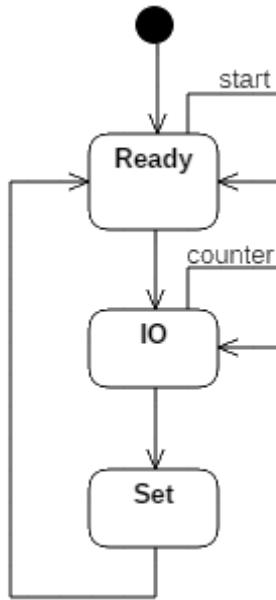
4-5 ábra IMU vezérlő működésének állapotdiagramja

Az inicializáló részt négy külön részre osztottam fel. Ennek az az oka, hogy amikor az eszköz valamelyik szenzorjától szeretnék olvasni adatot, abban az esetben az adott szenzort kiválasztó sínt aktiválnom kell, jelezvén, hogy adatot akarok küldeni vagy fogadni. A gyorsulásmérő és a giroszkóp ugyanazon a vezérlősínen, míg a magnetométer és a barométer külön-külön vezérlősínen helyezkednek el. A gyorsulásmérő és a giroszkóp vezérlősínjét külön választottam az automata belsejében, mivel így jobban el lehet határolni a kettőnek a vezérlését, és átláthatóbb a rendszer. Az eszköz inicializálása a 4-6 ábrán látható.



4-6 ábra IMU szenzor inicializálásának állapotdiagramja

Az inicializálási fázisban a szenzorokat sorban paraméterezem fel és indítom el. Például, az *Init_Accel* állapotban a vezérlőegység megnézi, hogy le van-e küldve az összes adat, és ha nincs, akkor megadja, hogy melyik a következő adatcsomag, amelyet az SPI interfész ki kell, hogy küldjön. Az SPI interfész külön modulként szerepel a rendszerben, ezért az automata vezérlő egysége meg kell, hogy várja minden esetben, amíg az SPI modul végez az adatcserével. Az SPI állapotdiagramja az 4-7 ábrán látható.



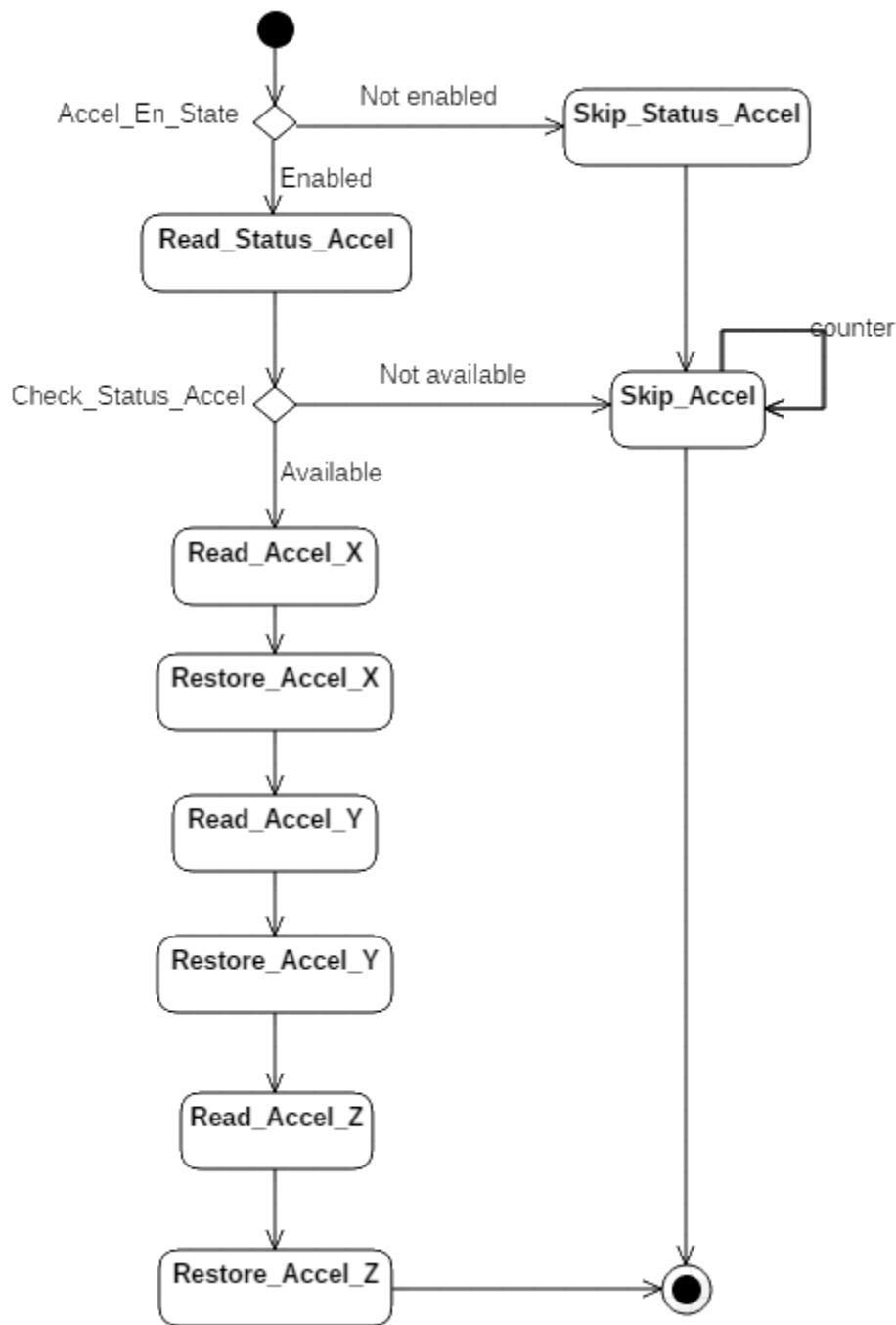
4-7 ábra SPI modul állapotdiagramja

Az SPI interfész modul bemenetei között található az órajel, reset, start, küldendő adatbitek, valamint egy adatút, amelyre az IMU küldi ki az adatokat (SDI), valamint egy engedélyező bitet, amely elindítja az adatok kiküldésének folyamatát. Kimenetek között található a kapott adat, egy jelzőbit, amely arra mutat, hogy megérkezett az adat, valamint egy adatvonal, amelyen keresztül a rendszer küldi az adatbiteket az IMU-nak (SDO). Az SPI nem foglalkozik azzal, hogy az adatbitek, amiket küld olvasásra, vagy írásra vonatkoznak, ez úgy van megépítve, hogy az első nyolc bit, amit kiküld, a között van a vezérlő bit, amelyet a vezérlő egység állít össze. Az SPI modul folyamatosan tárolja és küldi az adatokat. Abban az esetben, amikor a modul ír, akkor nem történik semmi különös, mivel bemenetként azt a bitet olvassa be folyamatosan, amelyet az IMU kitett a kimenetére utoljára, vagy amit beállított a küldés után. Olvasáskor az IMU beolvassa az első 8 bitet, a második 8 bitet pedig figyelmen kívül hagyja, és a kimenetre kiteszi az eredményt. Az SPI

vezérlő pedig az első 8 beolvasott bitet eldobja shift regiszter segítségével, mivel ekkor történik meg a kérés az olvasásra, és a megcímzés. Az ezt követő 8 bit van csak elmentve, melyet a tranzakció végén kitesz a kimenetére, és ezzel együtt jelzi a vezérlőnek, hogy készen van az adat küldése / fogadása.

Az IMU inicializálásához szükséges adatok egy ROM memóriában vannak eltárolva, ahol megtalálhatóak a szükséges konfigurációs adatcímek, valamint adatbitek, ezen felül azok az adatcímek, ahonnan le kell, hogy kérjük a kiválasztott szenzor adatait. Ez úgy van felépítve, hogy egy belső számláló segítségével az összes adaton végigmegy a modul, és az adatot kiteszi a kimenetre. A vezérlő egység jelet ad a modulnak, hogy mikor lehet küldeni a következő adatot. A modul felépítése úgy van megtervezve, hogy miután az konfigurációs adatok ki vannak olvasva, utána az egység csak olyan adatokat küld ki, amelyek címző bitek, ebben az esetben az adatokat lehet kiolvasni anélkül, hogy a vezérlőegység gondoskodna a ROM megfelelő megcímzéséről.

Visszatérve a vezérlő egységhez, minden konfigurációs adat leküldése után szerepel egy úgynévezett *Restore_Init* állapot, amelyet szükséges volt betenni a számláló lenullázása érdekében. Az inicializálást befejezően következik a kiolvasó rész, amely egy végtelen ciklusban van. A vezérlő bemenetei között található egy kapcsoló, amely segítségével el lehet indítani az adatok lekérését, valamint négy kapcsoló, amellyel kiválasztható, hogy mely szenzoruktól kellenek az eredmények. Ezen kívül bemenetek között található a kapott adat, amit az SPI vezérlőtől kap. Kimenetként egy start bitet tartalmaz. Ezen kívül még tartalmaz mindenik szenzornak, és szenzoronként mindenik koordinátatengelynek egy kimeneti adatsínt, mivel ez rakja össze a több bájtóból álló értékeket. A vezérlő egység nem adja meg, hogy írást vagy olvasást hajtunk végre, mivel ez már integrálva van a ROM modulba. Ez az adatsorozat első bitje. A magnetométer, és a barométer tartalmaz egy második vezérlőbitet, amely a második bit a sorozatban. Ennek segítségével elérhető, hogy egyszerre több adatot küldjön ki a szenzor egymás után. Az automata egyszerűségének kedvéért ez a funkció nincs alkalmazva. A 4-8 ábrán látható a gyorsulásmérő értékeinek beolvasása.



4-8 ábra Gyorsulásmérő adatai olvasásának állapotdiagramja

Abban az esetben, ha nincs szükség az adott szenzor adataira, akkor a már említett kapcsoló segítségével ez megtehető. Ekkor a vezérlő jeleket küld a ROM memóriának, hogy a következő adat szükséges, jöhet a következő. Ez addig megy, amíg teljesen átugorja a jelenlegi szenzor memóriarekesztét. Ezt követően jön a következő szenzor, mely ugyanígy kerül kiértékelésre.

Abban az esetben, amikor szükségünk van a szenzor adataira, akkor a modul előbb beolvassa ennek állapot, vagy más szóval státus regiszterét. Ebből kinyeri, hogy van-e újadat, vagy sem. Abban az esetben, ha nincs újadat, akkor a vezérlő úgy működik, mintha az adat nem érdekelne, más szóval átugorja ezt a részt. Ha van adat, akkor hozzájárul az adatok kinyeréséhez. Ha ezt befejezte, akkor következik a következő szenzor. Az adatok kinyerése esetében a fennebb említett szenzorok mellett még található két hőmérő is. Az egyik a magnetométerhez van hozzákapcsolva, a második pedig a barométerhez. Ezeket az adatokat is ki lehet ugyanúgy olvasni, mint az előbbieket.

4.3. Impulzust generáló modul

A rendszerbe be van építve több impulzus-generáló modul is. Ezeknek szerepe az, hogy különböző frekvenciájú jeleket generáljanak. A létrejövő impulzus frekvenciáját egy külső értékkel lehet meghatározni. Ez segít abban, hogy meghatározzuk, hogy milyen gyakran akarjuk elindítani a komplementer szűrőt, a szabályzót, és egyéb modulokat. Megvalósítás szempontjából ez a modul nagyon hasonlít ahhoz, mint ami a Pmod NAV IMU szenzor esetében volt használva, azzal a különbséggel, hogy itt az előírt frekvencia kívülről bármikor változtatható. Az impulzust generáló modul működését az Algoritmus 1 rész írja le.

```
Algoritmus 1 Impulzust generáló
    ameddig számláló<előírt érték
        számláló növelése
        vége
        impulzus kiküldése
        számláló lenullázása
    vége
```

4.4. Átlagoló IIR szűrő

A szenzorral elvégzett mérések során kiderült, hogy a zajok jelentős mértékben befolyásolják a gyorsulásmérő és a giroszkóp jeleit. Mivel a zajok jelentősen befolyásolták a szögek értékének meghatározását, amikor a motorok működésben voltak, ezért különböző szűrők voltak tesztelve a rendszeren. Elsősorban a komplementer szűrő alfa paramétere volt változtatva, viszont ez minimálisan segített a rendszer orientációjának a meghatározásában. Ezt követően a szenzorba beépített szűrők paraméterei voltak változtatva, viszont ez sem javított eleget az eredményeken. Utolsó sorban egy egyszerű átlagoló IIR szűrő is volt tervezve a rendszerhez. Az szűrő kimenetének kiszámítására alkalmas képlet a (4.1).

$$\text{adat}_{\text{új}} = \text{adat}_{\text{m\'ert}} * \alpha + \text{adat}_{\text{r\'egi}} * (1 - \alpha) \quad (4.1)$$

Mivel a rendszerben nem egy, hanem 6 különböző adatot kellett szűrni, ezért a párhuzamosítás lehetőségét elvetettem, mivel így 12 darab szorzó áramkör lenne felhasználva. A szűrő úgy van megvalósítva, hogy egy szorzó áramkört, egy összeadó, egy ideiglenes, a 6 eredménynek szükséges tárolót, valamint egy vezérlőt tartalmaz. Ennek segítségével erőforrások szabadulnak fel, amelyeket a későbbiekben lehet felhasználni. Az első órajel alatt a vezérlő beolvassa az első adatot, és összeszorozza az α -val. Ez idő alatt kiszámítja az $1 - \alpha$ értéket is. Következő ciklusban eltárolja az eredményt egy ideiglenes tárolóba. A harmadik órajel alatt összeszorozza a régi adatot az $1 - \alpha$ -val. A negyedik órajel alatt a két eredmény összeadódik, az ötödikben már tárolódik az eredmény, és ugyanebben az órajelben a következő adat már szorzódik is az α -val. Összességében kiszámítható, hogy a 10 ns-os periódusú órajel alatt, a vezérlő a 6 jel szűrésére 250 ns alatt kiszámítja az eredményt, ami több, mint 1000 gyorsabb a beolvasott értékek frekvenciájánál. Az átlagoló szűrő két bemeneti paraméterrel rendelkezik. Az egyik segítségével meg lehet adni, hogy milyen gyakran mintavételezzen a rendszer, a másik jel segítségével pedig az α értékét lehet beállítani. Az átlagoló IIR szűrő kapcsolási rajza a 10-4 függelékben található meg.

4.5. Komplementer szűrő

Amint az elméleti leírásban is említve volt, ennek az módszernek a szerepe, hogy a különböző hibákat, zajokat lecsökkentsük, és megkapjuk a szögpozíciók értékét. Ennek a megvalósítására két modul volt létrehozva. Egy egyszerű verzió, amely erőforrásigénye jelentős, valamint egy újabb változat, amely erőforrásigénye alacsonyabb, viszont több időbe telik, amíg végigfut. Az egyszerű verzió hátránya az, hogy 27 szorzó áramkört, és 5 Cordic modult igényel, ha mindenhol szögpozíciót ki akarjuk számolni ennek segítségével, nem megemlíthető az összeadó és kivonó áramkörök. A Cordic modulok segítségével lehet trigonometriai számításokat elvégezni, mint a szinusz, és a koszinusz függvények. Jelenlegi állapotában a komplementer szűrő segítségével csak két szögpozíciót számolok ki, és jelenleg a szűrő 13 szorzó áramkört, és egy Cordic modult használ fel. Tudva azt, hogy a projekt keretén belül alkalmazott Zybo FPGA mindössze 80 beépített szorzó áramkört tartalmaz, ez arra kényszerített, hogy újratervezzem ezt, a lehető legkevesebb erőforrás igénybevételével. Figyelembe véve az FPGA órajelének a frekvenciáját, és azt, hogy a szűrőt nincs értelme 10 KHz-es frekvenciával járatni, ezért nyugodtan alkalmazhatok egy hosszabb, időigényesebb automatát. [4]

Az egyszerűsített, erőforrás-igényes komplementer szűrő a szorzó, összeadó, kivonó és Cordic áramköri elemek mellett tartalmaz egy vezérlő egységet is, amely szabályozza, időzíti az egységeket, hogy mikor melyik működjön, és mikor mit csináljon a rendszer. A komplementer szűrő vezérlője jelen állapotában úgy van megvalósítva, hogy ez számolja ki az eltolásokat és ez skálázza az IMU szenzortól kapott szűrt méréseket. Ezt követően a gyorsulásmérő értéki skálázva vannak a Cordic modulnak megfelelően, mivel ez az egység csak egy jól meghatározott bemenetek alapján tud működni. Ezzel együtt a giroszkóp értékei is szorzódnak az eltelt idővel, ezáltal meghatározható a giroszkóp alapján, hogy a szögpozíció mennyit változott egységnyi idő alatt. Miután a Cordic modul kiszámította a gyorsulásmérő adatai alapján a szögeket, elkezdődik a gyorsulásmérő és a giroszkóp adatainak összesítése. Az adatok összesítése mindenkor szögsebesség esetén ugyanakkor történik. A szűrő működését az Algoritmus 2 mutatja be.

```
Algoritmus 2 Komplementer szűrő
    eltolások korrigálása
    értékek skálázása
    szögpozíció változásának kiszámítása (giroszkóp)
    szögpozíció változásának kiszámítása Cordic modul segítségével(gyorsulásmérő)
    szögpozíciók értékének összegzése
    vége
```

A komplementer szűrő kapcsolási rajza kiegészítve az átlagoló IIR szűrővel és az IMU szenzorral a 10-2 függelékben található meg.

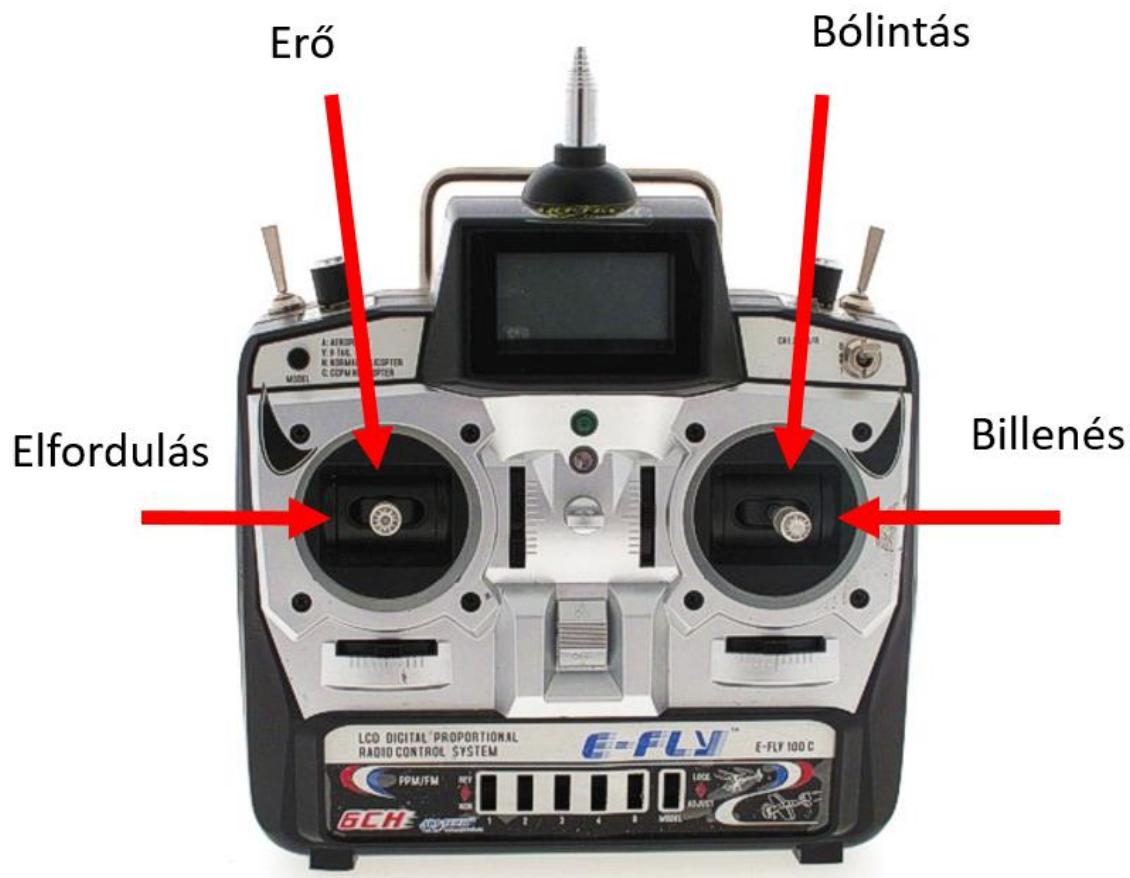
Amint az már az fennebb is meg volt említve, volt egy második verzió is készítve a komplementer szűrőből, mely sokkal kevesebb erőforrást igényel, viszont jóval több ideig tart, amíg kiszámolja a szögpozíciókat, viszont az FPGA sebessége kompenzálja ezt a hátulütőjét. Ez a modul úgy volt megtervezve, hogy a harmadik szögpozíciót is számolja, a függőleges tengely mentén. A szűrő megtervezésekor a vezérlő egységet két részre tagoltam. Az egyik rész vezérli a szögpozíciók számítását, a második pedig a függőleges tengely szögpozíciójának kiszámításához szükséges korrigáló paramétereket számolja ki. Szükséges a magnetométer adatait a Földdel párhuzamos síkba elhelyezni a rendszer, ezért van szükség a második kompenzáló egységre.[3] Ez így optimálisan működik, mivel mindenkor vezérlő rész tartalmaz egy Cordic egységet. A fő vezérlő egység, amellyel a szűrő számolja a szögpozíciókat, funkcionális szempontból ugyanúgy működik, mint az előbbi, erőforrás-igényes. Gyakorlatilag az egyszerűsített változat egyszerre csak egy műveletet hajt végre. Az egyszerűsített változat megtervezése után, mivel ez nem volt eleget tesztelve különböző esetekre, mint például fizikai rezgés, amikor a motorok forognak, vagy

hirtelen mozgások, ezért úgy döntöttünk, hogy a megbízhatóbb, erőforrás-igényesebb verziót használjuk a későbbiekben, mivel erre már elvégeztük a szükséges teszteket. Az egyszerűsített komplementer szűrő kapcsolási rajza pedig a 10-3 ábrán található meg.

4.6. Távirányító

A rendszernél alkalmazott távirányító az *Art-Tech* által kifejlesztett *E-fly 100C*. A vezérlő hat értéket térít vissza. Az első négy értékkel vezérelhető a repülő eszköz, a második kettővel pedig különböző feladatokat lehet elvégeztetni a rendszerrel, ha az illető eszköz úgy van bekonfigurálva. Ezen a vezérlőn be lehet állítani, hogy ezzel épp milyen eszközt szeretnénk vezérelni, amelyek között szerepel a repülőgép, a V farkú repülőgép, a CCPM helikopter, valamint a helikopter üzemmód. A távirányítón megtalálható LCD kijelzőn le lehet olvasni azt, hogy jelenleg milyen üzemmódban működik, mennyi ideje működik, mekkora az akkumulátorok töltöttségi szintje, valamint a kanálisok normál vagy fordított üzemmódba vannak állítva. A vezérlőnek van egy beépített jelző rendszere alacsony feszültség esetére. Az eszközön megtalálható öt kapcsoló, amelyek az alján helyezkednek el. Ezek arra szolgálnak, hogy amikor a karokat forgassuk valamely irányba akkor, a vezérlő által küldött értékek hogyan érkeznek, miként változnak bizonyos irányban. Ha az adott irányban megváltozott érték nem megfelelő, akkor ezekkel felcserélhessük. A gombsor mellett még van egy hatodik kapcsoló is, amely el van különítve a többitől, ezzel a jelenlegi beállításokat lehet rögzíteni, illetve tovább állítani. A vezérlőn még található három kapcsoló, két pótméter, két kar, amelyek két irányba is mozgathatók, valamint ezeknek megfelelő korrigáló csúsztatható gombok. A leírás szerint a bal kapcsoló az ötödik kanálisra van behatással. Megfigyelések alapján a bal kapcsoló a hatodik csatornára van behatással. Ezt a csatornát, a hatodikat, csak ez az egy gomb befolyásolja, ezért ez alkalmas lehet olyan kapcsolásoknál, ahol azonnal szükséges a váltás egy valamilyen más üzemmódba. A jobb felső gomb a harmadik, valamint az ötödik kanálisra van behatással. Ugyanígy erre a csatornapárosra van hatással a baloldalon lévő kar, amelyet függőleges irányba mozgatunk, valamint az ennek megfelelő csúsztatható gomb. A két tekerhető gomb az ötödik csatornát befolyásolják. A jobb alsó kart, a leírás szerint arra lehet használni, hogy skálázzák az értékeket. Ez a beállítás az első, második, és negyedik kanális értékeire vannak behatással. A vízszintes jobb oldali kar a neki megfelelő csúsztatával az első, a függőleges jobb oldali a második, a vízszintes bal oldali kar pedig a negyedik csatornára van hatással.

A távirányító jelen állásban a következőképpen van beállítva. Repülőgép üzemmódban működik, beállításai változtathatók. Az egyes, kettes, és hármas kanális REV, azaz fordított üzemmódban, a négyes és hatos NOR, vagyis normál üzemmódban. A jobb oldali vízszintes kart az billenésre, ugyanazon oldali függőlegest a bólintásra, a bal oldali vízszintes kart az elfordulásra, és végül ugyanazon oldali függőlegest a motoros sebességének beállítására használjuk. A rendszeren jelenleg a bal oldali kapcsoló van használatban, az, amelyik egyedül változtatja a csatorna értékeit. Ezt a kapcsolót pedig a rendszer azonnali leállítására használjuk. A 4-9 ábrán látható a távirányító és a karoknak megfelelő funkciók.



4-9 ábra Távirányító, és a karoknak megfelelő funkciók[22]

Az adó modulált PWM jelet küld a vevőnek. Mivel az FPGA órajele 100 MHz, a távirányító pedig alacsonyabb frekvencián továbbítja a jelet, ezért ebbe a modulba be van építve egy órajel-osztó, azért, hogy ne kelljen nagy értékekkel számolni a továbbiakban. Az érték kiolvasása úgy van megoldva, hogy 1-es bemeneti értékre egy automata számol addig, amíg a bemenet átvált 0-ra.

Amint megtörtént a váltás, a kimeneten megjelenik a vevő által küldött skálázott érték. Az új érték akkor fog megjelenni a kimeneten, ha a modul érzékel egy átmenetet 1-ről 0-ra, vagy pedig reset jelet kap. Reset jel esetében a rendszer kimenetén egy fix, pontos érték lesz látható. Ezzel elérhető az, hogy esetleges vészleállítás esetén olyan értékek jelenjenek meg a kimeneten, folytatáskor, amelyek akkor érkeztek. A távirányító értékeinek beolvasását az Algoritmus 3 írja le.

```

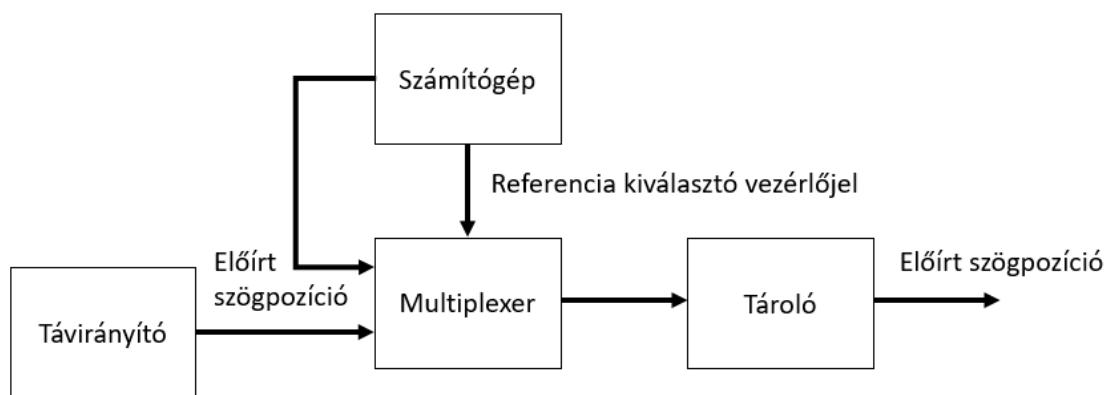
Algoritmus 3 Távirányító értékeinek beolvasása
    ha PWM_jel=1
        számláló növelése
        számláló értékének eltárolása
    másiképp
        számláló lenullázása
        tárolt számláló értékének kijelzése
    vége
vége

```

A távirányító kapcsolási rajza kiegészítve a skálázó résszel a 10-5 függelékben látható.

4.7. Referencia kiválasztó

A rendszer egy kiegészített változata tartalmaz egy referencia kiválasztó egységet is. Ekkor már az előírt értékek érkezhetnek egy felsőbb szintről, mint például a számítógéptől. A szabályozónak beadandó jelek közül a referencia egység keretén belül egy multiplexer segítségével lehet kapcsolatba hozni, hogy az értékeket a számítógéptől, vagy pedig a távirányítótól kapja. A referencia kapcsoló a számítógép oldalán van megvalósítva, így futás közben bármikor át lehet kapcsolni. A referencia kiválasztó egység tömbvázlata a 4-10 ábrán látható.



4-10 ábra Előírt érték kiválasztásának tömbvázlata

4.8. Quadkopter szabályozási módszerek

A quadkopterek távirányítóval való vezérlése esetében két módot különítenek el. Az egyik mód az szögsebesség alapú vezérlés, a második pedig a szögpozíció alapú vezérlés. Az szögsebesség alapú vezérlés esetében a karok megadják a tengely mentén a forgási sebességét a rendszernek. A kar elengedése után a rendszer próbálja megtartani azt az irányt. Ezzel ellentétben, a szögpozíció alapú vezérlés esetében a kar elengedése után a rendszer arra törekszik, hogy a rendszer újra beálljon a vízszintes helyzetbe. Vezérléskor pedig a karok megadják, hogy maximálisan mennyit fordulhat el a rendszer a vízszintes pontjától. Mindkét vezérlési módnak megvan a maga előnye, illetve hátránya. Abban az esetben, ha stabilan, egy helyben, biztonságos környezetben szeretnénk használni a rendszert, akkor a szögpozíció alapú vezérlés a legalkalmasabb, mivel ez nem fordul könnyen fejjel lefelé, ha jól paraméterezve van a rendszer. Mivel nem lehet a rendszert egykönnyen megfordítani, ezért ezeket a quadkoptereket nem alkalmas olyan helyeken alkalmazni, ahol a rendszer flexibilitásán van a lényeg, mármint nem lehet akrobatikus mozdulatokat végezni vele. Ezért használják ilyen helyeken az szögsebesség alapú vezérlési módot. Ennek hátránya viszont az, hogy nem lehet felügyelet nélkül hagyni a rendszert, mivel a rendszer csak akkor áll be vízszintes helyzetbe, ha kézileg visszaállítják, amit viszont csak megközelíteni lehet, ezért a rendszer folyamatosan sodródik. [10]

4.8.1. Módosított D csatornás PID algoritmus

A PID algoritmus kiválasztása esetében egy módosított D csatornás PID-re esett a választás. Ez az algoritmus nem a hibát deriválja, hanem a kimenetet. Tapasztalatok alapján a kimenetet szűrő D csatornával rendelkező PID szabályozók akkor alkalmazhatóak a leghatékonyabban, ha a kimeneten a jel csak kissé zajos. Az alkalmazott PID algoritmus megegyezik azzal, amit az *Elméleti megalapozás* fejezetben volt bemutatva. Mivel ez a megoldás folytonos esetre volt kidolgozva, ezért a rendszer mintavételezését követően, és a paraméterek átrendezésének következtében kapott képlet a (4.2)**Error! Reference source not found.**-nak felel meg.

$$\begin{aligned}
 u_k &= K_p * \left(e_k - T_d * \frac{y_k - y_{k-1}}{T} + \frac{T}{T_i} \sum_{i=0}^k e_i \right) \\
 u_k &= u_{k-1} + K_p * \left(e_k - e_{k-1} - T_d * \frac{y_k - 2 * y_{k-1} + y_{k-2}}{T} + \frac{T}{T_i} * e_k \right) \\
 u_k &= u_{k-1} + q0 * e_k + q1 * e_{k-1} + q2 * y_k + q3 * y_{k-1} + q4 * y_{k-2}
 \end{aligned} \tag{4.2}$$

A szabályozó algoritmus implementálásakor a cél ugyanúgy, mint az előbbiekbén, a minimális erőforrás szükséglet volt. A megtervezett PID algoritmus összesen egy szorzó és egy összegző áramköri elemet, két összehasonlító, egy negáló, és egy tároló egységet tartalmaz, ami mellé még bekerülnek multiplexerek, tárolók, és a vezérlő áramkör. A vezérlő áramkör segítségével lehet elérni azt, hogy hatékonyan lehessen kihasználni a rendelkezésre áramköri elemeket. Kezdetben a vezérlő áramkör lenullázza az összegző áramkört. Eközben a szorzó áramkör összeszorozza az első két elemet. Az első elem a PID paramétere, a második pedig egy érték, amely lehet a rendszer kimenete, a hiba, vagy pedig egy régebb kiadott szabályozójel. A második órajelre az összeadó áramkör a már meglévő értékhez hozzáadja az új értéket. Összesen 12 órajelre van szükség, amíg a szabályozójelet kiszámolja a szabályozó. Ezt követően az eredménye nem kerül ki a kimenetre, hanem előbb lekorlátozódik. Ezzel be lehet határolni azt, hogy a szabályozójelet mennyire korlátozzuk le. A szabályozójel egy összehasonlító áramkörön megy keresztül, amely egy kívülről megadott pozitív és negatív határérték között kell, hogy legyen. Ha ezek közül valamelyik értéket a szabályozójel átlépi, akkor a vezérlő a határértéket teszi ki a kimenetre, és jelzi, hogy készen van a számítással. A PID működésének leírása az Algoritmus 4 részben látható.

```

Algoritmus 4 PID
    ha start=1
        amíg számláló<6
            eredmény = eredmény + paraméter[számláló] * érték[számláló]
            számláló növelése
        vége
        ha szabályozójel>korlát
            szabályozójel = korlát
        vége
        számítások befejezésének jelzése
    vége
vége

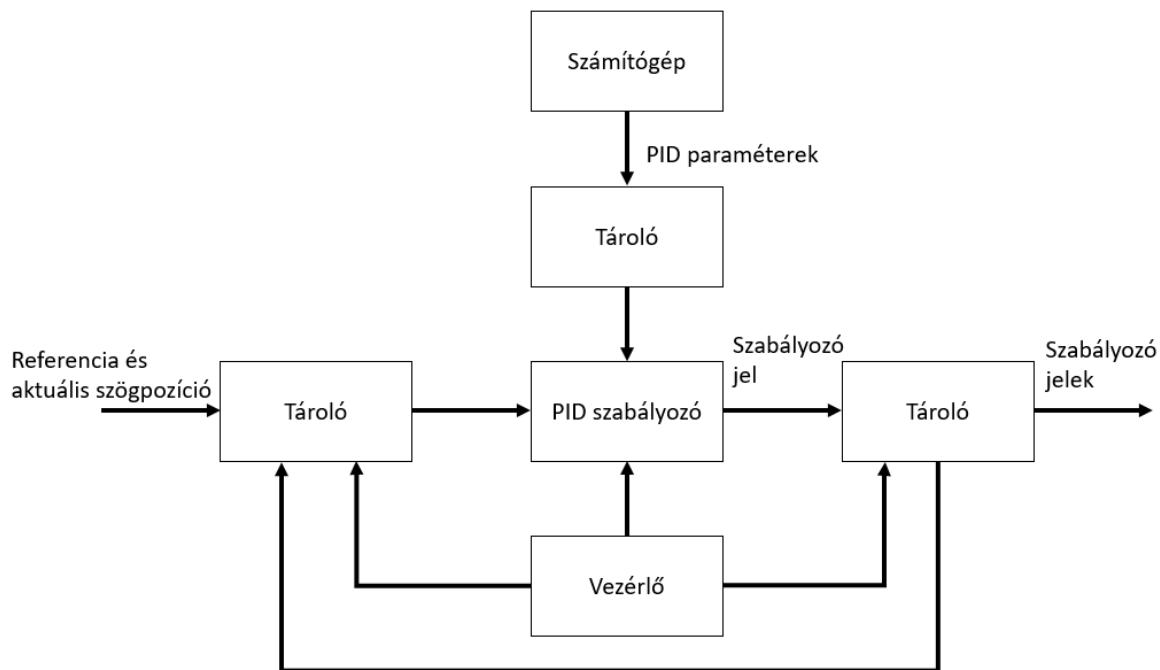
```

Ebből PID szabályozóból minimum két példányra lenne szükség. Mivel a második PID szabályozó is ugyanúgy töténik a paraméterezése, ezért elég egyet használni, kívülről pedig csak annyit kell megadni, hogy mikor mely bemenetekkel dolgozik a rendszer, és ezt egy vezérlő egység irányítja.

Ebben a részben egy újabb áramkör is megjelenik, amely az adatok eltolásáért felel. Mivel a PID algoritmus dolgozik olyan adatokkal, amelyek egy-két futással ezelőtt történtek, szükséges ezen adatok tárolása. Az adatok tárolása egyszerűen úgy van megvalósítva, hogy amikor a PID szabályozó kiszámította az összes szabályozójelet, akkor a PID-et vezérlő egység egy jelet küld a tároló egységnek. Ebbe a részbe egy nagyon egyszerű vezérlő egység van beépítve. Ennek a

feladata az adatok eltolása egyik regiszterből a másikba. A legrégebbi adatok minden elvesztődnek, az újak pedig bekerülnek ebbe a sorba. Ugyanitt történik a hiba kiszámítása is. Átgondolva azt, hogy a hiba ebben a ciklusban van újraszámolva, és ugyanekkor történik a szabályozójelek kiszámítása, meglehet, hogy jobb lenne a kettőnek a sorrendjét felcserélni, így mindenkor a legfrissebb hiba szerint történne a szabályozás. Ez az egység nem jelzi, hogy mikor van kész, mivel a szabályozójelek kiszámítása, és az értékek eltolásának ideje bőven 1 ms alatt van, és ilyen gyorsan nem lehet szabályozni ezt a quadkoptert, mivel a motorvezérlőknek a maximális vezérlőjel 400 Hz-es frekvenciával változhat.

Visszatérve arra a vezérlőre, amely a PID szabályozót vezérli, ezt is egy impulzust generáló modul működtet. Tehát a PID szabályozónak a számítási frekvenciáját állítani lehet. Ez jó, mivel össze lehet hangolni a motorvezérlőt a szabályozóval, mindenkor ugyanazon a frekvencián működik, szóval a szabályozó adja a szabályozójelet, a motorvezérlő pedig azonnal reagál erre. A vezérlő áramkör a PID szabályozóval és az eltoló-tároló egységgel a 4-11 ábrán láthatóak.

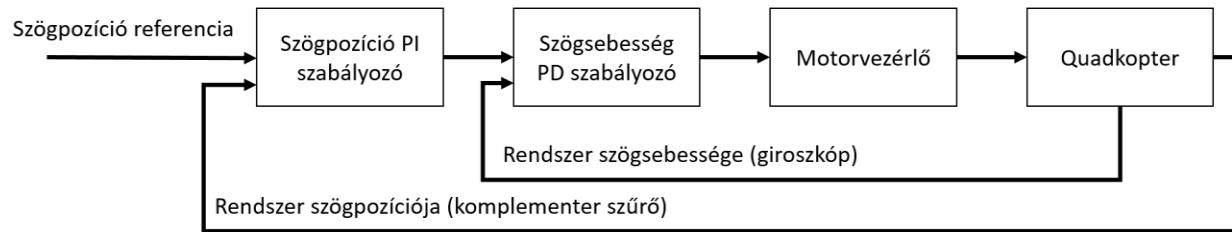


4-11 ábra PID szabályozó és a vezérlőegység tömbvázlata

A módosított D csatornás PID szabályozó kapcsolási rajza a 10-6 függelékben érhető el.

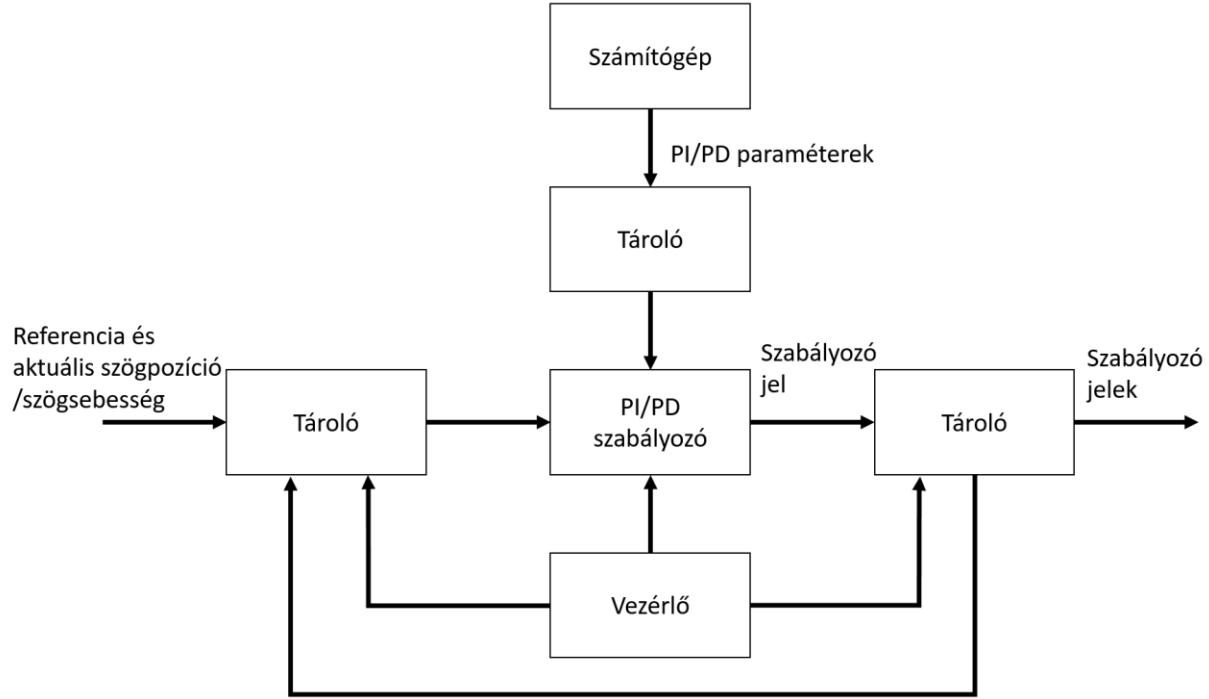
4.8.2. Kaszkád szabályozó

A rendszer szabályozására kaszkád PID szabályozó is volt tervezve, viszont ez nem volt kipróbálva. A szabályozó két hurkot tartalmaz, a külső hurok segítségével a rendszer orientációja, pozíciója van szabályozva, a belsővel pedig a szögsebesség. A külső hurokban egy PI szabályozó van alkalmazva, mivel az állandósult állapotbeli hibát célszerű nullába levinni, a belső hurok esetében egy PD szabályozó, hogy a hirtelen változásokra ne legyen annyira érzékeny a rendszer. A kaszkád PID szabályozó tömbvázlata a 4-12 ábrán látható.



4-12 ábra Kaszkád PID szabályozó tömbvázlata

A rendszer orientációjának szabályozására egy darab szabályozót implementáltam, így csökkentve az erőforrás szükségszükséget. Az FPGA sebességét kihasználva úgy készítettem el a szabályozót, hogy ez rendre számolja ki a motoroknak kiadandó vezérlőjelet. A PI és PD szabályozókat felosztottam két különálló részre. Az egyik maga a szabályozó, a másik pedig egy tároló, amely a szabályozónak szükséges értékeket tárolja. Ezt a két egységet egy automata vezérli, így a szabályozó egy adott szögnek megfelelő kimeneti értéket számol ki, amelyet követi a következő. Ennek tömbvázlata a 4-13 ábrán látható.



4-13 ábra PI/PD szabályozó és a vezérlőegység tömbvázlata

A kaszkád PID szabályozó kapcsolási rajza a 10-7 függelékben látható.

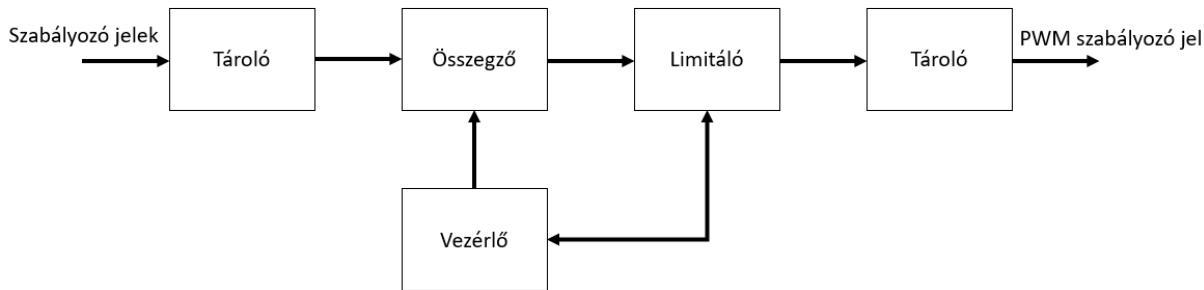
Az értékeket tároló egységen belül egy shift regiszter jellegű tároló található, melynek segítségével a régi és az új értékek menthetők el. Ez egy automata segítségével van megvalósítva. A tároló egység kapcsolási rajza a 10-9 függelékben található meg.

A PI és PD szabályozók szintén egy egyszerűsített megoldáson alapszanak, egy szorzó áramkörön és egy akkumulátoron. Az FPGA-k rendelkeznek beépített DSP modulokkal, így feltevődik a kérdés, hogy miért nem volt egy MAC egység alkalmazva, amely egyben tartalmaz egy szorzó és összeadó áramkört, különálló áramköri elemek helyett? A válasz az, hogy ennek a modulnak a használata esetén ugyanannyi időbe került volna a szabályozó kimenetének a kiszámítása, mivel az összeszorozott értékeket az előbbi értékhez kellett volna hozzáadni. Az egyszerű, regisztert nem használó alkalmazás esetében ez problémát okozott volna, mivel megtörténhet, hogy egy órajel alatt kétszer végződik el a MAC művelet. A végső érték még nem kerül ki a kimenetre. Előbb egy összehasonlításon megy keresztül, ami arra szolgál, hogy bizonyos korlátokat állítsunk fel a szabályozó kimenetének. Egy automata segítségével van megvalósítva, hogy mikor melyik

értékeket kell összeszorozni, valamit az is, hogy mi legyen a szabályozó kimenete. A kaszkád PID szabályozó PI szögpozíció szabályozójának kapcsolási rajza a 10-10 függelékben található meg.

4.9. Szabályozó jelek összesítése

A szabályozók által kiszámított szabályozójelek összesítését egy újabb automata valósítja meg. Ez is arra szolgál, hogy a lehető legkevesebb áramkori elem legyen felhasználva. Ez az áramkör is tartalmaz egy összehasonlító elemet, mivel szükséges lekorlátozni a motorokra adandó maximális és minimális vezérlőjelet. A PWM generáló egységnek köszönhetően nem szükséges a minimális értékkel foglalkoznia, ezért ez egyszerűen csak 0-ra van beállítva, és nem módosítható. Ennek a kapcsolási rajza nagyon hasonlít a PI és PD szabályozókéra, viszont ki van egészítve néhány szükséges áramkörrel. Ennek tömbvázlata a 4-14 ábrán látható.



4-14 ábra Szabályozójeleket összegző egység tömbvázlata

A szabályozójelek összegzésére alkalmazott képlet mindenik motor esetén a (4.3) részben található meg. Ezzel a képlettel az összegző összeadja a motorerőt, a jelenlegi bólintás vagy billenés, és az elfordulás szerint kiszámított szabályozójeleket.

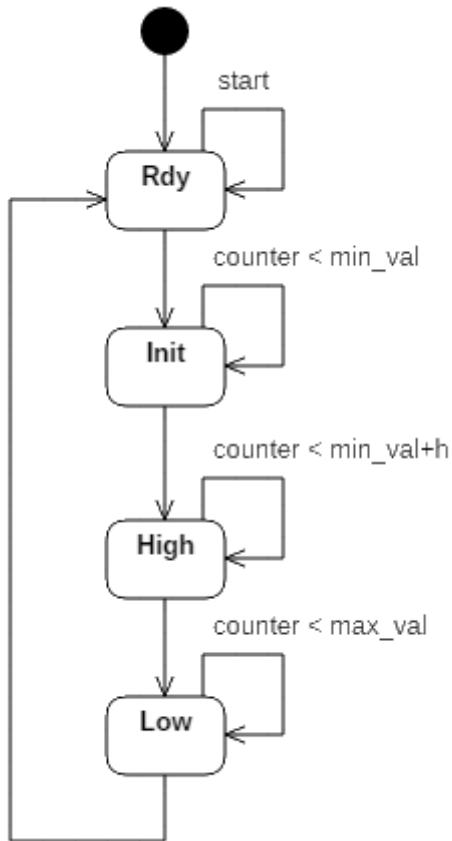
$$\begin{aligned}
 M_1 &= Throttle - u_{Pitch} + u_{Yaw} \\
 M_2 &= Throttle - u_{Roll} - u_{Yaw} \\
 M_3 &= Throttle + u_{Pitch} + u_{Yaw} \\
 M_4 &= Throttle + u_{Roll} - u_{Yaw}
 \end{aligned} \tag{4.3}$$

A szabályozójelek összegzőjének a kapcsolási rajza a 10-11 függelékben található meg.

4.10. PWM jelgenerálás

A PWM jelek megnevezése az angol *Pulse Width Modulation*-ból ered, amely impulzusszélesség modulációt jelent. Ezek a jelek négyszögjelek. Lényegük az, hogy egy adott periódusú jelnek a

kitöltési tényezőjét változtatva kapjuk ezt a jelet. Az általam alkalmazott PWM jelgenerátor állapotdiagramja a 4-15 ábrán látható. **Error! Reference source not found.**



4-15 ábra PWM modul állapotdiagramja

A modul bemenetei között található paraméterek között található a *div_val*, melynek segítségével az FPGA-nak a 100 MHz-es órajelét lehet leosztani egy alacsonyabb frekvenciára. Az újonnan kapott órajel frekvencia értéke a (4.4) képlettel számítható ki.

$$f_{new} = \frac{100 \text{ MHz}}{2 * div_val} \quad (4.4)$$

A *max_val* bemeneti értékkel megadható a rendszernek, hogy mekkora lesz a kimeneti jel periódusa. Minél nagyobb a *max_val* érték, a PWM jel periódusa annál kisebb, ugyanakkor annál nagyobb a felbontása. Ebből azt lehet lekövetkeztetni, hogy a *div_val* és a *max_val* értékekkel is a PWM jel frekvenciáját lehet meghatározni a (4.5) képlet alapján.

$$f_{new} = \frac{100 \text{ MHz}}{2 * div_val * max_val} \quad (4.5)$$

A *min_val* érték segítségével megadható, hogy mekkora lesz a minimális kitöltési tényező. Ezt az értéket %-ban kifejezve a (4.6) képlet segítségével lehet meghatározni.

$$n = \frac{100\% * min_val}{max_val} \quad (4.6)$$

A *h* érték segítségével megadható, egy érték, amely megadja, hogy mennyi ideig lesz 1-ben a kimenet. Abban az esetben, ha a felhasználó a *max_val* értékhez képest egy olyan *h* értéket ad meg, amely ennél nagyobb, abban az esetben nem áll fent az a probléma, hogy az automata egy állapotban hosszabb időt töltön el, mint amennyit lehet. A kitöltési tényező kiszámítására a (4.7) képlet használható, az eredményt pedig %-ban kapjuk meg.

$$n = \frac{100\% * (min_val + h)}{max_val} \quad (4.7)$$

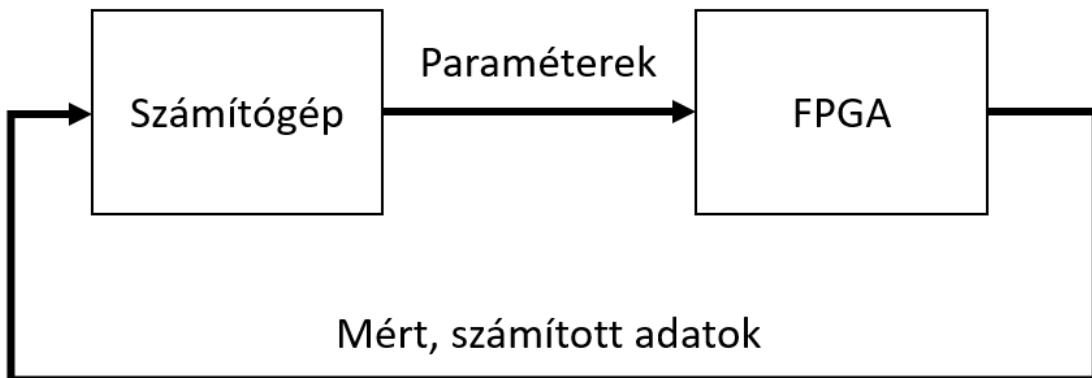
Ezen felül van egy *reset* jel is, amely segítségével a kimenet nem megy le konstans 0-ba, hanem a már meghatározott periódusú, minimális kitöltésű jel lesz a kiküldendő PWM jel.

A PWM jelgenerátor kapcsolási rajza a 10-12 függelékben található meg.

5. Mérési eredmények

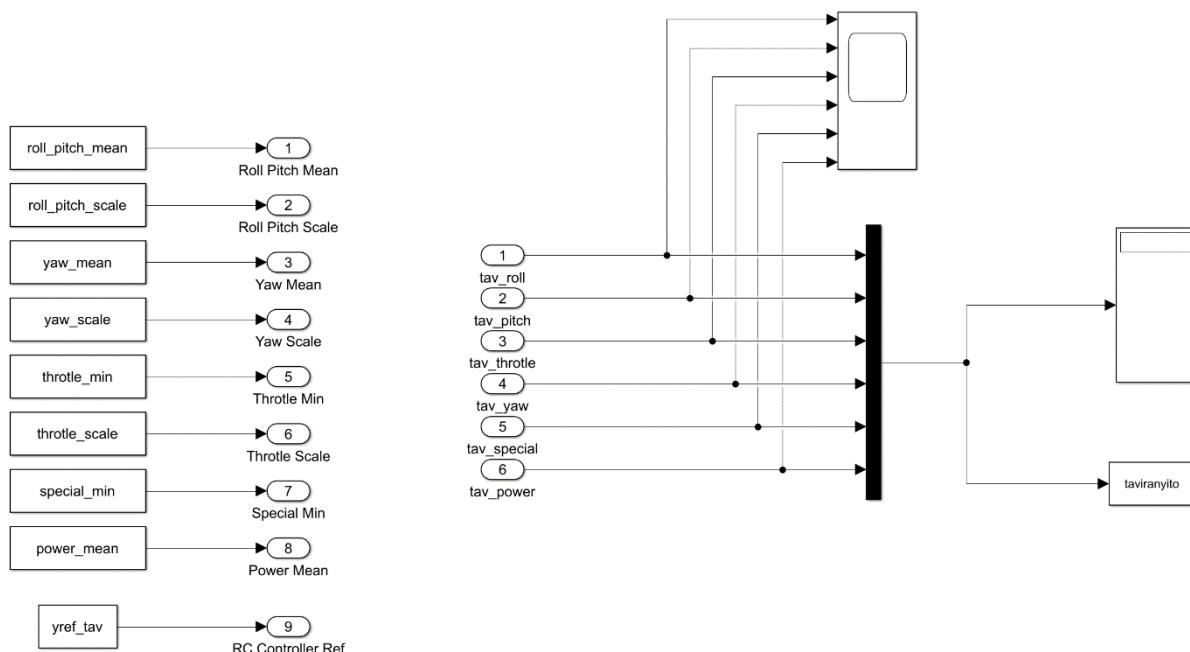
5.1. Mérési eredmények mentése hardver ko-szimulációt alkalmazva

Mivel a rendszer tartalmaz több modult, amelyektől kapott mérési eredményeket érdemes kimenteni, ezért a számítógépen belül a *Matlab Simulink* segítségével szétválasztottam a különböző moduloknak adandó és ezektől érkezendő jeleket külön egységekre, így könnyebben el lehet igazodni ezek között. Ahhoz, hogy a *Simulink*-et össze lehessen kapcsolni az FPGA-val, szükség van egy *System Generator* egység, amelyet a *Vivado* fejlesztőkörnyezet biztosít. Az 5-1 ábrán látható a hardver ko-szimuláció tömbvázlata. [24]



5-1 ábra A hardver ko-szimuláció tömbvázlata

Itt valós időben lehet követni, hogy a rendszer milyen értékeket küld vissza, és itt történik ezek mentése is. Az értékek kimentése úgy van megoldva, hogy minden futás végén a Matlab környezet automatikusan létrehozza a szükséges fájlokat, ezeket külön mappába helyezi, és automatikusan számozza őket. A felhasználónak ebben az esetben az a dolga, hogy kövesse a számokat, és jegyzeteket írjon hozzájuk, így minden mérés egyszerűen követve és rendszerezve van. Az 5-2 ábrán látható távirányítónak a kapcsolási rajza, hogyan vannak ezek az egységek összekötve és rendszerezve. Baloldalon vannak elhelyezve a távirányító konfigurációs paraméterei, jobb oldalon pedig a kapott értékek érkeznek vissza.

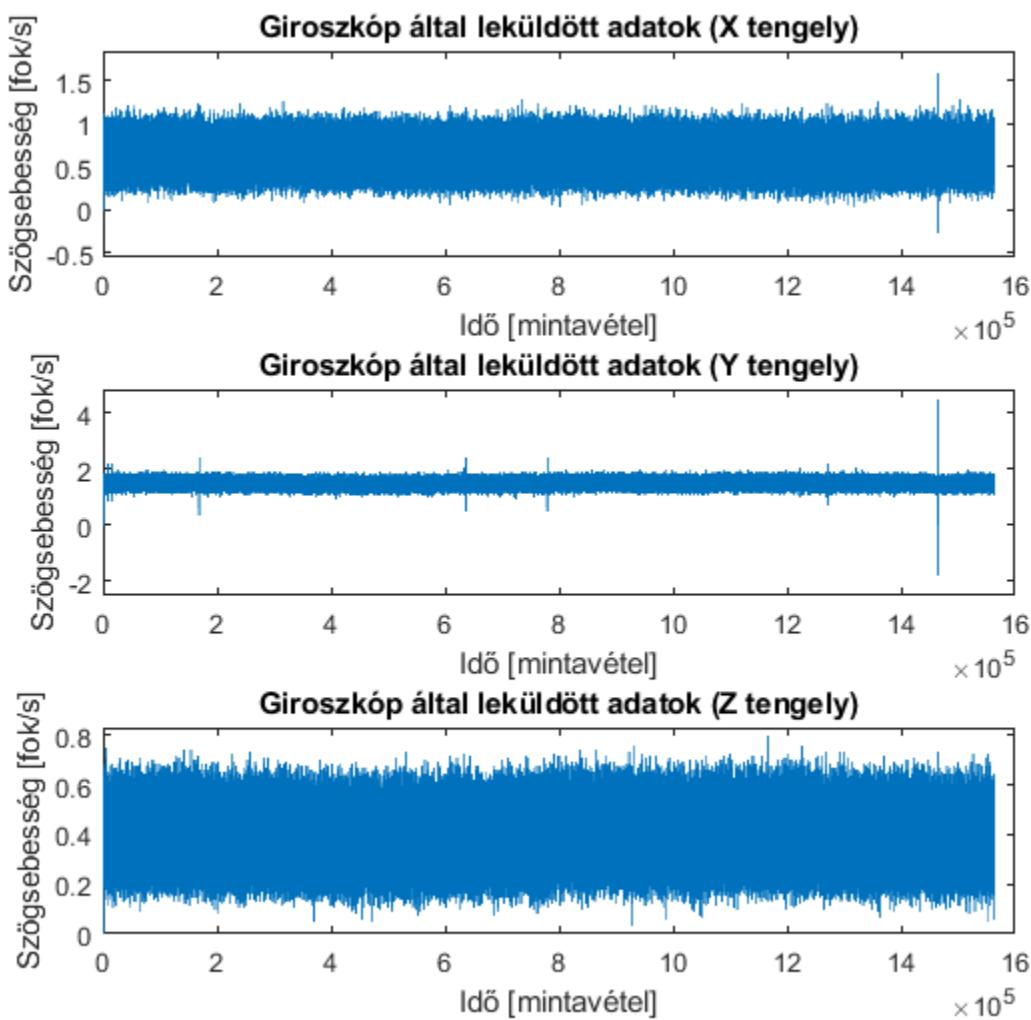


5-2 ábra Távirányítónak leküldendő paraméterek, és az általa visszaküldött eredmények mentése és ábrázolása Simulink környezetben hardver ko-szimulációt alkalmazva

A hardver ko-szimulációkor alkalmazott kapcsolási rajz a 10-13 függelékben található meg.

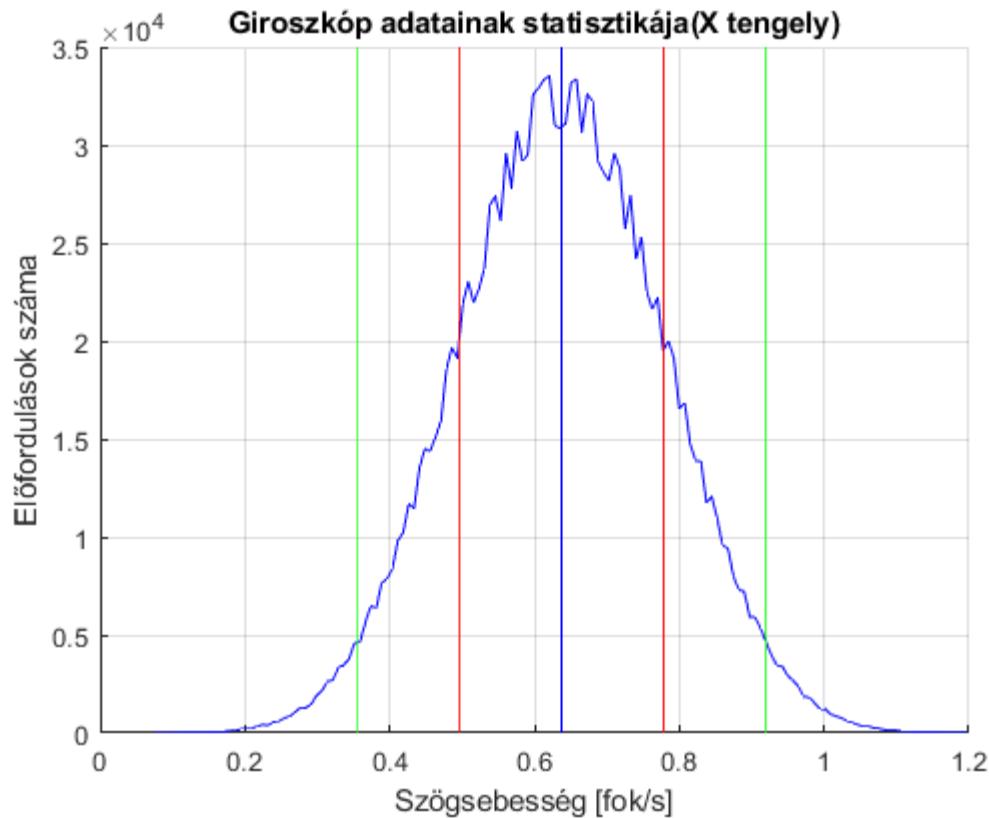
5.2. Pmod NAV IMU

Az 5-3 ábrán a giroszkóppal mért adatok látszanak a három különböző tengelyre nézve. A mérés stacionárius állapotban történt, mármint az eszköz mozdulatlan volt. Az eltolás abból ered, hogy a mérés hozzávetőlegesen 30 percet tartott.



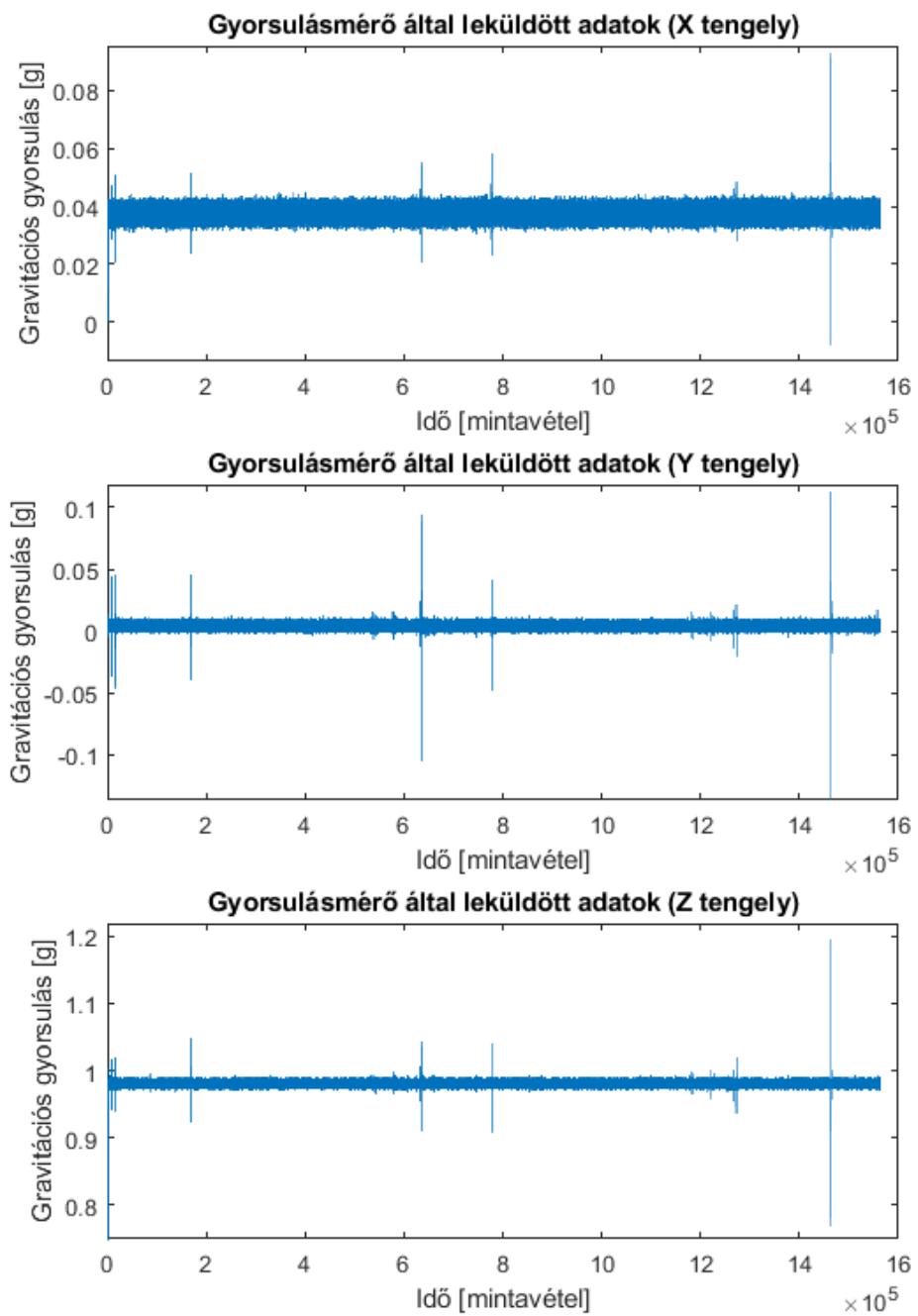
5-3 ábra Giroszkóp mérési adatai, stacionárius eszköz esetén

A giroszkóp eredményeit megfigyelve látható, hogy használat előtt kalibrálni kell a szenzort. A kalibrációs paraméterek meghatározására az eltolás úgy volt kiszámolva, hogy vettük az előbbi, hosszú mérést, összegeztük a mérési adatokat, majd elosztottuk a kapott eredményt az adatok számával. Mivel a rendszer stacionárius volt, ezért mindenik mérés ugyanazt az eredményt kellett volna, hogy visszaküldje. Ugyanakkor az adatok szórását is megvizsgáltuk. Az mérési eredmények eltolását, valamint egyszeres, illetve kétszeres szórását az 5-4 ábrán lehet látni.



5-4 ábra Giroszkóp mérési adatainak statisztikája

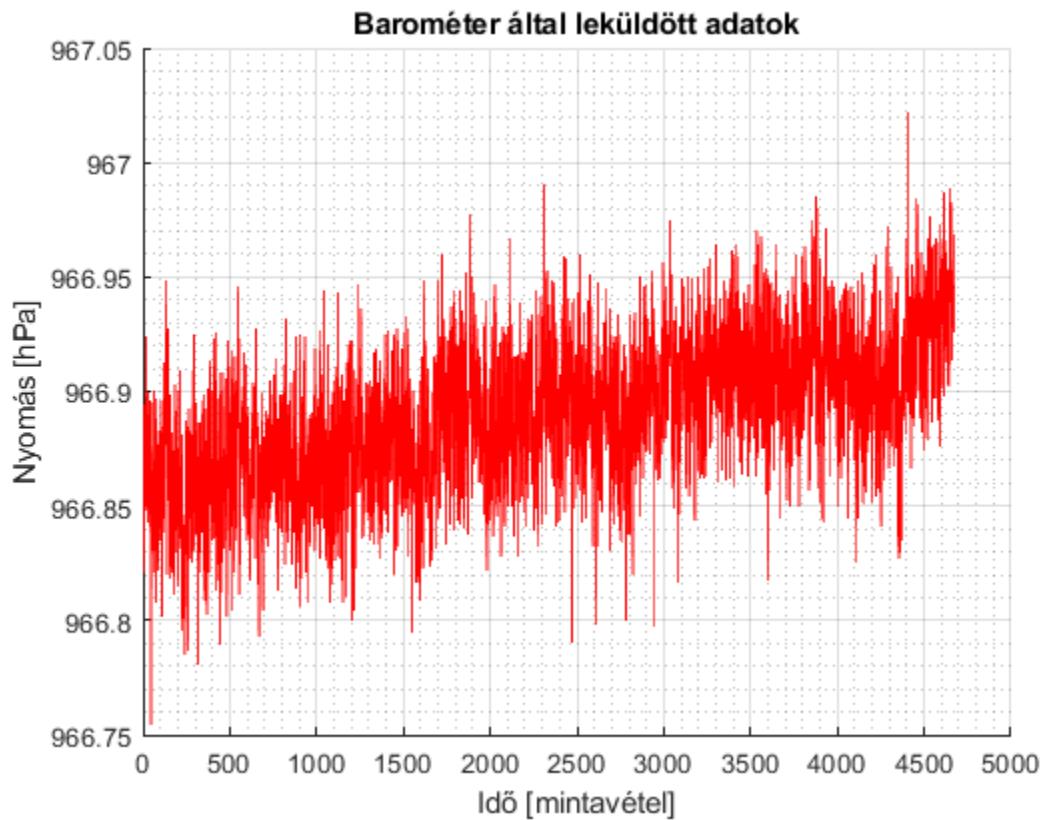
Az 5-5 ábrán pedig a gyorsulásmérőről kapott értékeket lehet látni. Ez a mérés is stacionárius állapotban történt, ugyanúgy 30 percig. A rajzokból jól kivehető, hogy helyenként a kapott értékek kiugranak. Ez lehet azért, mert hibásan volt leküldve az adat, vagy valamilyen más eszköz zavarta a szenzor mérését. Az olvasott érték 1 körül van, mivel a mérés a z tengely mentén volt végezve, tehát a szenzor az ábra szerint a gravitációs gyorsulást érzékelte.



5-5 ábra Gyorsulásmérő mérési adatai, stacionárius eszköz esetén

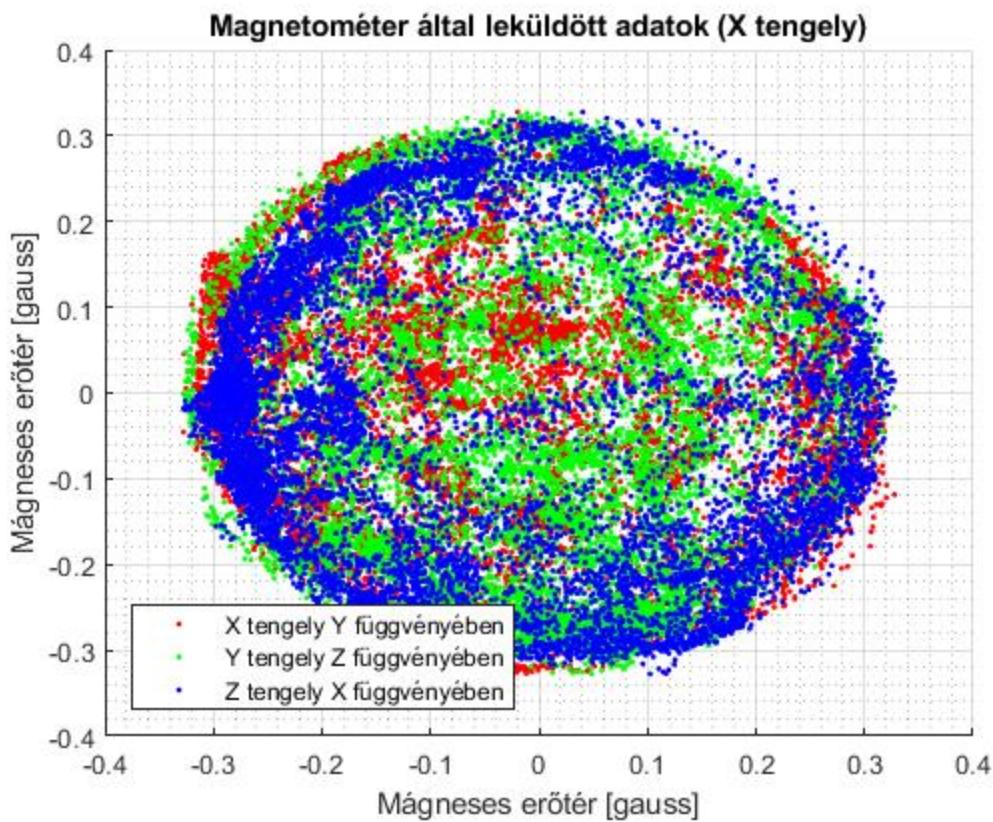
A barométerrel végzett mérések az 5-6 ábrán vannak szemléltetve. A mérés szintén huzamosabb ideig volt végezve. Ebből a rajzból jól látható, hogy habár nem telt el sok idő (néhány perc) az

adatok olyan szinten módosultak, hogy emiatt a magasság meghatározása meglehetősen bizonytalan, ha a rendszer huzamosabb ideig van a levegőben.



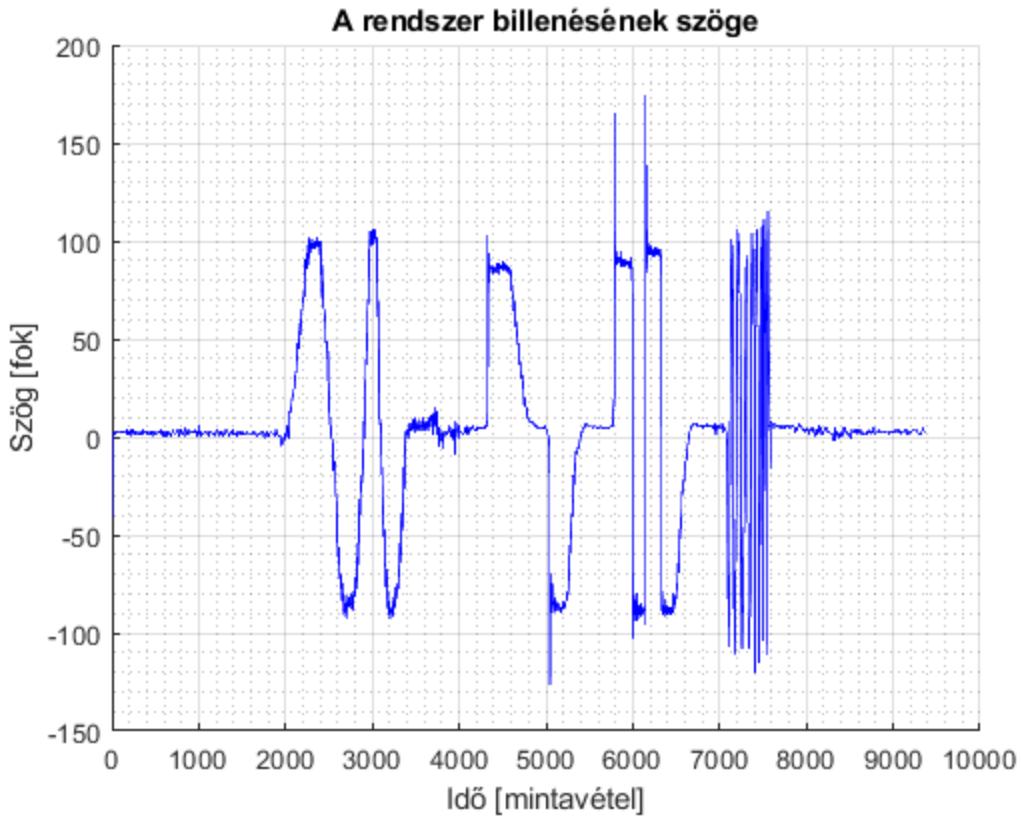
5-6 ábra Barométer mérési eredményei

Az 5-7 ábrán a magnetométerről szerzett mérési eredmények vannak megjelenítve feldolgozás után. A feldolgozás alatt azt kell érteni, hogy a különböző környezeti, mágneses, illetve elektromágneses hatások semlegesítve vannak. A zöld, piros, és kék pontok rendre a három tengely egymásra tevődését ábrázolja. A piros pontok az x tengelyen mért értékeket ábrázolja az y tengely mentén mért értékek alapján, a zöld az y -t a z -hez képest, a kék pedig a z -t az x -hez képest.



5-7 ábra Magnetométer forgatása, mágneses tér érzékelése

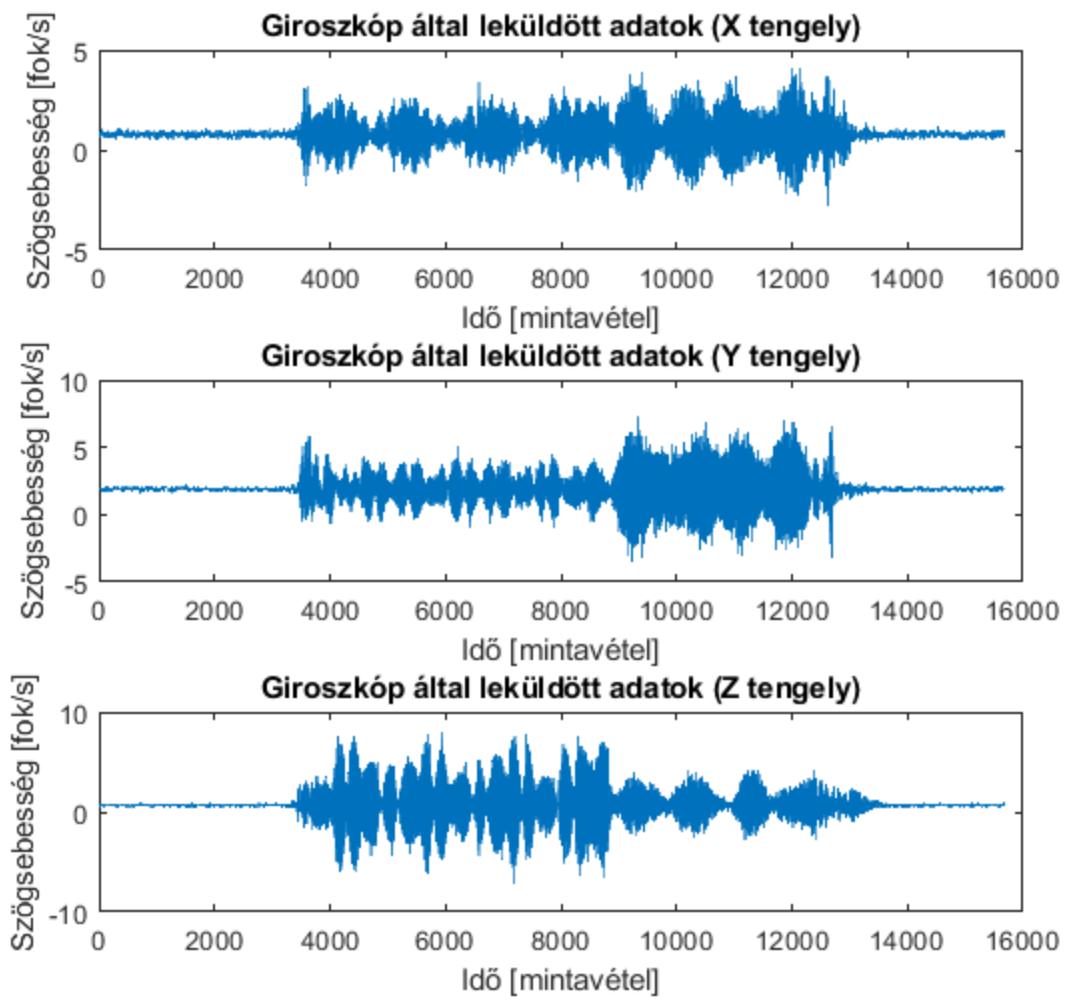
Az 5-8 ábrán a rendszer forgatása látható. Ekkor már használva van a komplementer szűrő. A rendszer billentve van egyik oldalról a másikra, $+/- 180$ fok-os szögben. Ennek a mérésnek a lényege az volt, hogy megvizsgáljuk, mennyire zajos a mérési eredmény a szögpozíciókat illetően.



5-8 ábra Rendszer billenésének mérése

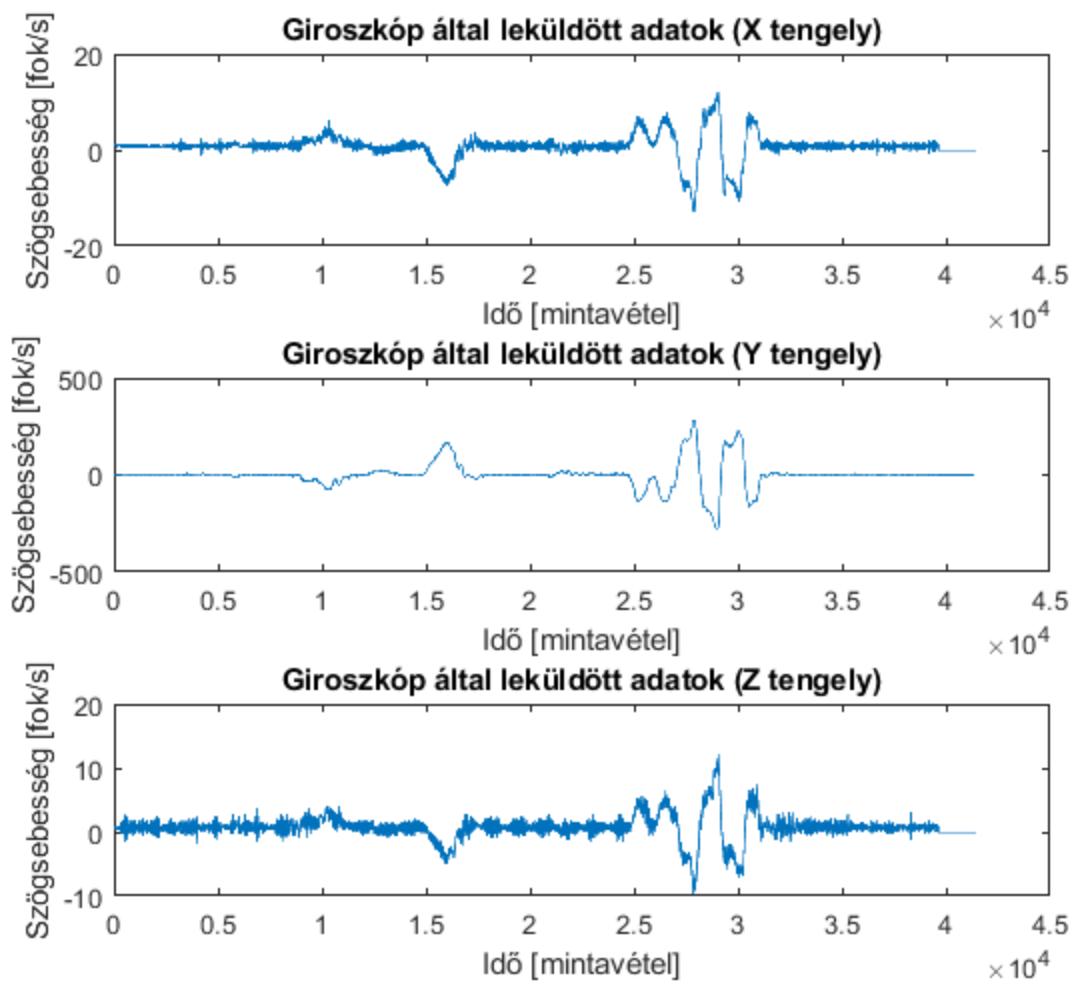
Az eddig bemutatott mérési eredmények mind zajmentes környezetben történtek. Ez alatt azt értem, hogy a szenzor közelében nem volt tápforrás, amely zavarná a jelek átvitelét, valamint a motorok sem forogtak a quadkopteron. A következőkben szemléltetett mérések zajos környezetben történtek, ami alatt azt értem, hogy a motorok egy bizonyos sebességgel forogtak propeller nélkül, és így történtek a mérések.

Az 5-9 ábrán látható a giroszkóptól kapott mérési eredmények. Ekkor a rendszeren nem volt semmilyen szűrő paraméter beállítva. A Z tengelyen kifejezetten jól látható, hogy a giroszkóp által leküldött értékek zajosak.



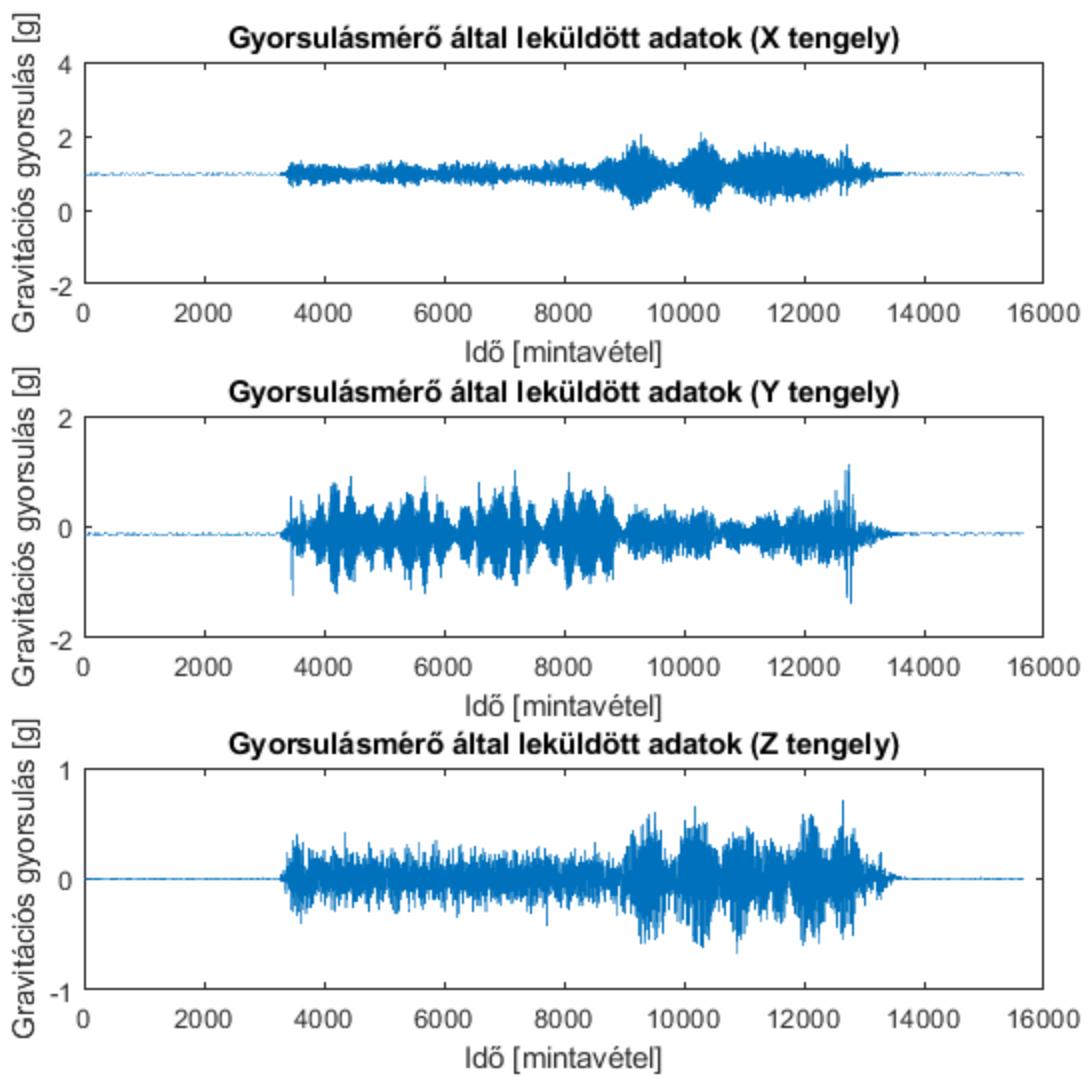
5-9 ábra Gyroszkóp segítségével mért mérési eredmények a szűrő jelenléte nélkül

Az 5-10 ábrán látható mérési eredmények az átlagoló szűrő jelenlétében történtek. Összehasonlítva az előbbi mérési eredménnyel jól látható, hogy a Z tengelyen meglehetősen kisebb a zaj. Ebben az esetben a rendszer mozgatva is volt. Ennek az az oka, hogy a változás közbeni zajokat is lehessen megfigyelni.

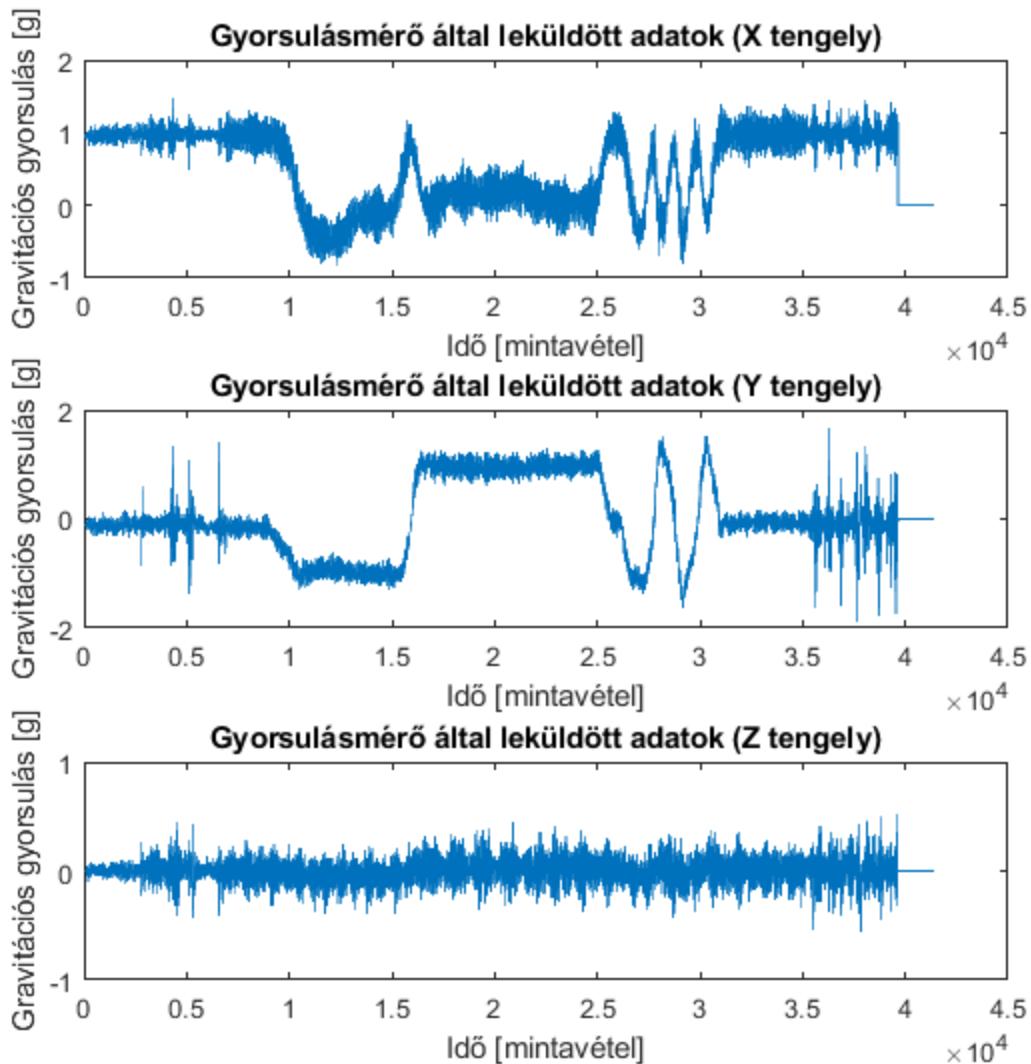


5-10 ábra Giroszkóp segítségével mért mérési eredmények a szűrő jelenlétében

Az 5-11 ábrán látható a gyorsulásmérővel mért mérési adatok. Ebben az esetben a szűrő nem volt jelen, és jól látható, hogy a zaj jelentős, könnyen érvénytelenítik a komplementer szűrő által kiszámított adatokat. Az érzékelőtől kapott nyers jelek után egy aluláteresztő szűrő alkalmazása után az 5-12 ábrán látható, hogy a gyorsulásmérőn megjelenő zaj jóval alacsonyabb.

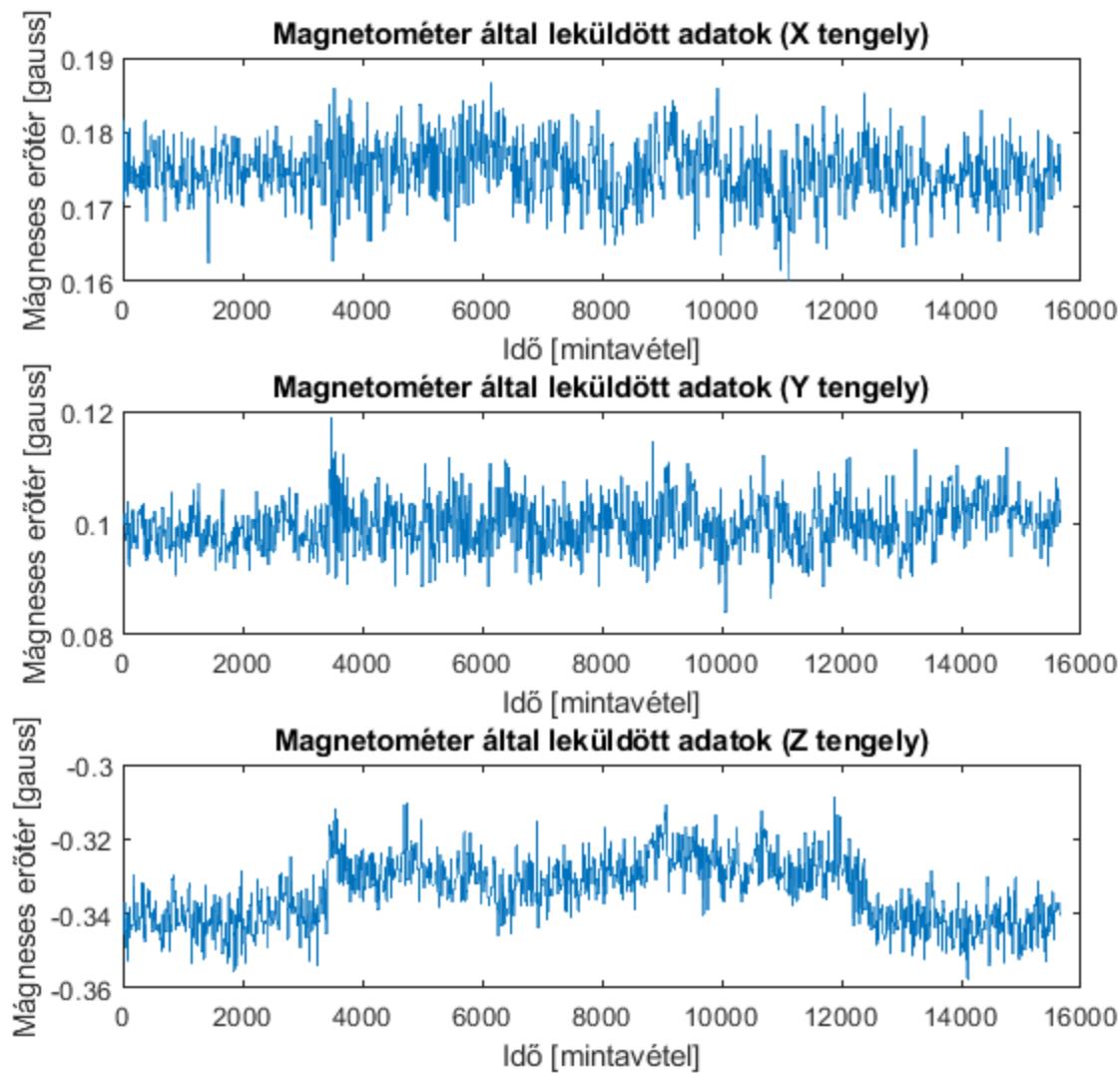


5-11 ábra Gyorsulásmérő segítségével mért mérési eredmények a szűrő jelenléte nélkül



5-12 ábra Gyorsulásmérő segítségével mért mérési eredmények a szűrő jelenlétében

A zaj azonosítása érdekében három lehetőség volt figyelembe véve. Az első probléma, ami figyelembe volt véve, az adatok átvitele a szenzortól az FPGA lapra. Ez oszcilloszkóp segítségével volt megfigyelve, viszont az adatok nem voltak zajosak átvitelkor. A második feltételezett ok a motorok által keltett elektromos zaj volt. Ezt könnyen le lehetett ellenőrizni a magnetométer segítségével. Az első zajos méréskor a magnetométer adatai is ki voltak mentve, és ezek a mérések voltak megfigyelve. Az 5-13 ábrán látható, hogy a magnetométer a motorok bekapcsolásakor csak minimális változást érzékelt, amiből arra lehet következtetni, hogy a mérési eredmények nem elektromos zajtól voltak olyanok, amilyenek.



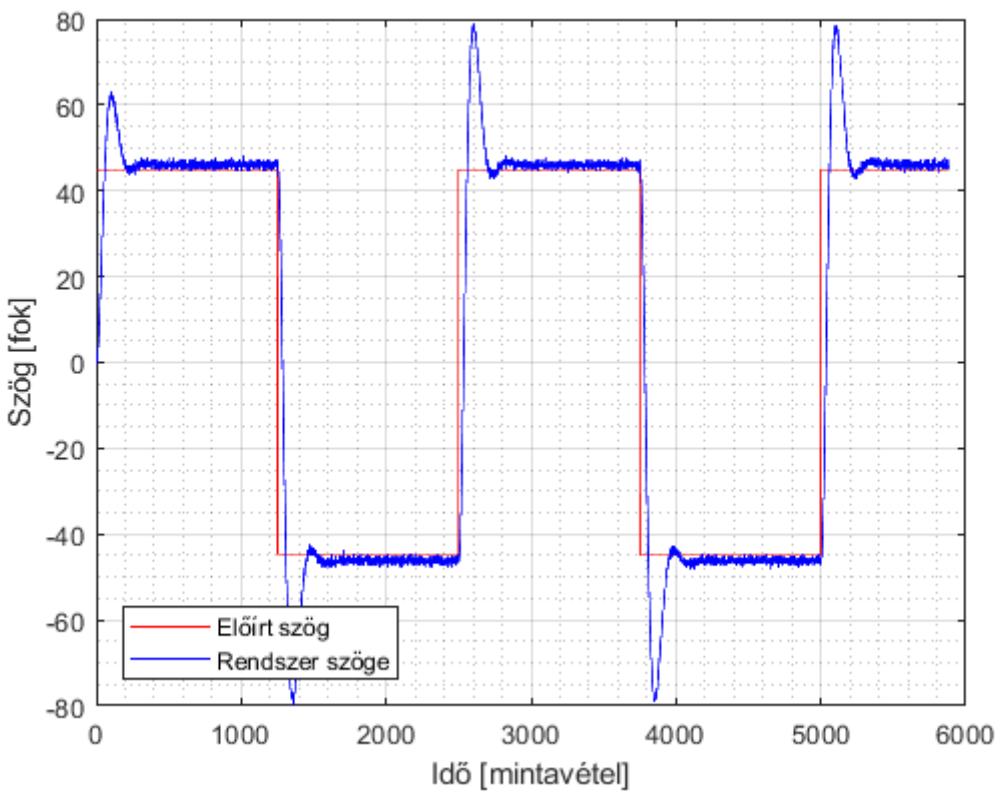
5-13 ábra Magnetométer segítségével mért adatok forgó motorok jelenlétében

A harmadik lehetőség, amely felmerült az a fizikai rezgés. Ezt a lehetséges könnyen lehetett tesztelni azzal, hogy a szenzor végét, amelyik nem volt rögzítve egyszerűen kézzel megfogtuk, és figyeltük a szenzor működését. Ebből a mérési sorozatból az jött ki eredménynek, hogy a mérési adatokon fizikai rezgések jelentkeztek.

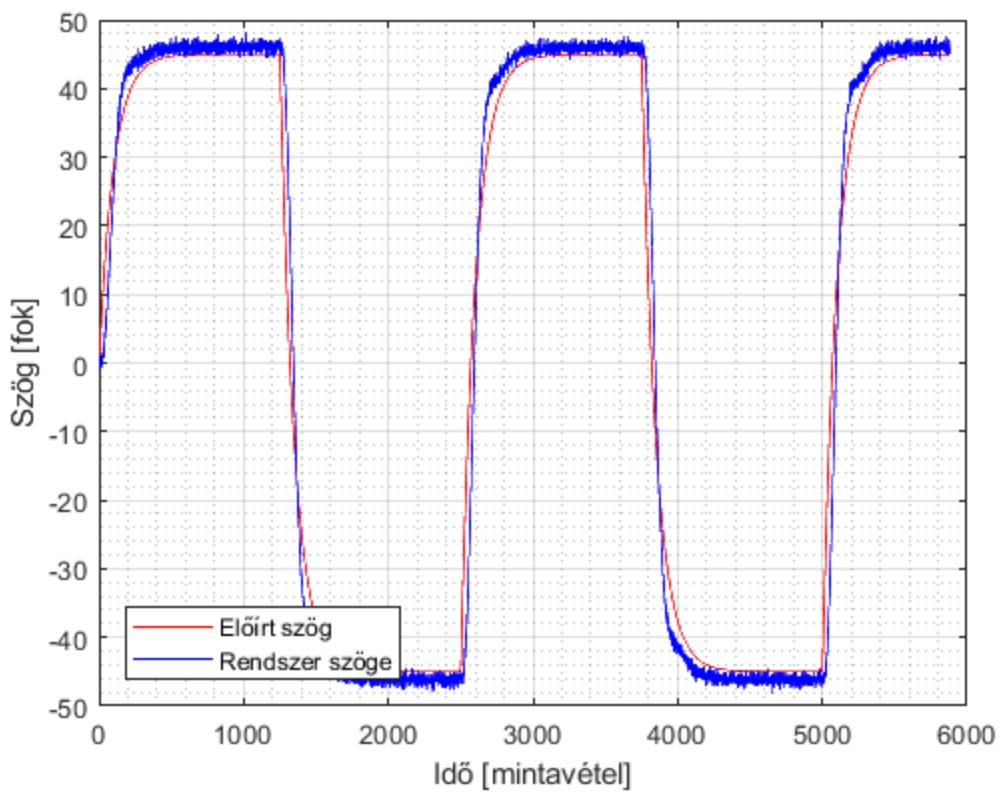
5.3. Módosított D csatornás PID

Az 5-14 ábrán látható a PID szabályozó működése szimulációs szinten. A felső ábrán látható a rendszer kimenete, míg az alsón az előírt referenciaérték. A rajzból az következtethető le, hogy

habár a PID paraméterei elég jól megközelítették a tökéletes értékeket, még így is rendelkezik a rendszer egy kis túllövéssel. Az 5-15 ábrán látható, hogy a referencia jel egy első fokú rendszerhez hasonlít az előbbihez képest. A módosítás abba áll, hogy a referencia jel előbb egy első fokú rendszeren van keresztülvezetve, ez által előírható a rendszernek egy viselkedés, mintha egy modell referencia alapú szabályozást akarnánk megvalósítani. Ebben az esetben a cél az volt, hogy nagy változások esetén, mint az előbbi esetben is, jól meghatározott PID paraméterek esetében a túllövést lenullázzuk.

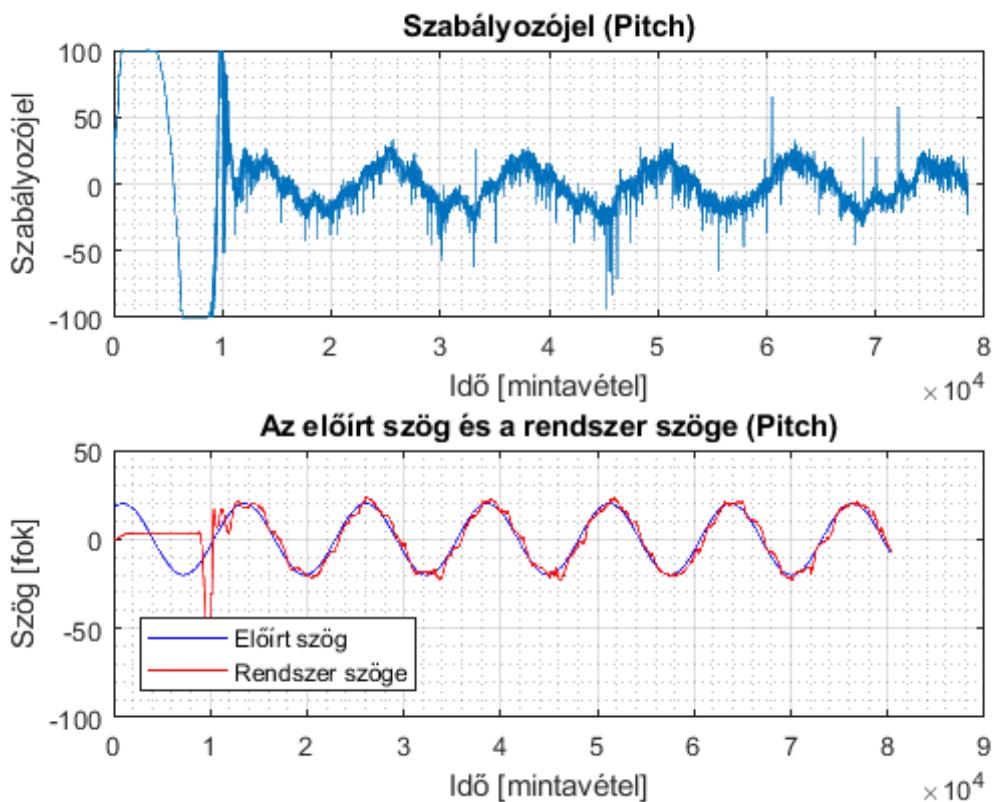


5-14 ábra PID szabályozó azonnal változó előírt értékkel



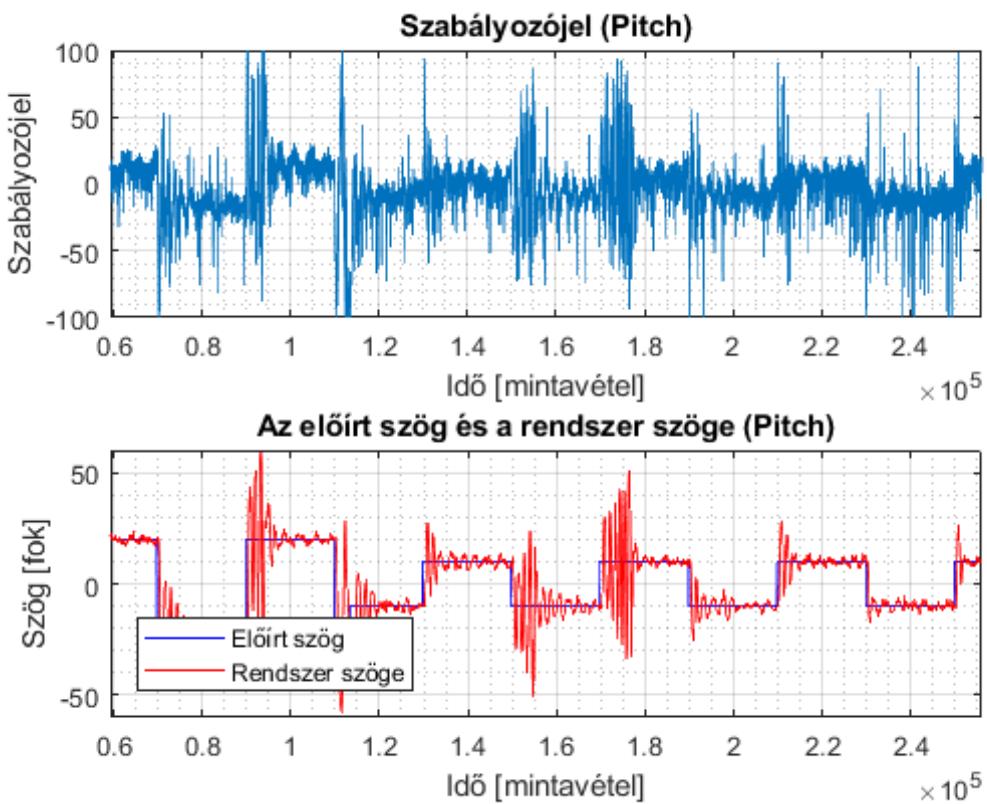
5-15 ábra PID szabályozó, fokozatosan változó előírt értékkel

Az 5-16 ábrán a PID szabályozó látható, melynek előírt referenciajára jelen esetben egy szinusz jel. A rajzon látható, hogy kezdetben a szabályozó kicsit nehezen hangolódik rá a jelre, viszont kis idővel eljut abba a pontba, ahol meglehetősen jól követi a referencia értéket. Kezdetben a szabályozó működött, a motorok viszont nem voltak felpörgetve, ez okozta az elején a hirtelen tüskét, és a lassan változó szabályozójelet.



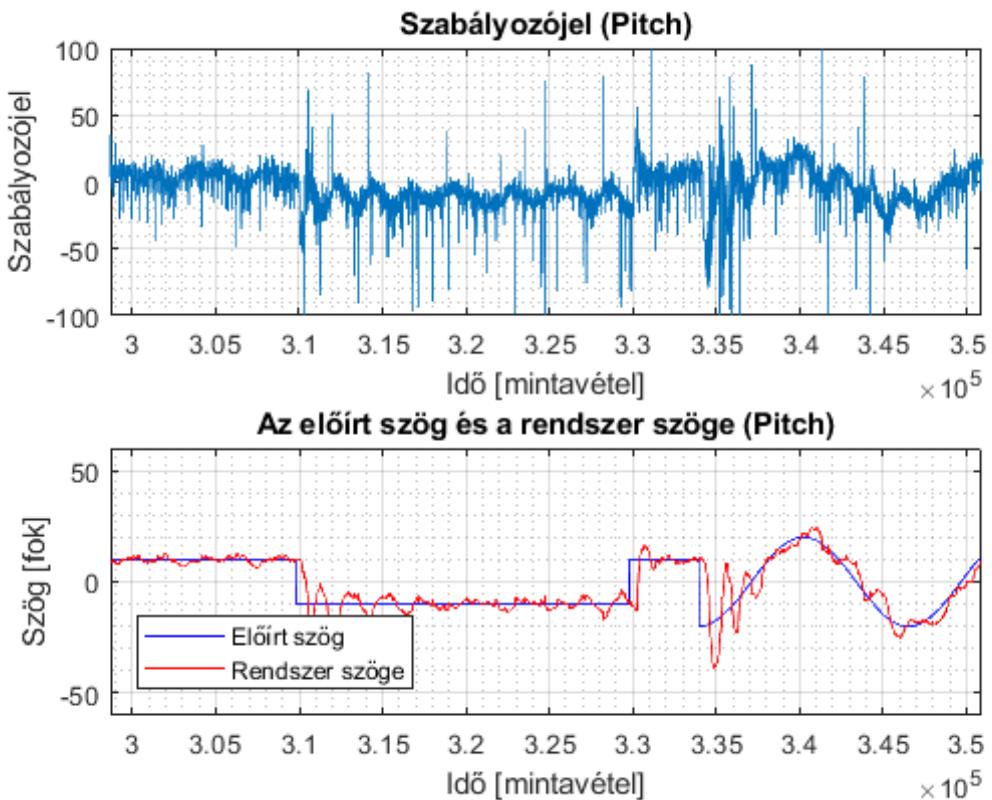
5-16 ábra Szögpozíció változása, és szabályozójel előírt szinusz jel alapján

Az 5-17 ábrán egy újabb mérés látható az előbb használt rendszerparaméterek alapján. Ebben az esetben négysszögjel volt a referencia, mivel a cél az volt, hogy hirtelen referencia változtatás, illetve hiba változása esetében megfigyelni hogyan reagál a rendszer. Az ábrán látható, hogy a hirtelen referenciaváltozás következtében a rendszer oszcillálni kezd, de idővel megáll, és követi a konstans jelet. Helyenként látható, hogy a referencia hirtelen változásakor a rendszer jobban berezeg, mint máshol. Ez annak tulajdonítható, hogy működés közben változtattuk a PID szabályozó deriváló tagját.



5-17 ábra Quadkopter szabályozása előírt négy szögjel esetén változó D paraméter értékekkel

Az 5-18 ábrán egy referencia jel váltás látható, ahol a négy szögjelről hirtelen átkapcsolunk szinusz jelre. Itt látható, hogy a rendszer nagyon berezeg. Ekkor szükséges volt a kézi beavatkozás, a rezgések csillapítására. Ez azért volt megoldható, mert a négy propellerből csak két szembe levő volt felszerelve biztonság és egyszerűség szempontjából.

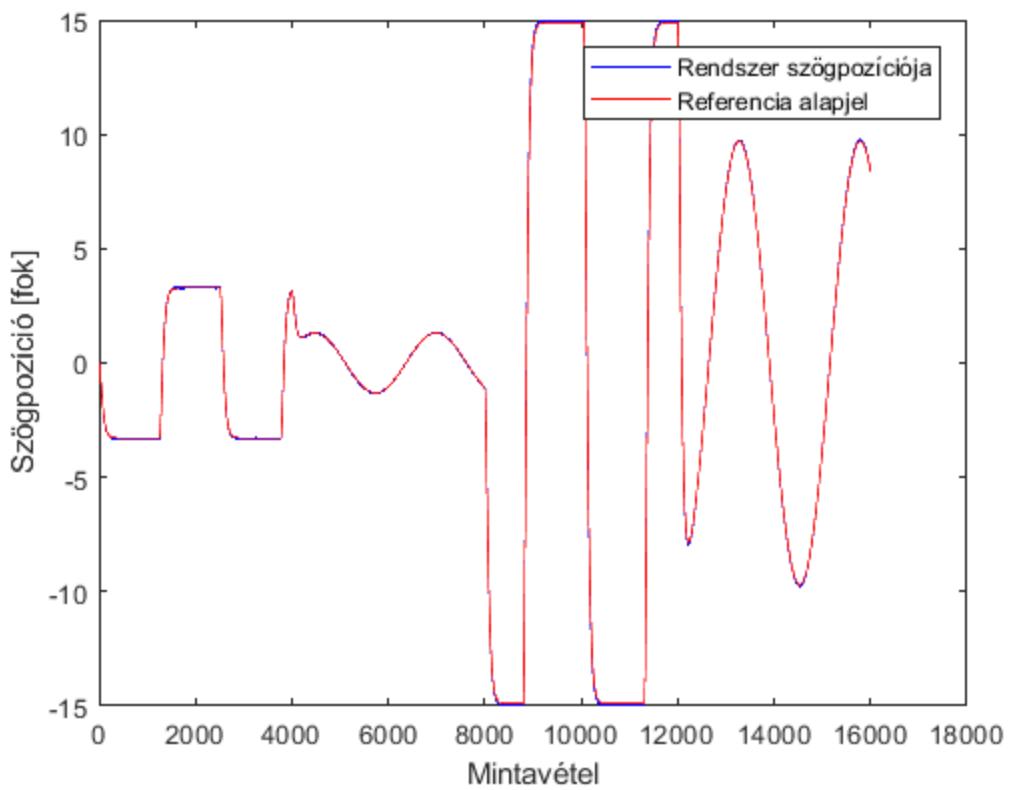


5-18 ábra Referencia megváltoztatása, és a rendszer válasza

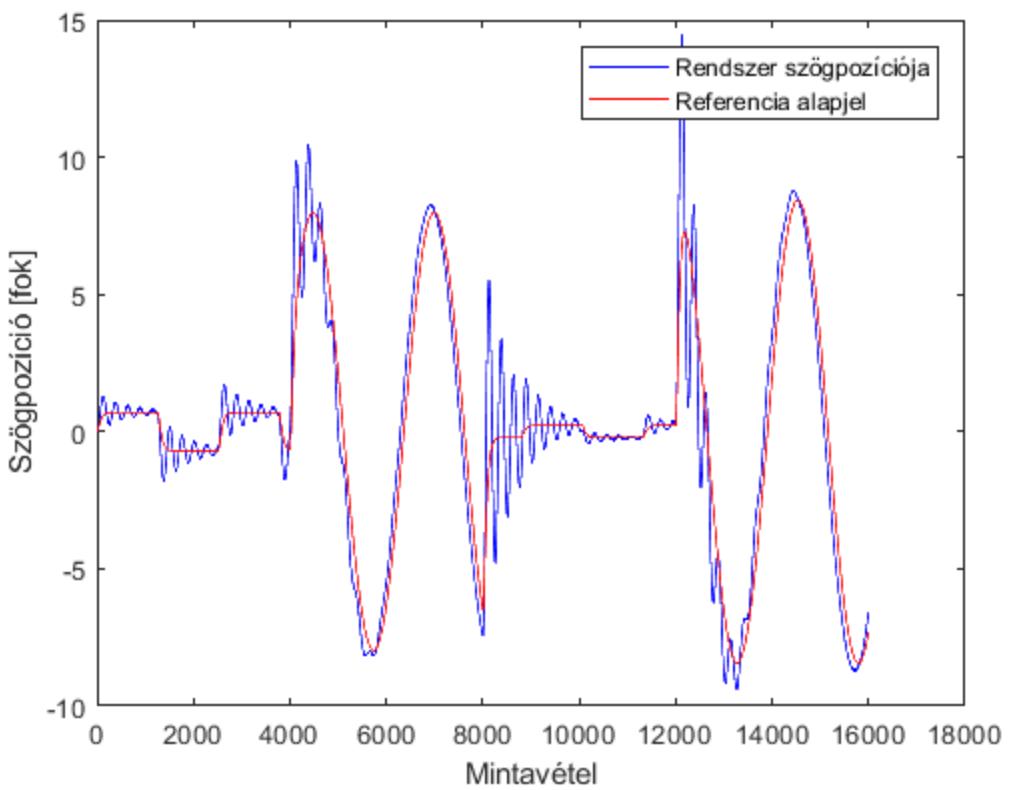
Ezekből a mérési eredményekből arra lehetett következtetni, hogy abban az esetben, ha a cél a referencia értékek folyamatos, hirtelen változtatása, akkor ilyen célra a jelenlegi szabályozó paraméterek nem megfelelőek, és ezeket tovább kell hangolni. Egy másik lehetőség az, hogy fokozatosan növelni az előírt értéket. Harmadik lehetőséggént az a lehetőség is fent áll, hogy egy szűrt D csatornás PID-el cseréljük le a jelenlegit, amely megoldaná a hirtelen referenciajel változtatás gondját.

5.4. RST szabályozó

Az 5-19 ábrán látható az RST szabályozó működése, miután a rendszer paraméterei meg vannak határozva. A rendszer paramétereinek meghatározásakor a rendszer kimenetén zaj is volt. Az 5-19 ábrán látható, hogy a becslő szinte tökéletesen megtalálta a rendszer paramétereit. Az 5-20 ábrán viszont az látható, hogy a zajos rendszer paramétereit nem tudta teljesen meghatározni, és ezért a szabályozó működése nem megfelelő e miatt.



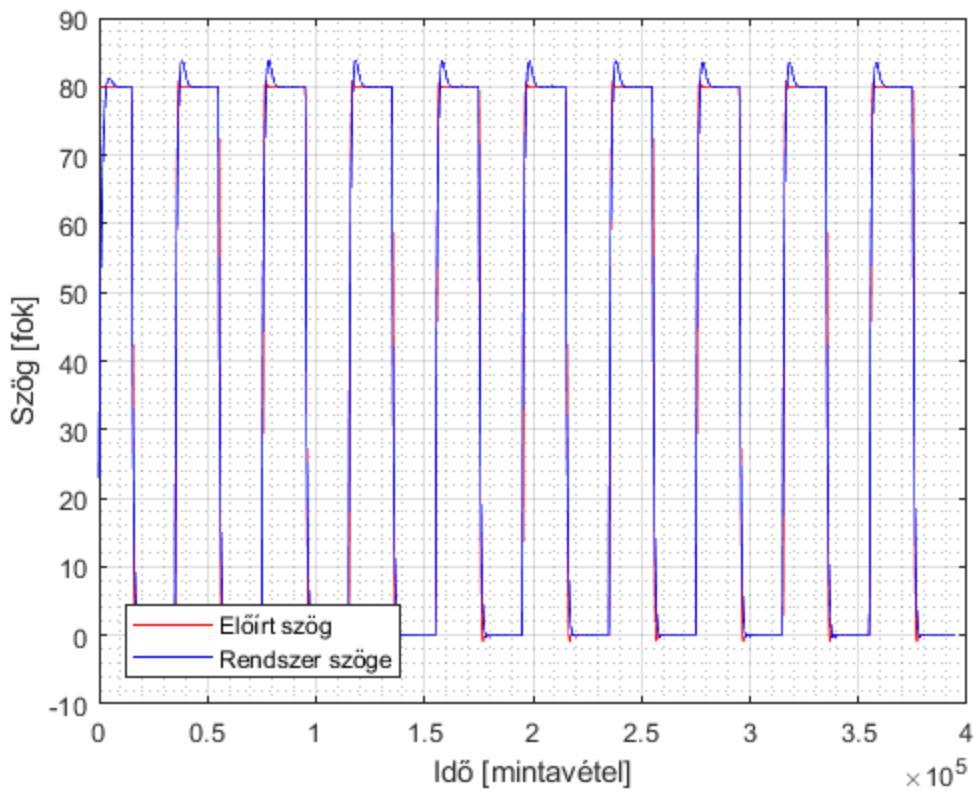
5-19 ábra RST szabályozó, jól megbecsült paraméterekkel



5-20 ábra RST szabályozó, rosszul megbecsült paraméterekkel

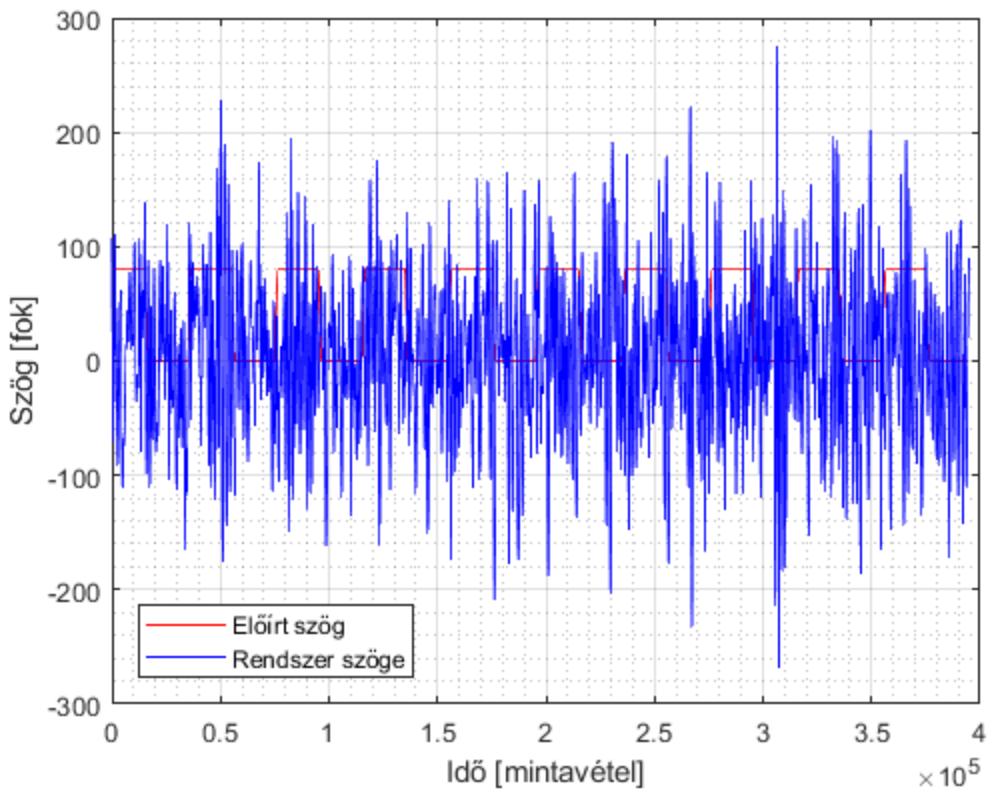
5.5. Adaptív MIT szabályozó

Az 5-21 ábrán látható az adaptív MIT szabályozó működése abban az esetben, amikor a rendszeren nincs zaj, valamint az előírt rendszer egy első fokú rendszer. ezen jól látható, hogy ez is szinte tökéletesen rááll, követi a referencia modell kimenetét a szimulációban.



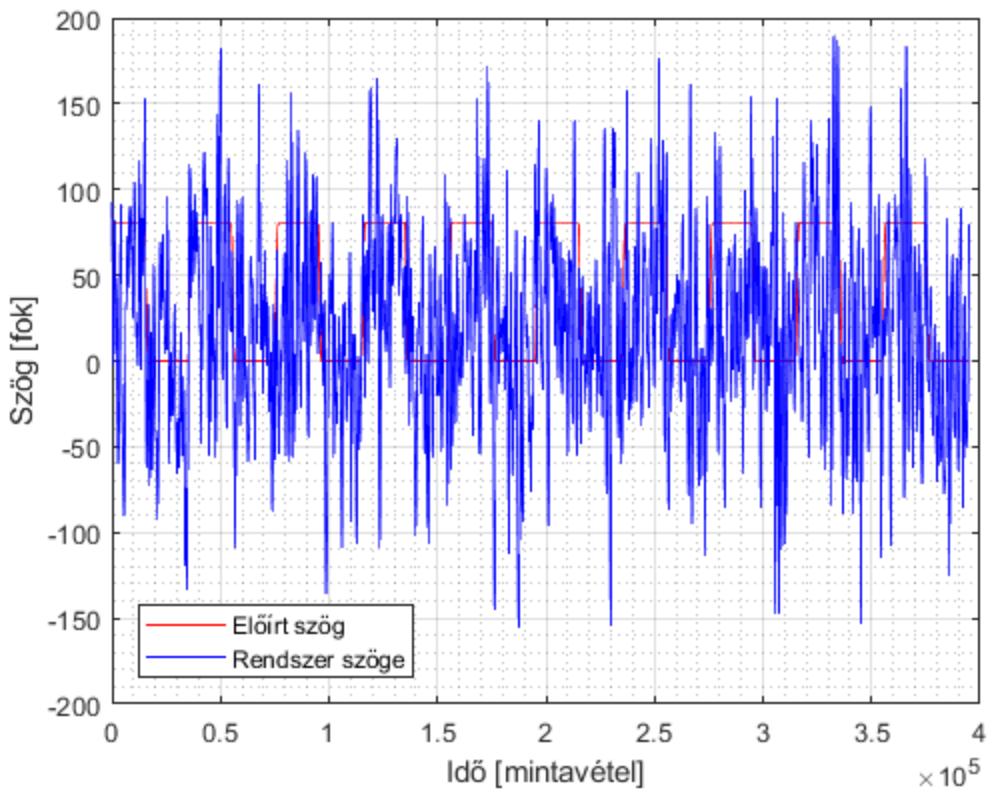
5-21 ábra Adaptív MIT szabályozó, zaj nélkül

Az 5-22 ábrán viszont ugyanaz az adaptív MIT szabályozó látható, viszont ebben az esetben a rendszer kimenetén egy maximális 3%-os zaj is van. A szabályozó kimenete e mellett korlátozva van, mivel nem lehet korlátlan nagyságú bemenetet adni a valós rendszernek. A rajzon látható, hogy a szabályozó folyamatosan túlszabályozza a rendszert, és nem tudja még megközelíteni sem az előírt rendszer működését.



5-22 ábra Adaptív MIT szabályozó, zajjal

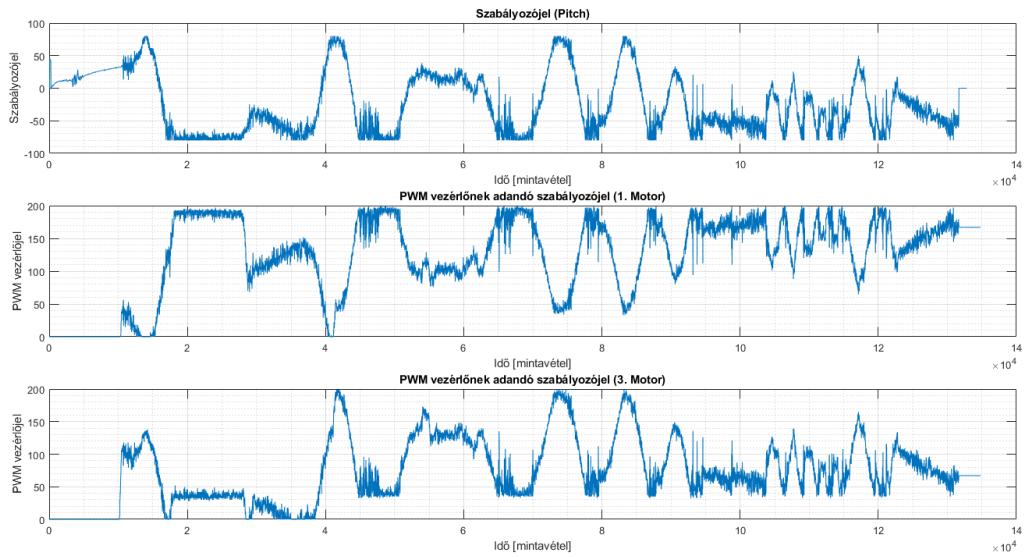
Az 5-23 ábrán ugyanaz a szabályozó, ugyanazokkal a paraméterekkel, a rendszerre került zaj viszont 0.5% alatti. Ebből a rajzból nehezen, viszont látható egyfajta tendencia, hogy a jel formáját próbálja követni valamilyen szinten. Az előbbihez képest nem annyira rossz, viszont ez még mindig nem megfelelő. További méréseket az adott szabályzóval nem érdemes végezni, mivel ennél kisebb zajt hiába is adnánk a rendszernek, mert ha szimulációs szinten a szabályozó jól működne, a valós rendszeren ezt nem lehetne alkalmazni, mivel ilyen minimális zajt nem lehet elérni.



5-23 ábra Adaptív MIT szabályozó, minimális zajjal

5.6. Vezérlőjelek összegzése

Mielőtt a propellerek rákerültek a rendszerre, szükség volt leellenőrizni, hogy a szabályozó jó irányba próbálja szabályozni a motorokat. Egy másik ellenőrzés, amit ugyancsak ekkor volt elvégezve az volt, hogy a szögszabályozó a megfelelő motorokat hajtsa-e meg. A motorpár és szögpozíció megfeleltetése úgy volt ellenőrizve, hogy amíg a szabályozó működtetve volt, a quadkoptert az egyik tengely mentén bedőltettem 30 fokkal. Hozzá kell tennem, hogy a motorok fordulatszáma alacsony volt, ami alatt azt értem, hogy amíg az egyik motor forgott, próbálta egyensúlyba hozni a rendszert, a másik vagy megállott, vagy olyan alacsony fordulaton forgott, hogy azt már könnyű volt elkülöníteni. Ez után a második tengelyt dőltettem egyik oldalról a másikra, és így figyeltem meg, hogy jó motorokat próbál meghajtani a szabályozó, és az irány is megfelelő-e. Mivel a harmadik szögpozíció jelenleg nincs implementálva a rendszerre, ezért ebben a mérésben az volt megfigyelve, hogy a motorvezérlő párokra kiadott jelek egy bizonyos szinten szimmetrikusak-e. Az 5-24 ábrán látható, hogy a motor fordulatszámának változtatásakor a motorokra kiadott vezérlőjelek szimmetrikusak.



5-24 ábra Motorokra adott vezérlőjeleknek a szimmetriája két szemközti motor esetén

6. Fejlesztés során felmerült problémák

Pmod NAV IMU szenzor esetében az a probléma jelentkezett, hogy amint feszültséget kapott a rendszer, az FPGA túl hamar kezdte el kiküldeni a konfigurációs paramétereket, aminek következtében a szenzortól csak rossz adatok jöttek. Ennek érdekében egy rövid ideig tartó reset áramkör van beiktatva, aminek köszönhetően az IMU szenzornak van ideje felkészülni a kapcsolat létesítésére. Ha ez az inicializáló fázis nincs kész, akkor az FPGA nem kap valós értékeket a szenzortól.

Egy második probléma, ami inkább figyelmetlenség volt az az, hogy a Xilinx által biztosított, *Matlab Simulink* alatt használandó *Cordic* modul nem volt megfelelően beállítva. A Cordic modul beállításakor szükséges megadni egy számot, amelynek megvan, hogy milyen biteket kell, hogy tartalmazzon, és hányat. Röviden, megvan, hogyan értelmezi a bemeneteket, és e szerint kellett ezeket kialakítani.

Egy harmadik probléma, ami adódott az a komplementer szűrőnek az alkalmazása volt, amelyet az IMU szenzor beállításai okoztak. Az volt a probléma, hogy a szenzortól kapott értékek adott mozgás esetén telítődtek, így a szűrő nem számított helyes eredményeket.

Egy újabb probléma, amin jelenleg is dolgozok az adaptív MIT szabályozás. A felmerült probléma az, hogy a rendszertől kapott szögpozíció értékek zajosak. A már fennebb bemutatott mérési eredményeknél látható, hogy nagyon kismértékű zaj esetén is a szabályozó bizonytalanná válik.

Egy újabb érzékteljesítő probléma az, hogy a rendszer becslésekor olykor a rendszerbecslő adott körülmények között megakad, és hosszabb idő elteltével sem tudja elégé megközelíteni a valós rendszer paramétereit. Ennek következtén megtörtént, hogy a rendszer becslésekor a becsült rendszer egyre csak rosszabb válaszokat adott. A kimenetet követni tudta, viszont a túllövés idővel növekedett a hibás becsült paraméterek miatt.

Az órajel-osztó során felmerült egy újabb említésre méltó probléma. Az általam használt Vivado fejlesztőkörnyezetben a jeleknek meg kell adni, hogy hány bitesek, illetve integer típusú jelre a maximális értékét. Az adott esetben, a bemenet 32 bitesnek volt megválasztva, a számláló típusa pedig integer-nek volt meghatározva. A számláló úgy volt létrehozva, hogy maximálisan 31-ig tudott elszámolni. A valós rendszeren ez problémát okozott, mivel 31- nél nagyobb értékekre az órajel-osztó nem működött, mivel az 5-bitesnek értelmezett számláló túlcordult.

7. Következtetések

Sikerült megvalósítani egy szögpozíciót követő módosított D csatornás PID szabályozót úgy, hogy a szenzorok beolvasása, szabályozójelek kiszámítása, és a motorok vezérlése is mind hardver szinten vannak megvalósítva, FPGA-ban. A komplementer szűrő egy erőforrás-igényes és egy egyszerűsített változata is ki van generálva. A rendszer jelen állapotában szükséges összekapcsolni a számítógéppel azért, hogy a paraméterek bekerüljenek a rendszerbe. Az elkészített terv előnye, hogy bármikor lehet változtatni a paraméterek negyrészét futás közben is. Az összegző, valamint a PID algoritmus is, ugyanúgy, mint a komplementer szűrő két változatban volt kigenerálva, így az erőforrás-szükségleteket sikeresen megfelelő mértékben lecsökkenteni.

Annak ellenére, hogy a PID szabályozó egy jól bevált módszer, szükség van a rendszer paramétereinek a behangolására, ami meglehetősen időigényes lehet, még abban az esetben is, ha olyan módszereket alkalmazunk, mint a Ziegler-Nichols módszer. Látható volt, hogy az adaptív MIT szabályozó habár jól megközelíti a rendszer működését, mégsem alkalmazható ilyen formában a zajok miatt. Az RST szabályozó esetében az egyedüli hátrány az, hogy szükséges

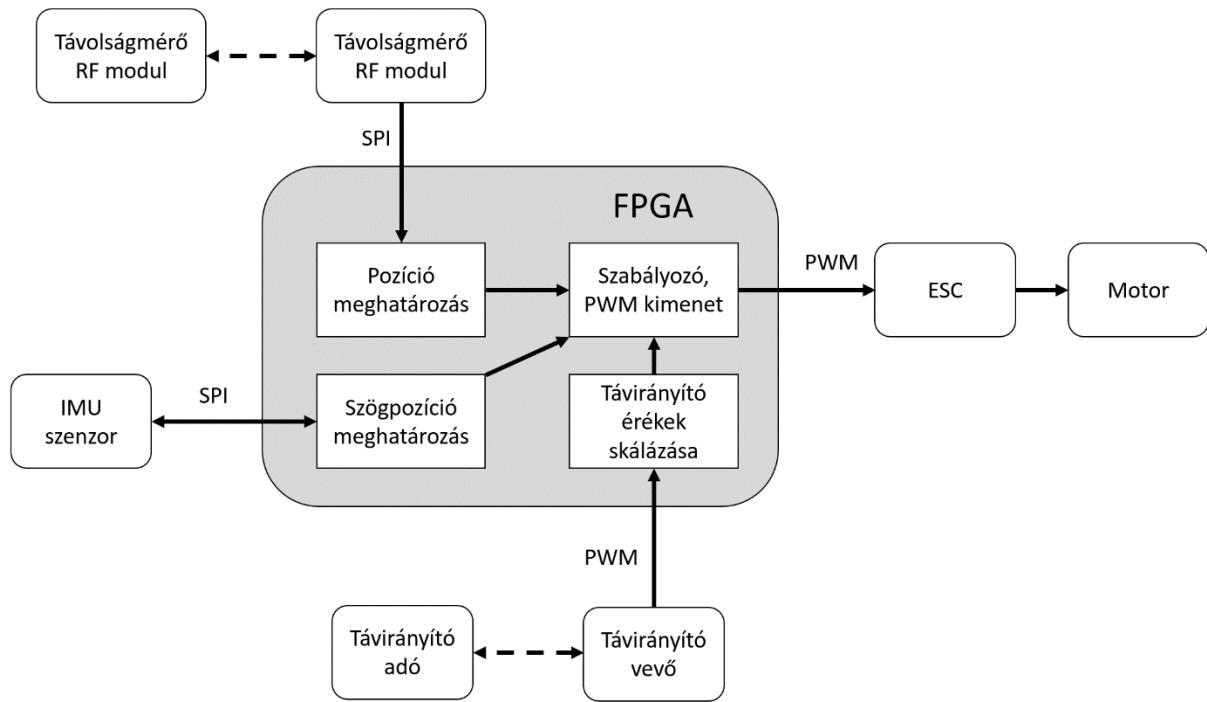
ismerni a rendszer paramétereit. Abban az esetben, ha nem ismertek, akkor becsülni kell őket, ami újabb erőforrásokat igényel.

8. Jövőbeli tervezek

A közeljövőben több minden szeretnénk tenni a jelenleg használt rendszerrel. Az egyik az, hogy a komplementer szűrőt szétszedjük két részre. Jelenleg az eltolások és a skálázások korrigálása is itt történik. Ezt tervezzük eltávolítani ebből a modulból, így ez különálló egységként működik majd. Ezzel az érhető el, hogy az értékek korrigálása és a komplementer szűrő is külön részben lesz, így modulárisabban lesz felépítve a rendszer. Egy másik terv, amit ugyanitt szeretnénk megtenni az, hogy leteszteljük több esetre is az újabb egyszerűsített változatban elkészített komplementer szűrőt, így erőforrás szabadulna fel, amit majd máshol modylok magvalósítására is lehet alkalmazni.

Amint azt már többször is említve volt, jelenleg egy módosított D csatornás PID szabályozó van megvalósítva. Szeretnénk tesztelni a rendszer szűrt D csatornás, valamint kaszkád PID szabályozóval is. Ezek mellett tervben van adaptív szabályozók kipróbálása, tesztelése is, mint például az RST szabályozó.

Abban az esetben, ha a rendszer elégé optimalizálva lesz, erőforrásigény szempontjából, akkor szeretnénk beépíteni egy lokalizáló egységet is a rendszerbe. Az eszköz, amellyel a pozíciót fogjuk meghatározni, speciális, erre a cérla kifejlesztett rádiófrekvenciás egységekkel van megvalósítva. [15] A rendszerre az a lokalizációs modul lesz alkalmazva, amelyet a 2018-as *Digilent Design Contest*-en[15], az MTDK-n[16] és TDK-n[17] mutattam be. A tervezett rendszer egyszerűsített tömbvázlata a 8-1 ábrán látható.



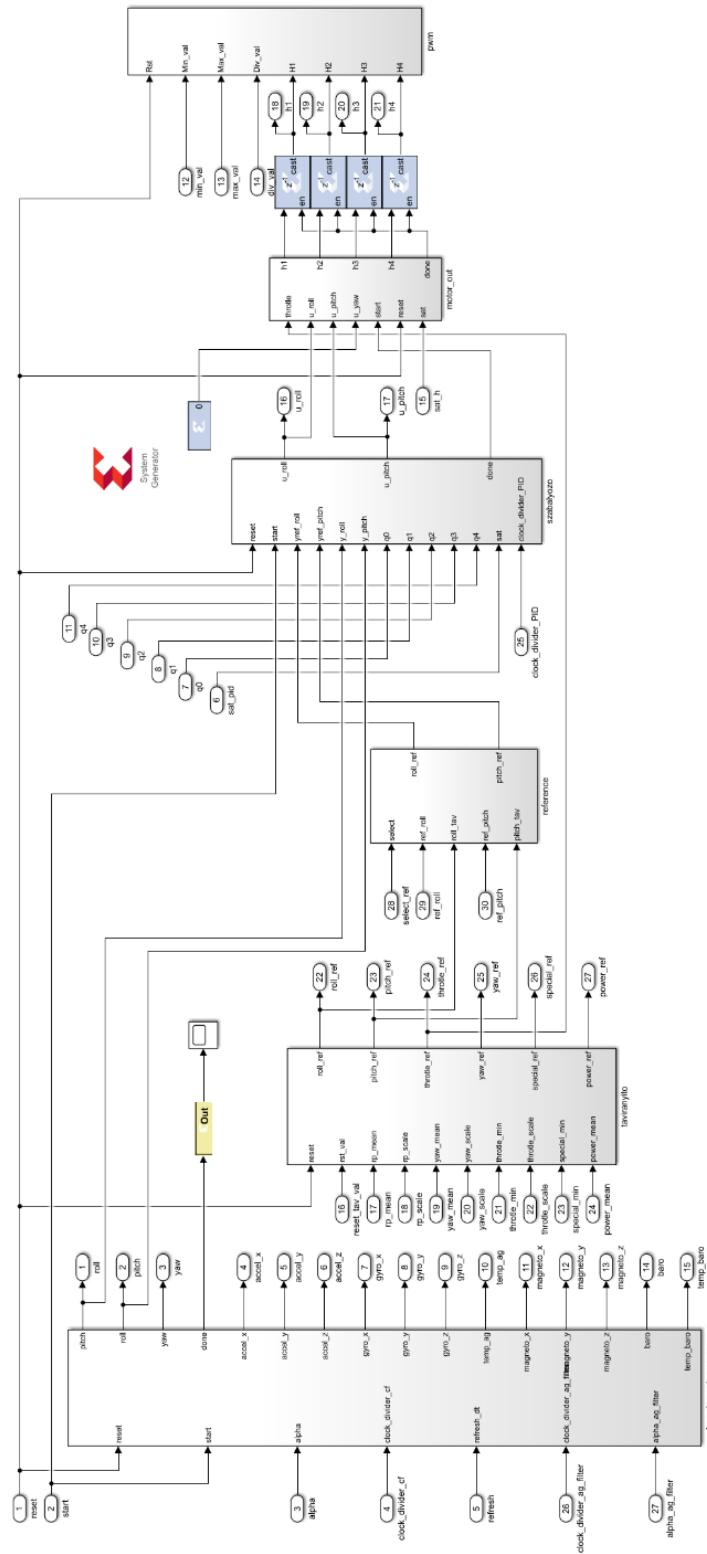
8-1 ábra Motorokra adott vezérlőjeleknek a szimmetriája két szemközti motor esetén

9. Könyvészet

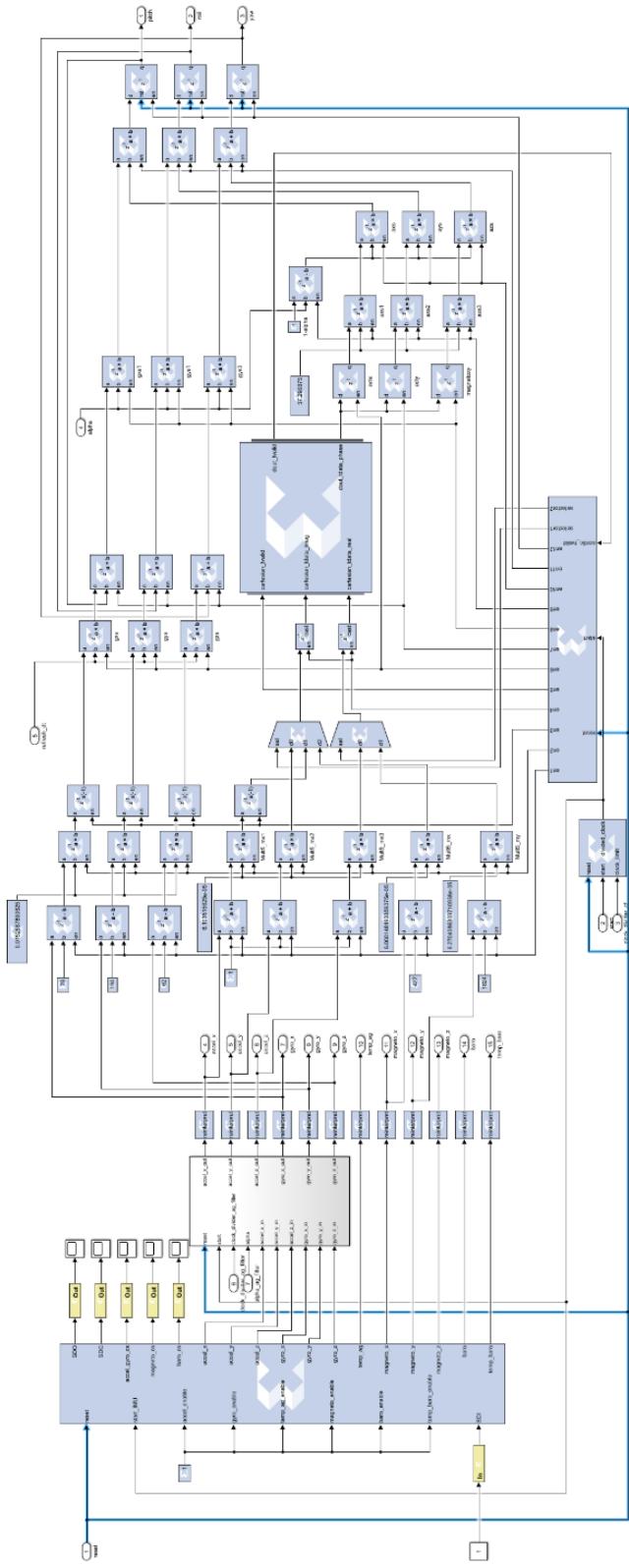
- [1]. Islam, T., Islam, M. S., Shajid-Ul-Mahmud, M., & Hossam-E-Haider, M. (2017, December). Comparison of complementary and kalman filter based data fusion for attitude heading reference system. In *AIP Conference Proceedings* (Vol. 1919, No. 1, p. 020002). AIP Publishing.
- [2]. Gaspar, J. A. (2008). Dead reckoning and magnetic declination: unveiling the mystery of portolan charts. *e-Perimetron*, 3(4), 191-203.
- [3]. Ozyagcilar, T. (2012). Implementing a tilt-compensated eCompass using accelerometer and magnetometer sensors. *Freescale semiconductor, AN*, 4248.
- [4]. Andraka, R. (1998, March). A survey of CORDIC algorithms for FPGA based computers. In *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays* (pp. 191-200). ACM.
- [5]. Márton Lőrinc, Irányítástechnika, Scientia, România, 2009
- [6]. Brassai Sándor Tihamér, Újrakonfigurálható digitális áramkörök tervezési és tesztelési módszerei, Scientia Kiadó, Kolozsvár, 2018, ISBN 978 606 975 020-9
- [7]. Zimmerman, N. M. (2016). Flight control and hardware design of multi-rotor systems.
- [8]. Lavretsky, E., & Wise, K. A. (2013). Robust adaptive control. In *Robust and adaptive control* (pp. 317-353). Springer, London.
- [9]. Jain, P., & Nigam, M. J. (2013). Design of a model reference adaptive controller using modified MIT rule for a second order system. *Advance in Electronic and Electric Engineering*, 3(4), 477-484.
- [10]. Gopalakrishnan, E. (2017). Quadcopter flight mechanics model and control algorithms. *Supervisor: Prof. Dr. Martin Hromčík. Prague*.
- [11]. György K., Dávid L., Identificarea sistemelor, Lucrări de laborator, Universitatea “Petru Maior”, Tg. Mureş, 2006
- [12]. Hajdú Sz., Brassai ST., Székely I, FPGA based angular stabilization of a quadcopter, MACRo 2017 - 6 th International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics, Marosvásárhely, 2017
- [13]. Bardócz Csaba, RC MODELL REPÜLŐ PÁLYA MEGHATÁROZÁSA, Diplomadolgozat, Sapientia EMTE, Automatika és alkalmazott informatika, 2017
- [14]. <https://digilentdesigncontest.com/2018-winners/>

- [15]. **Székely István-Zsolt**, LOCALIZATION BASED ON RF MODULES, Digilent Design Contest, Cluj-Napoca, 2018
- [16]. **Székely István-Zsolt**, RF modulokon alapuló helymeghatározás FPGA segítségével, XIX. MTDK, Temesvár, 2018
- [17]. **Székely István-Zsolt**, RF modulokon alapuló helymeghatározás FPGA segítségével, XVII. Sapientia TDK, Marosvásárhely, 2018
- [18]. <http://mathworld.wolfram.com/EulerAngles.html>
- [19]. https://hu.wikipedia.org/wiki/M%C3%A1gneses_elhajl%C3%A1s
- [20]. <https://www.renesas.com/us/en/solutions/key-technology/fpga-power-solutions/fpga-power-xilinx.html>
- [21]. <https://www.elprocus.com/fpga-architecture-and-applications/>
- [22]. <http://www.modellismo.com/2012/05/radiocomando-art-tech-e-fly-100c-6ch-e-ricevente-usato-modellismo/>
- [23]. https://www.researchgate.net/figure/Modern-Xilinx-FPGA-architecture-showing-different-basic-components_fig1_282398023
- [24]. System Generator for DSP, 2009, Xilinx
- [25]. <https://store.digilentinc.com/pmod-nav-9-axis-imu-plus-barometer/>

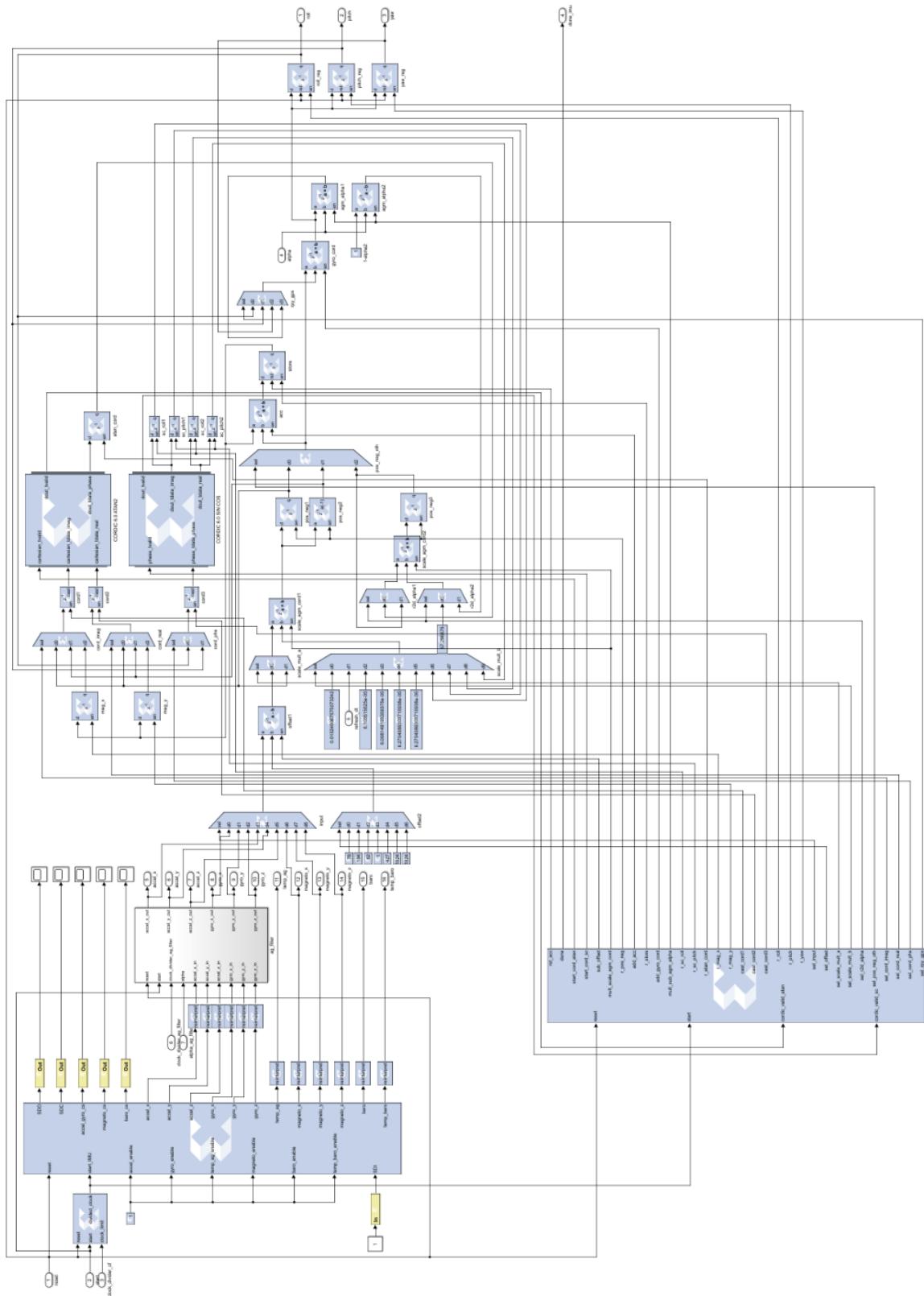
10. Függelék

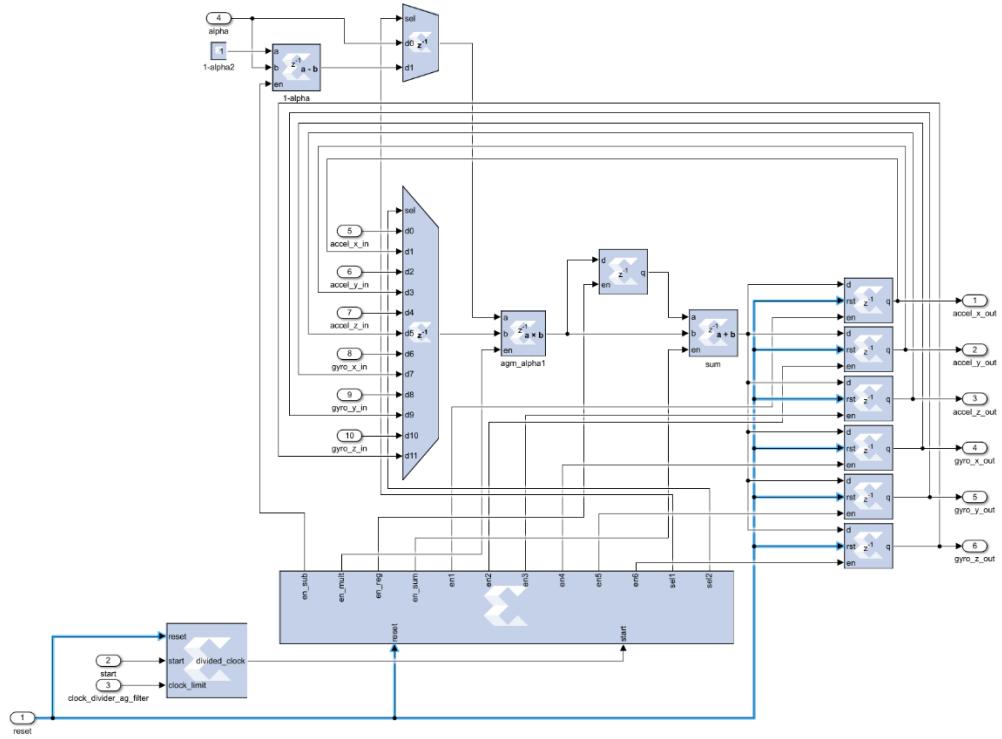


10-1 ábra A rendszer belső kapcsolási rajza

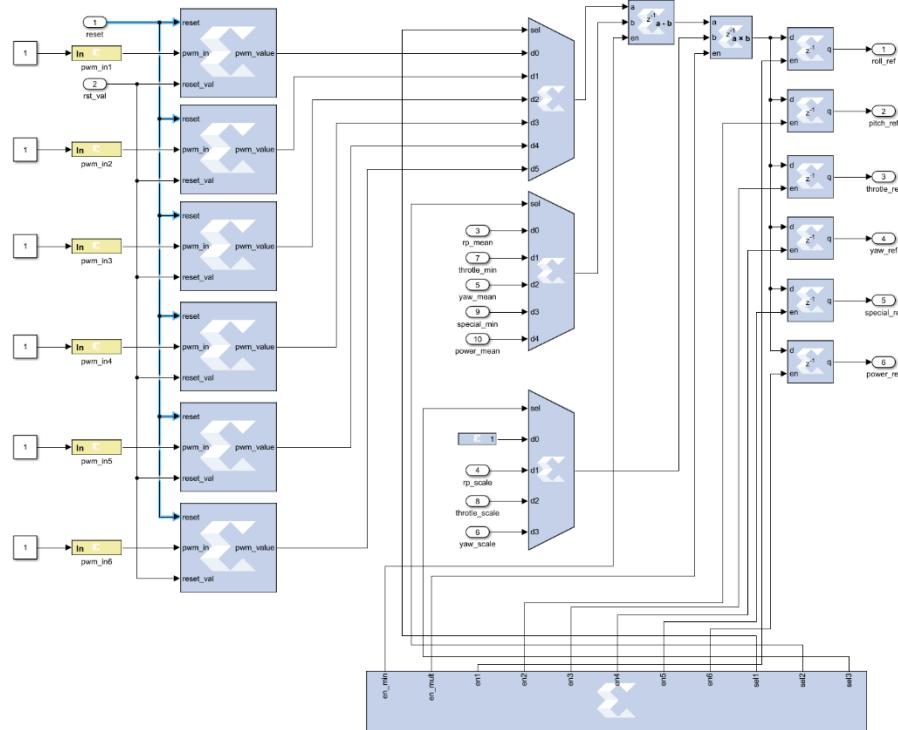


10-2 ábra IMU szenzor, átlagoló IIR szűrő, és a komplementer szűrő kapcsolási rajza

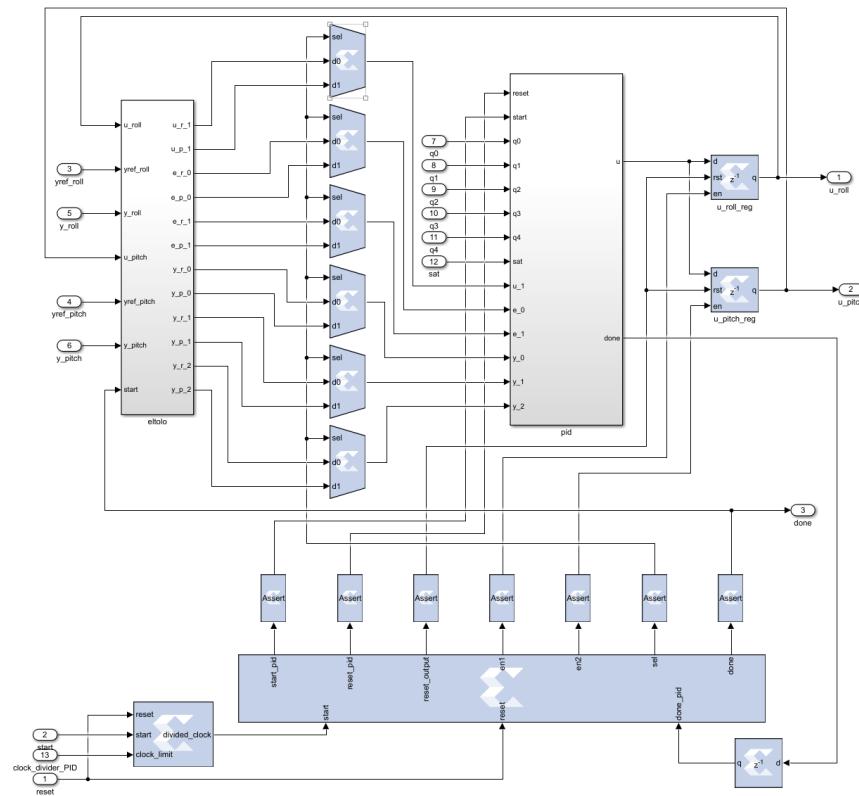




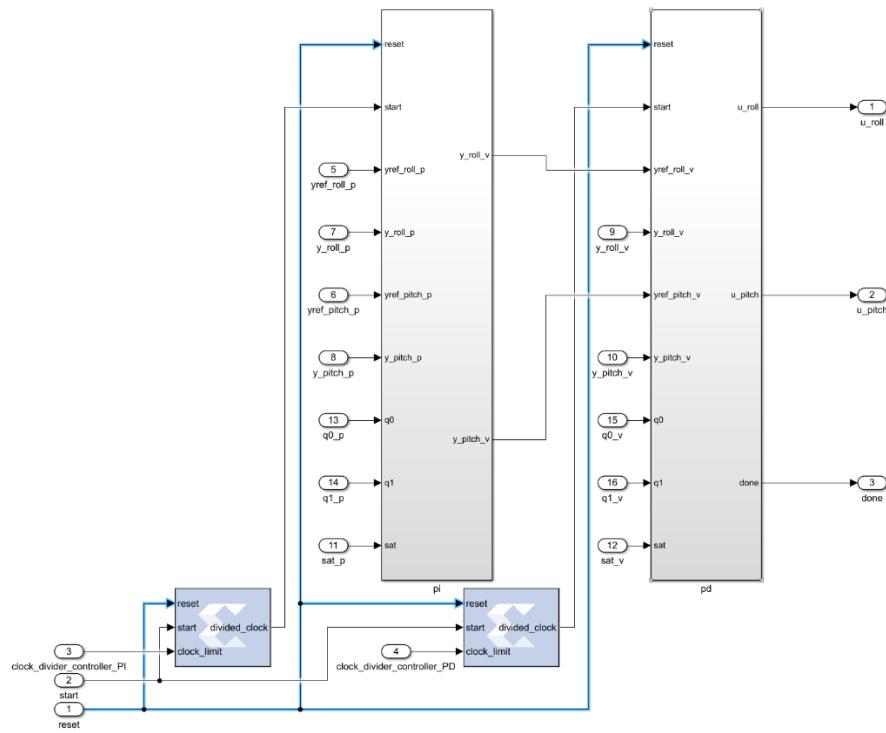
10-4 ábra Átlagoló IIR szűrő kapcsolási rajza



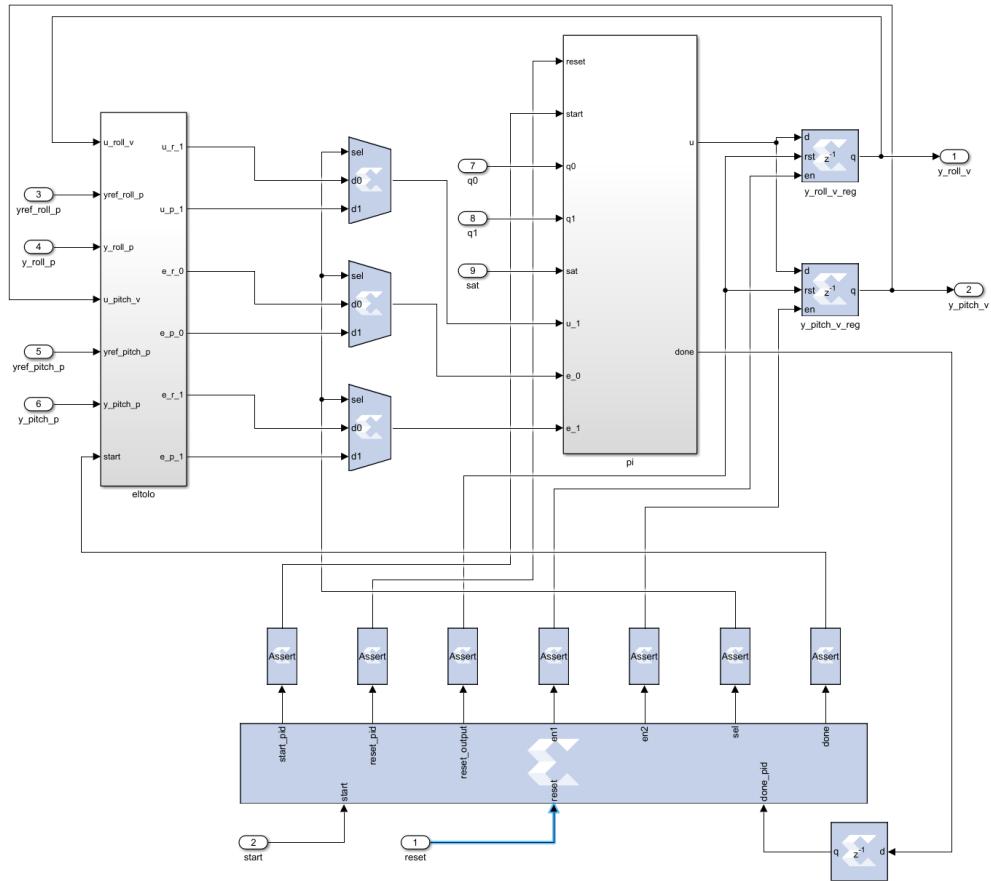
10-5 ábra Távirányító értékek beolvasásának és skálázásának kapcsolási rajza



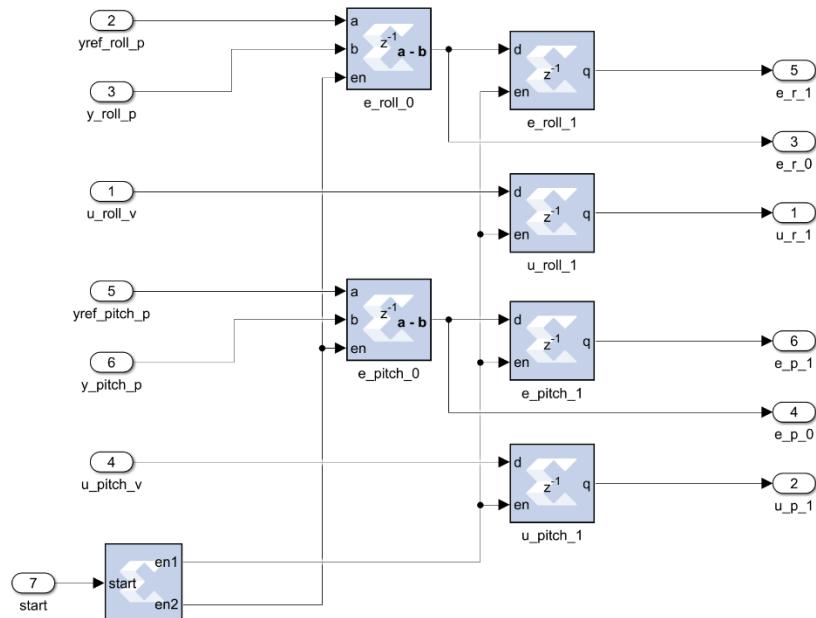
10-6 ábra Módosított D csatornás PID szabályozó, vezérlő, és tároló kapcsolási rajza



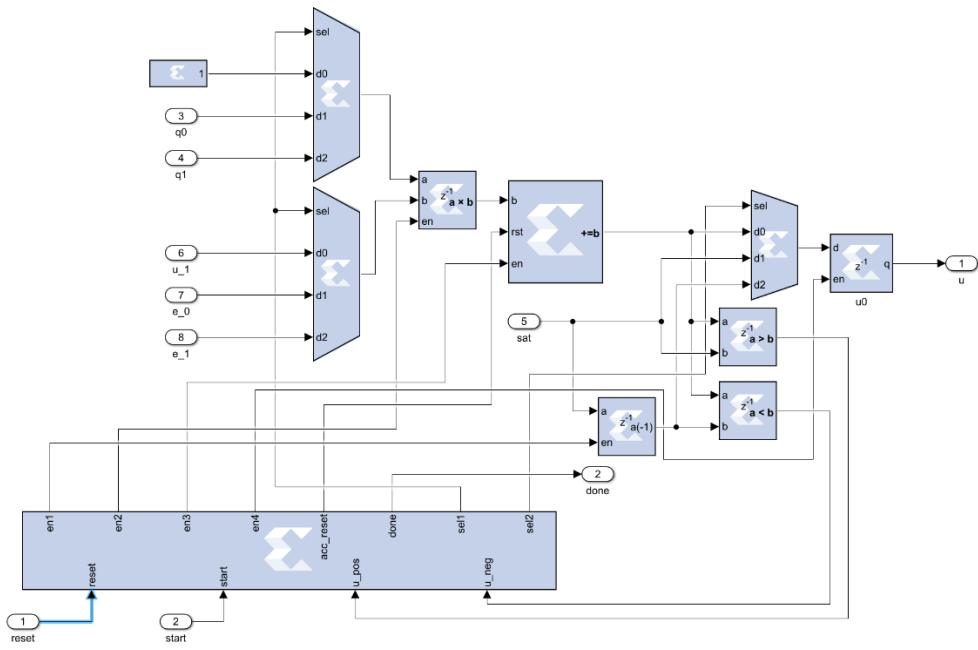
10-7 ábra Kaszkád PID szabályozó kapcsolási rajza



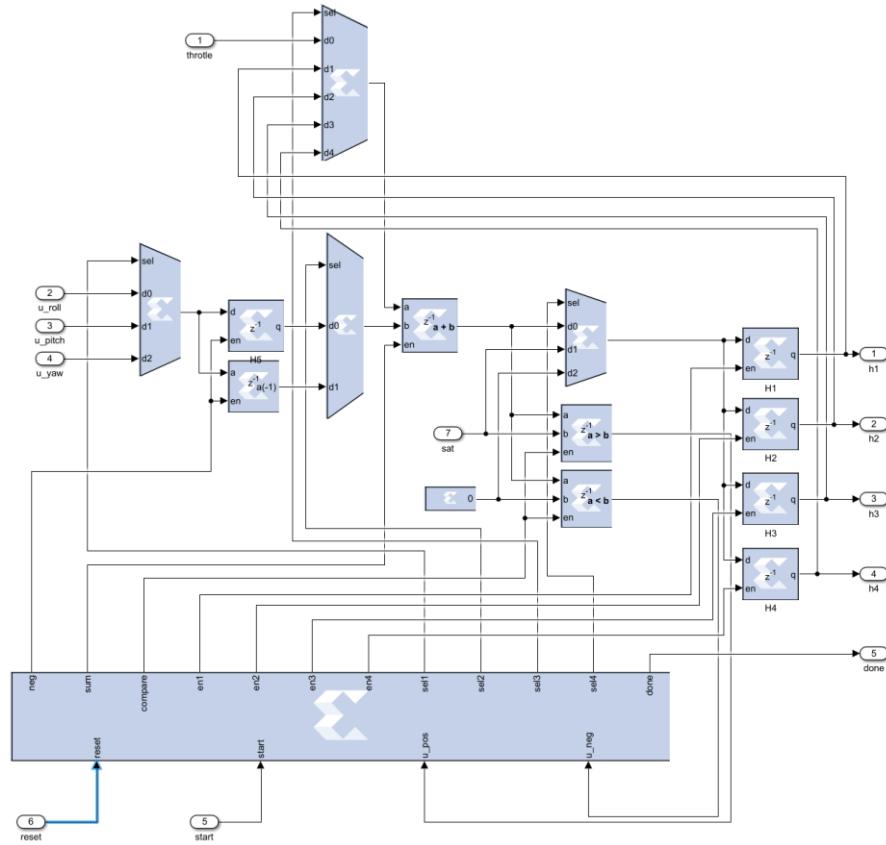
10-8 ábra PI szögpozíció szabályozó, vezérlő, és tároló egységének kapcsolási rajza



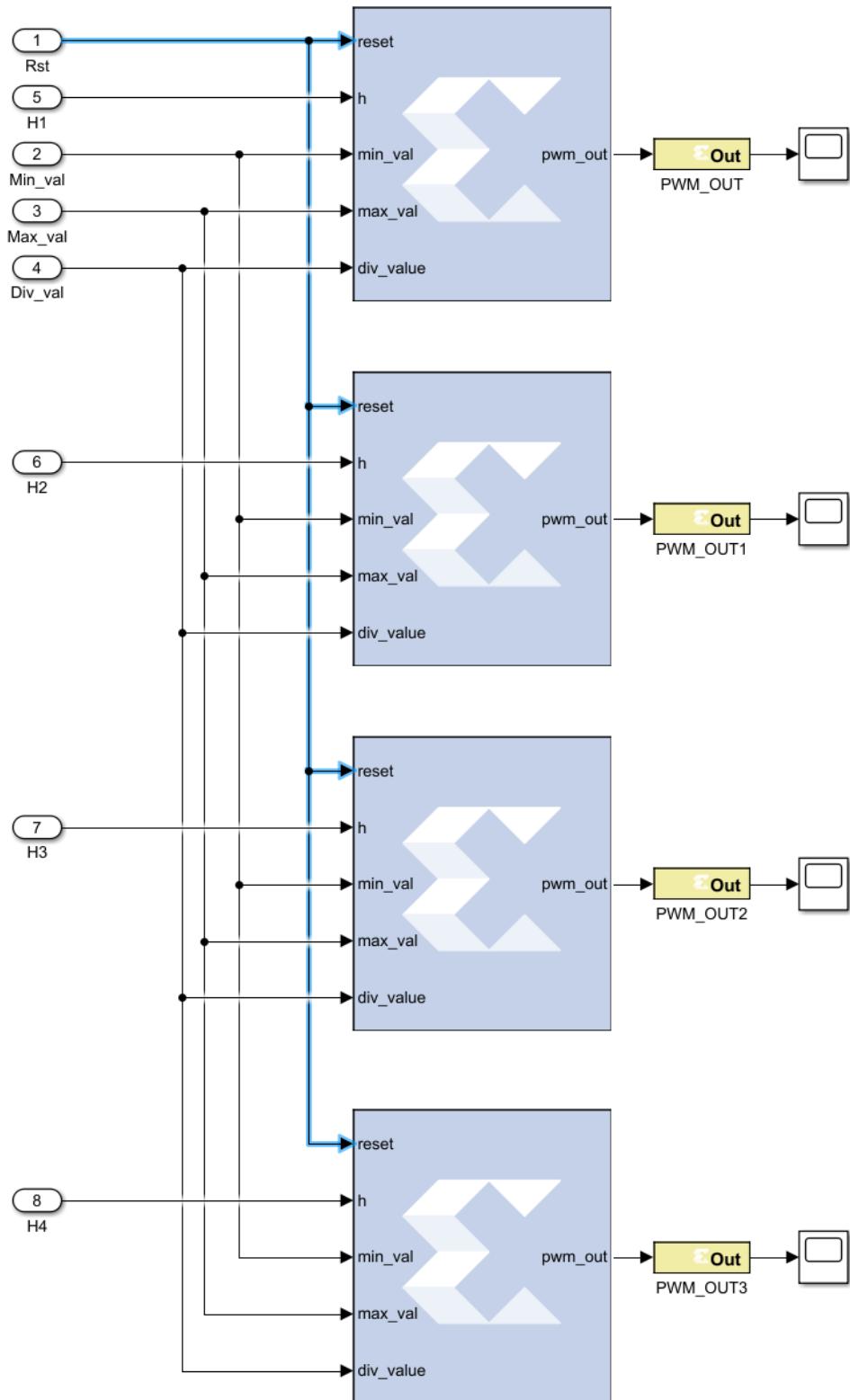
10-9 ábra Értékeket tároló egység kapcsolási rajza



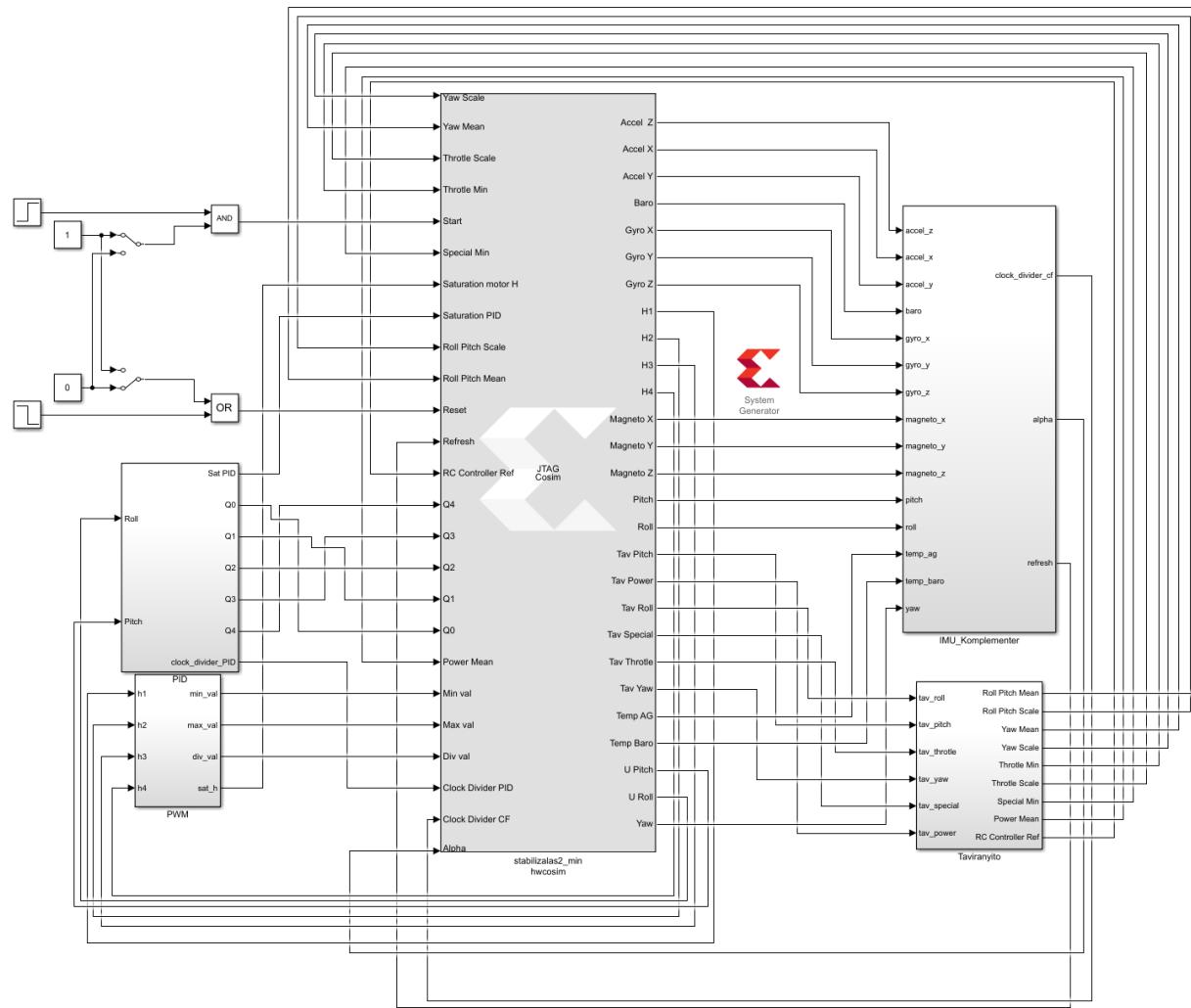
10-10 ábra PI szabályozó kapcsolási rajza



10-11 ábra Szabályozójelek összegzőjének kapcsolási rajza



10-12 ábra PWM jelgenerátor kapcsolási rajza



10-13 ábra Hardver ko-szimuláció kapcsolási rajza

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA

FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÂRGU MUREŞ

SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

Vizat decan

Ş. l. .Dr. ing. Kelemen András

Vizat director departament

Ş. l. Dr. ing. Domokos József