**Sapientia EMTE**

**Műszaki és Humántudományok Kar, Marosvásárhely**

**Villamosmérnöki Tanszék**

iRoobo

TDK dolgozat

Szakirányító tanár: Végzős hallgató:

Dr. Brassai Sándor Tihamér Gábor Szabolcs-László

adjunktus Automatizálás IV. év

**2015**

[Ábrák, táblázatok jegyzéke 4](#_Toc421979153)

[1 Bevevezető 7](#_Toc421979154)

[2 BIBLIOGRÁFIAI TANULMÁNY 9](#_Toc421979155)

[2.1 Hasonló FPGA fejlesztőrendszeren megvalósított PID szabályzók 9](#_Toc421979156)

[2.2 Inkrementális érzékelő 9](#_Toc421979157)

[2.3 Szögsebesség mérése FPGA segítségével 10](#_Toc421979158)

[2.4 Egyenáramú szervo motorok 10](#_Toc421979159)

[3 A RENDSZER TERVEZÉSE 12](#_Toc421979160)

[3.1 SZABÁLYOZÓK: 12](#_Toc421979161)

[3.1.1 Diszkrét Hardveres PID szabályozó 12](#_Toc421979162)

[3.1.1.1 Megvalósítás System Generátorban 13](#_Toc421979163)

[3.1.1.2 Szoftveres Szimulációs eredmények 14](#_Toc421979164)

[3.1.1.3 Q paraméterek számolása alapján. 16](#_Toc421979165)

[3.1.2 Mintavételi periódus jelének A generálása 16](#_Toc421979166)

[3.1.3 Pozíció Szabályzása 17](#_Toc421979167)

[3.1.3.1 A szabályozó felépítése: 17](#_Toc421979168)

[3.1.3.2 Szabályozó szimulálása 18](#_Toc421979169)

[3.1.4 Hardveres mérések 19](#_Toc421979170)

[3.1.4.1 Dc motor sebesség szabályzása mérőstandon 19](#_Toc421979171)

[3.1.4.2 Dc motor pozíció szabályzása mérőstandon 20](#_Toc421979172)

[3.2 Sebesség és pozíció szabályozót tartalmazó IP mag generálása System Generator-ban 21](#_Toc421979173)

[3.3 Szenzorok 25](#_Toc421979174)

[3.3.1 Inkrementális ÉrZékelő 25](#_Toc421979175)

[3.3.1.1 Optikai inkrementális vevő felépítése 25](#_Toc421979176)

[3.3.2 Inkrementális érzékelő jeleinek a feldolgozása FPGA áramkör segítségével 26](#_Toc421979177)

[3.3.2.1 Szimuláció System Generatorban 27](#_Toc421979178)

[3.3.2.2 Pozíció mérése Inkrementális adó segítségével 28](#_Toc421979179)

[3.3.2.3 Szögsebesség mérése Inkrementális adó segítségével 29](#_Toc421979180)

[3.3.2.3.1 Hardveres mérések 30](#_Toc421979181)

[3.4 MPU-6050 giroszkóp és gyorsulásmérő 32](#_Toc421979182)

[3.5 Beavatkozó elemek: 32](#_Toc421979183)

[3.5.1 Pwm Generátor megvalósítása FPGA áramkörön System Generator környezetben. 32](#_Toc421979184)

[3.5.1.1 Megvalósítás 32](#_Toc421979185)

[3.6 Elektronika 34](#_Toc421979186)

[3.6.1 Digitális Elektronika 34](#_Toc421979187)

[3.6.2 FPGA Rendszer Felépítése 35](#_Toc421979188)

[3.6.2.1 Zybo FPGA fejlesztőlap 35](#_Toc421979189)

[3.6.2.2 Spartan3e FPGA fejlesztőlap 38](#_Toc421979190)

[3.6.2.3 Kommunikációs protokollok 38](#_Toc421979191)

[3.6.2.3.1 SPI kommunikációs protokoll: 38](#_Toc421979192)

[3.6.2.3.2 Ethernet 39](#_Toc421979193)

[3.6.3 Feladatok Elosztása 39](#_Toc421979194)

[3.6.3.1 Zybo fejlesztőlap 39](#_Toc421979195)

[3.6.3.2 Spartan fejlesztőlap 40](#_Toc421979196)

[3.7 GUI felépítése 40](#_Toc421979197)

[3.8 Teljesítmény elektronika 40](#_Toc421979198)

[3.8.1 Bootstramp müködése 45](#_Toc421979199)

[3.9 Robot Modell 46](#_Toc421979200)

[4 Robot Mechanikai Felépítése 50](#_Toc421979201)

[5 Elért eredmények, magvalósítások: 52](#_Toc421979202)

[6 Bibliography 53](#_Toc421979203)

[7 FÜGGELÉK 54](#_Toc421979204)

Ábrák, táblázatok jegyzéke

[Kép. 2.1Inkrementális érzékelő jelek 9](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907504)

[Kép. 3.1 a Pozíció és a sebesség szabályzási hurok elvi strukturális felépítése 12](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907505)

[Kép. 3.2 Állapot automata, amely leírja a Diszkrét PID szabály ózót 13](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907506)

[Kép. 3.3 A PID felépítése System Generatorban 14](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907507)

[Kép. 3.4 PID szoftveres szimulációs model 14](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907508)

[Kép. 3.5 Szimulációs eredmény amely tükrözi a konstans bementre a számolási lépéseket 15](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907509)

[Kép. 3.6 PID minimális periódusa 15](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907510)

[Kép. 3.7 Mintavételi taktust generáló modul 16](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907511)

[Kép. 3.8 Szimulációs eredmények mintavételi jelgenerátor. 17](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907512)

[Kép. 3.9 A Pozíció szabályozó System generátoros felépítése 18](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907513)

[Kép. 3.10 a pozíció szabályzás moduláris felépítése System Genrator környezetben 19](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907514)

[Kép. 3.11 a pozíció szabályozó bemenete (pozíció hiba), valamint a Szabályozó kimenti jele. 19](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907515)

[Kép. 3.12 Sebesség szabályozás PID szabályzóval. 20](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907516)

[Kép. 3.13 Pozíció szabályzása a Kép. 3.9 látható kialakításban. 20](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907517)

[Kép. 3.14 A két szabályozót tartalmazó modul System generatoros felepitesben 22](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907518)

[Kép. 3.15 Pozíció Szab modul belső felépítése a Kép. 3.14 23](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907519)

[Kép. 3.16 Sebesség Szab modul felépítése a Kép. 3.14 24](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907520)

[Kép. 3.17 Optikai inkrementális vevő felépítése és elhelyezése 25](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907521)

[Kép. 3.18 Érzékelő tranzisztorok elhelyezése 25](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907522)

[Kép. 3.19 Idődiagram a Tárcsa paraméterei függvényében 26](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907523)

[Kép. 3.20 Rések és az Érzékelők közti kapcsolat 26](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907524)

[Kép. 3.21 Inkrementális Jelfeldolgozó modul1 érzékelő modul belső felépítése 27](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907525)

[Kép. 3.22 Inkrementális érzékelőtől érkező jelek átalakító irány és impulzus jelekre 27](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907526)

[Kép. 3.23 Szimulációs eredmények a lehetséges bemenetekről az Black Box1 modulba 28](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907527)

[Kép. 3.24 Inkrementális adóval mért pozicó, szimulálációs modelje Sytem Generatorban 28](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907528)

[Kép. 3.25 Sebesség mérő modul felépítése 29](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907529)

[Kép. 3.26 Dc motor sebességének mérése Fpga lapon 30](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907530)

[Kép. 3.27 Dc motor Sebesség mérése FPGA rendszeren, System generatorban megvalósítva 30](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907531)

[Kép. 3.30.a 8ms 31](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907532)

[Kép. 3.30.c 4ms 31](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907533)

[Kép. 3.30.b 80ms 31](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907534)

[Kép. 3.31 A PWM generátor System Generátorban megvalósított szerkezete 33](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907535)

[Kép. 3.32 a PWM generátor bemenő, kimenő illetve néhány belső jele (Scope1) 34](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907536)

[Kép. 3.33 rendszer elvi felépítése 35](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907537)

[Kép. 3.34 Kommunikációs csomagok és az FPGA áramkörökbe programozott modulok elvi felépítése 36](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907538)

[Kép. 3.35 ZYBO Core0 program folyamat árbája 37](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907539)

[Kép. 3.36 Spartan3e500, microblaze szoftver Folyamat ábrája 38](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907540)

[Kép. 3.37 Hip4082 alkalmazása H híd kapcsolásban. Forrás: http://www.intersil.com/en/products/space-and-harsh-environment/harsh-environment/half--full-bridge-and-three-phase-drivers/HIP4082.html 40](#_Toc421907541)

[Kép. 3.38 Két hídvezérlő árak kör kapcsolási rajza HIP4082 integrált áramkörrel megvalósítva 41](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907542)

[Kép. 3.39 Nem invertáló erősítő forrás [13] 42](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907543)

[Kép. 3.40 Dupla hídvezérlő áramkor vezérlő jelei JP2 csatlakozó a Kép. 3.36-n. 42](#_Toc421907544)

[Kép. 3.41 H híd tranzisztorainak a Gate vezetékei 43](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907545)

[Kép. 3.42 A négy Kép. 3.41 látható szalagvezeték jelenik meg a Buszvezetékben. 43](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907546)

[Kép. 3.43FPGA kimentének a védelme 43](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907547)

[Kép. 3.44 A robot energia ellátása valamint a hűtő rendszer elvi felépítése 44](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907548)

[Kép. 3.45 Vízpumpa és a ventilátor motorjának vezérlő teljesítmény elektronikai kapcsolása 45](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907549)

[Kép. 3.46 Bootstramp megoldás a felső tranzisztor Gate bemenetének a meghajtására 45](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907550)

[Kép. 3.47 Bootstramp kondenzátor feszültsége a W és W11 pontokban 46](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907551)

[Kép. 3.48 Robot kerekek sebsége és a robot mozgásának viszonya 46](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907552)

[Kép. 3.49 Robot 3D vektorábrája 48](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907553)

[Kép. 3.50 Oldalnézetek és Felülnéztet, jelölések szemléltetése 49](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907554)

[Kép. 4.1 Robot vázának Inventoros 3D Képe 50](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\Dolgozat_v4.docx#_Toc421907555)

[Táblázat. 3‑1 Manuálisan számolt értékek a szimuláció ellenőrzése kedvéért 15](#_Toc421907556)

# **Bevevezető**

A dolgozat célja mobilis tereprobot tervezése és megépítéséhez szükséges elemek tárgyalása. A mechanikai rendszer Autodesk Inventor-ban volt megtervezve, és az elkészített terv alapján kivitelezve. A következő részfeladat a vezérlő elektronika kialakításának a tervezése és a szenzoroknak a rendszerbe való integrálása volt. A rendszeren különböző szenzorok találhatók, amelyek közül talán a legfontosabb a, sebesség és pozíció mérésére alkalmazott inkrementális jeladó. A rendszeren megtalálható két FPGA fejlesztő lap, egy nagyobb erőforrásokkal rendelkező Zybo (beépített ARM processzorral), amely a matematikai számítások elvégzésére hivatott, és egy kisebb kapacitású FPGA lap (SPARTAN3e500) amely tartalmaz 8 hardveresen megvalósított szabályozót és egy MicroBlaze procesort, a szabályzók 12V DC motor sebességét vagy pozícióját szabályozzák. A Microblaze processzor feladata az adatok fogadása az és egy egyszerű feldolgozás után a megfelelő hardveres szabályozó osztott regiszterébe való írása. A SPARTAN3e500 laphoz van illesztve 8 db. inkrementális érzékelő, amelyek a motorok pozícióját illetve sebességét mérik. A szenzorok adatait a Spartan lap egy gyors SPI kommunikáción keresztül küldi tovább a Zybo lapnak. A gyors hardveres PID szabályozó megvalósítása FPGA fejlesztőlapon Xilinx System Generator tervezőeszközzel készült, a szimulációkat hardver co-szimulációval végeztem el.

A tervezést a mechanikai rendszerrel kezdtem Autodesk Inventor segítségével több változatot is megterveztem ameddig eljutottam a dolgozatban tárgyalt mechanikai struktúrához. A mechanikai rendszert, saját magam viteleztem ki a tervek alapján, a kivitelezés után tesztet végeztem, amely során a fogaskerék áttételeket teszteltem. Az eredmények azt mutatták, hogy a rendszert módosítására van szükség a motorok rögzítése miatt. A mechanikai rendszer felépítése és újratervezése két hónapba telt.

A szoftver és digitális hardver fejlesztésére FPGA rendszert választottam, mert könnyen fejleszthető a szoftver és a hardver közösen. A szenzorokat úgy választottam, meg hogy könnyen illeszthető legyen az FPGA rendszerhez, minden szenzor 3,3V feszültségszinten dolgozik.

Az inkrementális szenzorok jeleinek a feldolgozására szolgáló modult System Generátorban valósítottam meg. Miután tudtam mérni a pozíciót és a sebességet, megterveztem a rendszer működéséhez szükséges szabályozókat elsőként a PID szabályozót, megpróbáltam alkalmazni a sebesség és pozíció szabályzására is, de az eredmények arra vezettek, hogy a PID nem hatékony a pozíció szabályzás elvégzésére. A rendszer áttételében levő kotyogás miatt feleslegesen korrigálta a pozíciót, ezért kialakítottam egy másik szabályozó elgondolást, amely működő képesnek bizonyult.

A dolgozatban bemutatjuk PWM generátor, PID szabályozó, pozíció szabályozó inkrementális érzékelő adatainak a feldolgozó modulját, a megvalósítását System Generátor környezetben, és a modulokkal végzett hardveres és szoftveres szimulációkat.

A robotot robot vázához rögzíteni lehet nagyobb tömegű kiegészítő tartozékokat pl.: robotkar, fűnyíró, stb. Alkalmazhatósága elképzelhet a mezőgazdaságban, mint gyomtalanító gép, vagy akár a biztonság technikában, mint beavatkozó eszköz.

# **BIBLIOGRÁFIAI TANULMÁNY**

## Hasonló FPGA fejlesztőrendszeren megvalósított PID szabályzók

A [1] cikkben tárgyalt FPGA erőforráson megvalósított PID szabályozó, amelyet a nagyobb működési sebesség kedvéért FPGA alapon valósított meg. A szabályzó paraméterei fordításkor kerülnek, bele a hardverben kívülről nem tudja megadni, ami a hangolás szemszögéből nem elönyős.

A [2] PID szabályzó folytonos átviteli függvényéből indul ki, és levezeti a diszkrét átviteli függvényt, amelyből majd a rekurzív mintavételes szabályzót kapja meg.

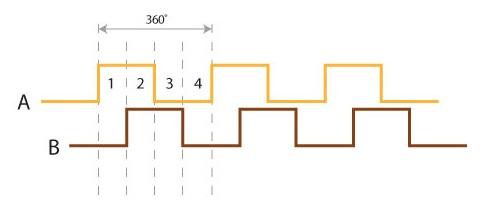
[2]

A Q paraméterek konstansak, és a rendszer kimenete megkapható három összeadás, és három szorzás elvégzésével. Az összefüggések a [3] irodalomban bemutatott elemekkel meglehet valósítani. Az általam használt FPGA fejlesztőrendszeren a [4] irodalomban találtam rá. A PID szabályozó követi a hagyományos három P,D,I tagokból álló elrendezést, amelyek csővezeték szerűen vannak illesztve egymáshoz. A szabályzó kimenete közvetlenül illesztve van egy PWM generátor modulhoz. A szabályozónak három órajelre van szükség a műveletek elvégzéséhez. A generátor képes a kettes komplemens értéket PWM jelé és egy irányjelé átalakítani, így téve lehetővé a teljes híd kapcsolás vezérlését, valamint meg egy engedélyező jelet is kivezet. A PID kimente egy 15 bites előjeles szám, és az előjel bit segítségével generálja ki az irányjelet, közvetlenül kivezet.

## Inkrementális érzékelő

Az inkrementális érzékelőknek két kimenete van jelölésük általában A, és B a két jel időbeni viszonya alapján tudjuk megállapítani a forgás irányát. A jelek generálódása egy dioda és egy optikai tranzisztor segítségével történik miközben egy tárcsa, amelyen ablakok találhatók, és mozgáskor

elhalad a dióda és a tranzisztor között A két jel időben 90 fokos késésben van egymáshoz viszonyítva. Az érzékelők alkalmasak szögsebesség, szögelfordulás mér érésére.



Kép. 2.1Inkrementális érzékelő jelek

Az elfordulással arányosan impulzusokat adnak vissza a kimenten, amelyeket számláló segítségével feldolgozhatunk. A [5] laboratóriumi gyakorlatban két mérési technikát említ meg: impulzusok számolása nagy fordulatszámokra javasolja, valamint az időzítéses ahol a két impulzus közti időt méri meg.

## Szögsebesség mérése FPGA segítségével

A sebesség szabályzásához mérni kell a sebességet, [6] dolgozatban két sebességmérő módszert említ meg, az első az inkrementális adó segítségével mért időalapú sebességmérés, amely abból áll, hogy: méri a két impulzus között eltelt időt egy számláló segítségével, amely az FPGA órajelére számol. A sebességet a következő összefüggéssel kapja meg:

Ahol a FPGA órajele Hz ben kifejezve, az inkremtális tárcsa felbontása, megszámolt órajelek a két impulzus között, egy szorzó (1,2,4). A módszer hátránya az, hogy minél nagyobb a fordulatszám a kvantálási hiba is nő a következő összefüggés szerint:

A másik módszer ahol megszámolja, az inkrementális adatol érkező éleket, idő alatt. Ahol a másodpercben, kifejezett idő.

A második módszer előnye hogy minél nagyobb a fordulat annál kisebb a hiba. A két módszert ötvözve használja.

Meglátásom szerint a két módszer bonyolulttá tenné a rendszert, mivel a számítások túl sok erőforrásokat igényelne, amelyekkel nem lehetne megoldani 8 motor szabályzó körét az FPGA rendszeren. Az általam alkalmazott módszer az hogy a diszkrét PID szabályzónak az előirt bemente egy mintavételben beérkező impulzusok száma lenne, amelyet könnyebben meg lehetne mérni mint az időt vagy a frekvenciát.

## Egyenáramú szervo motorok

Az egyenáramú motorokat használják általában nagy pontosságot igénylő hajtások megvalósítására, kisebb teljesít menyű motorok permanens mágnesből készült állorészel rendelkeznek, és tekercselt forgorészzel. A szervo motorok fő jellemzőik a gyorsaságuk, kicsi az elektromos és a mechanikai időállandójuk. A [7] alapján a rotort egy sorba kötött L induktivitással és egy R ellenállással modellezi. Ahol a rotoron átfolyó áram. A Biot Sav art és a Lenz törvények alapján: , . Ahol a c1 és c2 konstansok.

Az elektromos egyenlet mellé még felírja a mechanikai egyenleteket is:

– rotor inerciája, - motorban fellépő súrlódási erők, - külső nyomaték.

A motor dinamikus moteljéhez egyesíti a két egyenletet:

Ahol az a motor szög pizicíója.

A motor állapoteres moteljét is levezeti az általa választott állapotok:

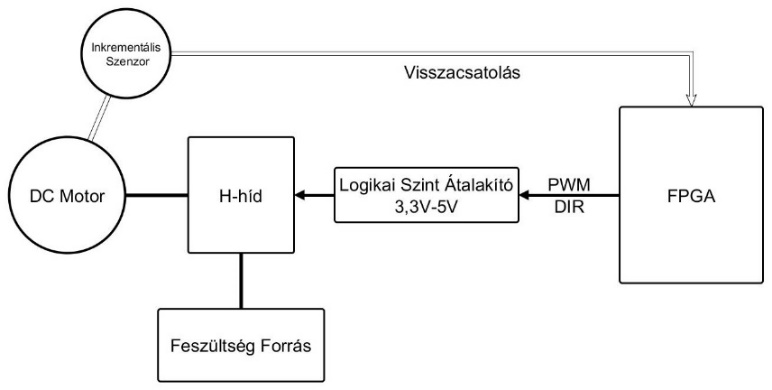
Ahol - állapotok, és a - bemenetek.

Mivel az általam használt rendszerben nem mérem a motorokon átfolyó áram nagyságát, ezért, és kültéri terepen tartható pontosság is elég kicsi ezért a modellt nem használom fel a rendszeremben.

# A RENDSZER TERVEZÉSE

## **SZABÁLYOZÓK:**

A DC motor és a hozzá csatolt mechanizmus pozíció és a sebesség szabályzására az egyhurkú kialakítást választottam Kép. 3.1. A feszültségben vezérelt Dc motorok beavatkozó jelét PWM generátorral állítom elő. A Visszacsatolást inkrementális érzékelő segítségével valósítottam meg.



Kép. 3.1 a Pozíció és a sebesség szabályzási hurok elvi strukturális felépítése

### **Diszkrét Hardveres PID szabályozó**

Napjainkban az egyik leghasználtabb szabályozótípus a PID, amely az átviteli függvénye a következő:

Az általam elkészített PID szabályozó hardveresen van megvalósítva FPGA áramkörben, a minél kisebb mintavételezési periódus elérése céljából. A fent látható összefüggések alapján egy adat utas automatát terveztem, amelyet majd System Generatorban építtettem meg. A PID szabályozó paramétereit, a Q paraméterek segítségével adhatjuk meg, amelyek függenek az ismert paraméterektől: - deriválási idő, -integrálási idő, mintavételezési periódus, valamint proporcionális erősítés. Az automata öt állapotot tartalmaz. Minden mintavételre, az automata végigpörög az állapotokon és majd visszatér a kiinduló állapotba. Az állapotokban végzet műveletet az FPGA fejlesztő lap órajelének a frekvenciájára hajtjuk végre, minden állapoton egy órajel periódus alatt lép át.

Minden állapotban egy (ÖSSZEGZŐ) regiszterhez adjuk, hozzá a műveletek eredményét és így valósul meg a fenti rekurzív összefüggés.

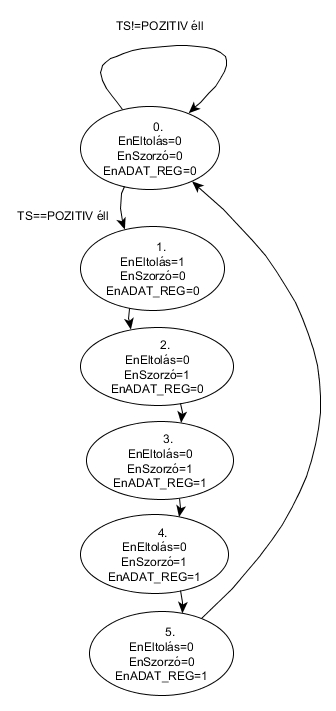
Az automata mindaddig 0. állapotban van, amíg a TS szignálon nem érkezik egy felfutó él. A 1. állapotban végrehajtja az regiszterek eltolását, azáltal hogy jelet logikai 1 re állítja, vagyis , és regiszterbe betölti az aktuális bemeneti értéket.

Az 2,3,4 állapotokban matematikai műveleteket végez, azáltal hogy EnSzorzó jellel a SZORZÓ modult aktívája. A szorzás elvégzésére egy órajel periódust vesz igénybe, és a következő periódusban használhatjuk csak az eredményt.

Az állapotokban végzet műveletek:

* 0. állapotban várakozika felfutó él érkezésére
* 1. állapotban elcsúsztatja az e regiszterek értékét
* 2. állapotban elindítja a, műveletet.

3. állapotban eltárolja a 2. állapotban elindított művelet eredményét és elindítja , műveletet .



Kép. 3.2 Állapot automata, amely leírja a Diszkrét PID szabály ózót

* 4. állapotban eltárolja a 3. állapotban elindított művelet eredményét és elindítja műveletet.
* 5. állapotban eltárolja a 4. állapotban elindított művelet eredményét.

#### Megvalósítás System Generátorban

Az adatút kiválasztására egy 2bit-es számlálót alkalmazunk (ADAT UT) amely, órajelre számol, ha az enable (en) bemenetén logikai 1 érték található, 2biten a számláló maximum 4 értéket vehet, ezért a számlálót úgy állítjuk, be hogy a maximális értéke 2 lehessen így 0,1,2 értékeket veheti fel. Az adat utakat két 16bites multiplexerrel MUXQ és MUXE válaszuk ki.

Bemeneti paraméterek a 16bit előjeles egész értékek, 16bit előjeles egész érték,-bool típusú.

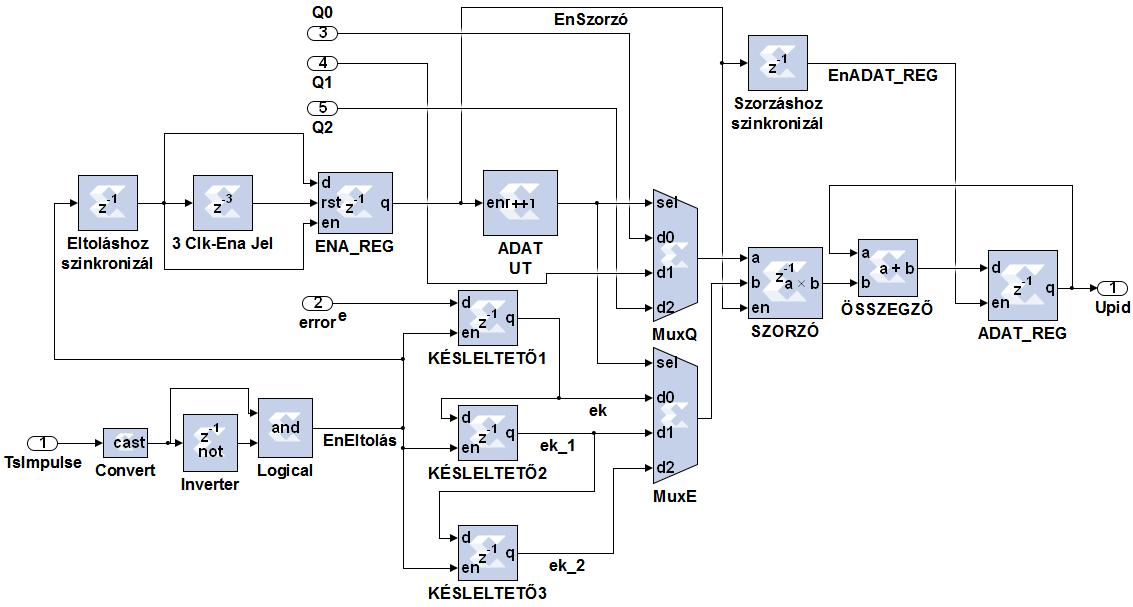
Kimenetek: 17bit előjeles egész érték. A MUXQ a Q paraméterek kiválasztásáért felelős, valamint a MUXE az időben késleltetett bemeneti értékek kiválasztásáért felelős. A Kép. 3.3 a „*SZORZÓ*” modul a két szelekciós multiplexertől kapott értéket összeszorozza, aztán hozzáadja az „*ADAT\_REG*” regiszter értékéhez.

Minden modulértéke szaturálódik abban az esetben, ha túlcsordulna akár negatív vagy pozitív irányba, így elkerülhetjük azt is, hogy az integráló tag változatlan hiba bemenete esetén túlcsorduljon és felborítaná a rendszer működését.

A Kép. 3.3 látható KÉSLELTETŐ regiszterek állítják elő múltbeli hiba értékeit, úgy hogy a három regiszter egymás után van láncolva és a felfutó élére a következő regiszterbe csúszik át az érték. A KÉSLELTETŐ1 regiszterbe kerül mindig az aktuális mintavételezett hiba értéke.

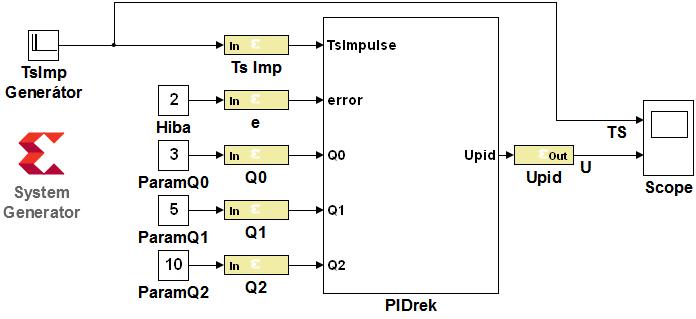
Az „*ADAT UT”* számláló csak a regiszterek elcsúsztatása után indul el, amelyet „*Eltoláshoz Szinkronizál*” késleltető elem valósít meg. Az „ENA\_REG” és a „3 Clk-Ena Jel” egyűt valósítják meg a három órajelig tartó logikai engedélyező jelet, amely a számlálót indítja el.

Kép. 3.3 A PID felépítése System Generatorban



Az él detektáló elemet egy tagadó kapu „*Inverter*” valamint egy „és” kapu (Logical) biztosítják, mégpedig úgy, hogy figyeljük egyazon jel előbbi periódusban az értéket és összehasonlítva a két értéket tudjuk detektálni a jel váltózását. A „*Szorzáshoz Szinkronizál*” modul segítségével tudjuk engedélyezni az „*ADAT\_REG*” bemenetét a „*Szorzó*” modul „*EnSzorzó*” engedélyező jeléhez képest késleltetve 1 órajelet, erre azért van szükség, mert a szorzás eredménye 1 órajelet késik az elindítást követően és az eredményt szeretnénk eltárolni.

#### Szoftveres Szimulációs eredmények



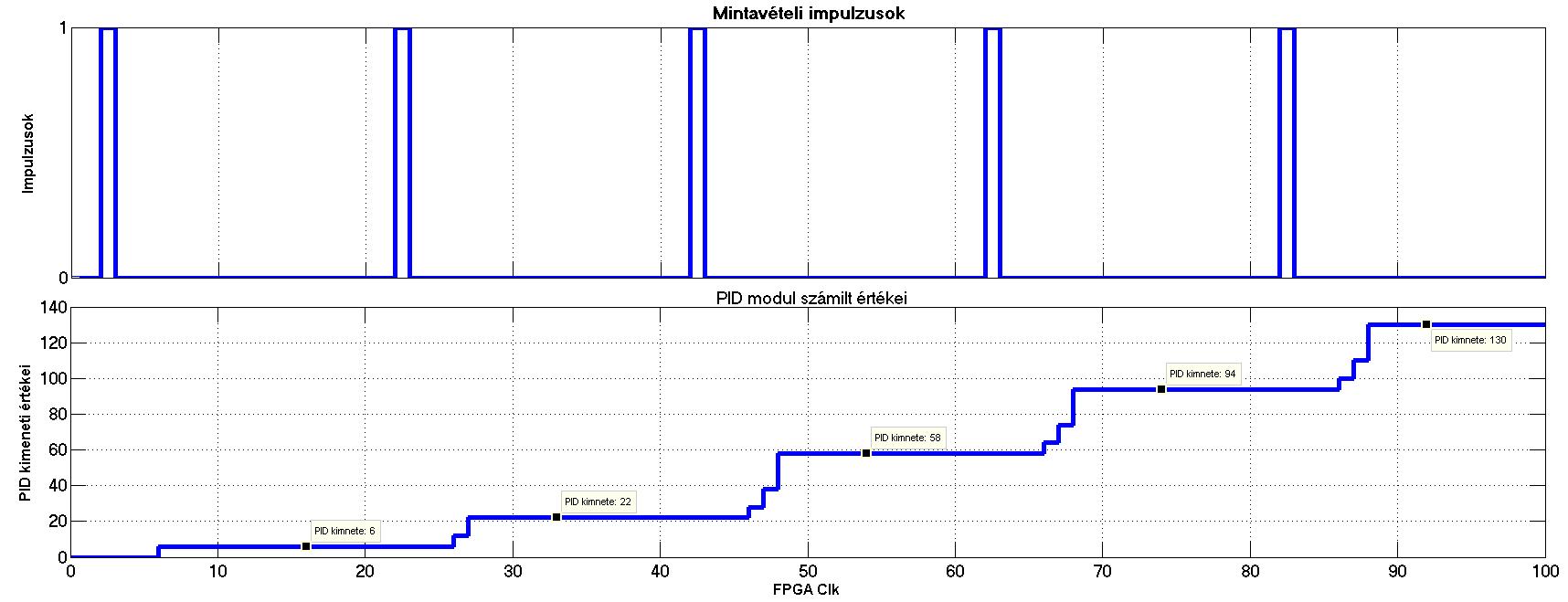
Kép. 3.4 PID szoftveres szimulációs model

A Kép. 3.4 a „PIDrek” modul tartalmazza a Kép. 3.3 képen látható modult, a bemenetekre és a kimentre illesztünk egy-egy konvertáló elemet mely segítségével adatokat közölhetünk, vagy nyerhetünk a megtervezett Xilinx Sytem Generátoros hardverrel.

A szimulációk során a számítások eredményét ellenőriztem le, amelyeket SYSTEM GENERATOR-ban végeztem el Simulink segítségével, az eredményeket majd összehasonlítottam a manuálisan számolt értékekkel (Táblázat. 3‑1).

Bemeneti paraméterek: , a bemenet konstans:

A Kép. 3.6 látható a szükséges 6 órajel a számítások elvégzésére. Megjegyzésként a szabályozó negatív bemeneti értékekre is működőképes.



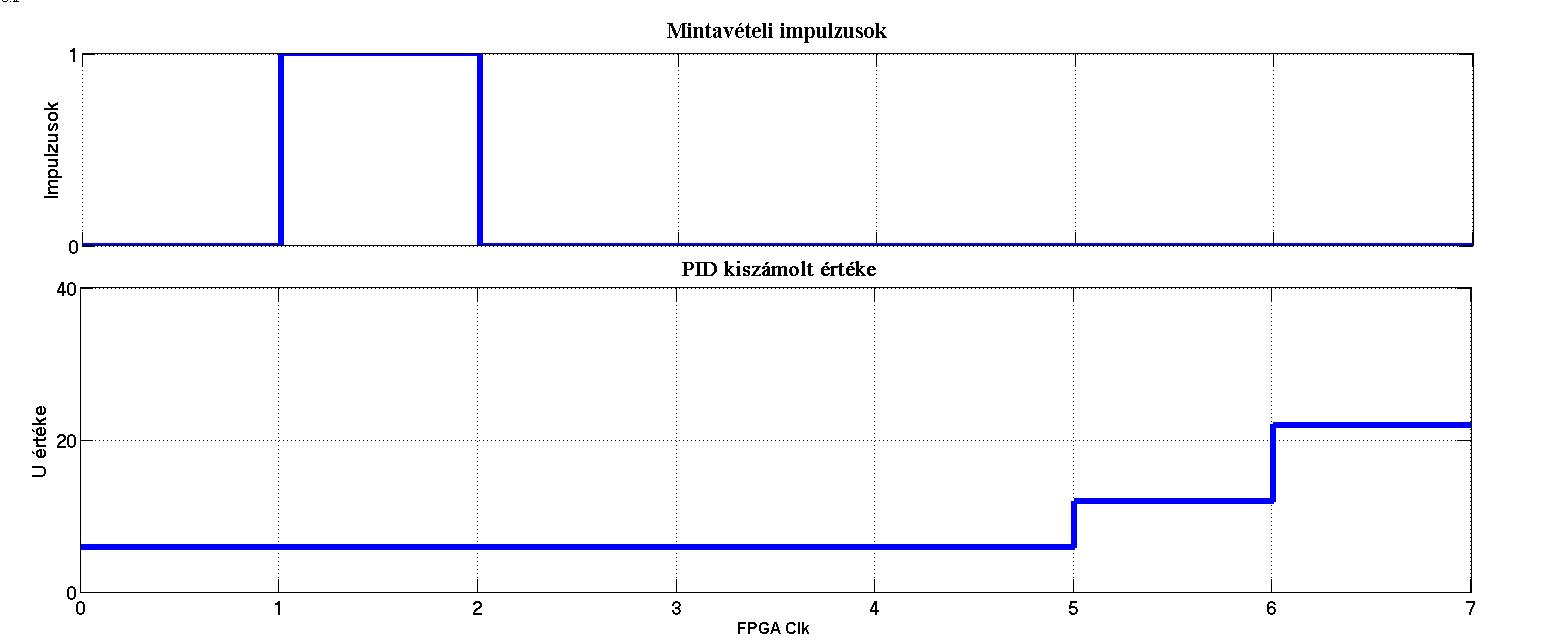
Kép. 3.5 Szimulációs eredmény amely tükrözi a konstans bementre a számolási lépéseket

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Órajel |  |  |  |  |  |  |  |
| 1 | 3 | 5 | 10 | 2 | 0 | 0 | **6** |
| 2 | 3 | 5 | 10 | 2 | 2 | 0 | **22** |
| 3 | 3 | 5 | 10 | 2 | 2 | 2 | **58** |
| 4 | 3 | 5 | 10 | 2 | 2 | 2 | **94** |
| 5 | 3 | 5 | 10 | 2 | 2 | 2 | **130** |
| 6 | 3 | 5 | 10 | 2 | 2 | 2 | **166** |

Táblázat. 3‑1 Manuálisan számolt értékek a szimuláció ellenőrzése kedvéért

A Szabályozó az elvártaknak megfelelő eredményeket térít vissza. A szimulációkból többek között megfigyelhető, hogy az elindítástól a végső eredmény megjelenéséig 6 órajelre van szükség. Ismerve a rendszer órajelét ki tudjuk számolni a szükséges időt ami, kell a számítások elvégzésére. Az órajel jelen esetben 50MHz, amiből következik, hogy egy periódus 20ns –ig. tart, és így a szükséges idő . Következés képpen a PID szabályozó maximális mintavételi periodusa 120ns

Kép. 3.6 PID minimális periódusa



#### Q paraméterek számolása alapján.

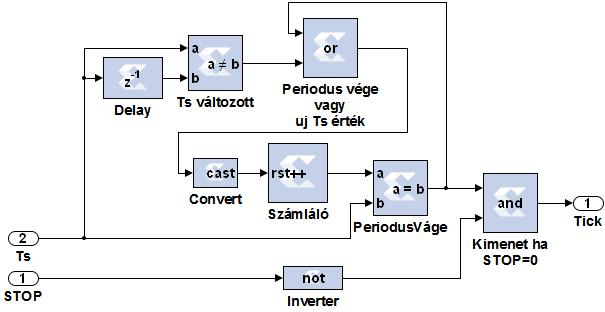
Adottak az összefüggések, látható hogy megjelenek a nevezőben így fennáll annak a veszélye, hogy 0-val való osztás történik. Ezért soha ne válaszuk a –t 0-nak, de lehet egy nullához közeli pozitív szám. A mindig nagyobb, mint nulla.

A paraméterek értékének beállításával ki tudjuk választani a szabályozó típusát is PI szabályozó esetén a paramétert válaszuk 0-nak, mert nem okoz számítási problémát. PD szabályozó esetén célszerű a -nek minél kisseb értéket beállítani, ami nem lehet egyenlő 0-val.

### Mintavételi periódus jelének A generálása

Az Kép. 3.7 Mintavételi taktust generáló modul belső felépítése. A modulban a „*Számláló*” 32bites számláló az FPGA órajelére számol, és az értékét összehasonlítjuk a „*Ts*” bemenet értékével. Ha az érték megegyezik, akkor generálódik egy impulzus, amely lenullázza a számlálót. A számláló akkor is nullázódik, ha megváltozott a „*Ts*” értéke, amelyet a „*Delay*” késleltető és a „*Ts* *változott*” egyenlőséget tesztelő modul valósít meg úgy, hogy összehasonlítja az előző órajel periódusban eltárolt értékével. Ha a két érték különbözik, akkor reset állapotba hozzuk a „Számláló” modult. A „*Periódus vége*” összehasonlító modul abban a pillanatban, amikor a számláló elérte a „*Ts*” bemenet értékét reset állapotba hozza a számlálót.

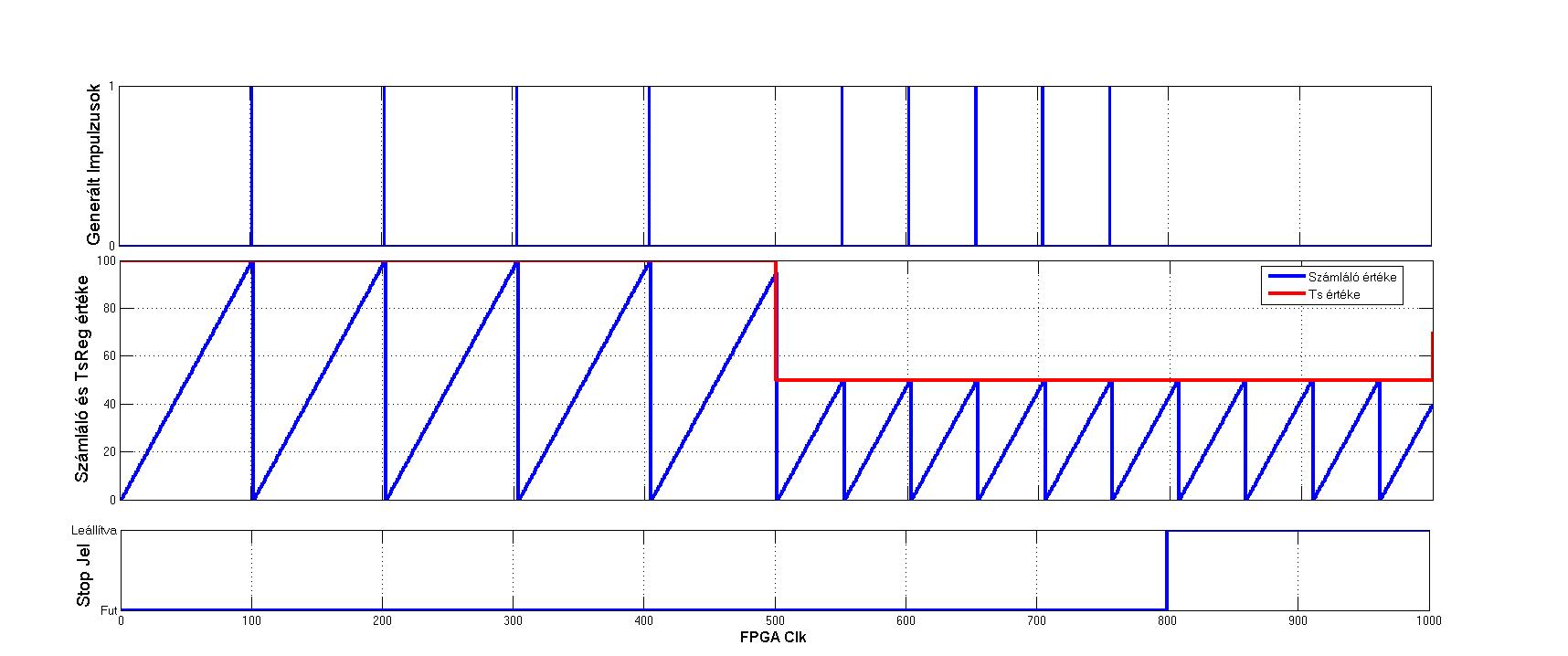
Kép. 3.7 Mintavételi taktust generáló modul



Az Kép. 3.8 látható, ahogy a pirossal jelölt „*Ts*” két különböző értékére hogyan történik az impulzusok generálása. A „Tick” kimenten az impulzusok 1 órajel periódusig tartanak.

Az impulzusokat a STOP bementen keresztül letilthatjuk, ha logikai 1 értéket adunk rá.

Impulzusok frekvenciája: Hz ben.



Kép. 3.8 Szimulációs eredmények mintavételi jelgenerátor.

### Pozíció Szabályzása

A mechanikai rendszer kialakításából adódóan, ha a hajtó motor leáll és a hajtott tengely terhelés alatt marad, a hajtott tengely a súrlódások miatt nem tud visszafele hajtani, ezért elegendő, ha a megfelelő időpillanatban a hajtó motort leálltjuk. Mivel a Dc motor polaritás váltásakor a motor forgási iránya is megváltozik elegendő, ha a maximális vagy minimális szabályozó jellel avatkozunk be a rendszerbe.

Az elkészített szabályozót a következő egyenletek írják le:

Elmondható a kimeneti szabályozó jel függ a hiba értékétől. A mechanikai rendszerben kotyogás van, és az ebből származó zajokat szeretnénk kiszűrni úgy, hogy ha a mechanizmus a megfelelő pozícióban van, akkor egy tartományban a szabályozót érzéketlené tesszük a bemenetre mindaddig, amíg a hiba ki nem lép a sávból vagy a referencia jel meg nem változik.

#### A szabályozó felépítése:

A bementek: „RefVal” –előírt pozíció impulzusban mérve, 16 bites előjeles érték, „AktVal” – aktuálisan mért pozíció impulzusban mérve, 16 bites előjeles, „U” –kimente 17 bites előjeles.

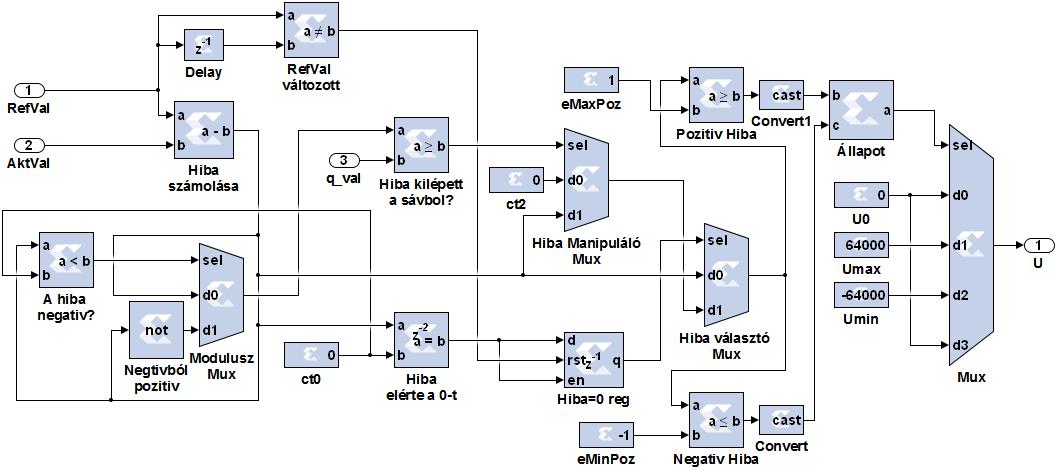
A „*Hiba Számolása*” modul végzi a hiba kiszámolását az aktuális és az előirt pozícióból. A hiba lehet negatív is ezért „*A hiba negatív?”* Komparátor segítségével eldöntjük, hogy negatív vagy pozitív a hiba, azután a „*Modulusz Mux*” segítségével kiválasztjuk a magát a számolt hibát, ha az pozitív, vagy a hiba tagadottját, ha az negatív így megközelítve a moduluszát.

Ha a hiba elérte a 0-t akkor a „Hiba=0 reg” értéke 1 lesz mindaddig amíg az előírt pozíció meg nem változik.

A „Hiba kilépett a sávból” modul megvizsgálja, hogy a hiba modulusza kisebb mint a sáv értéke ha igen akkor a „Hiba Manipuláló Mux” segítségével a továbbiakban a hiba 0 lesz.

A „Hiba választó Mux” a „Hiba=0 reg” irányítására válasza ki manipulált hibát vagy számolt hibát, amelye továbbmegy a háromállású szabályzóba.

Az „Umin”, „Umax”, „U0” 17 bites előjeles regiszterek segítségével kiválaszthatjuk a szabályzó maximális és minimális beavatkozó jelének értékét. A „Pozitív Hiba” „Negatív Hiba” eldöntik, hogy a hiba mely tartományba van. Három tartományt különböztetünk meg: negatív pozitív, és 0 hibát. A „Mux” kiválassza az aktuális állapotnak megfelelő vezérlő jelet.



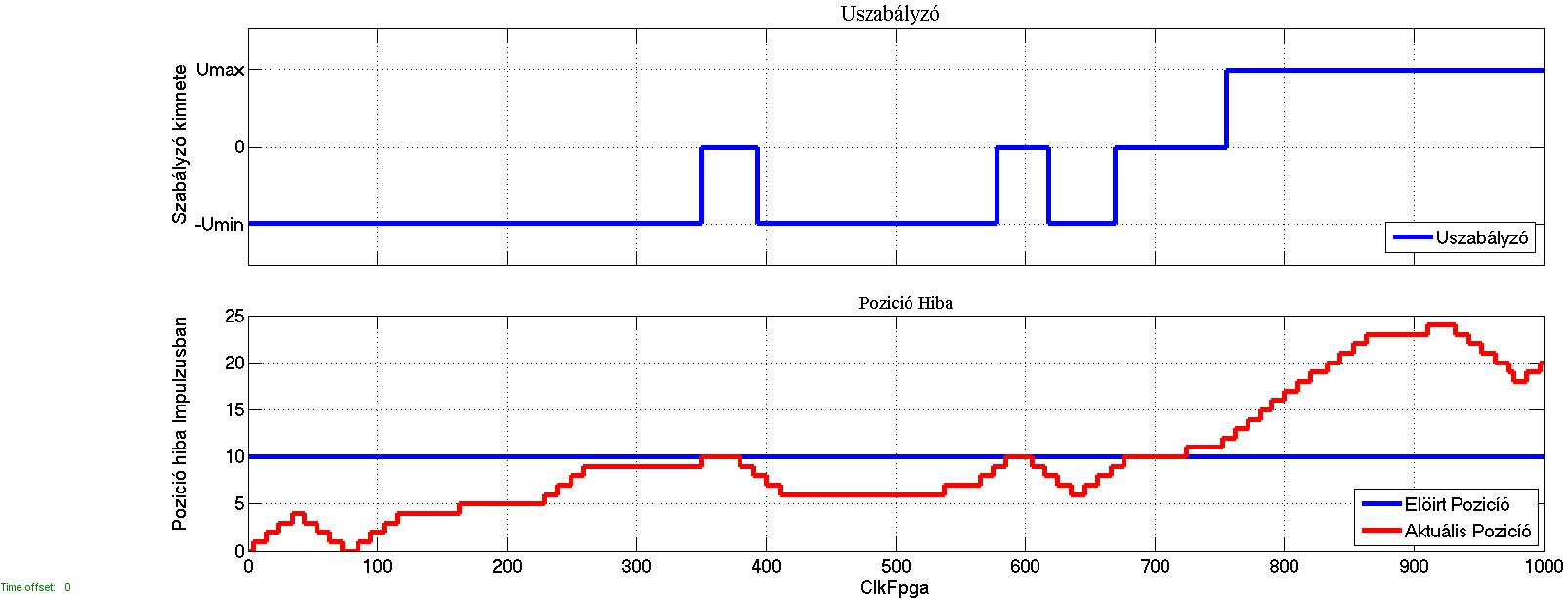
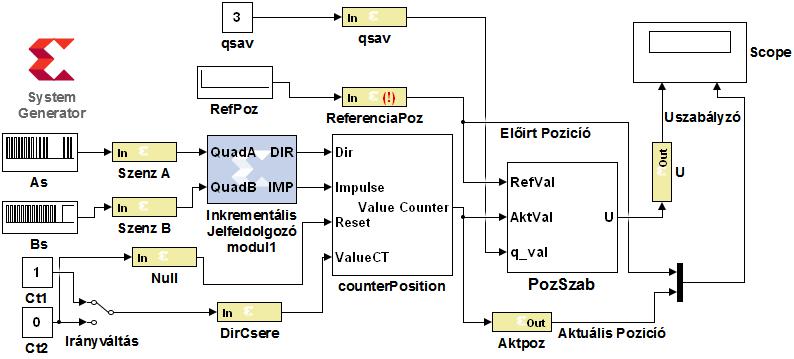
Kép. 3.9 A Pozíció szabályozó System generátoros felépítése

#### Szabályozó szimulálása

Az Kép. 3.10 látható a szimulációs logika, a „counterPosition” és a „Inkrementális Jelfeldolgozó modul 1” megtalálható a pozíció mérése inkrementális adó segítségével. A szabályozót a „PozSzab” modul tartalmazza, belső felépítése a Kép. 3.9 alapján.

A Kép. 3.11 megfigyelhető hogy a kimeneti jel miként változik a hiba függvényében. Látható, ha a hiba 0 környékén van a szabályozó kimenete 0 lesz, és csak akkor mozdul ki, amikor a hiba kilép a sávból. A referencia jel a szimuláció során konstans értékű, ezért a nem idézheti elő az érzéketlenségi sávból való kilépést.

Kép. 3.10 a pozíció szabályzás moduláris felépítése System Genrator környezetben



Kép. 3.11 a pozíció szabályozó bemenete (pozíció hiba), valamint a Szabályozó kimenti jele.

### Hardveres mérések

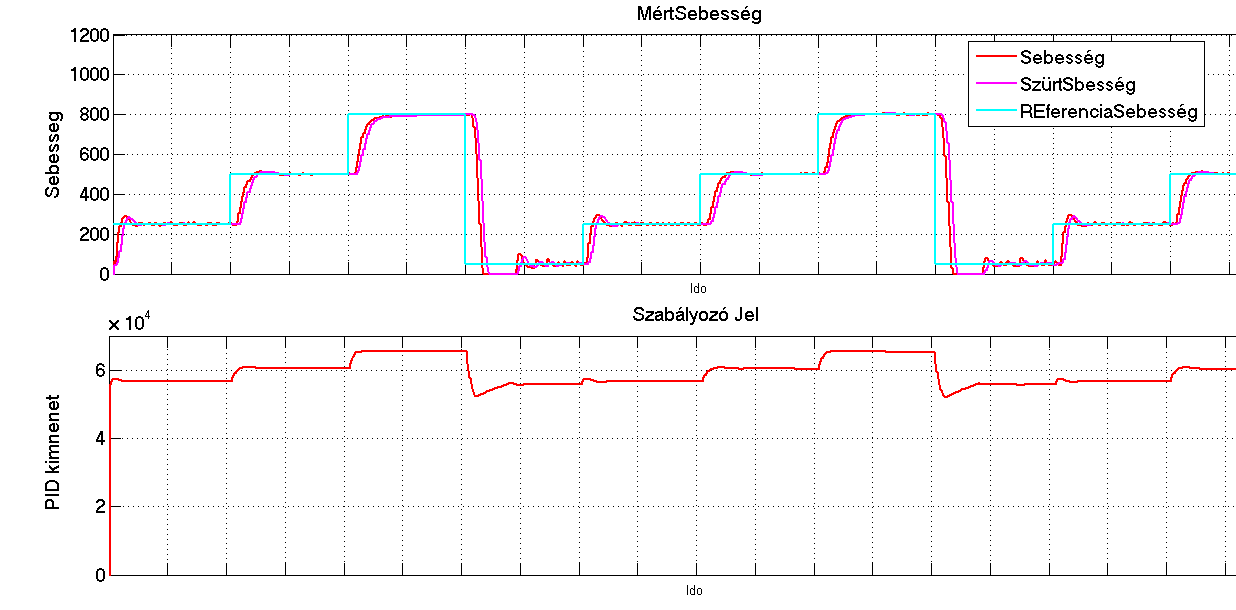
#### Dc motor sebesség szabályzása mérőstandon

A sebesség szabályzáshoz szükséges érzékelőt az általam megvalósított inkrementális jeladó segítségével oldjuk meg. A motor egy H hídba van kötve négy N csatornás MOSFET tranzisztor segítségével. A híd bemenetei 5V logikai szintű PWM (kitöltési tényezője maximálisan 99% lehet) és DIR (irány, segítségével megadhatjuk a motor forgási irányát.)

Az FPGA I/O kivezetései 3,3V logikai szinten vannak, ezért kell egy szintillesztést végeznünk 3,3V ról 5V-ra.

A szint illesztés csak egyirányú, az FPGA-tól kimeneti irányba. A motor tengelyére vagy a mozgatott mechanizmusra rögzített inkrementális tárcsa segítségével tudjuk mérni az elfordulást. A motor sebességét -ben vagyis impulzus per mintavételben mérjük, így a referencia sebességet is ebben a mértékegységben kell megadnunk, ezért átalakítást kell, végezünk a következő összefüggés szerint.

A fenti képletben mindig csak egy ismeretlen van attól függően, hogy referencia értéket kel számolnunk akkor az paramétert fejezzük ki, vagy az paramétert.

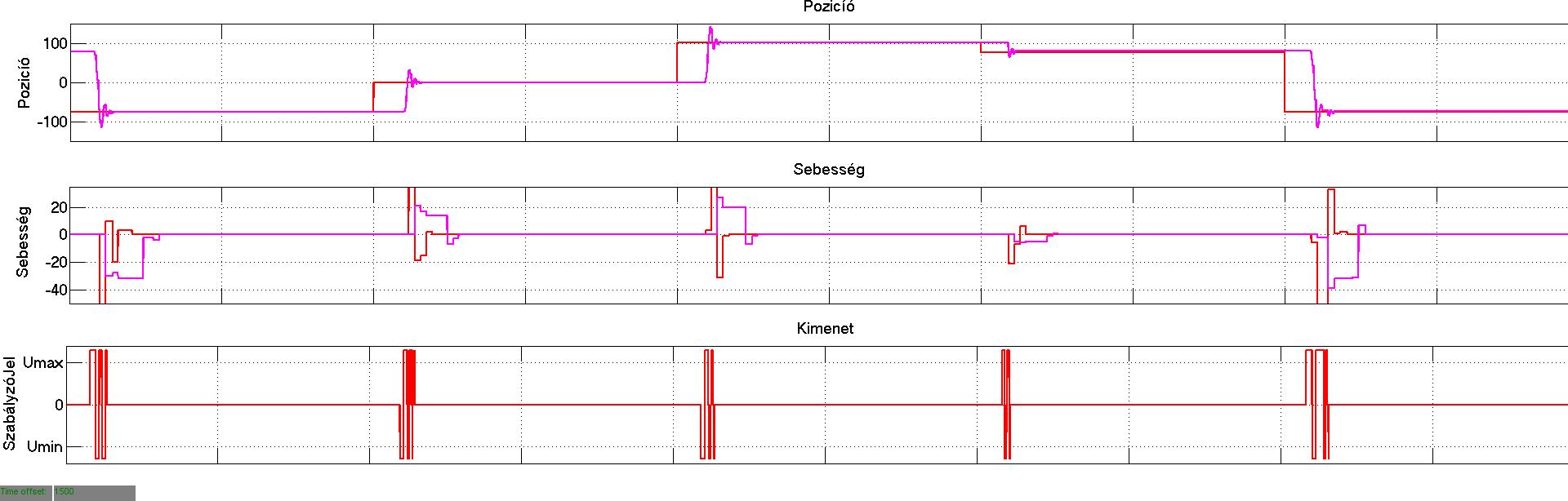


Kép. 3.12 Sebesség szabályozás PID szabályzóval.

A motor sebessége követi az előirt értéket, ugyanakkor megfigyelhető, hogy az alacsony sebességnél a mérések nagyon zajosak, mert a mérésre csak impulzus számolást alkalmaztam, a továbbfejlesztési lehetőségként kikell egészíteni a [6] cikkben bemutatott időmérési módesszerrel összekevert hibriddel.

#### Dc motor pozíció szabályzása mérőstandon

Mivel a mérőstand nem rendelkezik, csiga áttelel ezért a szabályozó enyhén oszcillál, de ez majd az áttétel jelenlétében nem fog fennállni.



Kép. 3.13 Pozíció szabályzása a Kép. 3.9 látható kialakításban.

## Sebesség és pozíció szabályozót tartalmazó IP mag generálása System Generator-ban

Az eddig megépítet, modulokat összekötve alkotunk egy nagy modult, amely tartalmazza a sebesség és a pozíció szabályzási hurkokhoz szükséges elemeket.

Bementek: „*SpeedA*”, „*SpeedB*”, „*PositionA*”, „*PositionB*” inkrementális érzékelőktől érkező 1bites fizikai jelek. „null”-pozicíó mérésénél használt referencia pozíció.

Kimenetek: „*SpeedPWM*”, „*SpeedDir*”, „*PosPWM*”, „*PosDir*” modul kimnő 1 bites fizikai jelek.

A „Sebesség Szab” modult tarlatmazza az látható modulokat és a sebesség szabályzására hivatott, míg a „Pozicíó Szab” tartalmazza a dfdsf látható modulokat és a pozíció szabályzózására hivatott.

A szabályozók referencia bemeneteit osztott regiszterekkel írjuk elő microBaze processzor segítségével. Mindkét szabályozó tartalmazza a PWM generátort, és a sebesség mérő modult (counterSebesseg). A két inkrementális érzékelő jeleit egyetlen modul segítségével dolgoztam fel „Inkrementális Jelfeldolgozó 2”, amely segítségével feldolgozunk két inkrementális érzékelőtől beérkező jeleket.

A „Config” osztott regiszter bitjeivel be vagy kikapcsolhatunk funkciókat.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 |
| Pozíció PWM engedélyezése | Null sáv engedélyezése | Sebesség PWM engedélyezése | Sebességek csatolása | Pozíció mérés irányváltás | Sebesség mérés irányváltás |

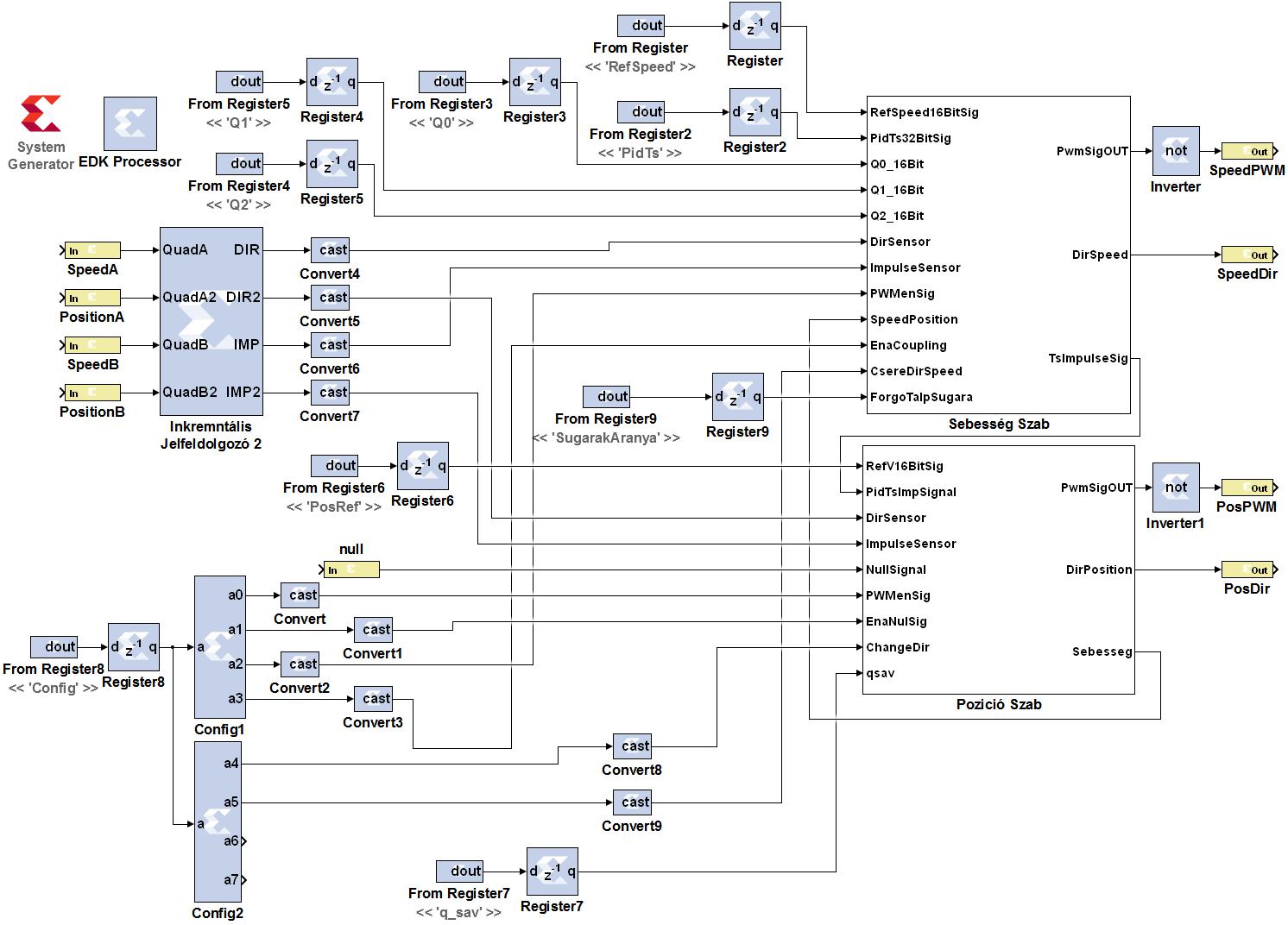
Tábla. 1 Konfig regiszter funkciói

A robot forgó talpának a pozíció változása a sebességet generál a kis keréknél, ezért megváltozik a lánctalp sebessége, a talajhoz viszonyítva. A sebesség szabályozó referencia bemenetéthez hozzá kell adni a pozíció változását megszorozva egy arányosáig tényezővel.

A 6.3 képen látható az 1 forgó talp sebességet generál az kör mentén, a robot lánctalpának az aktuális sebessége

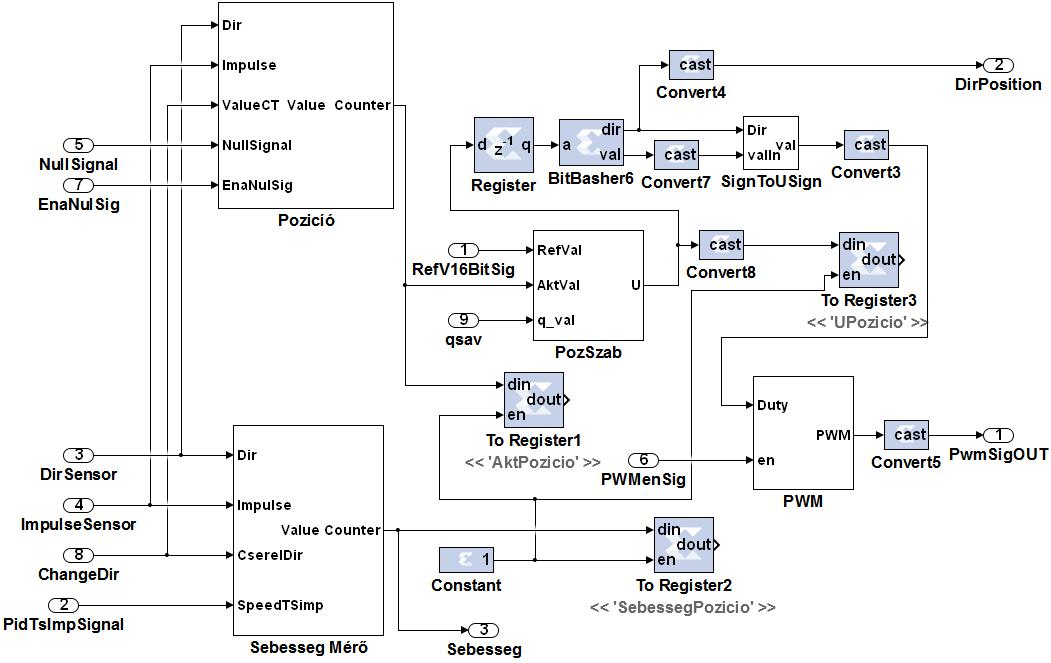
A sebeségek összeadását 1.10 képen látható Mult, Mux, AddSub1 végzik, az összeadást ki vagy be kapcsolhatjuk a Config regiszter negyedik bitjével.

A ToRegister típusú memóriák csak írhatók, míg a FromRegister típusúak csak olvashatók.



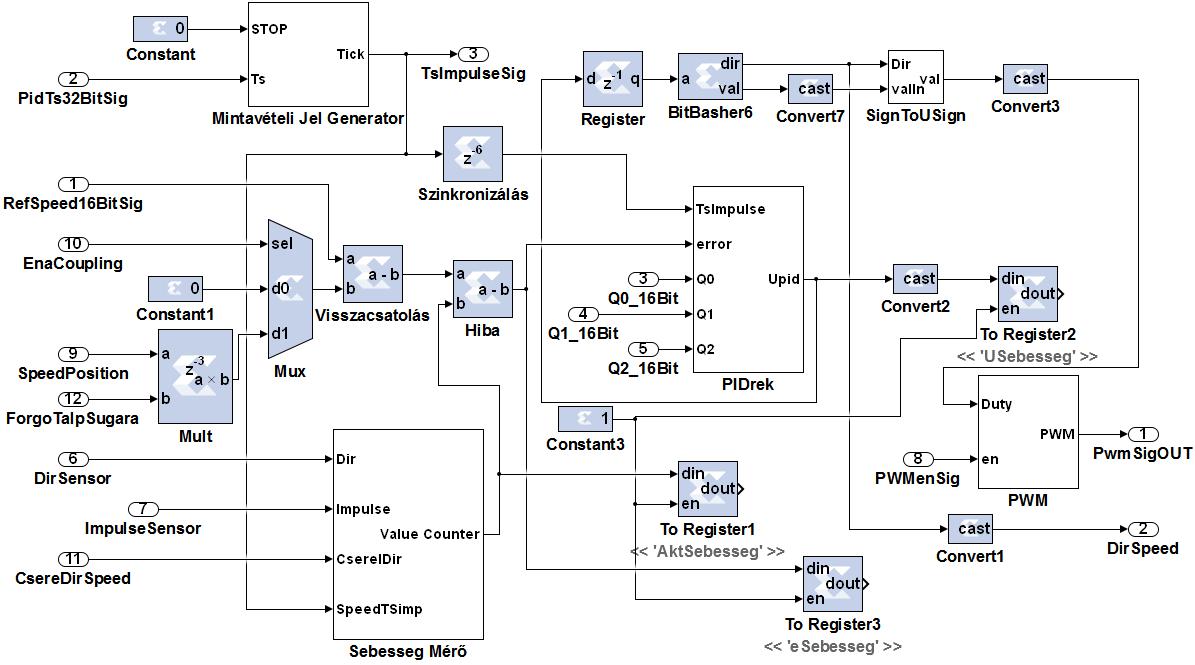
Kép. 3.14 A két szabályozót tartalmazó modul System generatoros felepitesben

A „*Pozíció*” nevű modul tartalmazza a Kép. 3.24 képen látható modulokat feladata a relatív pozíció mérése impulzusban, kimente egy 16 bites előjeles szám. A „*Sebesség Mérő*” nevű modul tartalmazza az Kép. 3.25 látható modulokat feladata a sebesség mérése impulzus per mintavételben, a kimenete egy 16 bites előjeles szám. A „*Pozíció Szab*” nevű modul tartalmazza a Kép. 3.9 látható modulokat, feladata a pozíció szabályozása. A „*PWM*” nevű modul tartalmazza Kép. 3.31 látható modulokat, feladata a pwm jel előállítása. A „*BitBasher6*”, és a „*SignToUsign*” nevű modulok átalakítják a szabályozótól érkező 17 bites előjeles számot egy 16 bites előjel nélküli számmá és egy 1 bites jellé, amely tartalmazza az a17 bites szám előjelét. A mért paraméterek osztott regiszterekbe kerülnek: „*UPozicio*”,-a beavatkozó jel, „*SebessegPozicio*”- pozíció deriváltja, „*AktPozicio*”- pozíció.



Kép. 3.15 Pozíció Szab modul belső felépítése a Kép. 3.14

A „Sebesség Szab” modulnak a felépítése hasonló a pozíció szabályzóéra, annyi eltérés van hogy itt nem jelenik meg pozíció mérés csak sebesség, és meg megjelenik a„Mintavételi Jel Generátor” amely tartalmazza a Kép. 3.7 látható elemeket és a feladata hogy biztosítja a mintavételi periódust a sebesség mérő modulok számára és a pid szabályzónak.



Kép. 3.16 Sebesség Szab modul felépítése a Kép. 3.14

A „PIDrek” nevű modul tartalmazza a pid szabályozót a Kép. 3.3 látható kialakításban.

Abban az esetben ha változtatjuk a karok pozícióját és vele egy időben konstanson szeretnénk tartani a lánctalpak sebességét akkor össze kell adni a két sebességet a megfelelő előjellel.

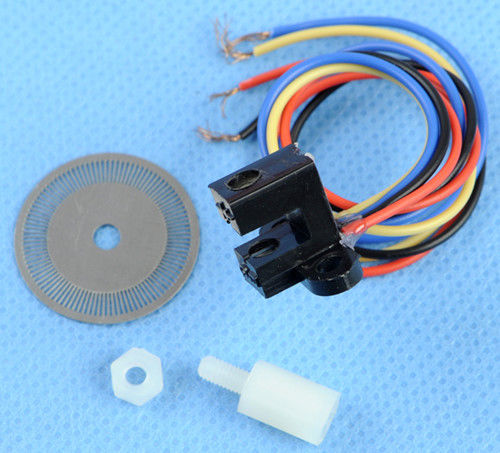
A Kép. 3.50 látható alsó ábrán a pozíció változása sebességet generál a kis keréknek, ha mi a sebességet szeretnénk tartani, akkor a következő összefüggést kell, előírjuk a sebesség szabályzónak ahhoz, hogy ne változón meg a sebesség:

, ahol a és cm, mérve.

## Szenzorok

### Inkrementális ÉrZékelő

#### Optikai inkrementális vevő felépítése

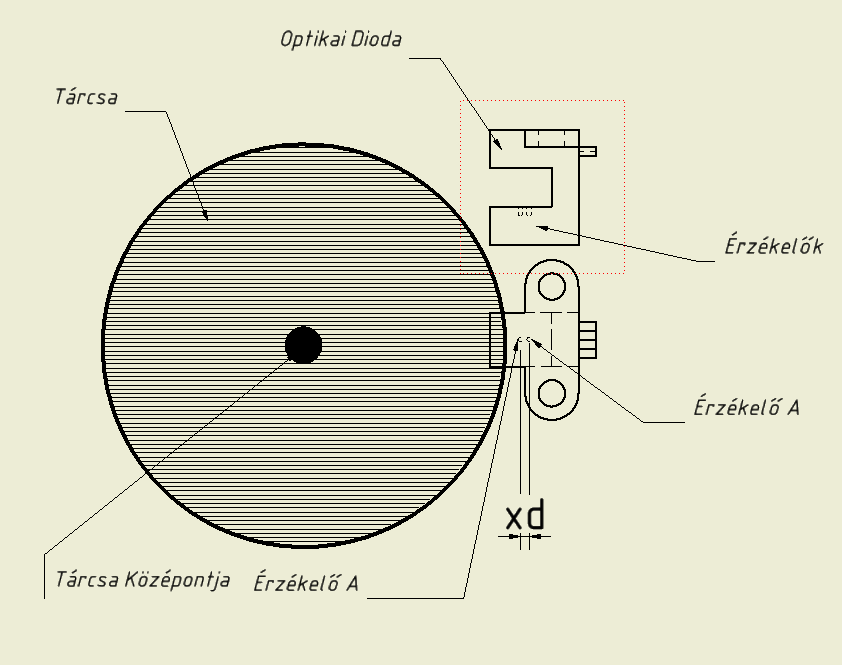
 Az optikai érzékelő két részből áll, egy optikai forrásból, és egy vevő részből, amelyen két optikai kapcsoló eszközt tartalmaz egymástól távolságra.

Két vezeték segítségével táplálhatjuk be a piros (3,3V-5V), fekete (GND), a sárga és a kék vezetékek, azok kimenti jelek az érzékelőtől.

A sárga vezetéken érkező jeleket nevezzük el A jelnek, míg a kék vezetéken érkező jeleket B-nek.

Az érzékelő számára a tárcsát a 2.1 képen látható módon kell illeszteni.

Könnyen belátható hogy a tárcsán a rések mérete és dőlés szöge befolyásolja az A, B jelek időbeni eltolását. A könnyebb kivitelezés kedvéért a tárcsákat lézeres nyomtató segítségével átlátszó fóliára szeretnénk nyomtatni.

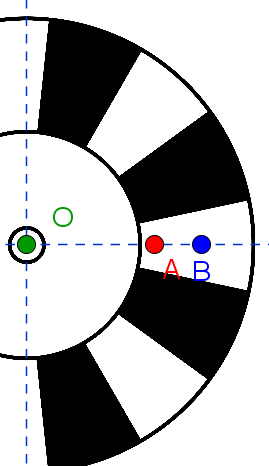


Kép. 3.17 Optikai inkrementális vevő felépítése és elhelyezése

Ha a 2.2 ábrán látható módon helyezzük el, sugár irányban nem jön létre késés a két jel között (A és B), így nem lehetne meghatározni a forgás irányát. Ezt elkerülendő a réseket meg kell dölteni egy alfa szöggel így kialakul a késés is.

Tekintsük az A és B pontokat az Érzékelő A és Érzékelő B pontjainak. Az AB szakasz hossza ismert, amely megadja az érzékelők közti távolságot.

Az pont az inkrementális tárcsa középpontja, amely körül Omega szögsebességgel forog.



Kép. 3.18 Érzékelő tranzisztorok elhelyezése

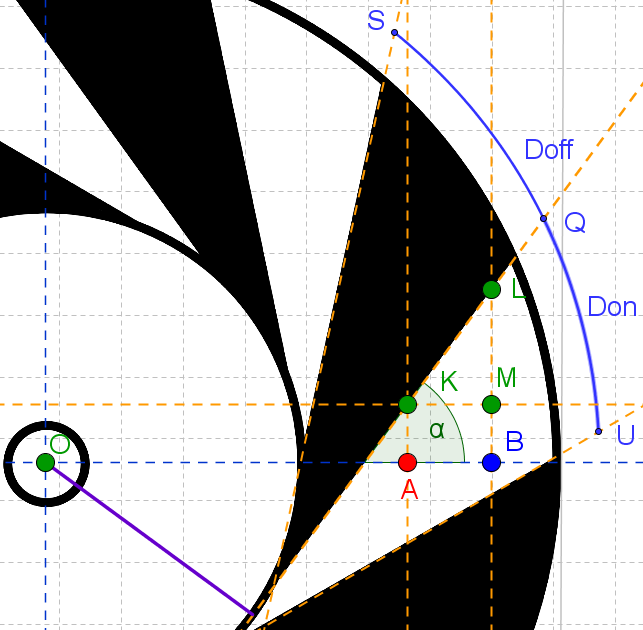
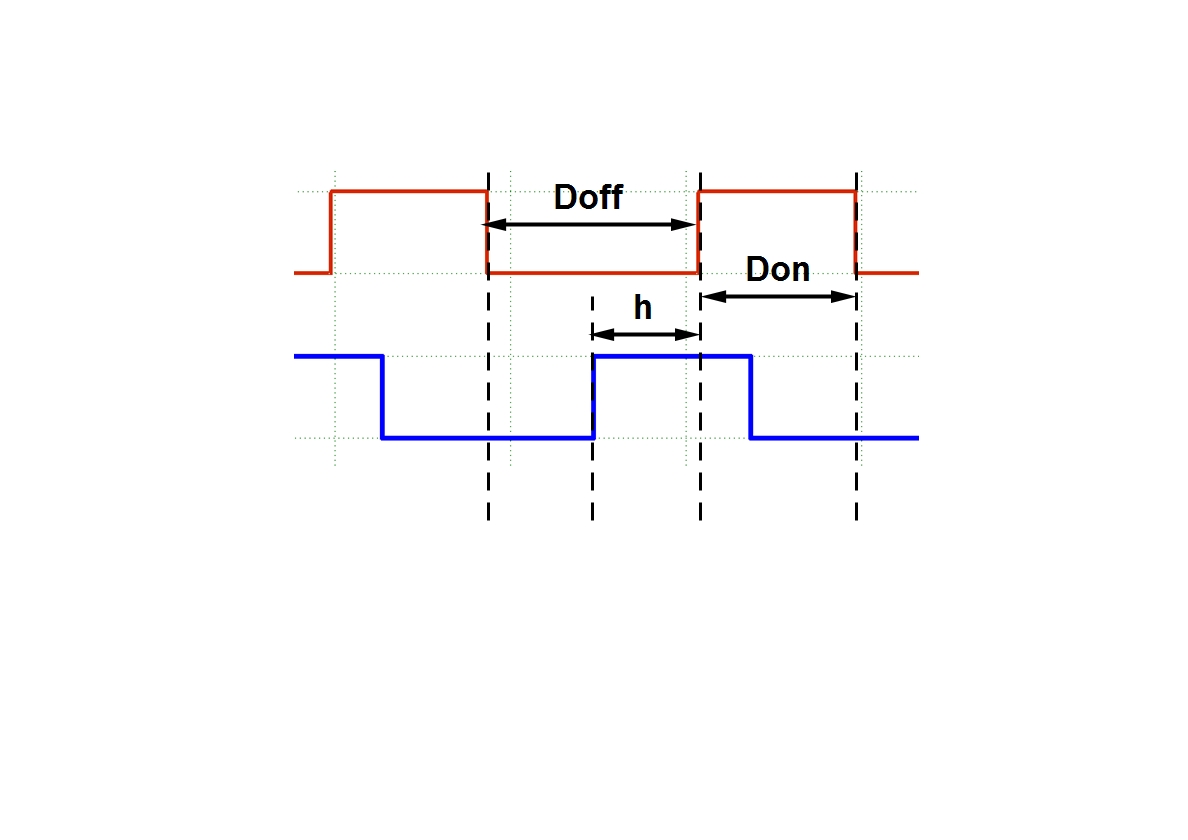
Az 2.3 képen a fehér mezők az inkrementális tárcsa réseit képviselik. A rések száma megadja, a tárcsa felbontását N.

Azokban a pontokban ahol a rések fedik az érzékelőket ott az érzékelő kimeneti jele logikai magas szinten, míg ahol nem fedik, ott logikai alacsony szinten van.

Ha az A, B, O pontok egy egyenesen találhatok (könnyebb az érzékelő felfogatása), akkor meg kel dölteni a réseket az A,B pontok által meghatározott egyeneshez képest szöggel (2.3 kép).

**Ismert adatok**:

Kép. 3.19 Idődiagram a Tárcsa paraméterei függvényében



Kép. 3.20 Rések és az Érzékelők közti kapcsolat

A magas állapot és alacsony állapot közti arány egyenesen arányos az és a szakaszok aranyával, látható a bal oldali ábrán.

Az érzékelő 2.3 képen látható módon van illesztve a tárcsához, akkor felírható az összefüggés, amely meghatározza a két jel közti késést.

,.

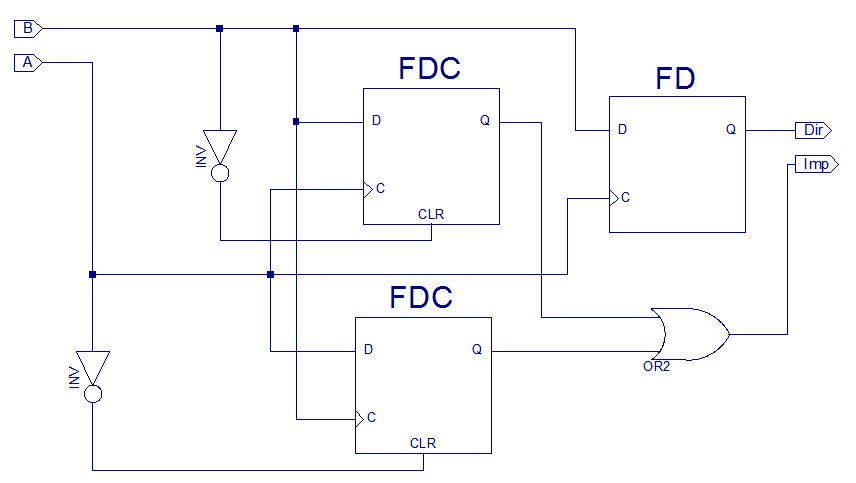
A minden esetben nagyobbnak kell lennie, mint a távolságnak.

### Inkrementális érzékelő jeleinek a feldolgozása FPGA áramkör segítségével

Az elkészített modulba bemenő jelek az A és B, amelyek az inkrementális érzékelőtől érkeznek az FPGA áramkörbe.

A modul VHDL programnyelven készült a Kép. 3.21 látható kialakítás szerint, BlackBox modul segítségével integráljuk a Simulink környezetbe (BLACK BOX1). A ki és bemeneti portókat illesztjük a Simulink környezetben található elemekhez, majd létrehozzuk a szimulációs bemeneti jelelt, amelyeket az A, B sárgával jelölt modulokon keresztül viszünk be a rendszerbe. Az FPGA áramkörben megtalálható modul segítségével a jeleket feldolgozzuk és két kimenő jelet generálunk a Dir (megadja a forgás irányát), valamint a Imp (minden ablak elhaladásakor generál egy felfutó élet).

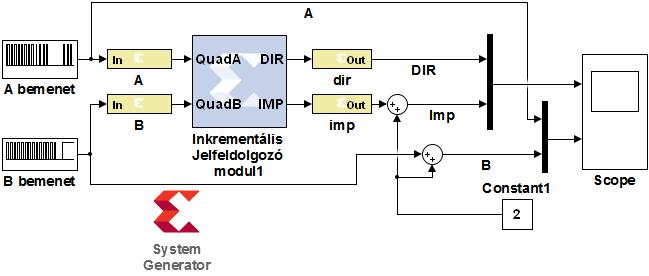
A beérkező A, és B jeleket egy tagadó kapu segítségével bekötjük a FDC tárolok CLR bemenetére, a 2.5 képen látható módon. Egy harmadik tároló segítségével meghatározhatjuk a forgás irányát. Az impulzus kimenete akkor lesz logikai 1, ha valamely FDC tároló Q kimenete is Logikai 1 lesz



Kép. 3.21 Inkrementális Jelfeldolgozó modul1 érzékelő modul belső felépítése

#### Szimuláció System Generatorban

A Kép. 3.23 látható a szimulációs eredmények az A és B bementi jelek (alsó ábra), Dirés Imp kimenti jelek(felső ábra).

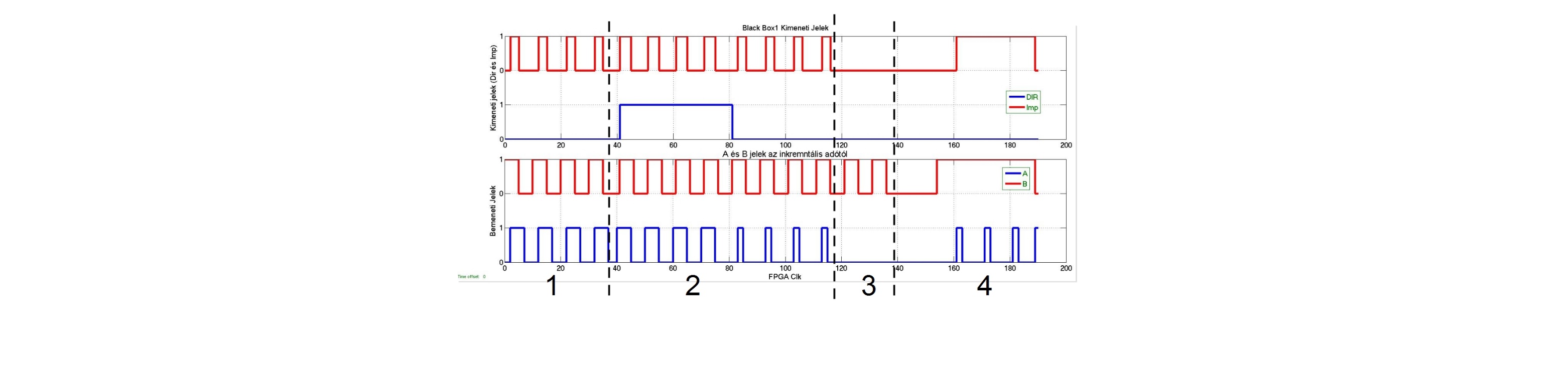


Kép. 3.22 Inkrementális érzékelőtől érkező jelek átalakító irány és impulzus jelekre

A bementi jelek négy kategoriába sorolhatók:

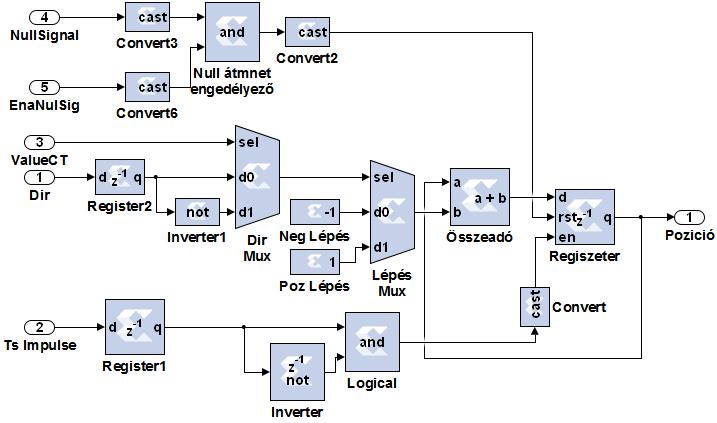
1. Az A jel késik a B jelhez képest, a kimenti jeleken látható ( érkező impulzusok és az irány).
2. Az B jel késik a A jelhez képest, látható hogy az irány megfordult.
3. Az A bemeneti jelen hibás jelek érkeznek, látható hogy ekkor nem történik impulzus generálas a kimeneten.
4. Az A bemeneti jelen ismét hibás adatok érkeznek, ez az eset akkor áll fen amikor a tárcsa forgási iránya azelőtt változik meg mielőtt elérte volna a sötét mező a B csatornát is.

Kép. 3.23 Szimulációs eredmények a lehetséges bemenetekről az Black Box1 modulba



#### Pozíció mérése Inkrementális adó segítségével

Az általam használt inkrementális tárcsák és érzékelő csak relatív pozíció mérésére alkalmasak, ezért alkalmazunk egy referencia tárcsát is. A referencia tárcsa ugyan arra a tengelyre van rögzítve, mint az inkrementális tárcsa, csak annyi különbséggel, hogy csak egy sötét foltot tartalmaz. Így egyetlen impulzust fog generálni. A pozíciót úgy tudjuk megmérni, hogy egy regiszter értékét, változtatjuk minden Imp jel felfutó élére. Növeljük vagy csökkentjük a Dir iránybit (logikai 1 vagy 0). függvényében.



Kép. 3.24 Inkrementális adóval mért pozicó, szimulálációs modelje Sytem Generatorban

Működési logikája alapján három osztályba sorolhatok. Az első az él detektáló, melynek feladata az impulzusok felfutó éleinek a detektálása, és egyetlen órajelig tartó impulzus generálása a tároló regiszternek (Regiszter) így engedélyezve az adatbevitelt a regiszterbe.

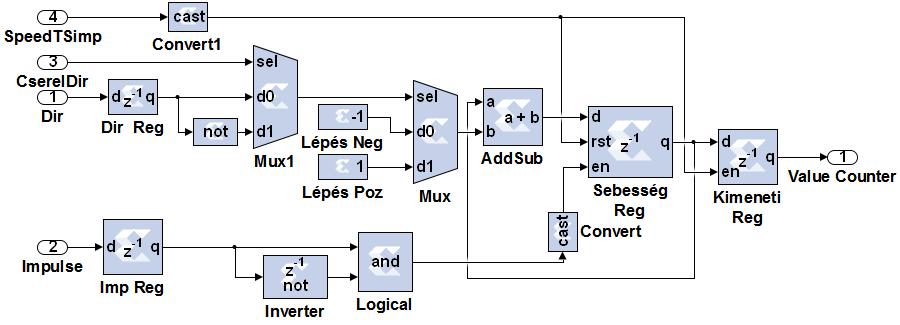
Tároló regiszter, feladata az aktuális érték tárolása, a típusa 16bites előjeles egész értékű a kezdőértéke mindig 0. A rst bemeneten érkező jel 0 értékre hozza a regiszter tartalmát, ez akkor történik meg, ha a „*NullSignal*” bemeneten impulzus érkezik. Azt mondhatjuk, hogy a pozíciót a 0 állapothoz mérjük, amely a Null átmeneti tárcsa határozza meg.

A számláló logika feladata, hogy növelje vagy csökkentse eggyel a regiszter tartalmát, annak függvényében, hogy a Dir jel milyen értékű. A „*ValueCT*” bemenet segítségével meg tudjuk fordítani a számolás irányát.

#### Szögsebesség mérése Inkrementális adó segítségével

A sebesség mérésénél hasonlóképpen járunk el, mint a pozíció mérésénél. A sebességet, időegység alatt érkező impulzusok számát mérjük. Az impulzusok az inkrementális adó jeleinek a feldolgozó moduljától érkeznek.

A modulban megtalálható a pozíció mérésénél kifejtet számláló logika, tároló logika és él detektáló logika, valamint itt még jelen van egy 5 mintás átlagoló szűrő. A szűrő tartalmaz öt regisztert, amelyek FIFO láncot alkotnak. A regiszterek és az összeadó modulok típusa megegyezik a tároló regiszter típusával. A mért sebesség értékek bekerülnek A FIFO struktúrába, minden Ts impulzus érkezésekor a FIFO elejére illesztődik a mért érték. Minden impulzuskor összegződik a FIFO tartama.

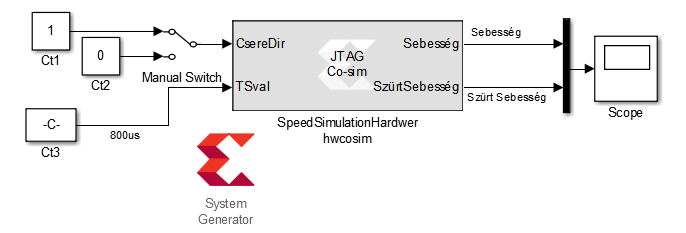


Kép. 3.25 Sebesség mérő modul felépítése

Az összeget elosztjuk a regiszterek számával (5) Mivel a regiszterek száma időben nem változik, ezért alkalmazhatjuk a 1/(regiszterek Száma) való szorzást. A tároló regiszter minden mintavételi impulzus érkezésekor reset állapotba kerül, így az értéke 0 lesz. A modulból kivezetjük mind a szűrt, mind a szűretlen értékeket.

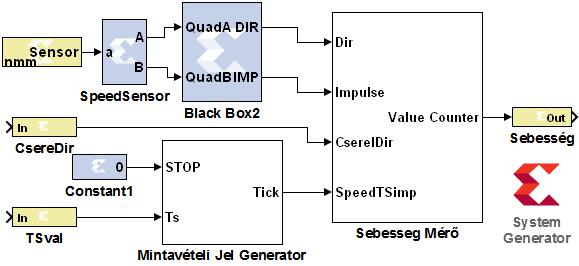
##### Hardveres mérések

A mérések során Kép. 3.30 látható egy DC motornak a tengelyén mért fordulatszáma, miközben a motor maximális sebességen pörög. Ahhoz hogy eltudjuk végezni a szimulációkat az inkrementális szenzort hardveresen illesztenünk, kell az FPGA kivezetéseire, amelyet a „*Sensor*” modul old meg.



Kép. 3.26 Dc motor sebességének mérése Fpga lapon

A Kép. 3.27 látható a Hardveres szimulációhoz szükséges logika, Sensor modul tartalmazza az FPGA azon kivezetéseit, amelyekre az érzékelő fizikailag kapcsolva van. Az érzékelő jelei egy SpeedSensor (csak a jelek bekötésében segít) nevű modulon keresztül a feldolgozó modulba érkeznek be a jelek. Ugyanakkor még megtalálható egy SampleTime Generator1 nevű modul is amelynek a feladata ( ) periódusú impulzusok generálása, a periódust bemeneten adhatjuk meg. A kiszámolható ms-ban az alábbi összefüggéssel.



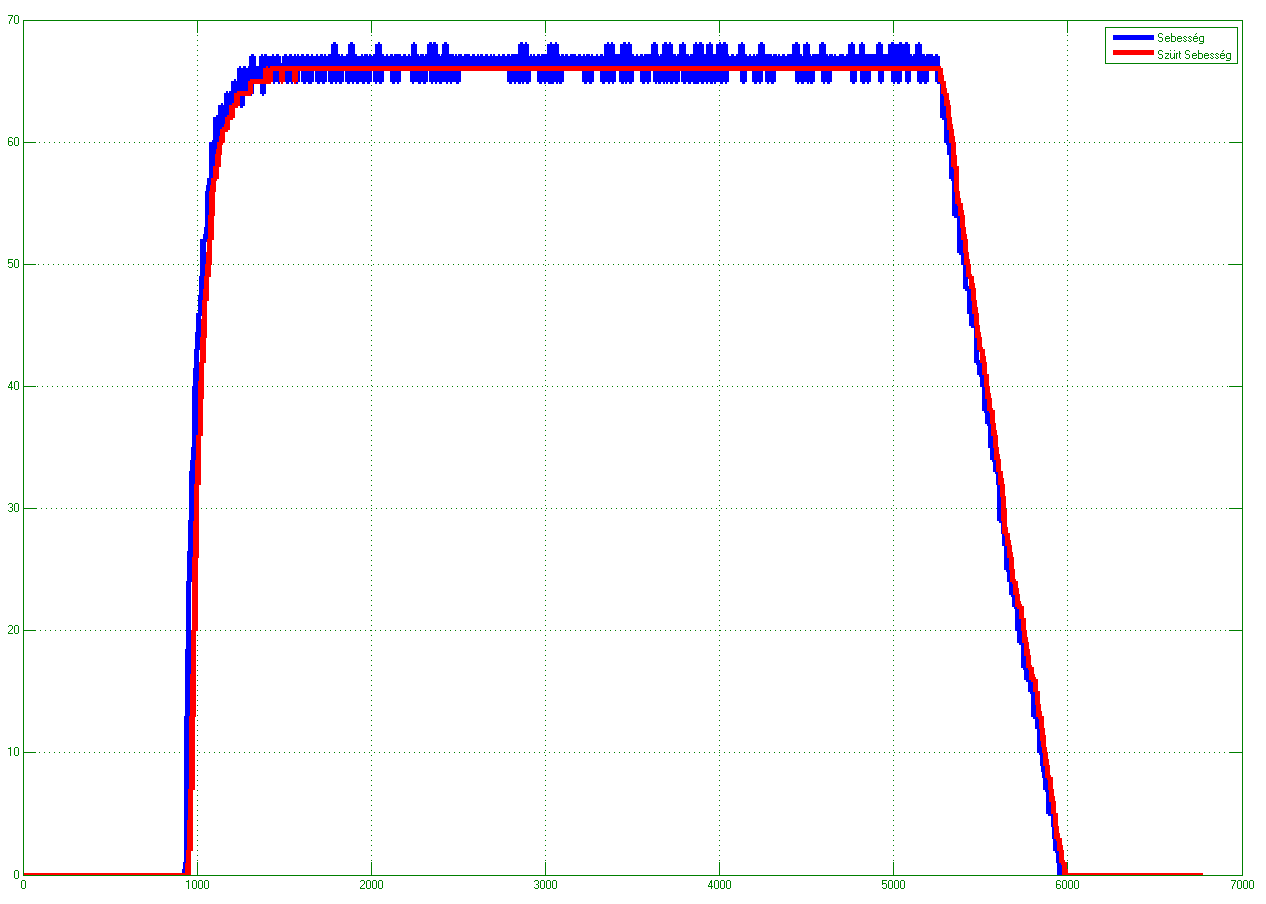
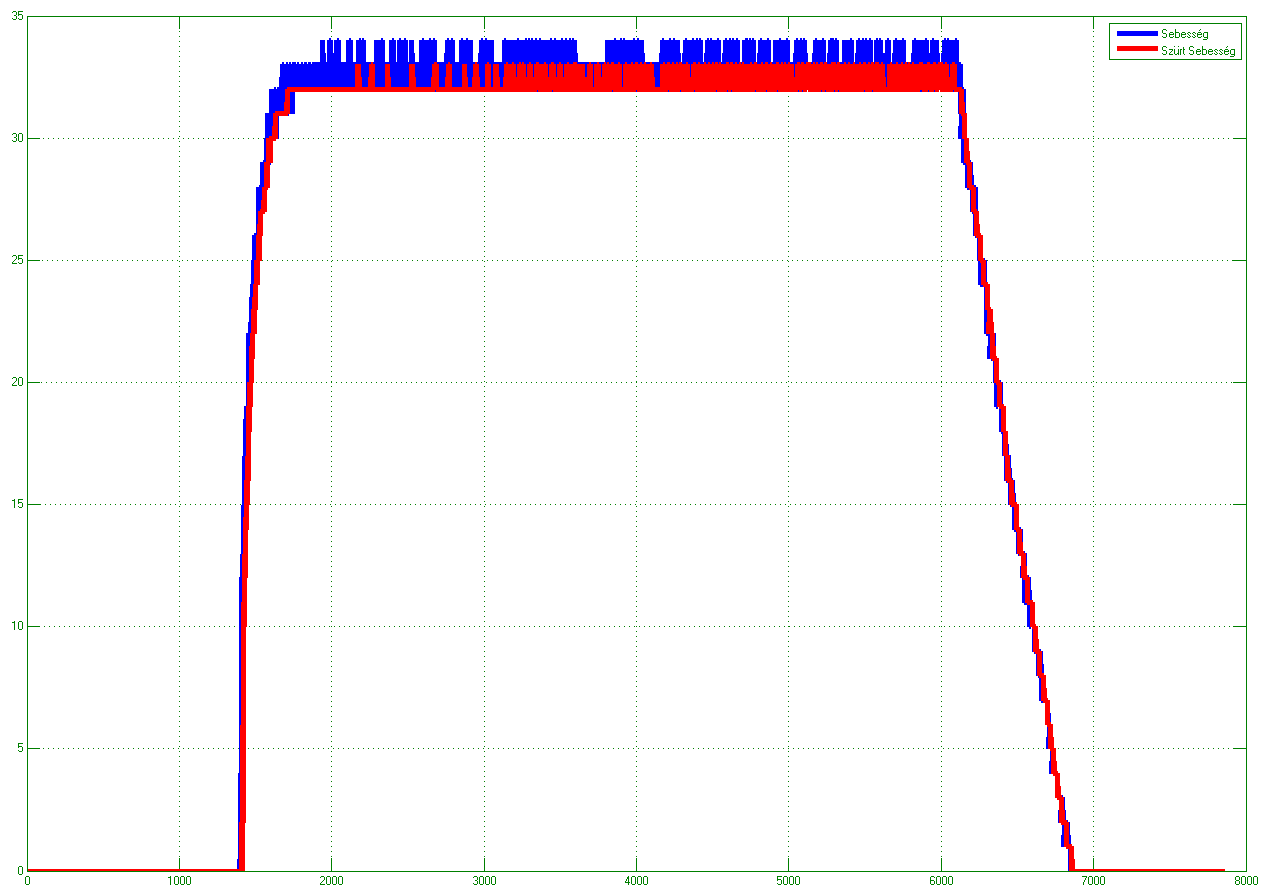
Kép. 3.27 Dc motor Sebesség mérése FPGA rendszeren, System generatorban megvalósítva

A terv kigenerálása után kapunk egy újabb modult SpeedSimulationHardwer hwcosim elnevezéssel.

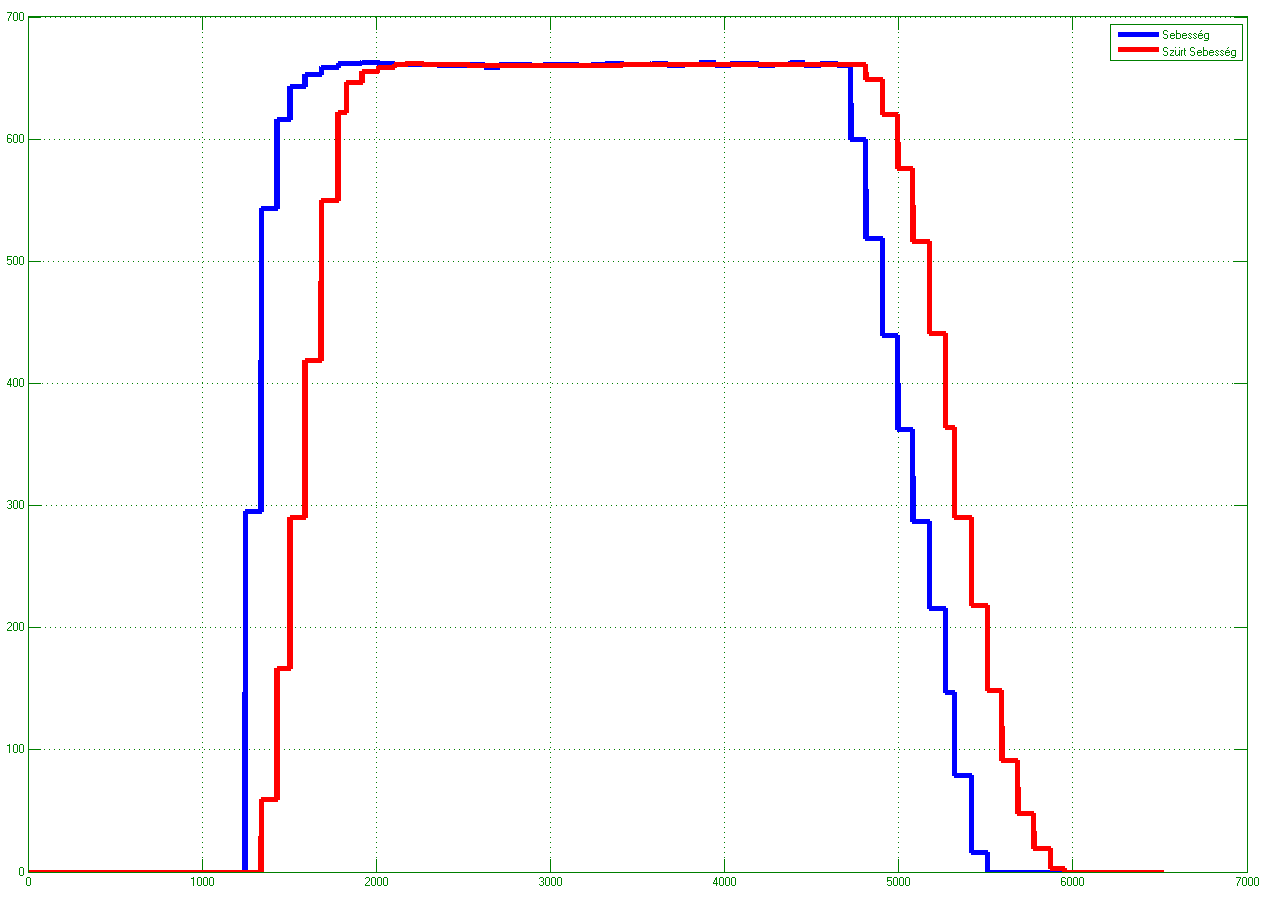
A 2.17 képen látható az újonnan generált modul bemenő adatainak a megadása illetve a mért értékek ábrázolása. A Ct3 nevű konstansba megadhatjuk a mintavételi időt ms ban.

A sebességet adott időegység alatt beérkező impulzusok számával mérjük.

**Eredmények**: lenti képeken látható a motor adott Ts mintavételi periódusokban érkező impulzusok száma, illetve a szűrt sebesség. Ahhoz hogy megkapjuk a sebességet RPM ben átkel alakítani.



Kép. 3.30.a 8ms



Kép. 3.30.c 4ms

Kép. 3.30.b 80ms

Következtetések: A fenti példában a az maximális fordulat, azért mert a motort a maximális megengedett feszültséggel volt táplálva.

Ha ismerjük a motor maximális fordulatszámát, ami megadja a mérés felbontását is.

A fenti példa esetében a felbontás 65 mivel egy időegység alatt 0 és 65 közötti értékeket vehet fel, amelyek csak természetes számok lehetnek.

Ha növelni szeretnénk a felbontást () növelnünk kell a mintavételi időt. Az alábbi összefüggés szerint kiszámíthatjuk az optimális, mintavételi időt ismerve a maximális fordulatszámot percenként, és a tárcsa adatait, .

A 2.21.B képen a mintavételi időt 80ms növelve, megnőtt a felbontás is megközelítőleg 650 re, A 2.21.C képen csökkentjük a mintavételi időt, és ez megközelítőleg 33-ra csökkentette a rezolúciót.

## MPU-6050 giroszkóp és gyorsulásmérő

A szenzorban megtalálható háromtengelyes gyorsulásmérő, és giroszkóp. Az eszközzel i2c komunikácíós protokollon keresztül kommunikálhatunk. Az eszköz rendelkezik egy saját címmel, amelyet mi választatunk ki egy ellenállás segítségével az AD0 bementén. Abban az esetben, ha az AD0 bemenetet egy ellenállás segítségével GND re kötjük a címünk 0x68 lesz, ha pedig Vcc re kötjük ellenállás segítségével a cím 0x69 lesz. A [8] adatlap alapján a következő bealításokat végeztem el:

* FIFO memóriák kikapcsolása FIFO\_EN=0x00
* Gyorsulásmérők inditása ACCEL\_CONFIG=0xE7
* PWR\_MGMT\_1 =0x00 bealítjuk a szenzort ciklikus működésre és 8MHz órajelre.

## Beavatkozó elemek:

### Pwm Generátor megvalósítása FPGA áramkörön System Generator környezetben.

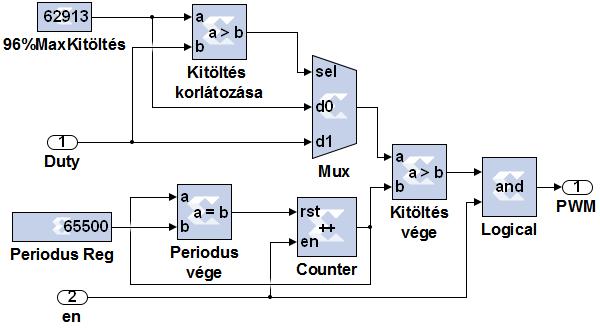
#### Megvalósítás

Egy hardveres számláló segítségével, amely az FPGA órajelére számol, egy komparátor segítségével összehasonlítjuk a számláló értékét az általunk megadott kitöltési tényező értékével, mindaddig, amíg a számláló értéke kisseb a kitöltési tényező értékénél a PWM kimenetén logikai 1-es kimenet lesz, amikor meghaladta a számláló akkor pedig 0 lesz. A számláló típusa egy 16bites pozitív egész értékű.

A PWM jel frekvenciáját egy „*Periodus Reg*” nevű regiszteren keresztül adhatjuk meg, a regiszter értéke össze van hasonlítva a számláló értékével, és amikor a számláló értéke megegyezik a regiszter értékével a számlálót reset állapotba hozzuk.

A PWM kitöltési tényezőjét egy „*Duty*” nevű bemeneten keresztül vihetjük be a rendszerbe, amelynek a típusa megegyezik a számláló típusával. A Duty értékének szüksége van egy skálázási eljárásra, amely segítségével illesszük a frekvenciához a Duty értékét. Ugyanakkor a felépítésében be van iktatva egy korlátozás, amely segítéségével nem engedjük meg egy bizonyos százalék fölötti kitöltési tényezőt, „96%MaxKitöltés” nevű konstansba írhatjuk be 0 és 65535 közötti értékben. A Duty bemenet maximálisan 0 és 16biten felírható maximális érték között lehet (65535). Az alábbi egyenlet segítségével kiszámíthatjuk a kitöltési tényező regiszterének az értékét.

Kép. 3.31 A PWM generátor System Generátorban megvalósított szerkezete

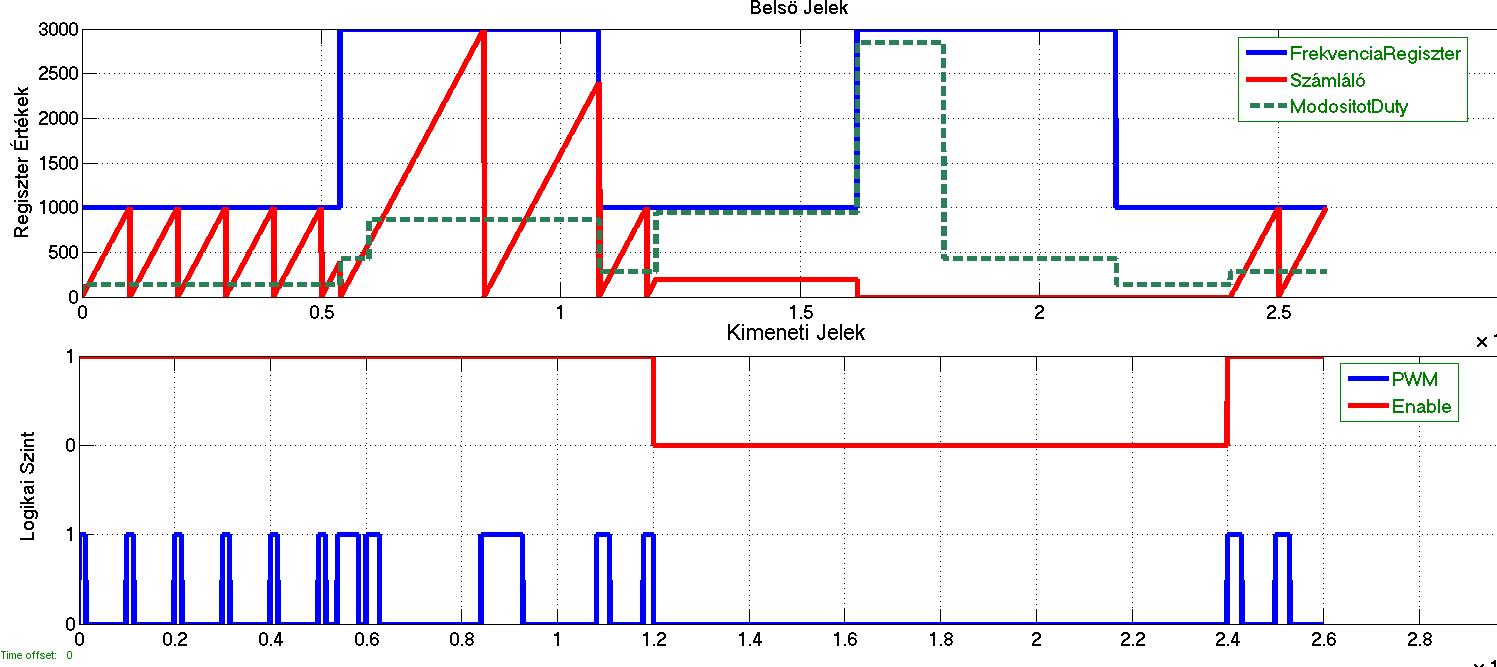


A PWM generátor kimenetét és a számláló működését letilthatjuk az „*en*” bemenetre adott logikai 0 értékkel, máskülönben engedélyezve vannak.

A frekvencia megadásához ismernünk kell az FPGA órajelének a periódus idejét, amely jelen esetben

A értéke ugyanakkor meghatározza a felbontást is, vagyis egy teljes periódust a PWM jelben hány részre tudunk felbontani. Látható hogy fordított arányosság áll fen a frekvencia és a között, így ha növeljük a frekvenciát, csökkeni fog a rezolúció.

A Kép. 3.32, a felső ábrán látható a kékkel jelölt frekvencia regiszter értéke, pirossal jelölt a számláló értéke, a zöld szaggatott a skálázott kiötlési tényező értéke. A kép alsó részén látható a kékkel jelelt PWM jel, illetve papirossal jelölt Enable jel.



Kép. 3.32 a PWM generátor bemenő, kimenő illetve néhány belső jele (Scope1)

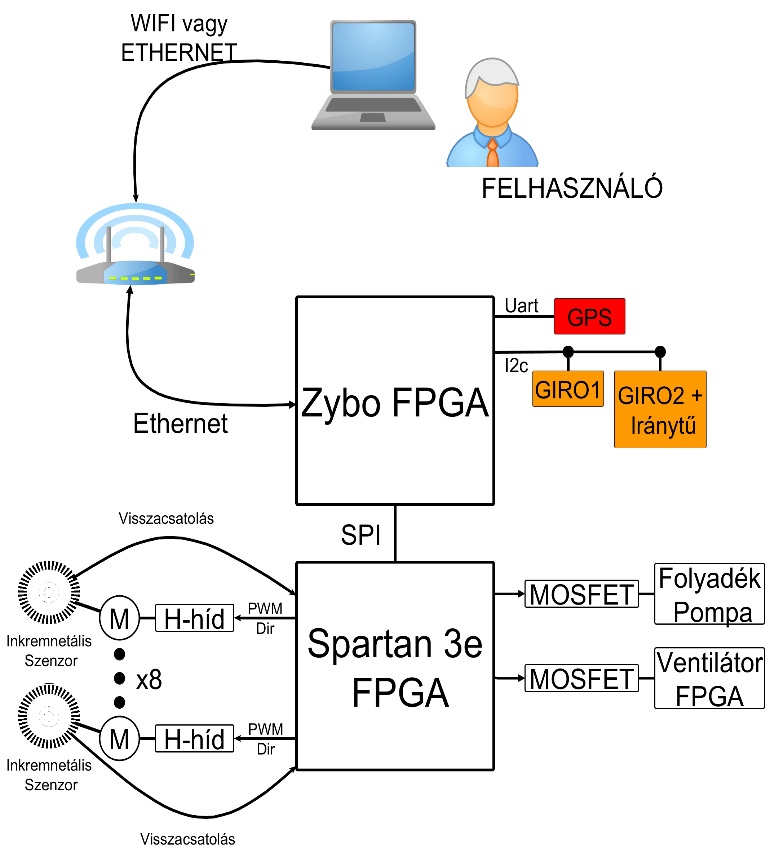
## Elektronika

### Digitális Elektronika

Az szenzorok adatainak a gyűjtésére és a beavatkozó jelek számítására FPGA rendszert használtam a gyorsaságuk és a flexibilitásuk miatt. FPGA rendszeren, könnyedén kivitelezhetjük az általunk tervezett hardveres elemeket és hozzákapcsolhatjuk egy beépített processzorhoz, és osztott regisztereken keresztül adatokat nyerhetünk, illetve küldhetünk az általunk megtervezett hardveres elemeknek.

### FPGA Rendszer Felépítése

A rendszeren megtalálható két FPGA fejlesztő lap, egy ZYBO amely nagyobb erőforrással rendelkezik, de kevés a kivezetéséinek a száma, és egy Spartan3e chippel rendelkező fejlesztőlap, amely kevés erőforrással bír, de 120 kivezetést tartalmaz.



Kép. 3.33 rendszer elvi felépítése

A ZYBO fejlesztőlapon levő ZYNQ 7010 chip tartalmaz két beépített ARM Cortex A9 processzort, a processzor mellet található egy újrakonfigurálható mag, és egy előre elkészített periférikus elemeket tartalmazó rész. A processzorok a körülöttük levő elemekkel az AXI busznak nevezet sin rendszeren keresztül tudnak kommunikálni.

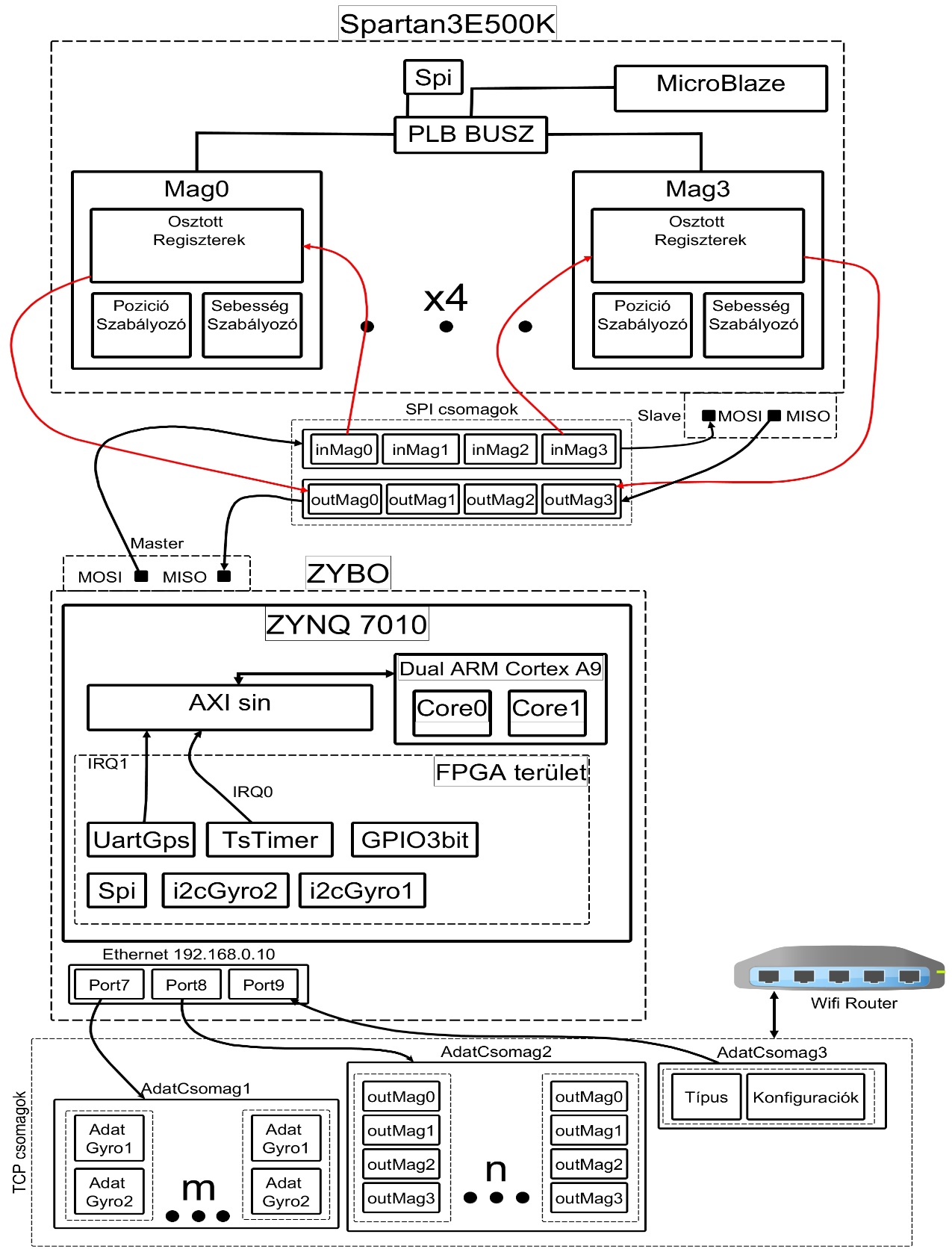
Az Spartan FPGA-ba kialakítunk egy 32 bites microProceszort (microBlaze), és a hozzá szükséges PLB sin rendszert, a sin rendszere illesztünk egy SPI kommunikációs egységet melynek feladata a ZYBO fejlesztőlappal való fizikai kommunikációs réteg kialakítása. A PLB buszra illesztünk még négy darab SebességÉsPozició szabályozóra IPmagot, amelyeket a System Generátorban készítünk el és generálunk ki.

A szabályozókat tartalmazó IPmag osztót regisztereken keresztül állíthatjuk be a paramétereit vagy olvashatunk ki értékeket, amelyek a PLB buszon találhatók. A Zybo lapon található Eternet modulon keresztül kapcsolódunk egy Wifi routerhez, amely Access pontként működik. A routerhez még csatlakoztathatunk három más vezetékes vezetékes eszközt, amelyek lokális hálózatba lesznek kötve a Zyboval.

#### Zybo FPGA fejlesztőlap

A két beépített processzorok (Core0, Core1) között munkamegosztást kell kialakítani a hatékonyabb működés elérése céljából.

A Core0 processzor feladatai között szerepel a megszakítások lekezelése, a legfontosabb megszakítása a mintavételi periódust generáló számlálótól érkező megszakítás, amelynek bekövetkeztekor a processzor begyűjti, az adatokat az szenzoroktól (Giroszkóp 1 és 2). Lekezeli a megszakítást, amelyek az UART modultól érkezik és a GPS adatait tartalmazza. Az adatok begyűjtése után elindítja a matematikai modell kiszámítását, amely a Core1 processzoron történik. Az Ethernet kommunikációhoz szükséges szervereket futatja.



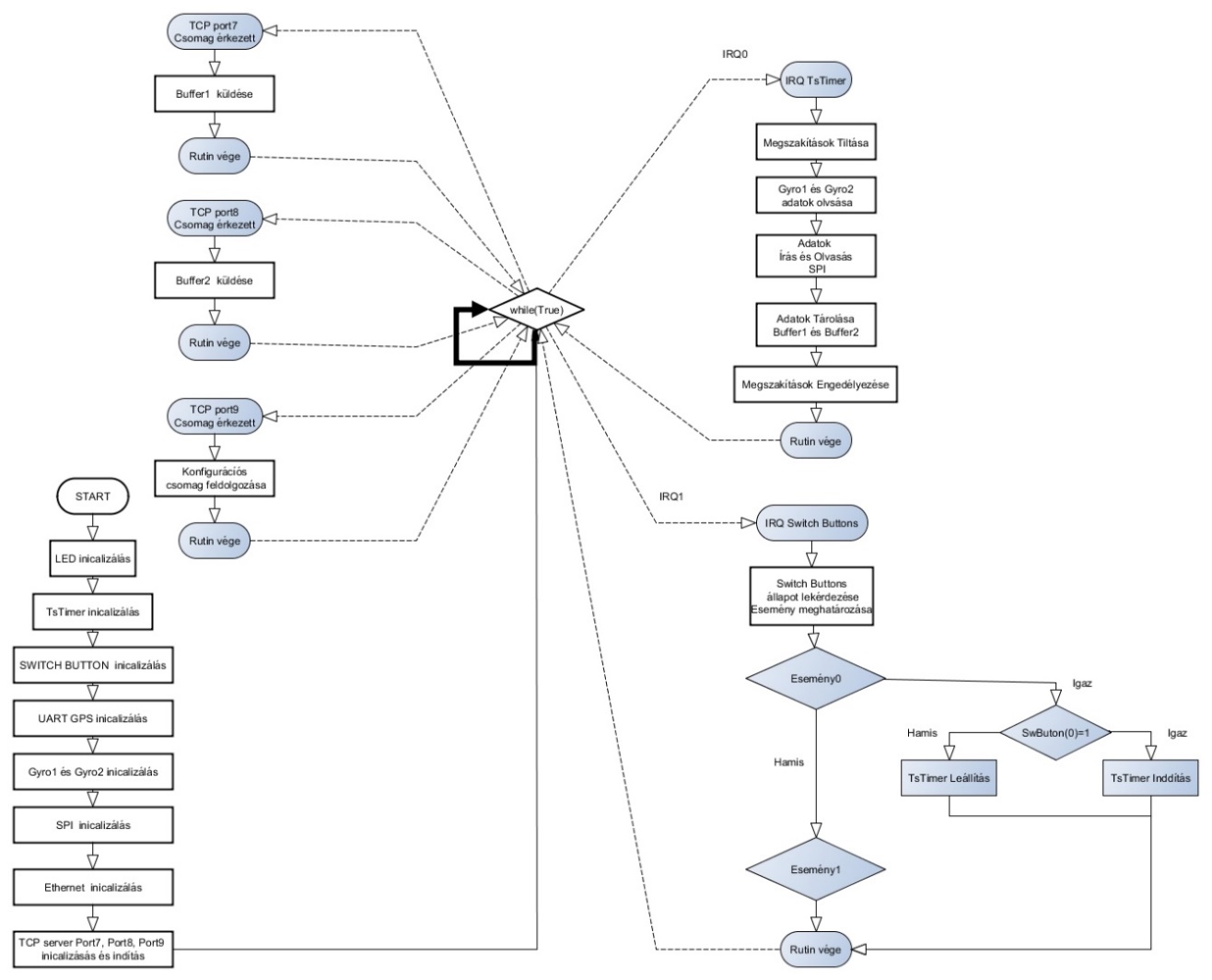
Kép. 3.34 Kommunikációs csomagok és az FPGA áramkörökbe programozott modulok elvi felépítése

Miután végzett a Core1 a matematikai számításokkal az SPI kommunikáción keresztül elküldi a szabályozók referencia értékeit a Spartan fejlesztőlapnak.

A szoftver a 3.3. képen látható folyamatábra szerint működik. Az indítás után a program elvégzi az eszközök előkészítésest és a beállításaikat, majd egy végtelen ciklusba lép bele. A ciklust bármikor megszakíthatja a TsTimer megszakítása, amely a legnagyobb prioritással bír. A megszakítás kiszolgálása előtt letiltjuk a megszakításvektort így nem érkezhet megszakítás a kiszolgáló rutin végrehajtásakor.

Switch Buttonok megszakítása nem nagy prioritású, célja a manuálisan bealítható paraméterek futás közben változtathatóak. A megszakítás érkezésekor lekérjük a Switch Button kapcsolok állapotait, majd generálunk egy eseményt annak függvényében, hogy mely kapcsoló állapota változott meg.

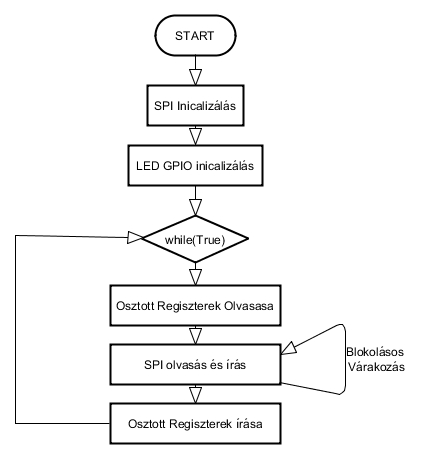
A program tartalmaz három TCP szervert is, amelyek a megadott port számon várják a kéréseket, minden kérés beérkezte után elküldik a pufferekben tárolt adatokat a kérést küldő kliensnek, és kiürítik a puffereket.



Kép. 3.35 ZYBO Core0 program folyamat árbája

#### Spartan3e FPGA fejlesztőlap

Feladata kezeli a SPI kommunikáción beérkező adatokat és eljutatja a megfélő osztott regisztereken, valamint a visszaküldi a szabályozók adatait a Zybo rendszernek.



Kép. 3.36 Spartan3e500, microblaze szoftver Folyamat ábrája

Abban az esetben, ha megszakad az SPI kommunikáció egy adott időn belül letiltja a PWM modulok kimenetét így állítva meg a rendszert.

A 8 hardveres szabályzó függetlenül működik a programtól, a programtól csak paramétereket kap. Abban az esetben, ha a program leáll vagy lefagy a szabályozók, akkor is tovább működnek.

A microBlaze processzoron futó program legfőbb feladata az SPI kommunikáció és az osztott regiszterek kezelése. Az Spi olvasás blokkolásos művelet, mivel a blokkolásból csak akkor lép ki ha lezajlott az adatcsere a Zybo fejlesztőlappal, ezután a kapót adatokat beírja a megfelelő regiszterekbe, és kiolvassa a szabályozók adatait amelyet a következő adatcserekor küld majd tovább.

A mintavételezési időt a Zybo határozza meg, a TsTimer segítségével.

#### Kommunikációs protokollok

##### SPI kommunikációs protokoll:

A Zybo mester míg a Spartan szolga egységként működik, a kommunikáció szinkron típusú az órajel frekvenciája 1.56 MHz. Miközben a mester adatokat küld a szolga irányába a szolga egység is továbbit párhuzamosan adatokat a mester irányába. Az spi legkisebb csomagja minimum 32bit. Az spi kommunikáció a Kép. 3.34 látható SPI csomagszerkezeten keresztül történik.

A struktúrák, amelyeket küldünk vagy fogadunk, mérete mindkét esetben megegyezik, egy IPmag 13\*4 byte adatot kap és küld minden adatcserekor. Mivel 4 IPmag van a rendszerben így a teljes csomag 13\*4\*4, összesen 208byte hosszú.

Az ipmag csomagok mellet megtalálható két regiszter, amely segítségével a nem kritikus modulokat tudjuk megcímezni és adatokat továbbítani nekik pl: ventilátor pwm modul, pumpa pwm modul.

s32 addresReg; - cél regiszter szoftveres címe

s32 dataReg; - cél regiszter adata

**outMag0**

s32 USebesseg0;

s32 UPozicio0;

s32 SebessegPozicio0;

s32 AktPozicio0;

s32 SzurtSebessegPozicio0;

s32 AktSebesseg0;

s32 SzurtSebesseg0;

s32 eSebeseg0;

s32 n20;

s32 n30;

s32 n40;

s32 n50;

s32 n60;

**inMag0**

s32 Config0;

s32 RefPos0;

s32 RefSeb0;

s32 Q0\_0;

s32 Q1\_0;

s32 Q2\_0;

s32 Ts0;

s32 TsL0;

s32 Egyeb0;

s32 PwmFrekREgH0;

s32 PwmFrekRegL0;

s32 sugarakAranya0;

s32 qSav0;

A outMag0 csomag tartalmazza egy beépített Ipmag kimenő adatait: aktuális sebesség, pozíció, sebesség, és pozíció szabályozó beavatkozó jele, valamint n20-n60-ig üres csomagok, azért van rá szükség hogy a outMag0 mérete megegyezzen a inMag0 méretével.

Az inMag0 tartalmazza azokat az adatokat, amelyek segítségével betudjuk konfigurálni a szabályzókat, és a paramétereket tudjuk frissíteni.

##### Ethernet

A kommunikáció három TCP serveren keresztül keresztül zajlik. Az első szerver a port7 várja a kéréseket, minden kérésre elküldi a giroszkópok adatait tároló puffert, amely tartalmazza az utolsó lekérdezéstől gyűjtött adatokat. A második szerver a port8 várja a kéréseket, minden kérésre úgy, mint az előző, elküldi az IP MAGOKTÓL beérkezett adatokat. A harmadik szerver segítségével konfigurációs parancsokat küldhetünk a rendszernek, amelyeket értelmez és végrehajtja.

### Feladatok Elosztása

#### Zybo fejlesztőlap

A két beépített processzorok (Core0, Core1) között munkamegosztást kell kialakítani a hatékonyabb működés elérése céljából.

A Core0 processzor feladatai között szerepel a megszakítások lekezelése, a legfontosabb megszakítása a mintavételi periódust generáló számlálótól érkező megszakítás, amelynek bekövetkeztekor a processzor begyűjti, az adatokat az szenzoroktól (Giroszkóp 1 és 2). Lekezeli a megszakítást, amelyek az UART modultól érkezik és a GPS adatait tartalmazza. Az adatok begyűjtése után elindítja a matematikai modell kiszámítását, amely a Core1 processzoron történik. Az Ethernet kommunikációhoz Szükséges Socketeket is kezeli.

Miután végzett a Core1 a matematikai számításokkal az SPI kommunikáción keresztül elküldi a szabályozók referencia értékeit a Spartan fejlesztőlapnak.

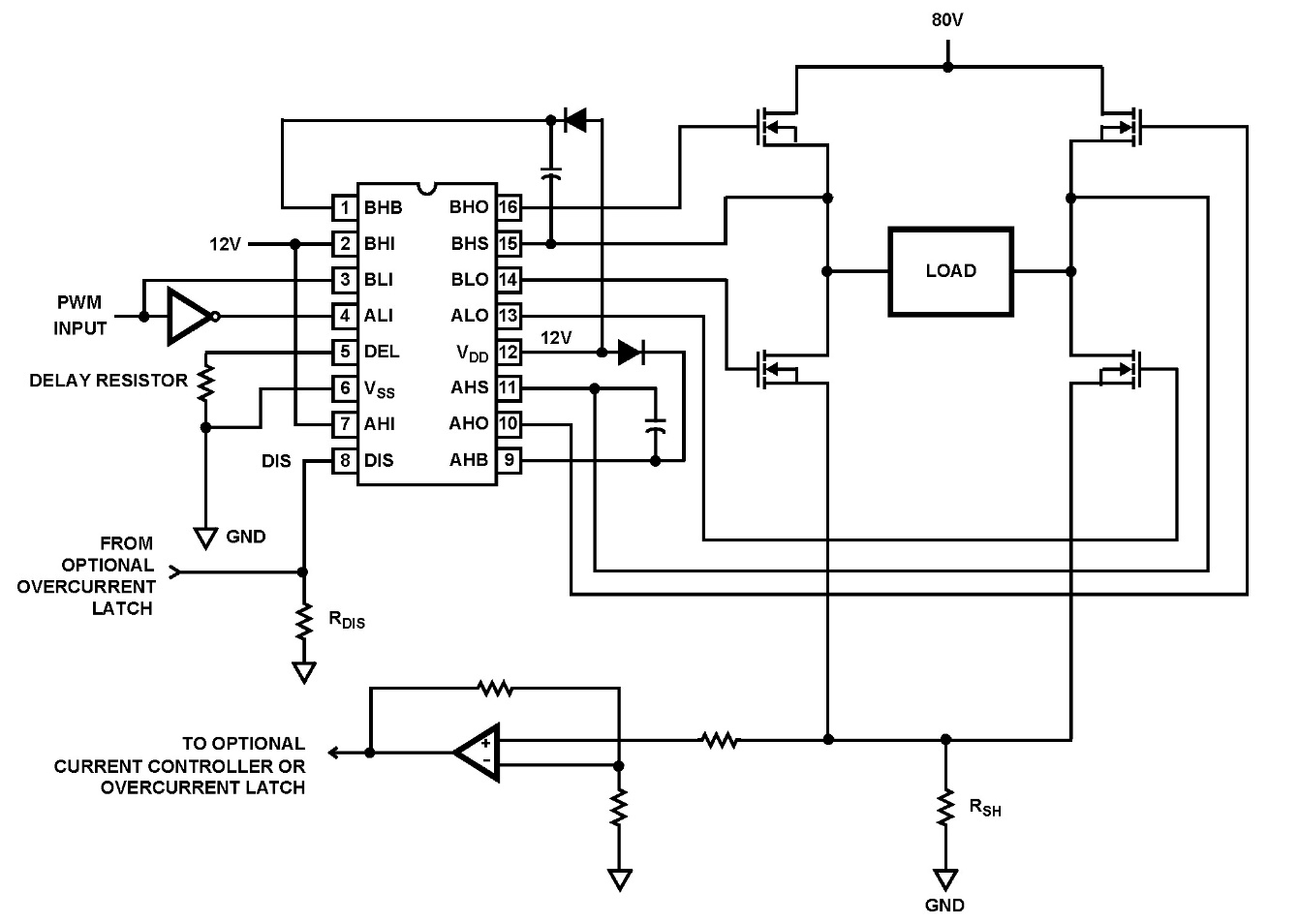
#### Spartan fejlesztőlap

Feladata kezeli a SPI kommunikáción beérkező adatokat és eljutatja a megfélő osztott regisztereken, valamint a visszaküldi a szabályozók adatait a Zybo rendszernek.

Abban az esetben, ha megszakad az SPI kommunikáció egy adott időn belül letiltja a PWM modulok kimenetét így állítva meg a rendszert.

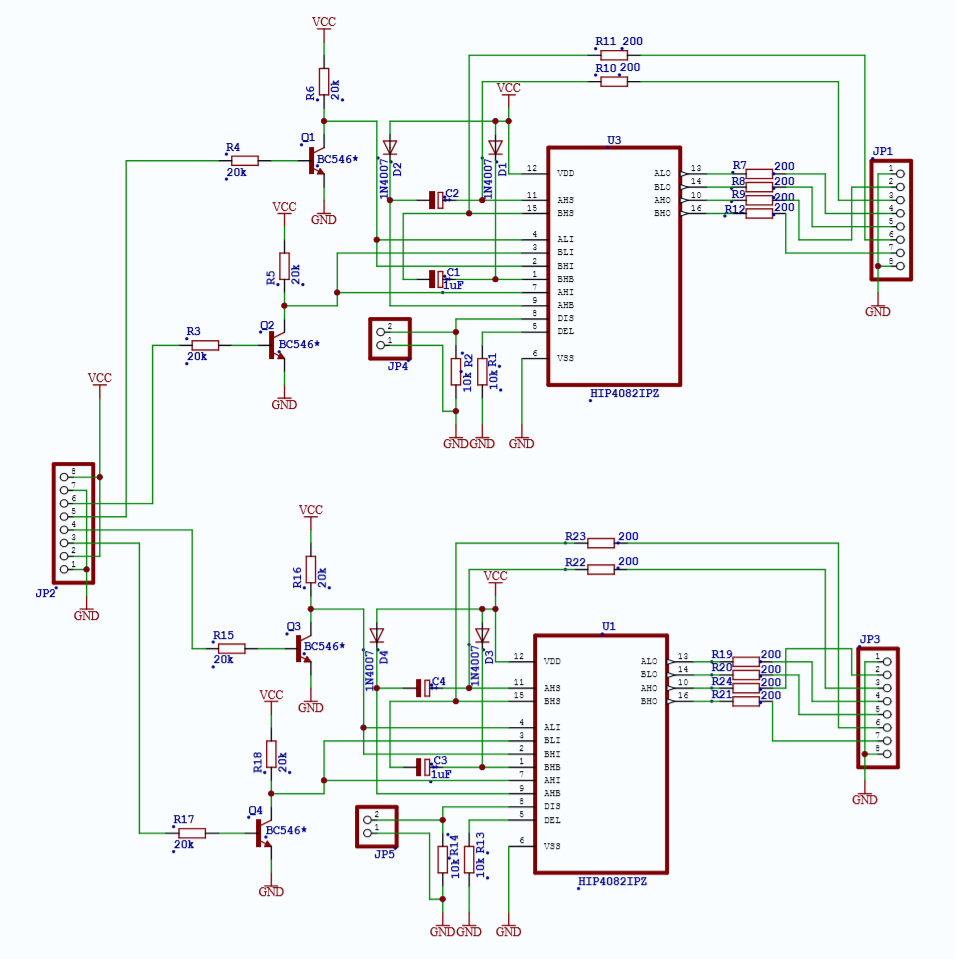
## GUI felépítése

## Teljesítmény elektronika

A DC motorok betáplálására 12V-16V egyenfeszültséget alkalmazunk a motorokat feszültségben pwm beavatkozó jel segítségével vezéreljük H-hidon keresztül. A H-hidakat N-csatornás MOSFET tranzisztor segítségével valósítottam meg, mert az N csatornás MOSFET tranzisztornak kisebb az DS ellenállása kinyitott állapotban. A tranzisztorokon nagy áramok is átfognak haladni akár 20A is. A tranzisztorok hűtése vízzel történik, a tranzisztorokat egy réz hűtőlemezre fogatva, amely közvetlen kapcsolatban van a hűtő folyadékkal. A tranzisztorok és a rézlemez között elektromos szigetelés van létesítve egy hővezető, de elektromos szigetelő segítségével. 

Kép. 3.37 Hip4082 alkalmazása H híd kapcsolásban. Forrás: <http://www.intersil.com/en/products/space-and-harsh-environment/harsh-environment/half--full-bridge-and-three-phase-drivers/HIP4082.html>

Az alkalmazott N csatornás MOSFET tranzisztorok megnevezése IRFB7437, adatlapi adatok szerint az 195A áramot tud vezetni maximálisan, 40V feszültséget bír el, valamit a teljesen kinyitott állapotban az ellenállása kisebb, mint 2mΩ.

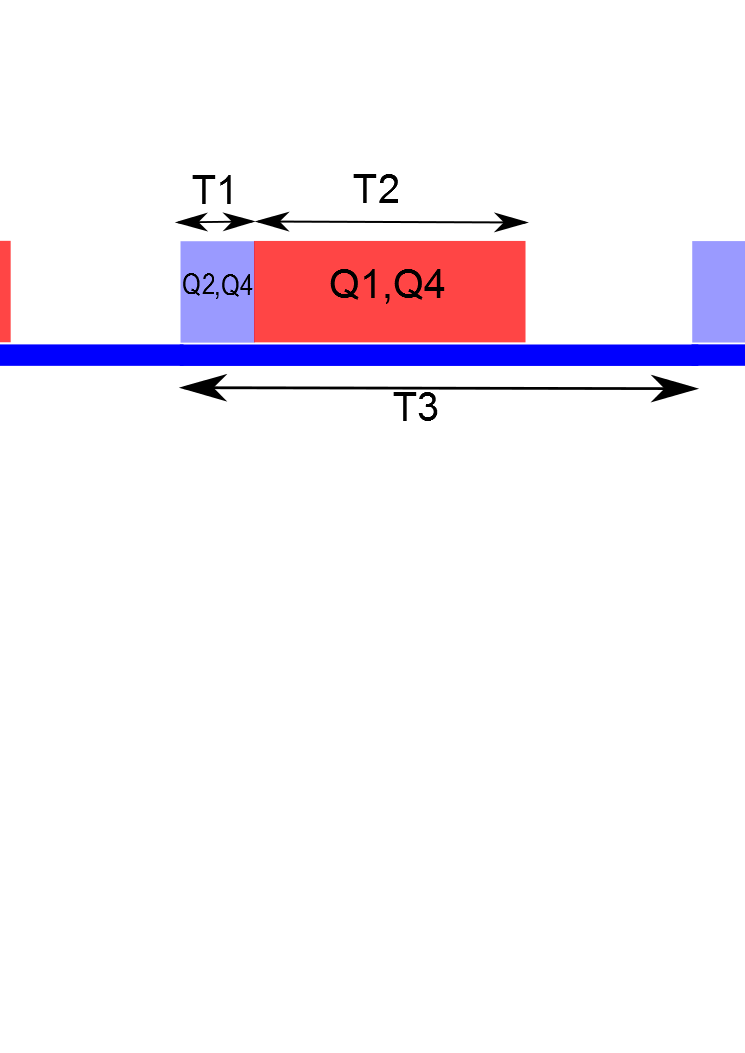


Kép. 3.38 Két hídvezérlő árak kör kapcsolási rajza HIP4082 integrált áramkörrel megvalósítva

Az N csatornás MOS tranzisztorokat pozitív feszültséggel tudjuk bekapcsolni a S (Source) nagyon befolyásolja. Ezért egy szinteltolást kell alkalmazni a felső tranzisztorok meghajtására Bootstrapp megvalósítás). Kivezetéséhez viszonyítva. Mivel a felső két tranzisztor S kivezetése a földhöz képest libeg, a terhelés miatt, ezért Boost megoldást alkalmazunk a felső két mosfet tranzisztor Gate bementének a vezérlésére. A megoldás egy diódát és egy kondenzátort tartalmaz, és feltételezi a szaggatást így oldva, meg hogy a kondenzátor töltődjön fel és majd a HIP4082 icn keresztül tudjuk rákapcsolni a felső tranzisztorok Gate bemenetére. A [9] adatlap alapján AHB, BHB az ic azon bemenetei, amelyeknek feszültsége elvan tolva az AHS,BHS közös pontokhoz képest, a H hídban egy kondenzátor feszültségével. Az AHO, BHO azok a pontok, amelyek a felső tranzisztorokat vezérli, az ALO, BLO az alsó tranzisztorok vezérlő kivezetései. A HIP4082 be megtalálható beépítve egy késleltető, amely garantálja, hogy ne alakuljon ki rövidzárás a hídban, abban az esetben, amikor a felső és az alsó tranzisztorokat kapcsoljuk át ugyan azon a fel híd oldalon.

A [9] adatlap alapján az, ha ALI, AHI, BLI, BHI bemeneteket, amelyek segítségével vezérelni tudjuk a tranzisztorokat. Az alsó két tranzisztor az ALO, BLO prioritást élveznek a felső AHO, BHO tranzisztorokkal szemben. Ami abból áll, ha bekapcsoljuk az alsó tranzisztorokat a felső tranzisztorok, ha bekapcsolt állapotban vannak, akkor kikapcsolja, és ha kikapcsolt állapotban vannak, és beszeretnénk kapcsolni akkor nem, engedi a bekapcsolásukat.

A Kép. 3.39 látható a pwm jel egy periódusának a felbontása. A pwm jel alapfrekvenciája 700Hz, így a T3=1,42ms. T1=0.5%T3=71us. A pwm jel kitőltése Duty=T1+T2.



Kép. 3.39 PWM és a tranzisztorok kapcsolása

A DIS bemeneten áramkorlát vagy más védelmeket iktathatunk be a rendszerbe, a [9] adatlapban ajánlott megoldás szerint méri az áramot az ellenálláson keresztül, amelyet egy műveleti erősítő segítségével, amelyet nem invertáló erősítő alapkapcsolásban használ így dönti el, hogy áramkorlátban van vagy nincs. A DIS bemenet logikai 1 ben van, ha 2.5V fölött van, illetve logikai 0, ha 1V alatt van.

A Kép. 3.39 látható a nem invertáló erősítő kapcsolás a Ube bemeneti feszültség az áramérő ellenállástól érkező feszültség. A kapcsolás erősítése:



Kép. 3.40 Nem invertáló erősítő forrás [13]

Ha , 10A szeretnénk az áramot korlátozni akkor a , ahol az a H hídon átfolyó áram nagysága.

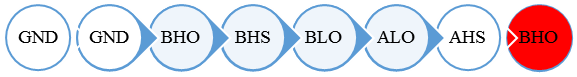
Az erősítés: . A műveleti erősítős áramkorlát nem jelenik meg a megvalósításba, de a továbbfejlesztési lehetőség megvan rá. A Kép. 3.38 képen látható JP4 és JP5 jumper segítségével elengedhetjük a DIS bennetet földhöz képest így kiiktatva az áramkorlátot, de illeszthető hozzá a jumperek helyére Kép. 3.39 képen látható áramérő és áramkorlátozó kialakítás.

A Kép. 3.38 képen látható kapcsolási rajz tartalmaz két HIP4082 hídvezérlő áramkört, az áramköre a bemenet a JP2 bemeneten történik, amelyen keresztül betápláljuk 12V feszültséggel, és 4 PWM jelet segítségével megtudjuk hajtani a két hidat. A jeleket szalagkábel segítségével csatoljuk az áramkörhöz. A szalagkábel 8 vezetékből tevődik össze, rendeltetésük szerint:

Kép. 3.41 Dupla hídvezérlő áramkor vezérlő jelei JP2 csatlakozó a Kép. 3.38-n.

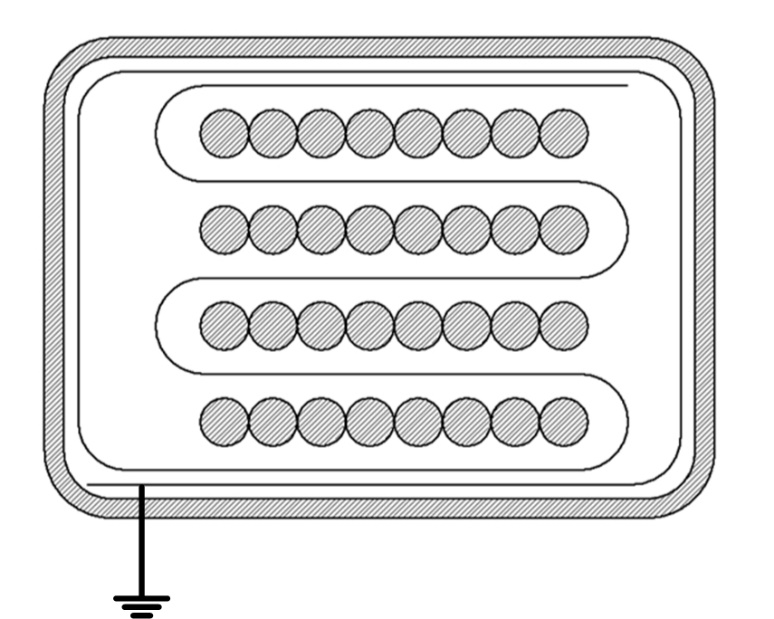
A pirossal megjelölt vezeték az 1 számú. A PWM4 egy 3,3Vpwm jel, amely egy NPN (Q2) tranzisztoron keresztül kapcsolja az BLI, AHI bemeneteket, a tenisztor a jelet megtagadja, ezért majd a FPGA pwm moduljába illesztünk egy tagadó, kaput hogy oltsa ki egymást a két kapu. A PWM3 hasonlok épen az előzőhöz csak a ALI, és BHI bementeket vezérli a (Q1) tranzisztorokon keresztül.

A Kép. 3.44 látható 4H-híd B és A dobozok tartalmaznak négy H hidat, a hidak kétesével rögzítve vannak egy rézlemezre, amelyeken keresztül tudunk egy vizet keringetni egy réz csővezeték segítségével így hűtve a tranzisztorokat. A tranzisztorok galvanikusan levannak választva a lemeztől egy elektromos szigetelő segítségedével, de ugyanakkor a szigetelő jó hővezető is. Egy hídban megtalálható tranzisztorok vezérléséhez szükséges vezetékek a Kép. 3.41 láthatok.



Kép. 3.42 H híd tranzisztorainak a Gate vezetékei

A négy híd vezérlésére négy Kép. 3.41 szalagkábelre van szükségünk, amelyeket a Kép. 3.42 látható módon rendeztem el és árnyékoltam le a zavarforrásoktól. A szalagvezetékek között és körül alumínium fólia található, amelyek földpotenciálon vannak. A legkívül borítás műanyag szigetelő, amely véd a fizikai behatásoktól.

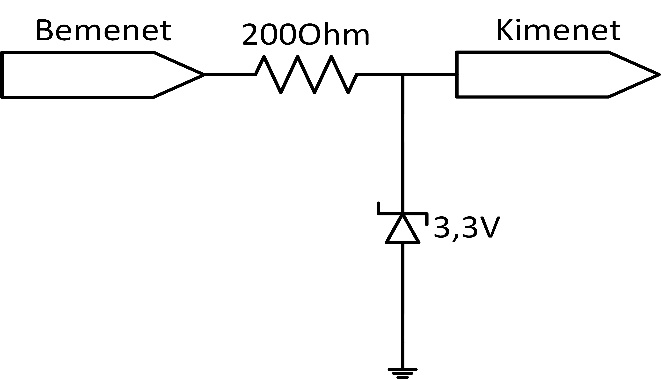


Műanyag szigetelő

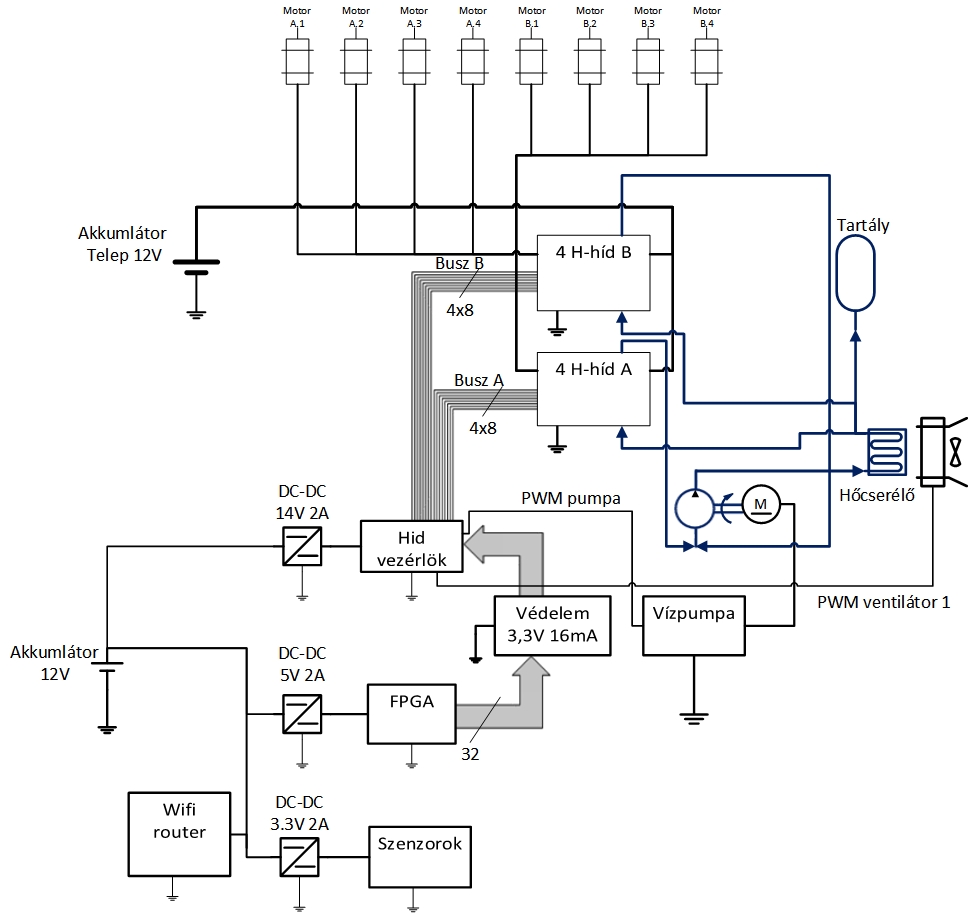
Alumínium fólia árnyékolás

Kép. 3.43 A négy Kép. 3.41 látható szalagvezeték jelenik meg a Buszvezetékben.

A Kép. 3.44 látható 32 bites buszvezeték, amely az FPGA rendszertől érkezik, és megtalálható benne a 8 motor hajtásához szükséges pwm beavatkozó jelek, amelyek az amplitúdója 3,3V. A busz szalagvezeték segítségével van kialakítva, és megtalálható benne egy védelem is amely megvédi az FPGA rendszert az esetleges visszahatásoktól. A védelem Kép. 3.43 látható, 3,3V ós zenre dióda segítségével történik, melynek feladata megakadályozza a 3,3V-ál nagyobb feszültségek az FPGA rendszerbe történő továbbhaladását. A diódával sorba van egy ellenállás is, amely az áramot korlátozza mivel az FPGA a bemenetén 16mA áramot visel el.



Kép. 3.44FPGA kimentének a védelme



Kép. 3.45 A robot energia ellátása valamint a hűtő rendszer elvi felépítése

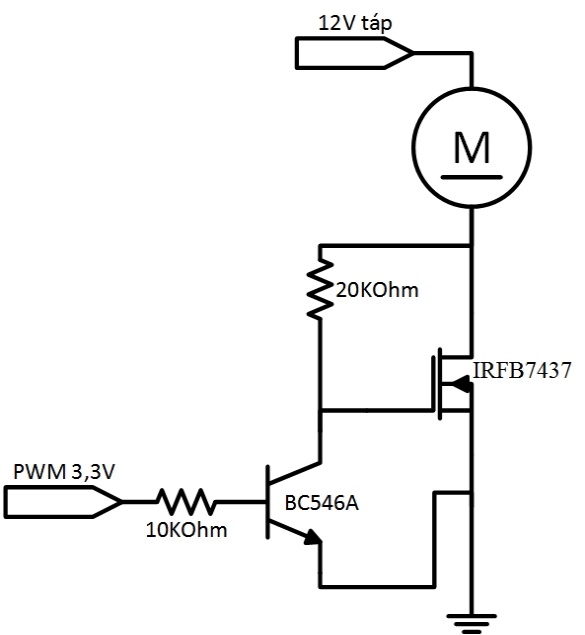
A Kép. 3.44 látható a robot energiaellátásának a terve. Az energia akumlátorokkal fog történi, rendeltetésük szerint két csoportba oszthatók: egy 12V akkumlátor gondoskodik a rendszer digitális áramköreinek az ellátásáról. A digitális elemeket DC-DC konverteren keresztül táplálom be

melyeknek a feszültsége alítható a minimális feszültség ami szükséges a konvertereknek 3V és a kimenetet állíthatjuk 3-30V-ig, és maximálisan 2A.

A wifi router modulban megtalálható a beépített konverter.

A másik energiaforrás egy több akkumlátorból álló telep lesz, amelyek párhuzamosan lesznek kapcsolva, és a H hidakat táplálják be energiával.

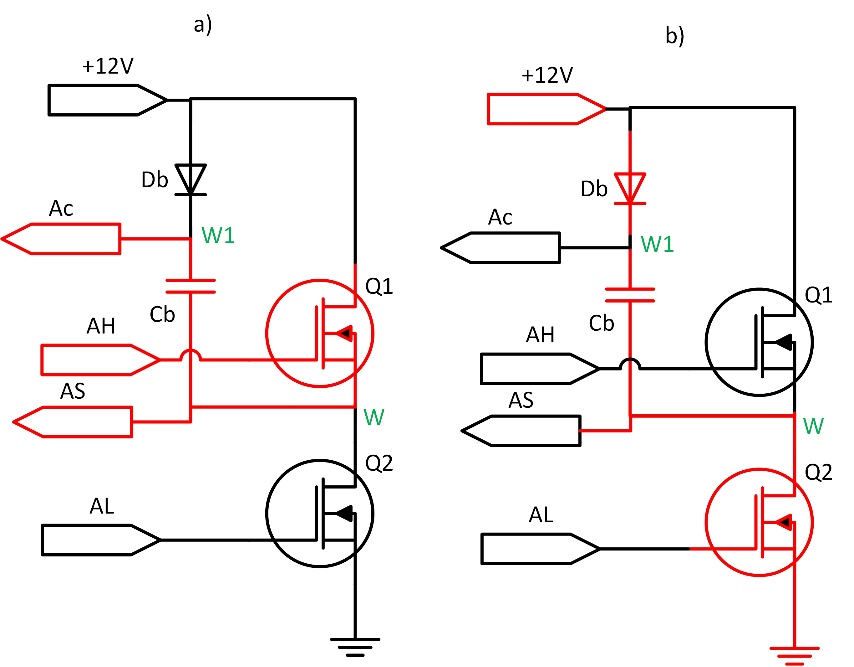
A vízpumpa és a ventilátorok motorja PWM jel segítségével van vezérelve egy N csatornás MOSFET tranzisztor segítségével, amelyet Kép. 3.45 ábra szemléltet.



Kép. 3.46 Vízpumpa és a ventilátor motorjának vezérlő teljesítmény elektronikai kapcsolása

### Bootstramp müködése

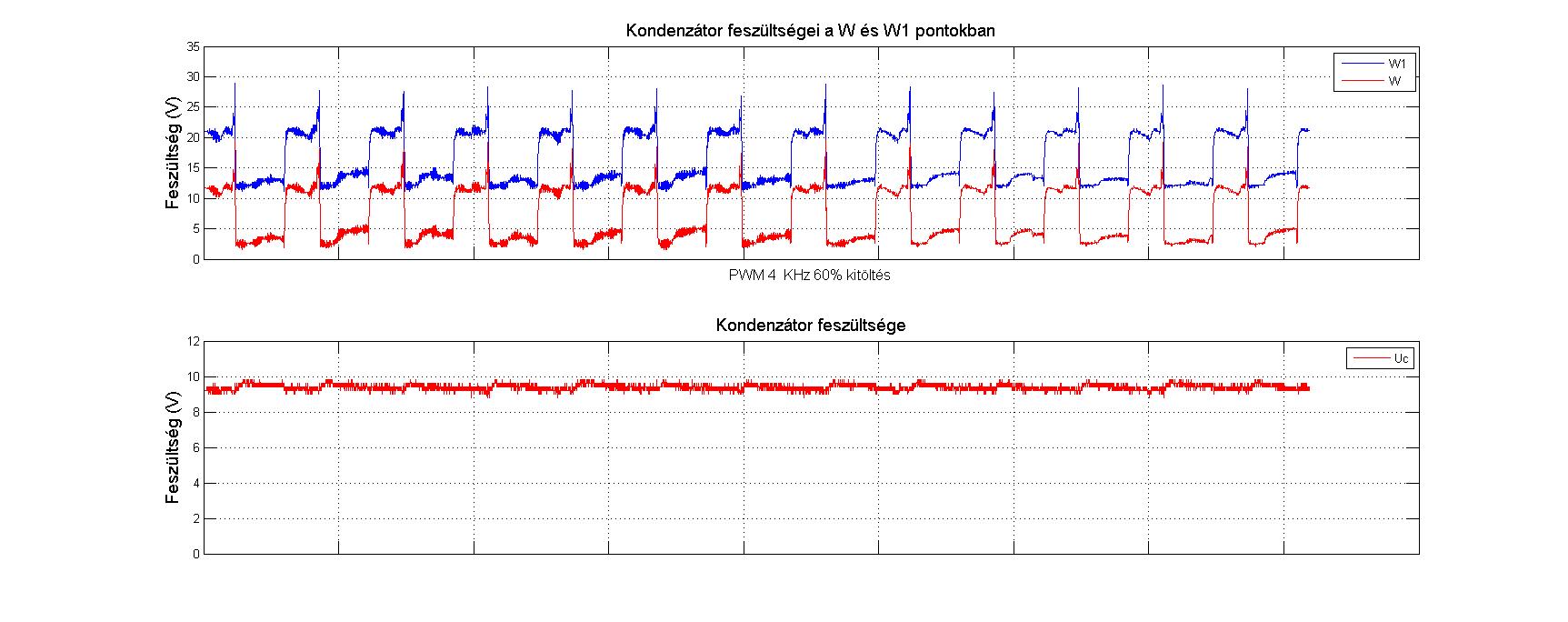
A [10] dokumentum alapján a bootstrap megoldás a Kép. 3.46 képen látható módon történik. A Cb kondenzátort töltjük fel a Db dióda segítségével azokban a pillanatokban, amikor a Q2 tranzisztor kinyitott állapotban van és a w potenciál elég alacsony ahhoz, hogy a Db dióda kinyitáson és így feltöltve majdnem 12V feszültségre a kondenzátort. A Kép. 3.39 látható T1 időintervallumban történik a kondenzátorok töltése.



Kép. 3.47 Bootstramp megoldás a felső tranzisztor Gate bemenetének a meghajtására

A Kép. 3.46 látható b) ábrán látható amint a Q2 tranzisztor tölti a Cb kondenzátort, a) képen a AS és Ac feszültségek bevanak vezetve a HIP4082 ic be amely majd az a vezérlő jel hatására rákapcsolja a Ac bemenet feszültségét a AH kimenetre.

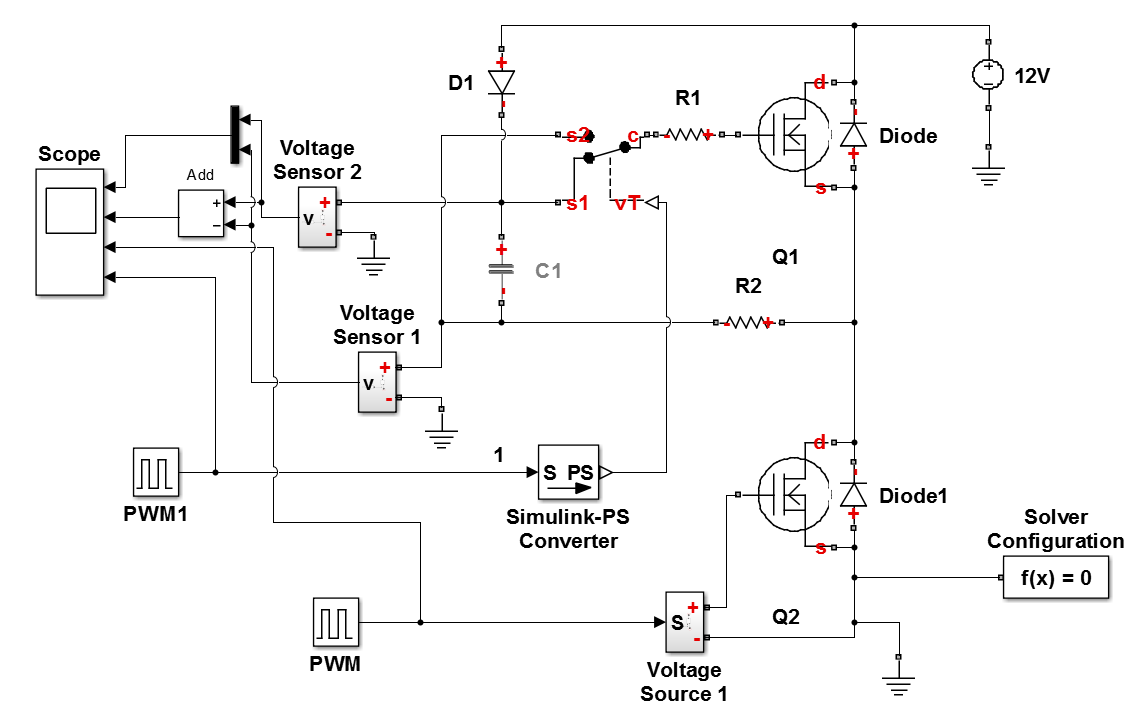
A Kép. 3.47 látható a mérése a Cb kondenzátor feszültségének, a méréseket oszciloszkop segítségével végeztem el és mentetem ki az adatokat, amelyeket majd ábrázoltam Matlab programmal.



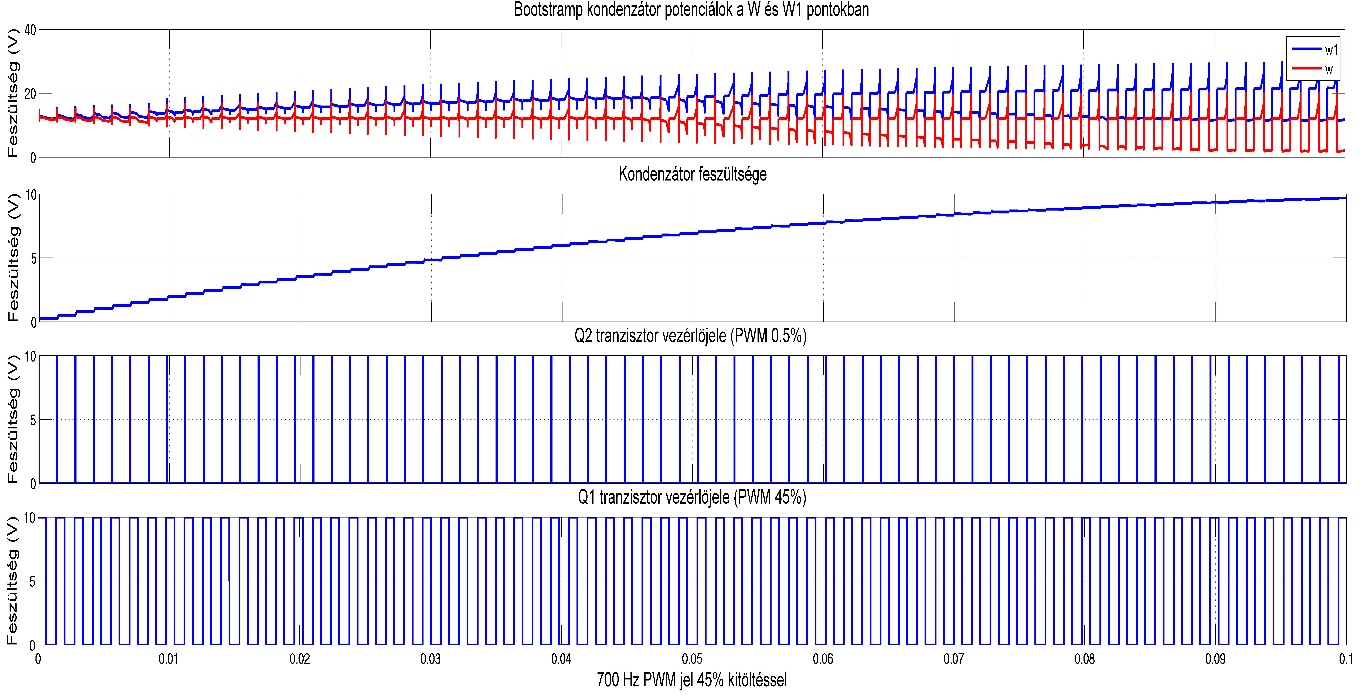
Kép. 3.48 Bootstramp kondenzátor feszültsége a W és W11 pontokban

#### Szimuláció simulink környezetben

A szimuláció során előalítottam a Kép. 3.47 látható méréseknek megfelelő környezetet. Az eredmények szerint a Kép. 3.49 látható, ha a kondenzátor kezdeti feszültsége nulla, akkor a feszültség lassan kezd el felfutni rajta, emiatt a felső Q1 tranzisztor nem nyit ki teljesen és ezért veszteségek jelentkeznek rajta. A leg optimálisabb az lenen, ha egyszer feltöltenénk a kondenzátort és csak azután kezdenénk el a motor indítását. Amelyet úgy érhetünk el ha a Kép. 3.38 kapcsolási rajzot nézve a mindkét bemenetre 0V adunk, mert a tranzisztorokból kialakított tagadó kapu megtagadja és így a hídba mindkét alsó tranzisztor kinyitott állapotba kerül.



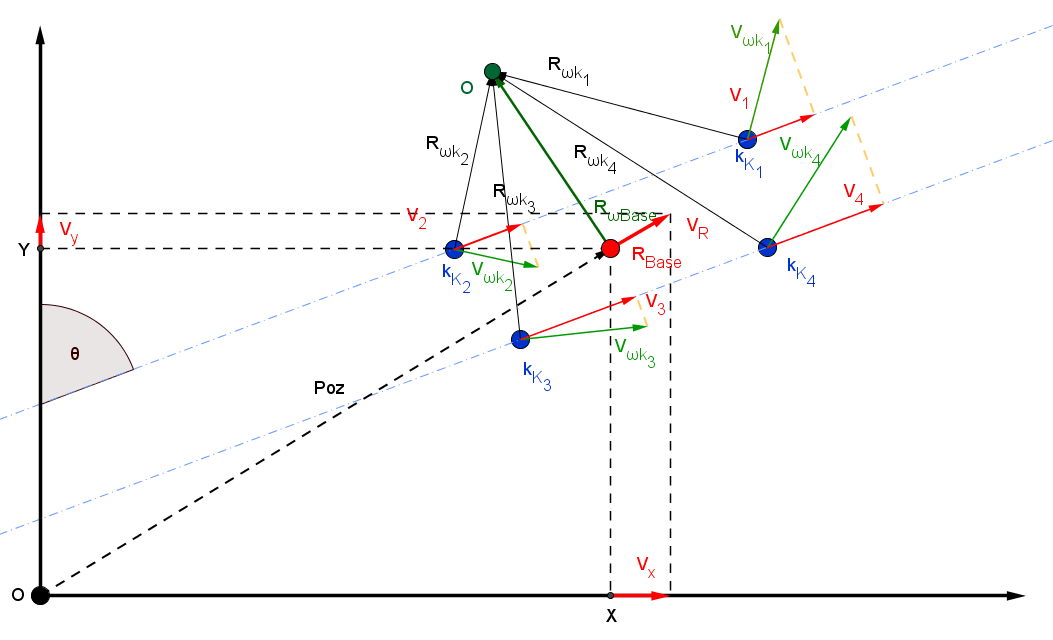
Kép. 3.49 Bootstramp működése, szimulációs modell MATLAB/SIMULINK környezetben



Kép. 3.50 Szimulációs eredmények Bootstramp

## Robot Modell

A [11] cikkben leírja egy négykerekű mobilis robot kinematikus és dinamikai moteljét. A rendseren hasonlok, épen lehetne alkalmazni a tárgyalt modellt annyi eltéréssel, hogy ebben az esetben az azok a pontok, amelyekben a robot érintkezik a talajjal nem szimmetrikusak.



Kép. 3.51 Robot kerekek sebsége és a robot mozgásának viszonya

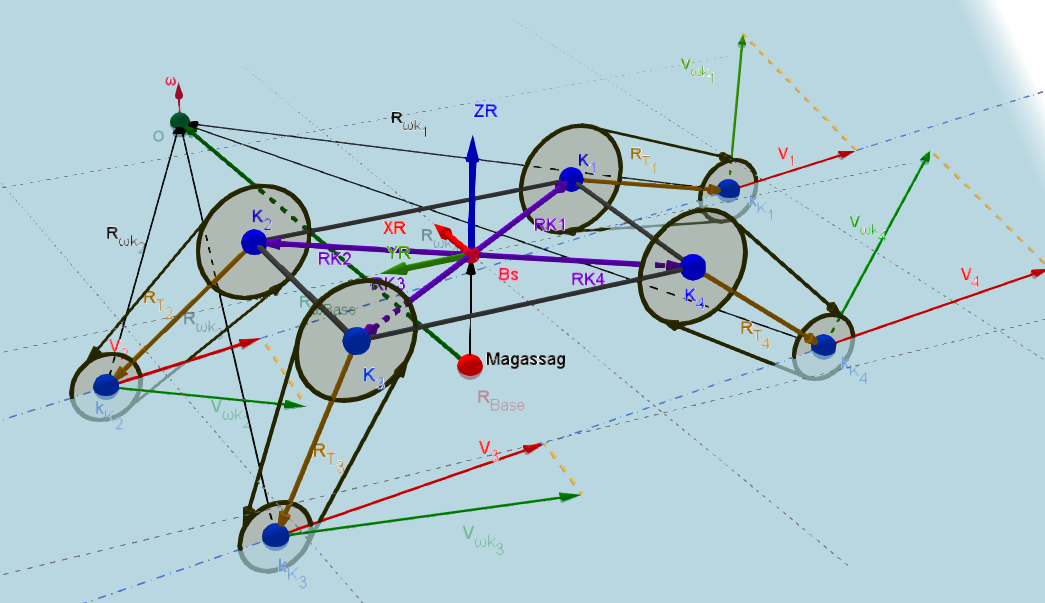
**Jelölések**:

Szeretnénk, ha a robotunk egy adott körpályát írna le egy pont körül egy adott sebességgel. Jelen esetben O pont körül és, szögsebességgel.

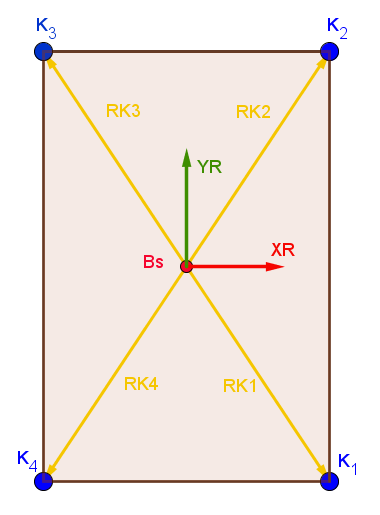
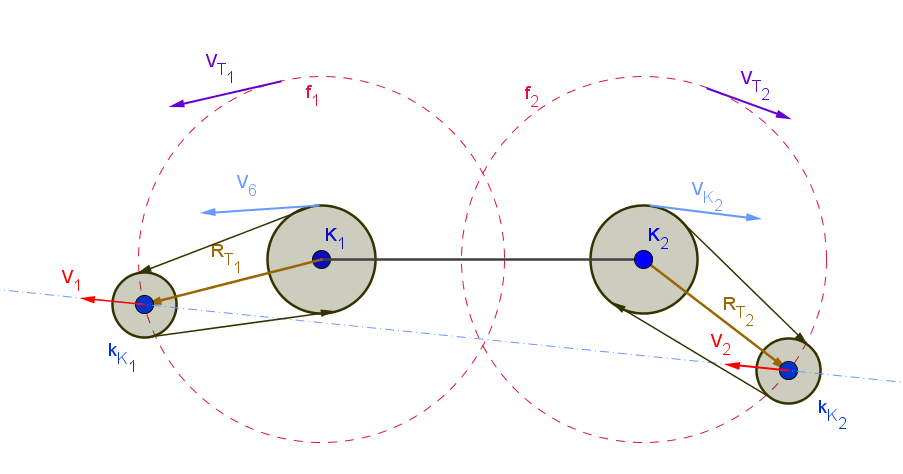
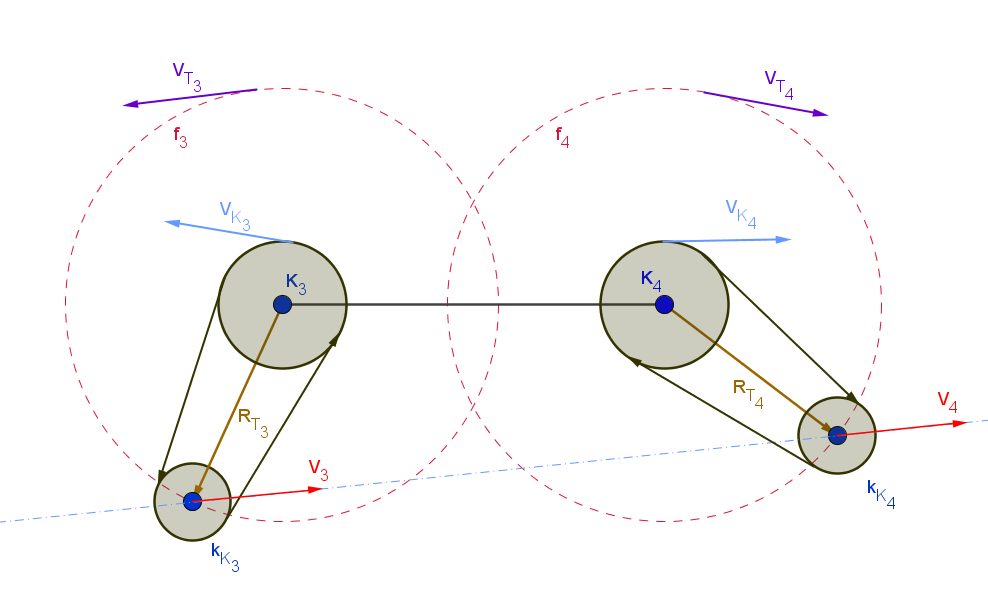
Az **Error! Reference source not found.** alapján felírhatók a következő összefüggések a vektorok között:

Ha ismerjük a , és kitudjuk számolni a sebességeket. Tudva hogy a rendszer csak az YR tengelye mentén tud sebességet generálni így:

A robot síkbeli mozgását



Kép. 3.52 Robot 3D vektorábrája



Kép. 3.53 Oldalnézetek és Felülnéztet, jelölések szemléltetése

# Robot Mechanikai Felépítése

A robot alapját képezi egy masszív váz, amely könnyű fémprofilokból áll össze és hegesztésekkel rögzítjük egymáshoz, az elemeket. A váz és az egész rendszer szimmetrikus két tengelyre nézve is ezért a továbbiakban csak a rendszer negyedét részletezzük. A 7.1 képen látható a rendszer vázának Autodesk Inventorban elkészített terve.

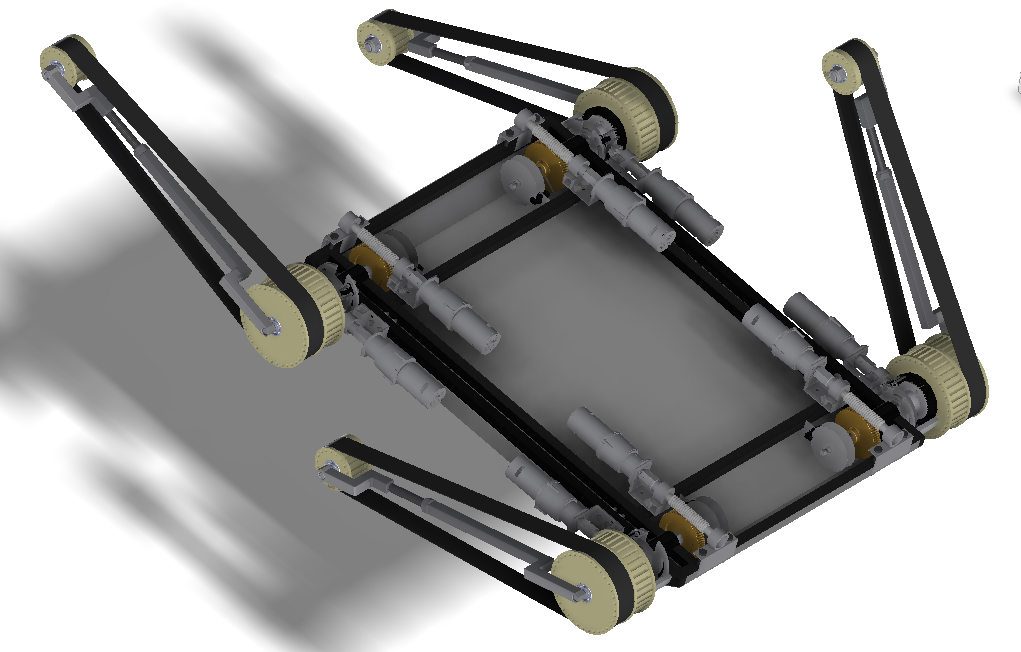
DC motrok betáplálási feszültsége: 12V, maximális terhelés alatt 10A áramot is felvehet.

A 7.1 képen látható Kup kerék áttételen keresztül hajtjuk meg a lánctalpat, a talpak mozgatására orsós áttételt használtam több okból is:

* Nagy nyomaték kifejtésére képesek, jelen esetben 40:1 az áttételi arány. Egy motor által kifejtett nyomaték névlegesen 30-40Nm között van így a karok végén tengelyre kifejtett nyomaték elérheti a 160Nm-t.
* A terhelés nem képes visszafele hajtani, mert a mechanizmus lezárja, így akár a motort teljesen ki is kapcsolhatjuk, ha nem szeretnénk megváltoztatni a talp pozícióját.

A forgó talpak 360 fokban körbeforgathatók, a NagyKerék tengelye körül.

A NagyKerék két csapágy segítségével illesztve van a talp tengelyéhez, így a kerék szabadon fut a tengelyen. A nyomatékot a NagyKerék-ről a KisKerék-re bordásszíj segítségével adódik át. A NagyKerék-re rögzítve van egy fogaskerék, amelyet a hajt meg egy másik fogaskeréken keresztül. A a csiga áttételen keresztül változtatja a lánctalpak szögét. A mechanikai rendszer terve az alábbi ábrákon van szemléltetve.



Kép. 4.1 Robot vázának Inventoros 3D Képe

ForgóTalp1

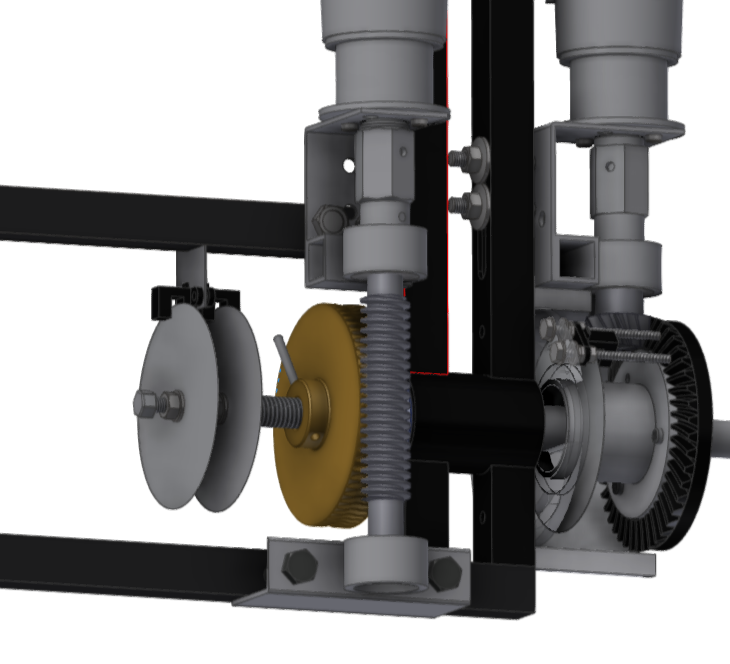
ForgóTalp4

ForgóTalp3

ForgóTalp2

KisKerék

NagyKerék



Null átmenti érzékelő tárcsája

Inkrementális szenzor tárcsa

# Elért eredmények, magvalósítások:

* Autodesk Invnetor megterveztem a mechanikai rendszert
* A mechanikai rendszer megépítettem az Inventoros terv alapján
* Inkrementális tárcsa tervezése.
* Hardveres pozíció Szabályozó megvalósítása
* Hardveres PID szabályozó megvalósítása
* DC motor mérőstand megépítése
* Hardver alapkonfigurációs kialakítása Xilinx Platform Studio-val a két fejlesztőrendszeren
* A beágyazott processzorokon futó programok megvalósítása Xilinx Software Development Kit eszközzel
* Ipmag generálása (Sebesség+Pozíció Szabályozó)
* Az egyes alegységek Simulink System generatorban való szimulációja

# Bibliography

|  |  |
| --- | --- |
| [1] | D. K. K. D. R. P. S. Prof. Vikas Gupta, "Efficient FPGA Design and Implementation of Digital PID Controllers in Simulink," 2013. |
| [2] | M. Lőrinc, Irányítástechnika, Kolozsvár: Scientia, 2009. |
| [3] | xilinx, „http://www.xilinx.com/,” [Online]. Available: http://www.xilinx.com/support/documentation/sw\_manuals/xilinx14\_5/sysgen\_gs.pdf. [Hozzáférés dátuma: 26 01 2015]. |
| [4] | R. T. R. G. Rajesh Nema, „Design & Implementation of FPGA Based On PID Controller,” 2013. |
| [5] | „http://www.ms.sapientia.ro/elektronika,” [Online]. Available: http://www.ms.sapientia.ro/elektronika/fileok/jelerzekelok/szt\_lab08\_inkrementallis\_ado.pdf. |
| [6] | A. G. K. Krisztián LAMÁR, „IMPLEMENTATION OF SPEED MEASUREMENT FOR ELECTRICAL DRIVES EQUIPPED WITH QUADRATURE ENCODER IN LabVIEW FPGA,” 2013. |
| [7] | M. Lőrinc, „http://www.ms.sapientia.ro/~martonl/index.htm,” [Online]. Available: http://www.ms.sapientia.ro/~martonl/Docs/Labs/IRI\_L1.pdf. [Hozzáférés dátuma: 11 6 2015]. |
| [8] | I. Inc, „www.olimex.com,” [Online]. Available: https://www.olimex.com/Products/Modules/Sensors/MOD-MPU6050/resources/RM-MPU-60xxA\_rev\_4.pdf. [Hozzáférés dátuma: 11 6 2015]. |
| [9] | intersil, „http://www.intersil.com/,” [Online]. Available: http://www.intersil.com/content/dam/Intersil/documents/hip4/hip4082.pdf. [Hozzáférés dátuma: 11 6 2015]. |
| [10] | S. labs, „http://www.silabs.com/,” [Online]. Available: http://www.silabs.com/Support%20Documents/TechnicalDocs/AN486.pdf. [Hozzáférés dátuma: 11 6 2015]. |
| [11] | D. P. I. J. K. Kozłowski, „Modeling and control of a 4-wheel skid-steering”. |
| [12] | M. Trojnacki, „Dynamics Model of a Four-Wheeled Mobile Robot for Control Applications – A Three-Case Study,” in *Intelligent Systems'2014*, Springer, 2014, p. 111. |
| [13] | L. Lajos, „http://www.ms.sapientia.ro/,” [Online]. Available: https://moodle.sapidoc.ms.sapientia.ro/pluginfile.php/2771/mod\_resource/content/1/Losonczi\_Lajos\_-\_Analog\_Aramkorok\_3\_V1.pdf. [Hozzáférés dátuma: 11 6 2015]. |

# FÜGGELÉK