**Sapientia EMTE**

**Műszaki és Humántudományok Kar, Marosvásárhely**

**Villamosmérnöki Tanszék**

Kültéri mobilis robot tervezése és megvalósítása FPGA alapú hardver erőforrás alkalmazásával

Szakirányító tanár: Végzős hallgató:

Dr. Brassai Sándor Tihamér Gábor Szabolcs-László

adjunktus Automatizálás IV. év

**2015**

[Ábrák,táblázatokjegyzéke 4](#_Toc422064087)

[1 Bevevezető 7](#_Toc422064088)

[2 BIBLIOGRÁFIAI TANULMÁNY 9](#_Toc422064089)

[2.1 Hasonló FPGA fejlesztőrendszeren megvalósított PID szabályzók 9](#_Toc422064090)

[2.2 Inkrementális érzékelő 9](#_Toc422064091)

[2.3 Szögsebesség mérése FPGA segítségével 10](#_Toc422064092)

[2.4 Egyenáramú motorok 10](#_Toc422064093)

[3 A RENDSZER TERVEZÉSE 12](#_Toc422064094)

[3.1 SZABÁLYOZÓK: 12](#_Toc422064095)

[3.1.1 Diszkrét Hardveres PID szabályozó 12](#_Toc422064096)

[3.1.1.1 Megvalósítás System Generátorban 13](#_Toc422064097)

[3.1.1.2 Simulink szimulációs eredmények 14](#_Toc422064098)

[3.1.1.3 Q paraméterek számolása alapján. 16](#_Toc422064099)

[3.1.2 Mintavételi periódus jelének A generálása 16](#_Toc422064100)

[3.1.3 Pozíció Szabályzása 17](#_Toc422064101)

[3.1.3.1 A szabályozó felépítése: 18](#_Toc422064102)

[3.1.3.2 Szabályozó szimulálása 18](#_Toc422064103)

[3.1.4 Hardveres mérések 19](#_Toc422064104)

[3.1.4.1 Dc motor sebesség szabályzása mérőstandon 19](#_Toc422064105)

[3.1.4.2 DC motor pozíció szabályzása mérőstandon 21](#_Toc422064106)

[3.2 Sebesség és pozíció szabályozót tartalmazó IP mag generálása System Generator-ban 21](#_Toc422064107)

[3.3 Szenzorok 26](#_Toc422064108)

[3.3.1 InkrementálisÉrZékelő 26](#_Toc422064109)

[3.3.1.1 Optikai inkrementális vevő felépítése 26](#_Toc422064110)

[3.3.2 Inkrementális érzékelő jeleinek a feldolgozása FPGA áramkör segítségével 27](#_Toc422064111)

[3.3.2.1 Szimuláció System Generatorban 28](#_Toc422064112)

[3.3.2.2 Pozíció mérése Inkrementális adó segítségével 29](#_Toc422064113)

[3.3.2.3 Szögsebesség mérése Inkrementális adó segítségével 30](#_Toc422064114)

[3.4 MPU-6050 giroszkóp és gyorsulásmérő 33](#_Toc422064115)

[3.5 Beavatkozó elemek: 33](#_Toc422064116)

[3.5.1 Pwm Generátor megvalósítása FPGA áramkörön System Generator környezetben. 33](#_Toc422064117)

[3.5.1.1 Megvalósítás 33](#_Toc422064118)

[3.6 Elektronika 35](#_Toc422064119)

[3.6.1 Digitális Elektronika 35](#_Toc422064120)

[3.6.2 FPGA Rendszer Felépítése 35](#_Toc422064121)

[3.6.2.1 Zybo FPGA fejlesztőlap 36](#_Toc422064122)

[3.6.2.2 Spartan3e FPGA fejlesztőlap 38](#_Toc422064123)

[3.6.2.3 Kommunikációs protokollok 39](#_Toc422064124)

[3.6.3 Feladatok Elosztása 42](#_Toc422064126)

[3.6.3.1 Zybo fejlesztőlap 42](#_Toc422064127)

[3.6.3.2 Spartan fejlesztőlap 42](#_Toc422064128)

[3.7 GUI felépítése 43](#_Toc422064129)

[3.8 Teljesítmény elektronika 43](#_Toc422064130)

[3.8.1 Bootstrampmüködése 49](#_Toc422064131)

[3.8.1.1 Szimuláció simulink környezetben 50](#_Toc422064132)

[3.9 Robot Modell 52](#_Toc422064139)

[4 Robot Mechanikai Felépítése 55](#_Toc422064140)

[5 Elért eredmények, magvalósítások: 57](#_Toc422064141)

[6 Következtetések: 57](#_Toc422064145)

[7 Bibliography 59](#_Toc422064146)

[8 FÜGGELÉK 60](#_Toc422064147)

Ábrák,táblázatokjegyzéke

[Kép. 2.1Inkrementális érzékelő jelek 10](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064026)

[Kép. 3.1 a Pozíció és a sebesség szabályzási hurok elvi strukturális felépítése 13](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064027)

[Kép. 3.2 Állapot automata, amely leírja a Diszkrét PID szabály ózót 14](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064028)

[Kép. 3.3 A PID felépítése System Generatorban 15](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064029)

[Kép. 3.4 PID Simulink szimulációs model 15](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064030)

[Kép. 3.5 Szimulációs eredmény amely tükrözi a konstans bementre a számolási lépéseket 16](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064031)

[Kép. 3.6 PID minimális periódusa 17](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064032)

[Kép. 3.7 Mintavételi taktust generáló modul 17](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064033)

[Kép. 3.8 Szimulációs eredmények mintavételi jelgenerátor. 18](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064034)

[Kép. 3.9 A Pozíció szabályozó System generátoros felépítése 19](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064035)

[Kép. 3.10 a pozíció szabályzás moduláris felépítése System Genrator környezetben 20](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064036)

[Kép. 3.11 a pozíció szabályozó bemenete (pozíció hiba), valamint a Szabályozó kimenti jele. 20](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064037)

[Kép. 3.12 Sebesség szabályozás PID szabályzóval. 21](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064038)

[Kép. 3.13 Sebesség szabályozás PID szabályzóval. 21](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064039)

[Kép. 3.14 Pozíció szabályzása a Kép. 3.9 látható kialakításban. 22](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064040)

[Kép. 3.15 Sebesség és pozíció szabályózást tartalazó Ipmag System generátoros felépítése 24](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064041)

[Kép. 3.16 Pozíció Szab modul belső felépítése a Kép. 3.14 25](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064042)

[Kép. 3.17 Sebesség Szab modul felépítése a Kép. 3.14 26](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064043)

[Kép. 3.18 Optikai inkrementális vevő felépítése és elhelyezése 27](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064044)

[Kép. 3.19 Érzékelő tranzisztorok elhelyezése 27](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064045)

[Kép. 3.20 Idődiagram a Tárcsa paraméterei függvényében 28](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064046)

[Kép. 3.21 Rések és az Érzékelők közti kapcsolat 28](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064047)

[Kép. 3.22 Inkrementális Jelfeldolgozó modul1 érzékelő modul belső felépítése 29](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064048)

[Kép. 3.23 Inkrementális érzékelőtől érkező jelek átalakító irány és impulzus jelekre 29](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064049)

[Kép. 3.24 Szimulációs eredmények a lehetséges bemenetekről az Black Box1 modulba 30](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064050)

[Kép. 3.25 Inkrementális adóval mért pozíció, szimulációs modellje SytemGeneratorban 30](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064051)

[Kép. 3.25 Inkrementális adóval mért pozíció, szimulációs modellje SytemGeneratorban 31](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064052)

[Kép. 3.26 Sebesség mérő modul felépítése 31](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064053)

[Kép. 3.27Dc motor sebességének mérése Fpga lapon 32](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064054)

[Kép. 3.28Dc motor Sebesség mérése FPGA rendszeren, System generatorban megvalósítva 32](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064055)

[Kép. 3.31.a8ms 33](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064056)

[Kép. 3.31.c4ms 33](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064057)

[Kép. 3.31.b80ms 33](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064058)

[Kép. 3.32 A PWM generátor System Generátorban megvalósított szerkezete 35](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064059)

[Kép. 3.33 a PWM generátor bemenő, kimenő illetve néhány belső jele (Scope1) 36](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064060)

[Kép. 3.34 rendszer elvi felépítése 36](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064061)

[Kép. 3.35 Kommunikációs csomagok és az FPGA áramkörökbe programozott modulok elvi felépítése 37](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064062)

[Kép. 3.35 Kommunikációs csomagok és az FPGA áramkörökbe programozott modulok elvi felépítése 38](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064063)

[Kép. 3.36 ZYBO Core0 program folyamat árbája 39](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064064)

[Kép. 3.37 Spartan3e500, microblaze szoftver Folyamat ábrája 39](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064065)

[Kép. 3.38Hip4082 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 44](#_Toc422064066)

[Kép. 3.39 Két hídvezérlő árak kör kapcsolási rajza HIP4082 integrált áramkörrel megvalósítva 45](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064067)

[Kép. 3.42 PWM és a tranzisztorok kapcsolása 46](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064068)

[Kép. 3.40 Nem invertáló erősítő forrás [13] 46](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064069)

[Kép. 3.41 Dupla hídvezérlő áramkor vezérlő jelei JP2 csatlakozó a Kép. 3.38-n. 47](#_Toc422064070)

[Kép. 3.42 H híd tranzisztorainak a Gate vezetékei 47](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064071)

[Kép. 3.44FPGA kimentének a védelme 48](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064072)

[Kép. 3.43 A négy Kép. 3.41 látható szalagvezeték jelenik meg a Buszvezetékben. 48](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064073)

[Kép. 3.45A robot energia ellátása valamint a hűtő rendszer elvi felépítése 49](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064074)

[Kép. 3.46 Vízpumpa és a ventilátor motorjának vezérlő teljesítmény elektronikai kapcsolása 50](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064075)

[Kép. 3.46 Vízpumpa és a ventilátor motorjának vezérlő teljesítmény elektronikai kapcsolása 50](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064076)

[Kép. 3.47Bootstramp megoldás a felső tranzisztor Gate bemenetének a meghajtására 50](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064077)

[Kép. 3.48Bootstramp kondenzátor feszültsége a W és W11 pontokban 51](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064078)

[Kép. 3.49 Bootstramp működése, szimulációs modell MATLAB/SIMULINK környezetben 52](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064079)

[Kép. 3.50 Szimulációs eredmények Bootstramp 52](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064080)

[Kép. 3.47Bootstramp megoldás a felső tranzisztor Gate bemenetének a meghajtására 53](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064081)

[Kép. 3.48Bootstramp kondenzátor feszültsége a W és W11 pontokban 53](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064082)

[Kép. 3.49Robot kerekek sebsége és a robot mozgásának viszonya 53](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064083)

[Kép. 3.50 Robot 3D vektorábrája 54](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064084)

[Kép. 3.51 Oldalnézetek és Felülnéztet, jelölések szemléltetése 55](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064085)

[Kép. 4.1 Robot vázának Inventoros 3D Képe 56](C:\\Users\\laca\\Desktop\\Allamvizsga\\Dolgozat\\Allamvizsgadolgozat\\GaborSzabolcsLaszlo_v3.docx" \l "_Toc422064086)

[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 AutodeskInventor-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 processzort, a szabályzók 12V DC motor sebességét vagy pozícióját szabályozzák. A Microblaze processzor feladata az adatok fogadása 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 AutodeskInventor 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 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 sikerültmérni 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 vannak meghatározva, a hardverben kívülről nem lehet megadni, ami a hangolás szemszögéből nem előnyös.

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

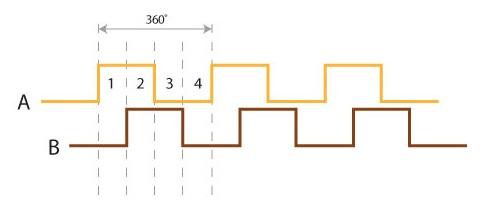
[2]

A Q paraméterek konstansak, és a rendszer kimenete kiszámítható 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,amelyekcsővezetékszerűenvannak 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 ketteskomplemens 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..

## 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] dolgozat két sebességmérő módszertemlít 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 határozza 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 adótól 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 erőforrást 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 motorok

Az egyenáramúmotorokat használják általában nagy pontosságotigénylő hajtások megvalósítására, kisebb teljesítményűmotorokpermanens mágnesből készült állórésszel rendelkeznek, és tekercselt forgórésszel. 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 Savart 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 állapotteres motelje, a választott állapotok:

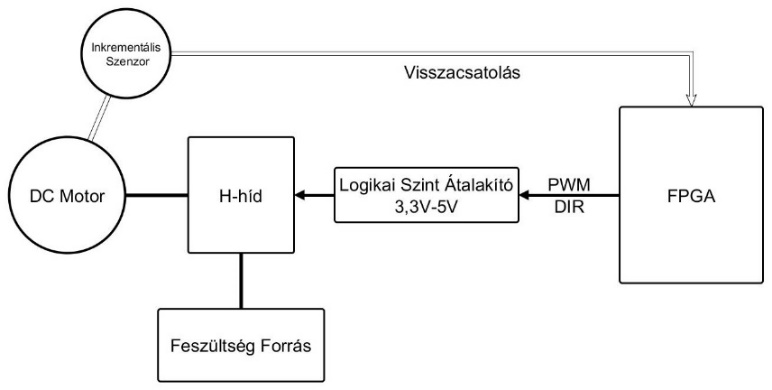
Ahol - állapotok, és a - bemenetek.

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 választottamKép. 3.1. A feszültségben vezéreltDc 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 rekurzív egyenlete a következő:

[2]

Az általam elkészített PID szabályozó hardveresen van megvalósítva FPGA áramkörben, a minél kisebb mintavételezési periódus elérése céljából. A fent látható összefüggésekalapján egy adat utas automatátterveztem, 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, mintavételezésiperiódus, valamintproporcionális 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 alattlép át.

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

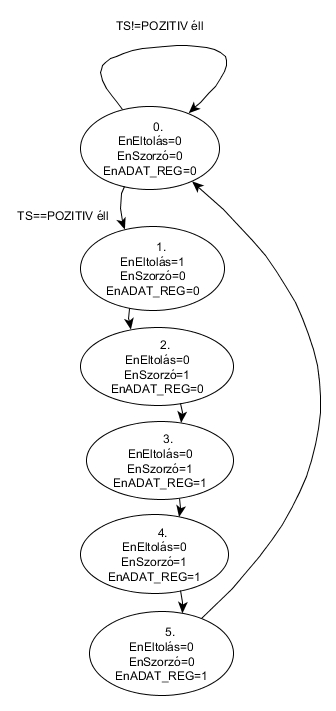
Az automata mindaddig 0. állapotban van, amíg a TS szignálon nem érkezik egy felfutó él.A1. állapotban végrehajtja az regiszterek eltolását, azáltal hogy jelet logikai 1 re állítja, vagyis, és regiszterbe betölti az aktuális bemeneti értéket.

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 számlálótúgyállítjuk, 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.

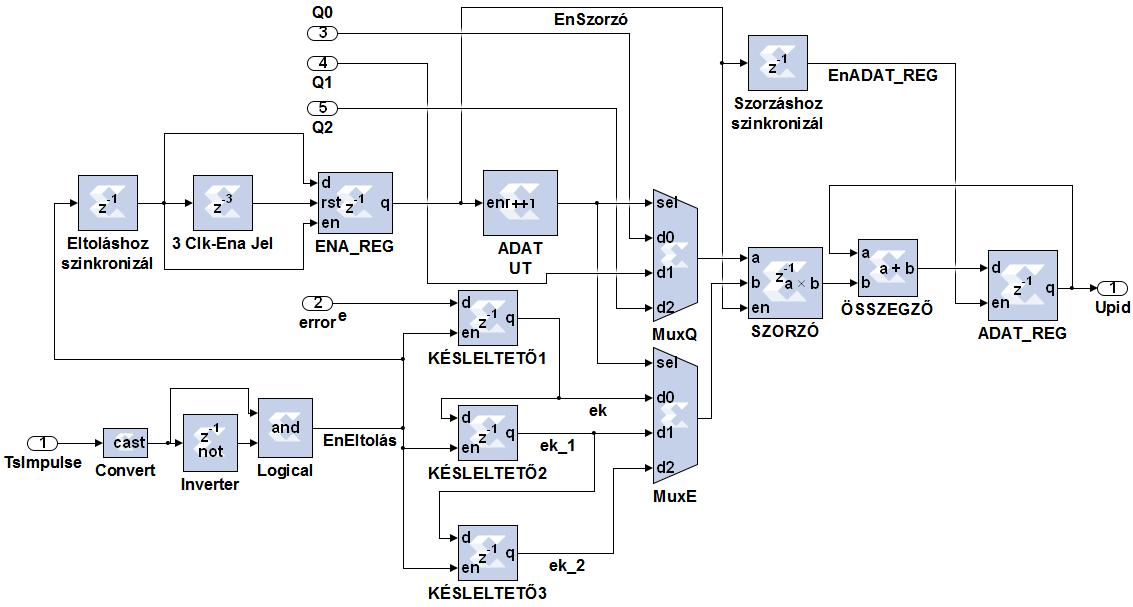
Kimenetek: 17bit előjeles egész érték.A MUXQ a Q paraméterek kiválasztásáért felelős, valamint a MUXE az időben késleltetett bemeneti értékek kiválasztásáért felelős. A Kép. 3.3a„*SZORZÓ*” modul a két szelekciós multiplexertől kapott értéket összeszorozza, aztán hozzáadja az „*ADAT\_REG*” regiszter értékéhez.

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 akövetkező regiszterbe csúszik át az érték. A KÉSLELTETŐ1 regiszterbe kerül mindig az aktuális mintavételezett hiba értéke.

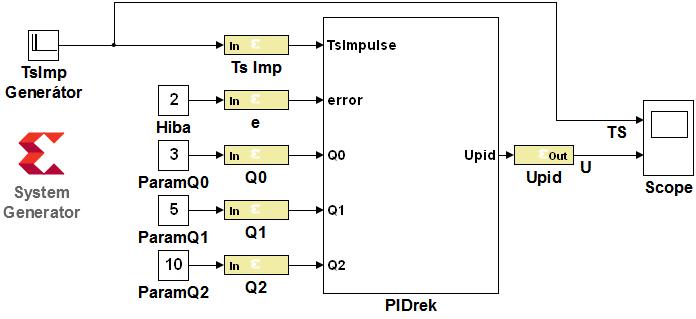
Az „*ADAT UT”* számláló csak a regiszterek elcsúsztatása után indul el, amelyet „*Eltoláshoz Szinkronizál*” késleltető elem valósít meg. Az „ENA\_REG” és a „3 Clk-Ena Jel” egyűt valósítják meg a három órajelig tartó logikai engedélyező jelet, amely a számlálót indítja el.

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 figyeljükegyazon jel előbbiperiódusban 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.

#### Simulink szimulációs eredmények



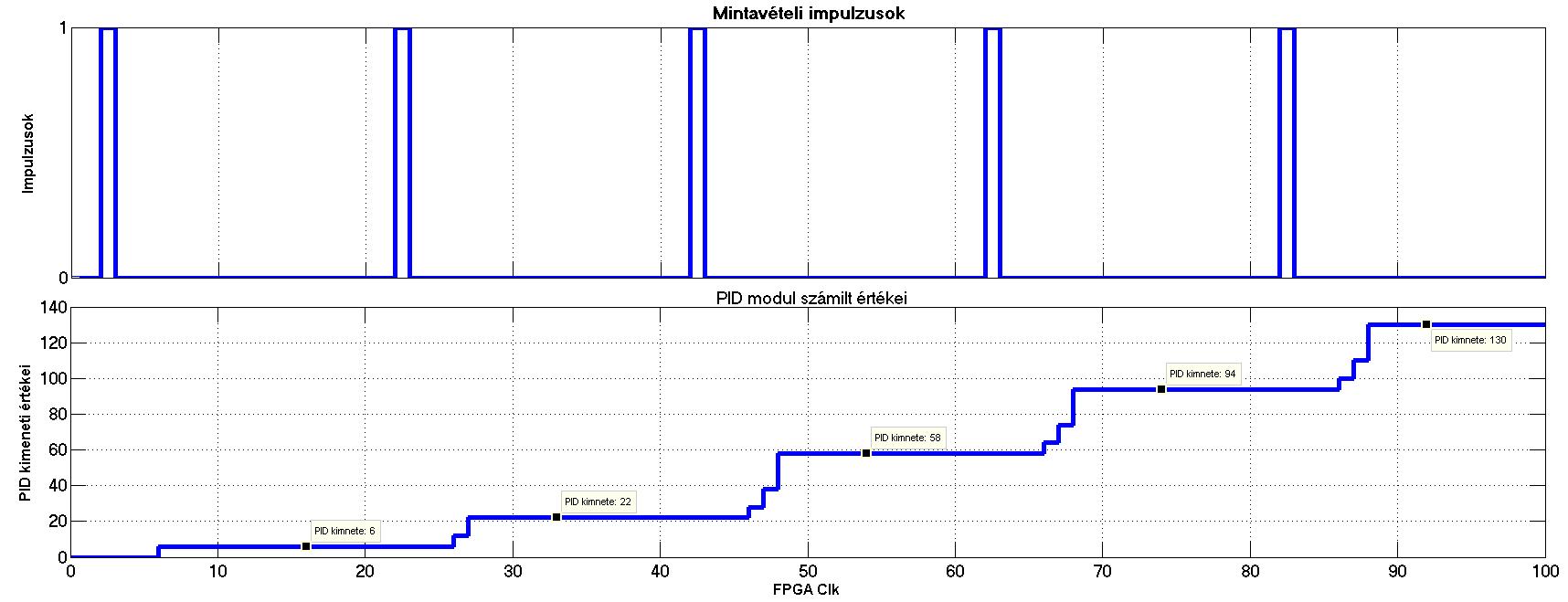
Kép. 3.4 PID Simulink szimulációs model

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 XilinxSytem 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.



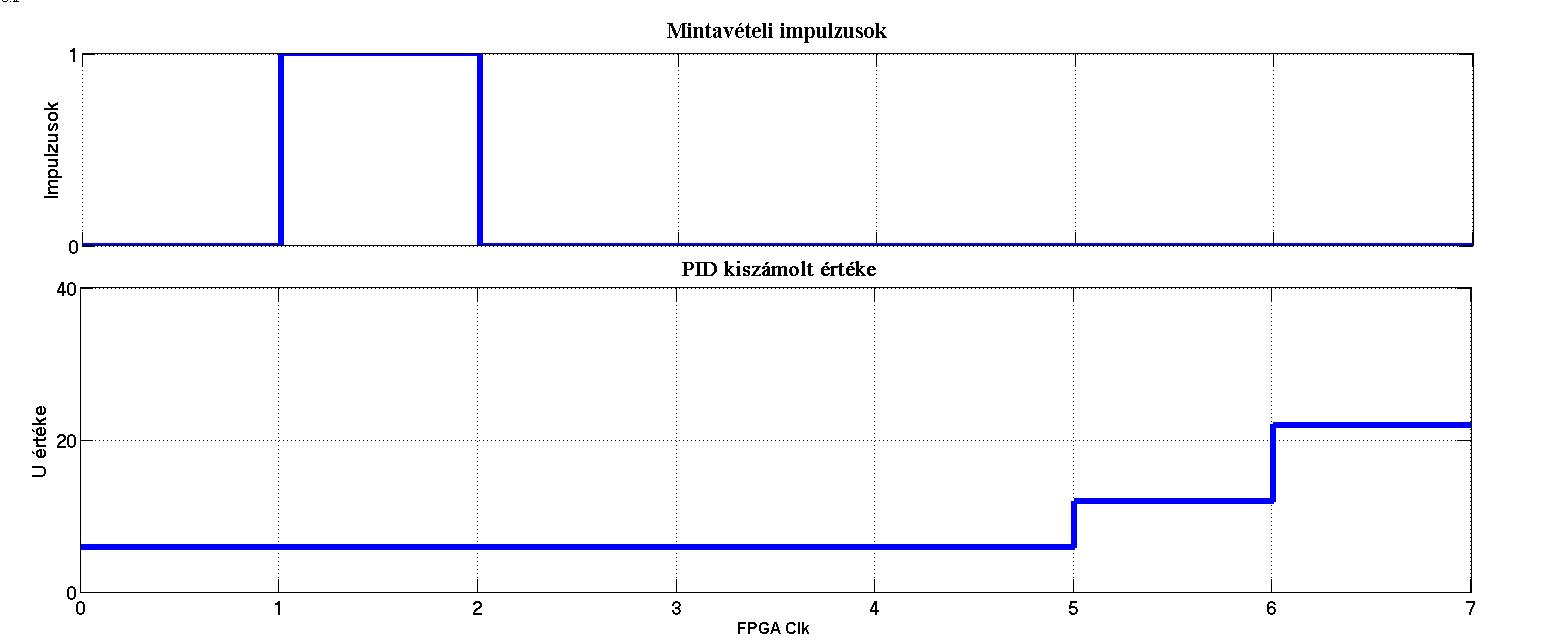
Kép. 3.5 Szimulációs eredmény amely tükrözi a konstans bementre a számolási lépéseket

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 ellenőrzésére

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.

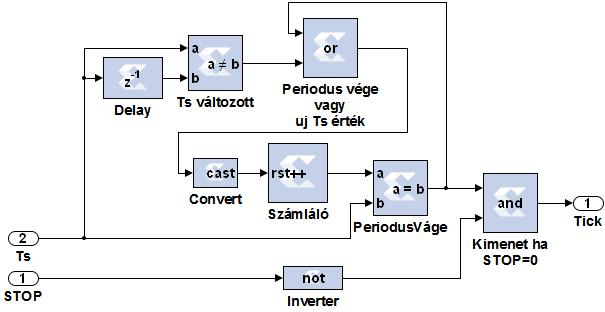
Adottak az összefüggések, látható hogy megjelenek a nevezőben így fennáll annak a veszélye, hogy 0-val való osztás történik. Ezért soha ne válaszuk a –t 0-nak, de lehet egy nullához közeli pozitív szám. A mindig nagyobb, mint nulla.

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

### Mintavételi periódus jelének A generálása

Az Kép. 3.7 Mintavételi taktust generáló modulbelső felépítése. A modulban a „*Számláló*” 32bites számláló az FPGA órajeléreszámol, ésaz 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 „*Tsváltozott*” 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.

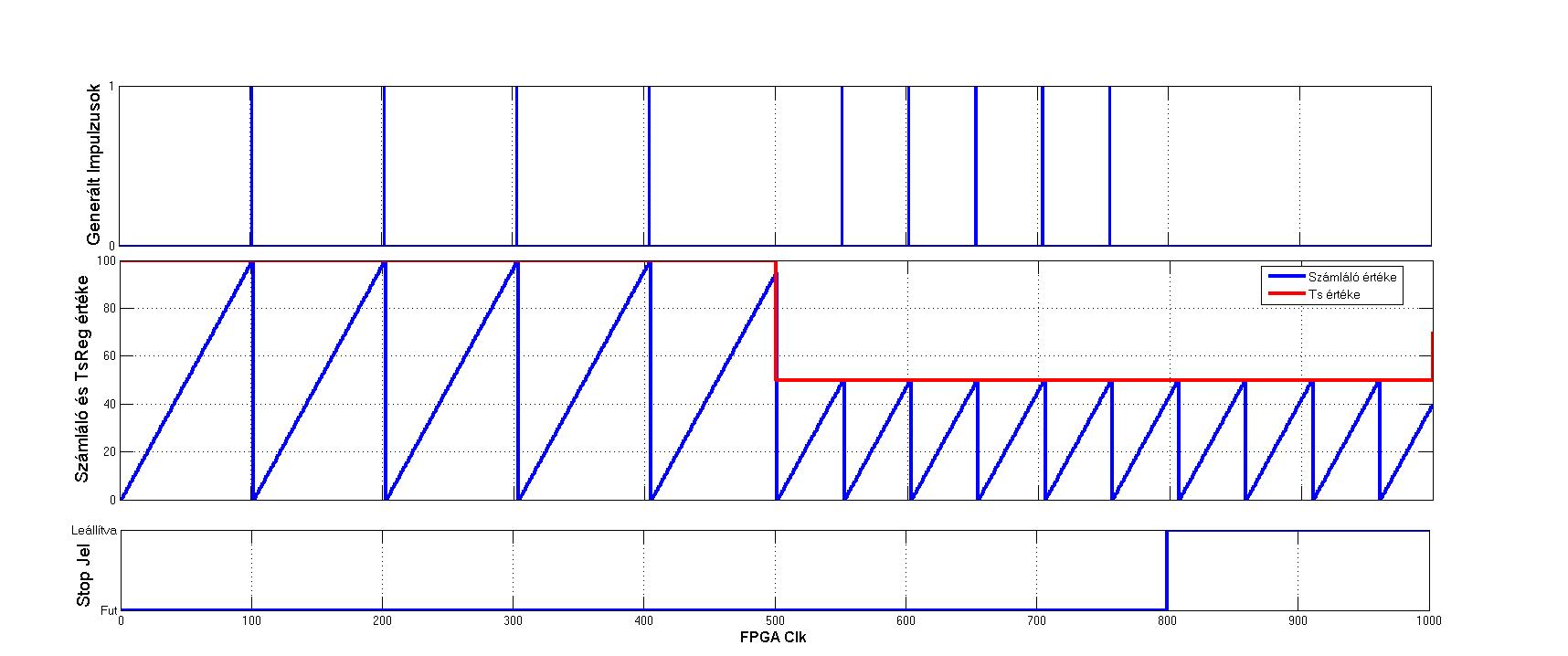
Kép. 3.7 Mintavételi taktust generáló modul



Az Kép. 3.8látható, 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.

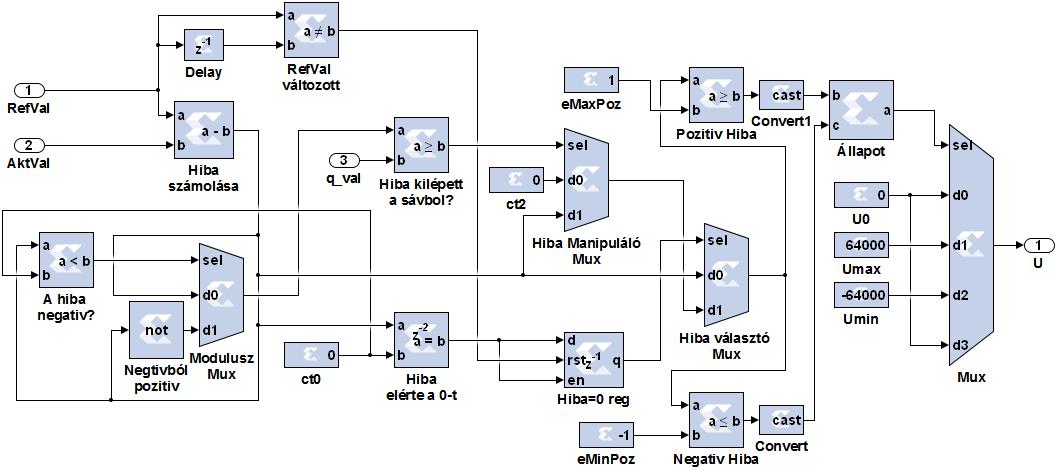
A „*Hiba Számolása*” modul végzi a hiba kiszámolását az aktuális és az előirt pozícióból. A hiba lehet negatív is ezért „*A hiba negatív?”* Komparátor segítségével eldöntjük, hogy negatív vagy pozitív a hiba, azután a „*ModuluszMux*” segítségével kiválasztjuk a magát a számolt hibát, ha az pozitív, vagy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



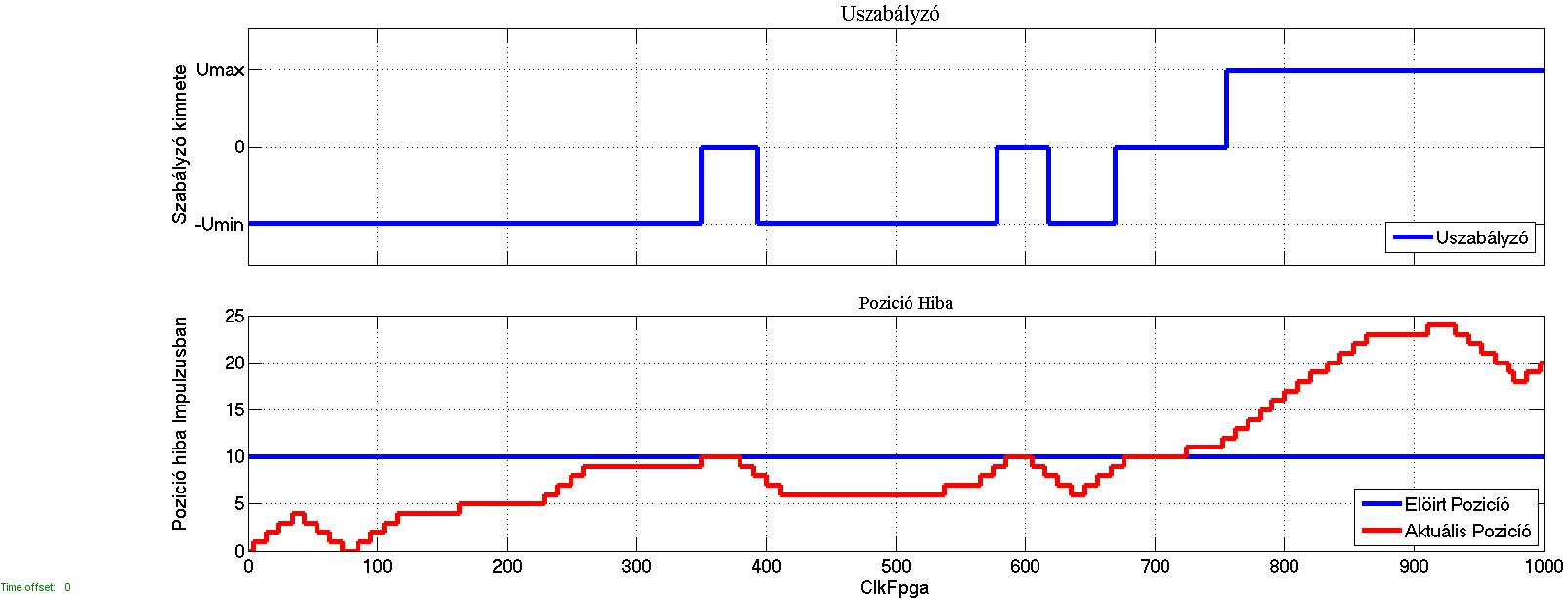
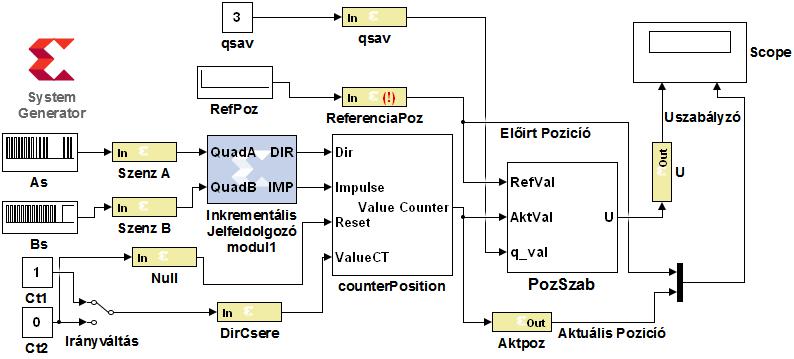
Kép. 3.9 A Pozíció szabályozó System generátoros felépítése

#### Szabályozó szimulálása

Az Kép. 3.10látható 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. Az előírt referencia jel a szimuláció során konstans értékű, ezért az 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.

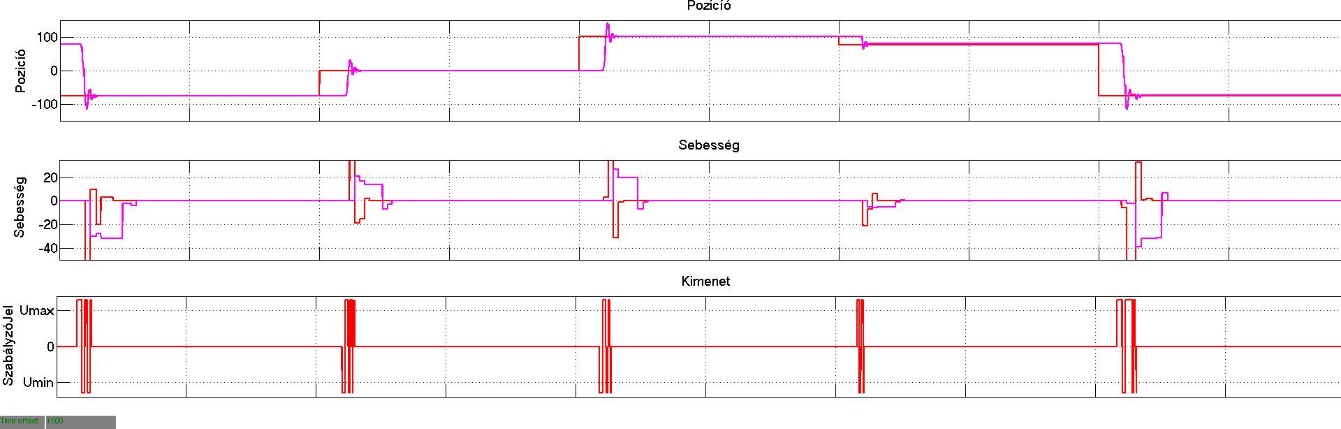
A szint illesztés csak egyirányú, az FPGA-tól kimeneti irányba. A motor tengelyére vagy a mozgatott mechanizmusra rögzített inkrementális tárcsa segítségével tudjuk mérni az elfordulást. A motor sebességét -ben vagyis impulzus per mintavételben mérjük, így a referencia sebességet is ebben a mértékegységben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

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

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 módszerrel 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. 3.14 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 Kép. 3.14, a „Sebesség Szab” modult tartalmazza Kép. 3.16 látható modulokat és a sebesség szabályzására hivatott, míg a „Pozicíó Szab” tartalmazza a Kép. 3.15 látható modulokat és a pozíció szabályzózására hivatott.

A szabályozók referencia bemeneteit osztott regiszterekkel írjuk elő a MicroBazeprocesszorból. 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 tartalmaz a .Kép. 3.21 látható modult.

Fel kellene sorolni milyen funkciókkal rendelkezik a modul és hivatkozni a táblázatraA „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. 1Konfig regiszter funkciói

A robot forgó talpának a szög pozíciójának deriváltja, a kis keréknél sebesség jön létre, ezért megkell változtatni a lánctalp sebességét, hogy a talajhoz viszonyítva a sebesség robot sebessége ne változón meg. A sebesség szabályozó referencia bemenetéthez hozzá kell adni a pozíció változását megszorozva egy arányosáig tényezővel.

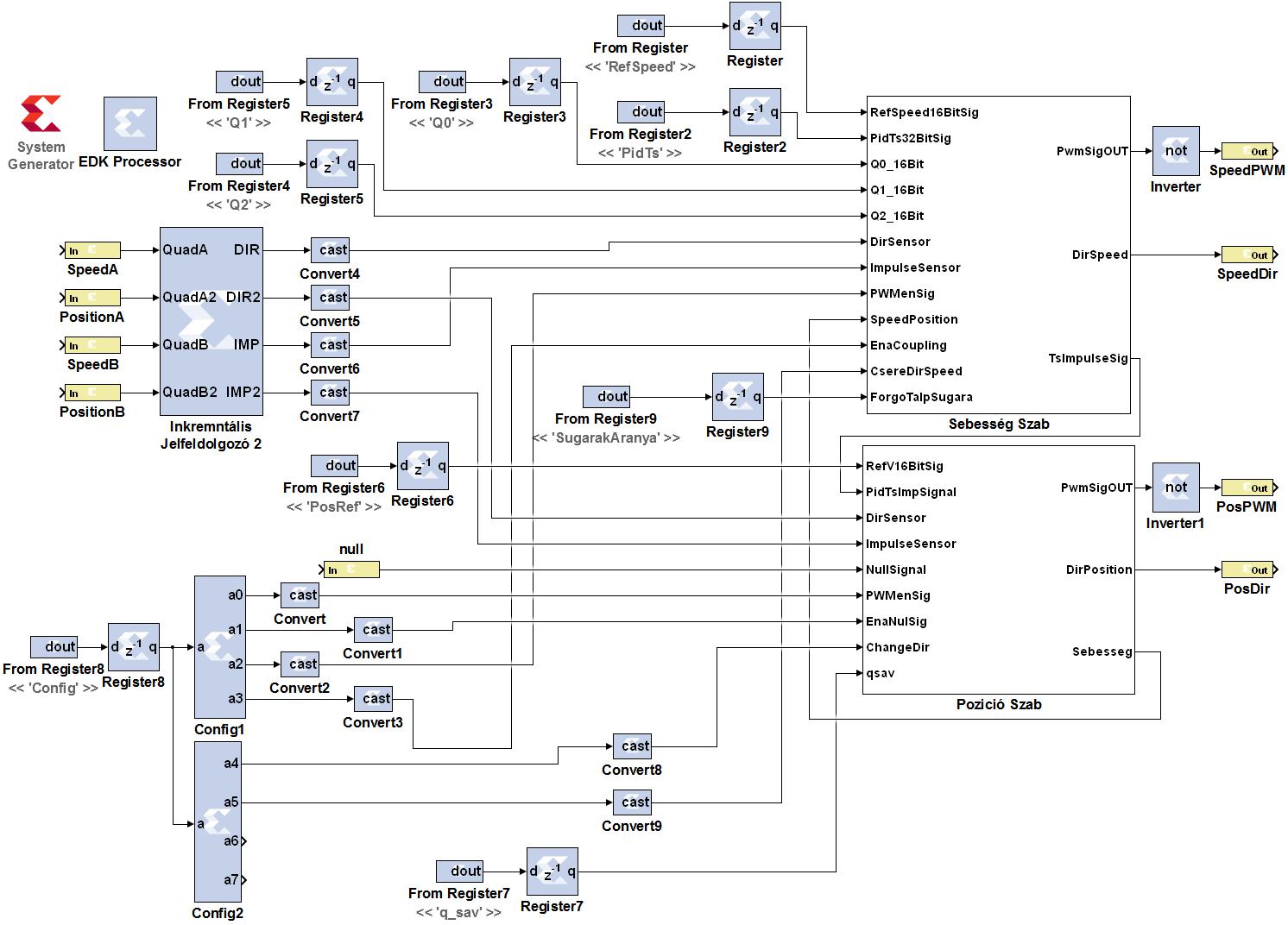
A Kép. 3.50 látható az 1 forgó talp sebességet generál az kör mentén, a robot lánctalpának az aktuális sebessége

A sebességek összeadását Kép. 3.16 látható Mult, Mux, Viszacsatolás modulok végzik el, az összeadást ki vagy be kapcsolhatjuk a Config regiszter negyedik bitjével.

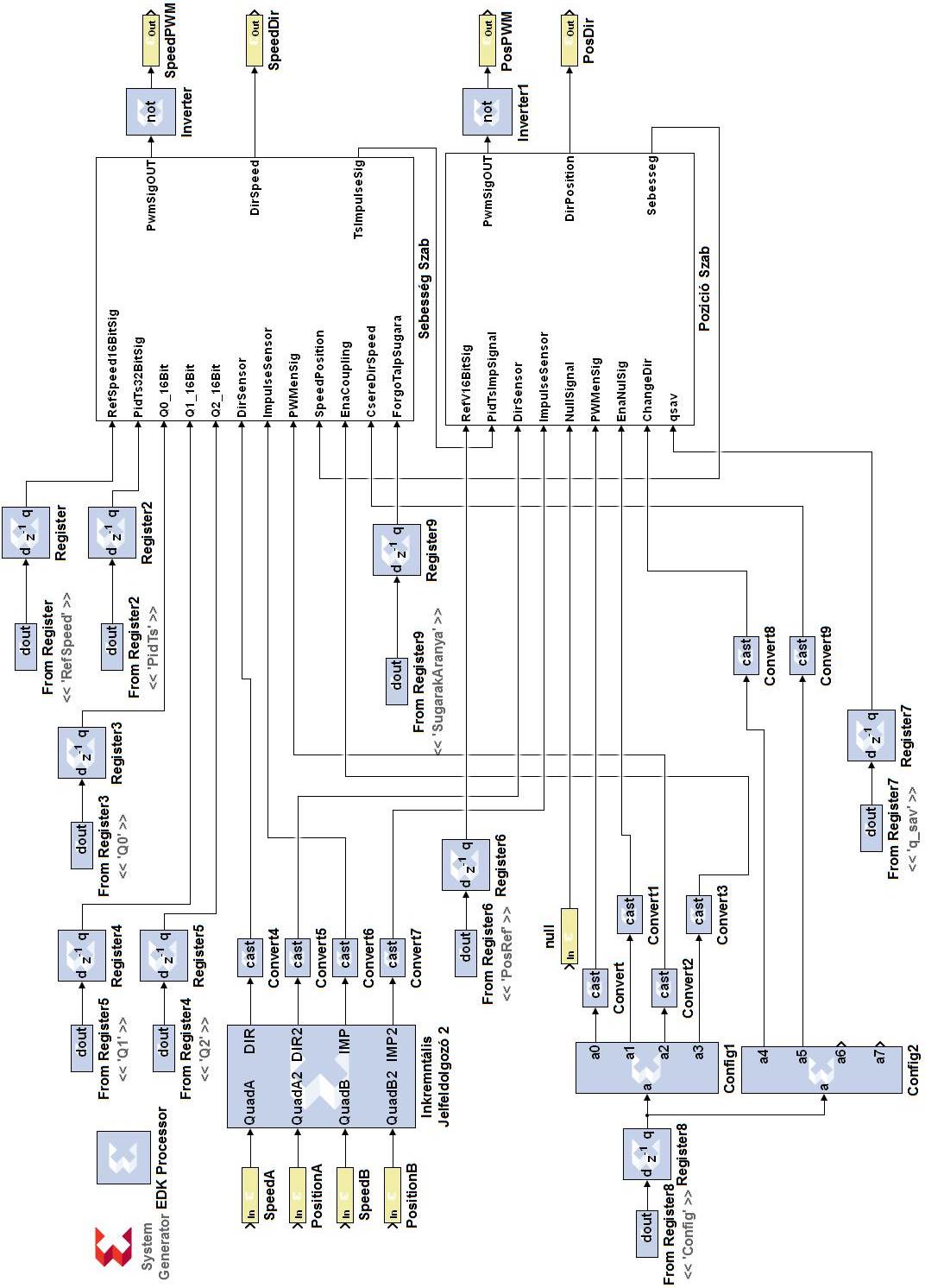
Az osztott regiszterek olyan hardveres elemek amelyeket az FPGA ba hozunk létre, és szoftveresen a memóriába mapolt cimekkel rendelkeznek, amelzeket irhatunk vagy olvashatunk. A System generátorban a ToRegister illetve FromRegister elemekkel tudjuk kivitelezni. A ToRegister típusú regiszterek csak írhatók a szoftvertől tudunk adatokat leküldeni a hardvernek, míg a FromRegister típusúak csak olvashatók, vagyis adatokat tudunk felvinni a szoftvernek, ha a hardver oldatról mezük. A szoftver általában egy microBlaze processzoron fut vagy egy beépített mikroprocesszoron. A regisztereknek betudunk állítani típusokat amely, a rendszerben a könnyebb kezelhetőség érdekében minden osztott regiszter típusa 16bit vagy 32bit nagyságú előjeles vagy előjel nélküli. Szoftveresen pedig egy meóira művelettel tudunk kinyerni vagy beírni az adatokat.

Memória műveletek: **Xil\_Out32**(*regiszter címe,változó neve*), az utasítás egy 32 bites értéket olvas ki a megadott címről a megadott változóba.

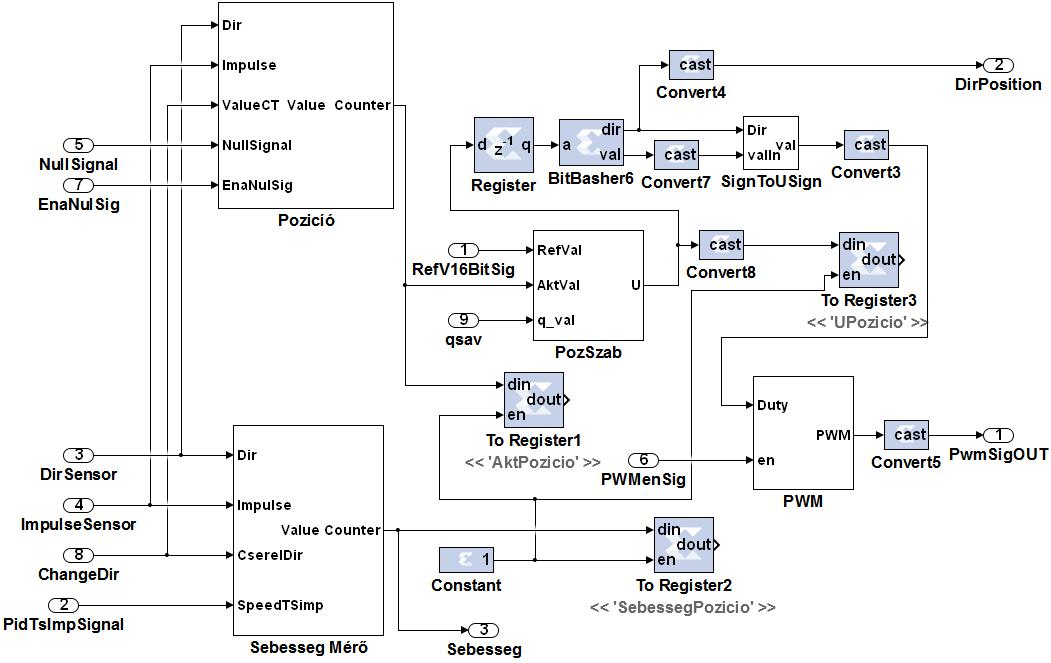
*változó neve* **=Xil\_In32**(*regiszter címe*), utasítással egy 32 biten értelmezett értéket olvasunk ki a megadott címről a megadott változóba. 16bites értékek esetén hasonloan járunk el anyi külömbséggel hogy a **Xil\_Out16, Xil\_In16** utasításokat használjuk**.**



Kép. 3.15 Sebesség és pozíció szabályózást tartalazó Ipmag System generátoros felépítése

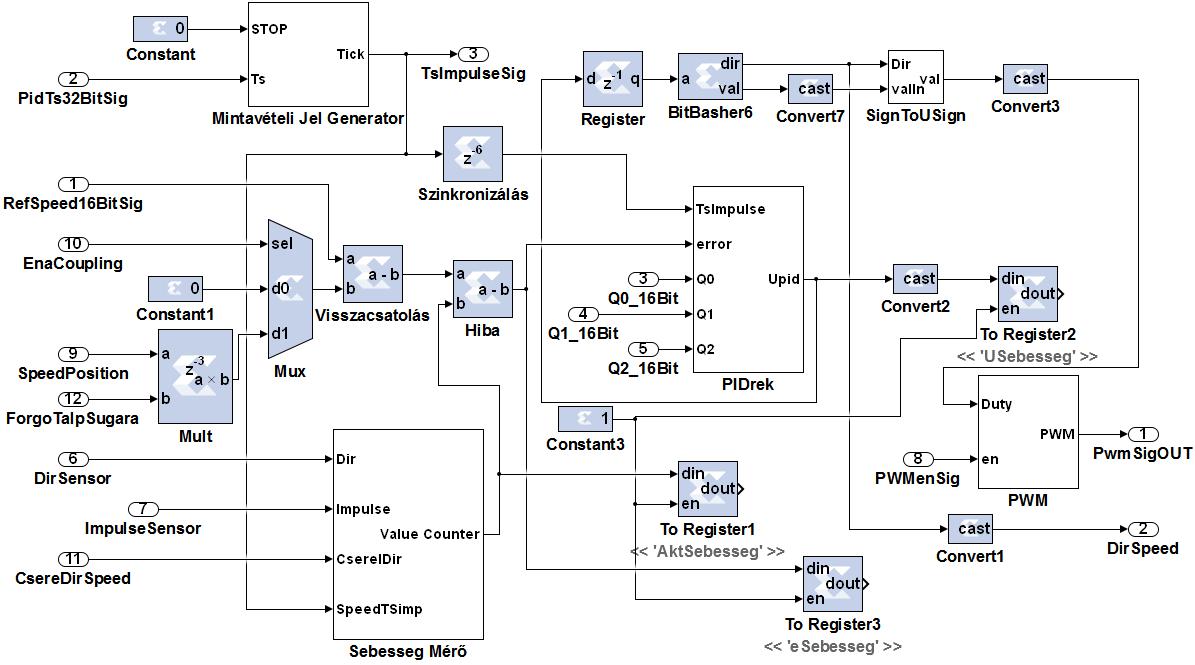


A „*Pozíció*” nevű modul tartalmazza a Kép. 3.24 képen látható modulokat, feladata a relatív pozíció mérése impulzusban, kimente egy 16 bites előjeles szám. A „*Sebesség Mérő*” nevű modul tartalmazza a Kép. 3.25 látható modulokat feladata a sebesség mérése impulzus per mintavételben, a kimenete egy 16 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. 3.16 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. 3.17 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.

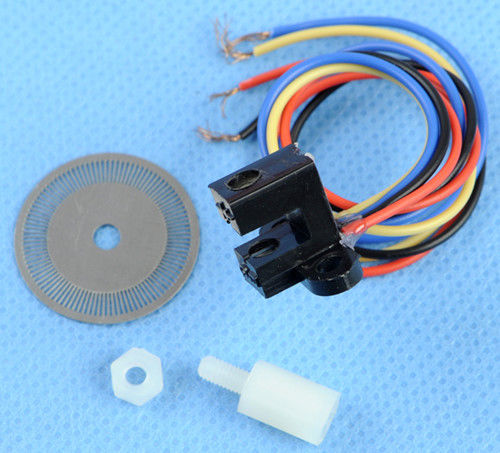
A Kép. 3.50 látható alsó ábrán a pozíció változása sebességet generál a kis keréknek, ha mi a sebességet szeretnénk tartani, akkor a következő összefüggést kell, előírjuk a sebesség szabályzónak ahhoz, hogy ne változón meg a sebesség:

, ahol a és cm, mérve.

## Szenzorok

### InkrementálisÉrZékelő

#### Optikai inkrementális vevő felépítése

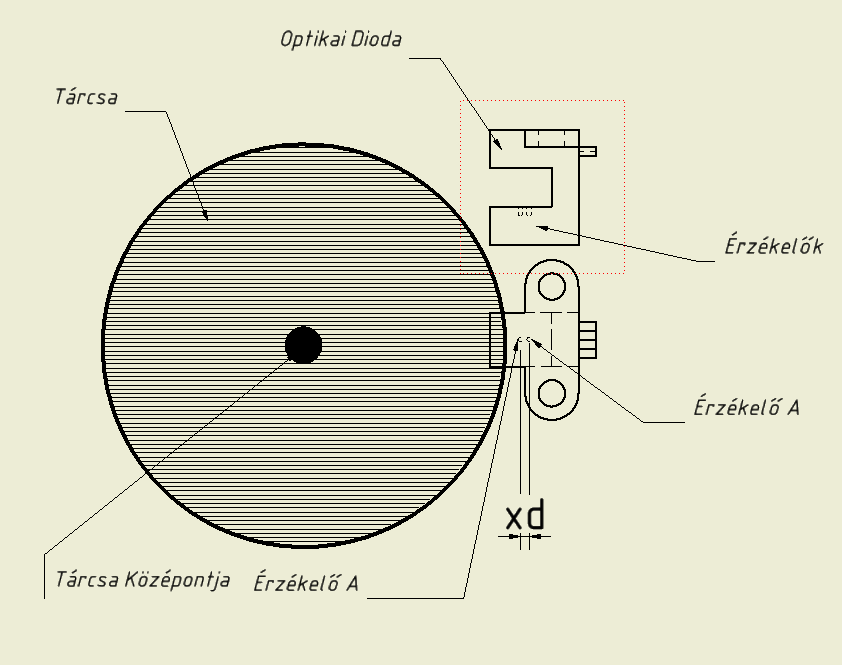
 Az optikai érzékelő két részből áll, egy optikai forrásból, és egy vevő részből, két optikai kapcsoló eszközt tartalmaz egymástól távolságra.

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.

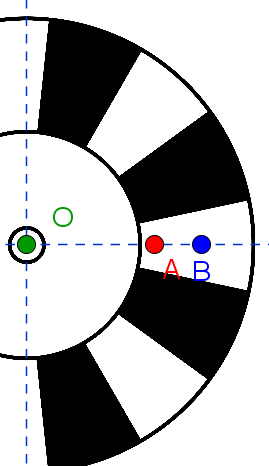


Kép. 3.18 Optikai inkrementális vevő felépítése és elhelyezése

Ha a 2.2 ábrán látható módonhelyezzük 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. 3.19 Érzékelő tranzisztorok elhelyezése

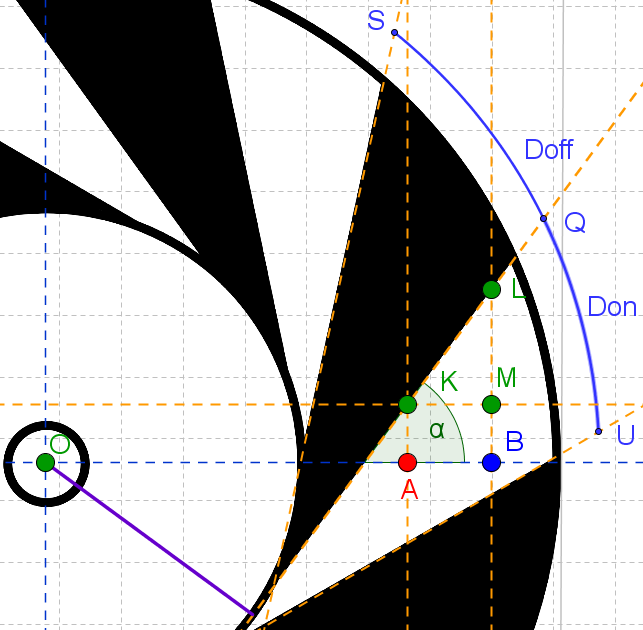
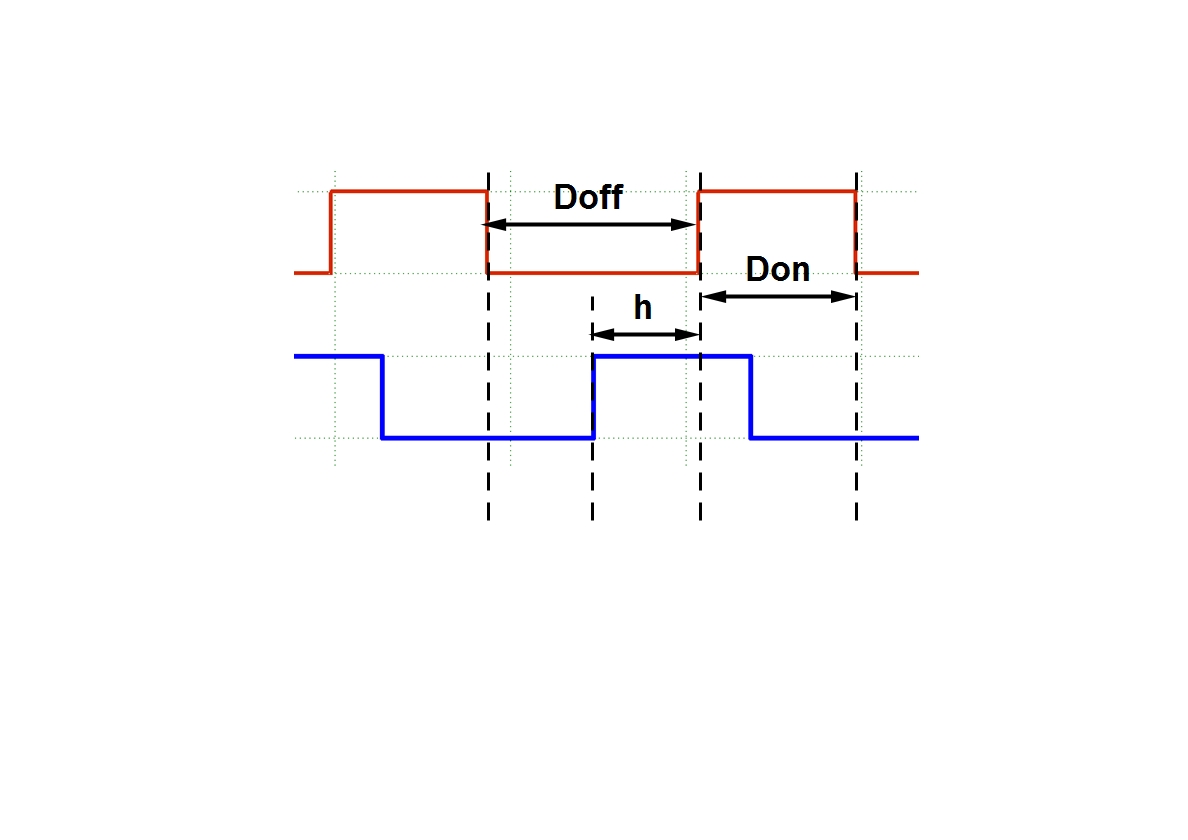
Az2.3 képen a fehér mezők az inkrementális tárcsa réseit képviselik. A rések száma megadja, a tárcsa felbontását N.

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).

**Ismertadatok**:

Kép. 3.20 Idődiagram a Tárcsa paraméterei függvényében



Kép. 3.21 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, akkorfelírható az összefüggés,amely meghatározza a két jel közti késést.

,.

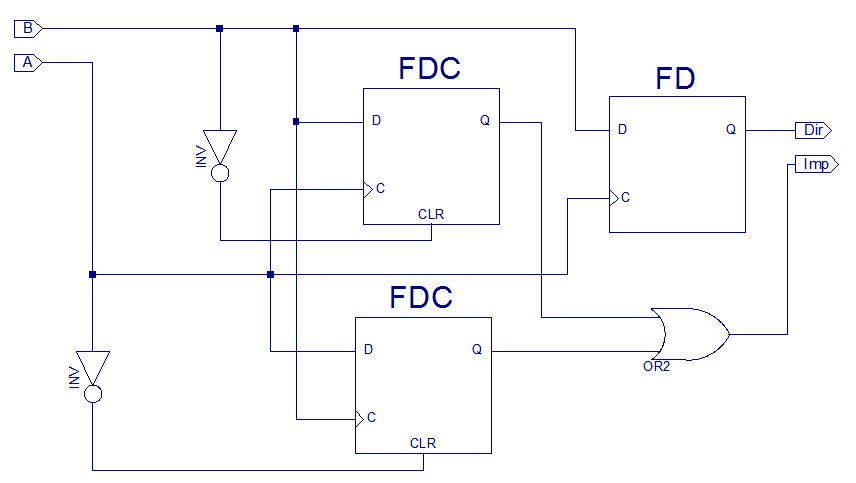
A minden esetben nagyobbnak kell lennie, mint a távolságnak.

### 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. 3.21látható 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 aImp (minden ablak elhaladásakor generál egy felfutó élet).

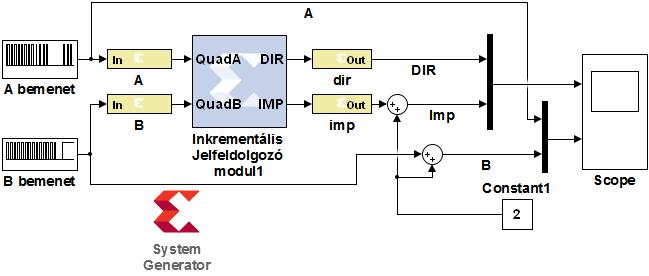
A beérkező A, és B jeleket egy tagadó kapu segítségével bekötjük a FDC tárolok CLR bemenetére, a 2.5 képen látható módon. Egy harmadik tároló segítségével meghatározhatjuk a forgás irányát. Az impulzus kimenete akkor lesz logikai 1, ha valamely FDC tároló Q kimenete is Logikai 1 lesz



Kép. 3.22 Inkrementális Jelfeldolgozó modul1 érzékelő modul belső felépítése

#### Szimuláció System Generatorban

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

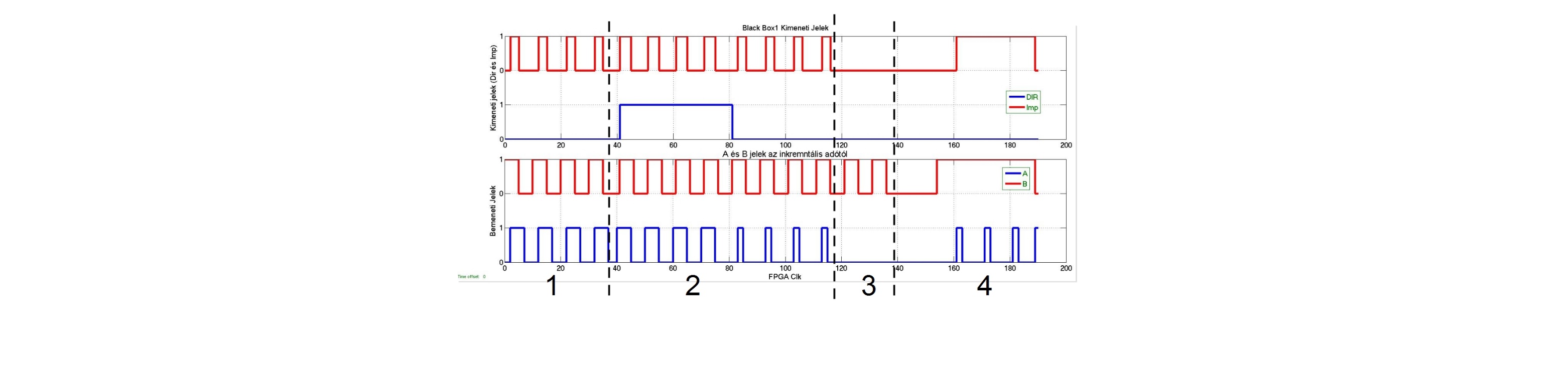


Kép. 3.23 Inkrementális érzékelőtől érkező jelek átalakító irány és impulzus jelekre

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. 3.24 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 impulzust generál fordulatonként. 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.

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 adatírást a regiszterbe.

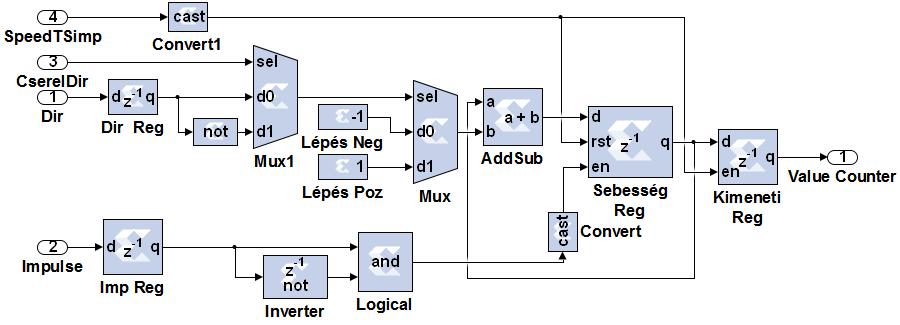
Tároló regiszter, feladata az aktuális érték tárolása,a típusa 16bites előjeles egész értékű a kezdőértéke mindig 0. A rstbemeneten érkező jel 0 értékre állítja a regiszter tartalmát, ez akkor történik meg, ha a „*NullSignal*” bemeneten impulzus érkezik. Azt mondhatjuk, hogy a pozíciót a 0 állapothoz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

A számláló logika feladata, hogy növelje vagy csökkentse eggyel a regisztertartalmát, annak függvényében, hogy a Dirjelmilyen 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 impulzus felfutó élére összegződik a FIFO tartalma.

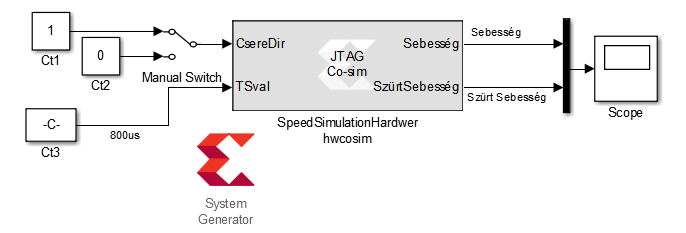


Kép. 3.26 Sebesség mérő modul felépítése

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.

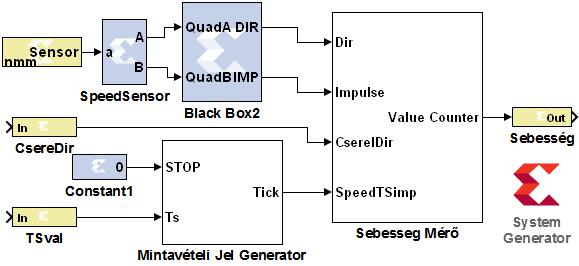
##### Hardveres mérések

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



Kép. 3.27Dc motor sebességének mérése Fpga lapon

A Kép. 3.27látható 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.



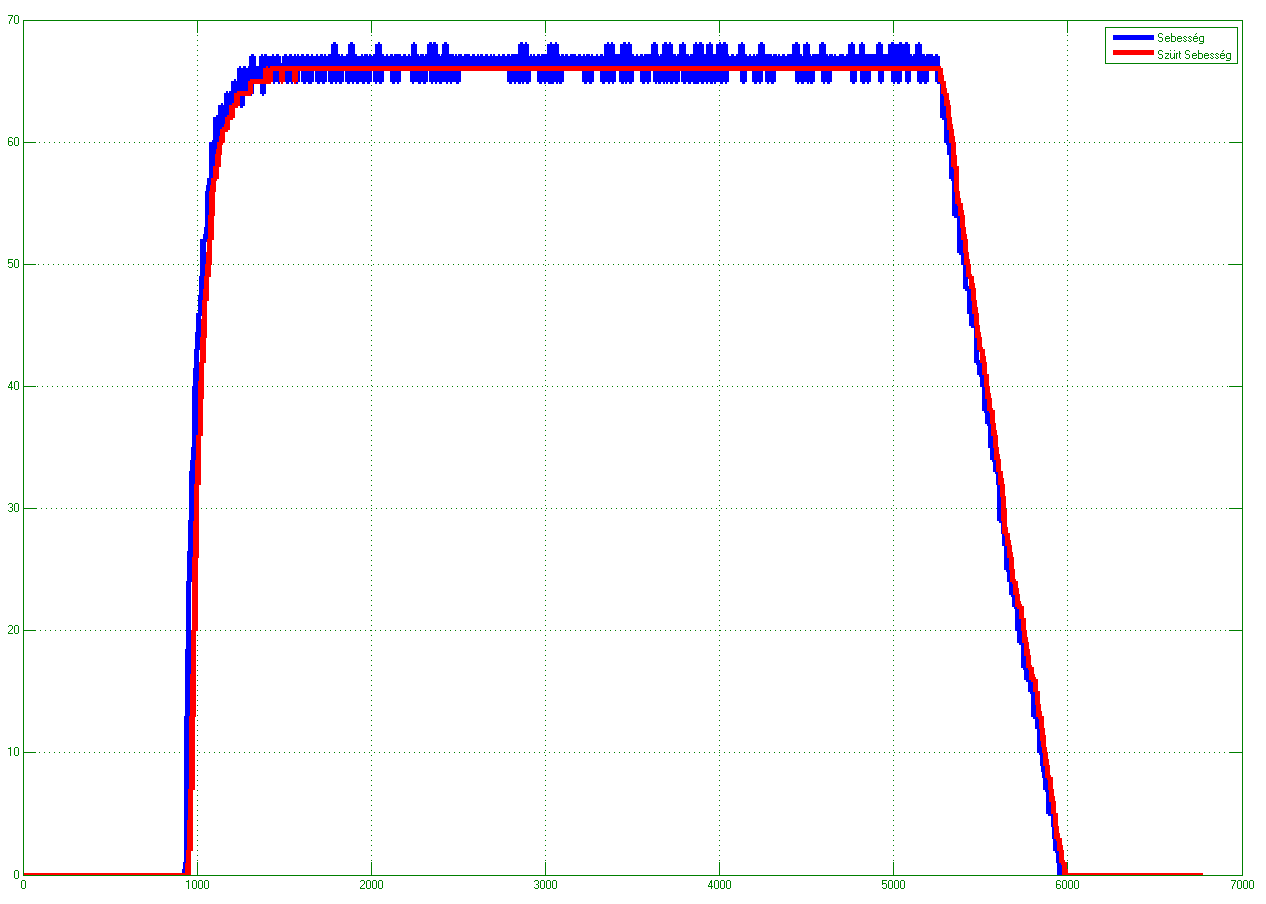
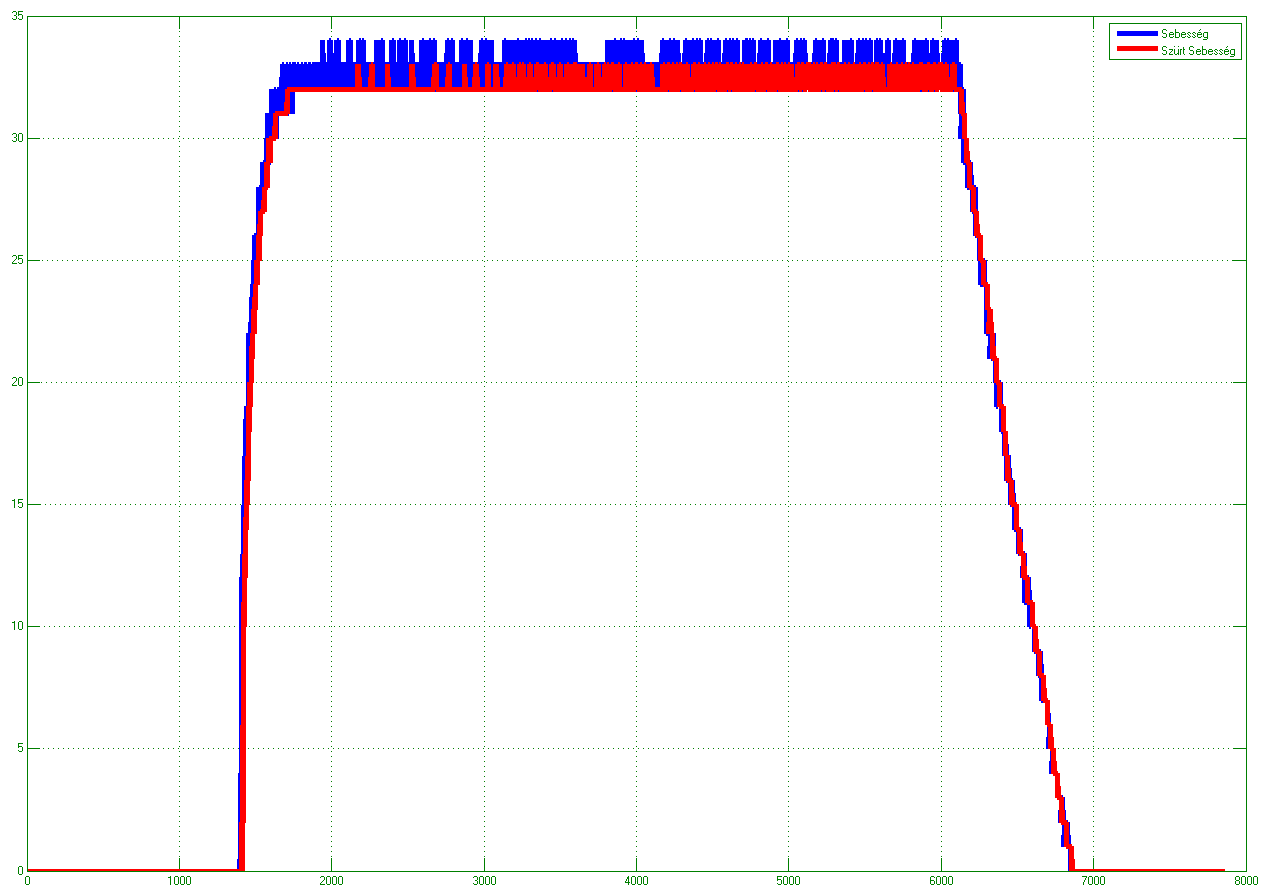
Kép. 3.28Dc motor Sebesség mérése FPGA rendszeren, System generatorban megvalósítva

A terv kigenerálása után kapunk egy újabb modult SpeedSimulationHardwerhwcosim elnevezéssel.

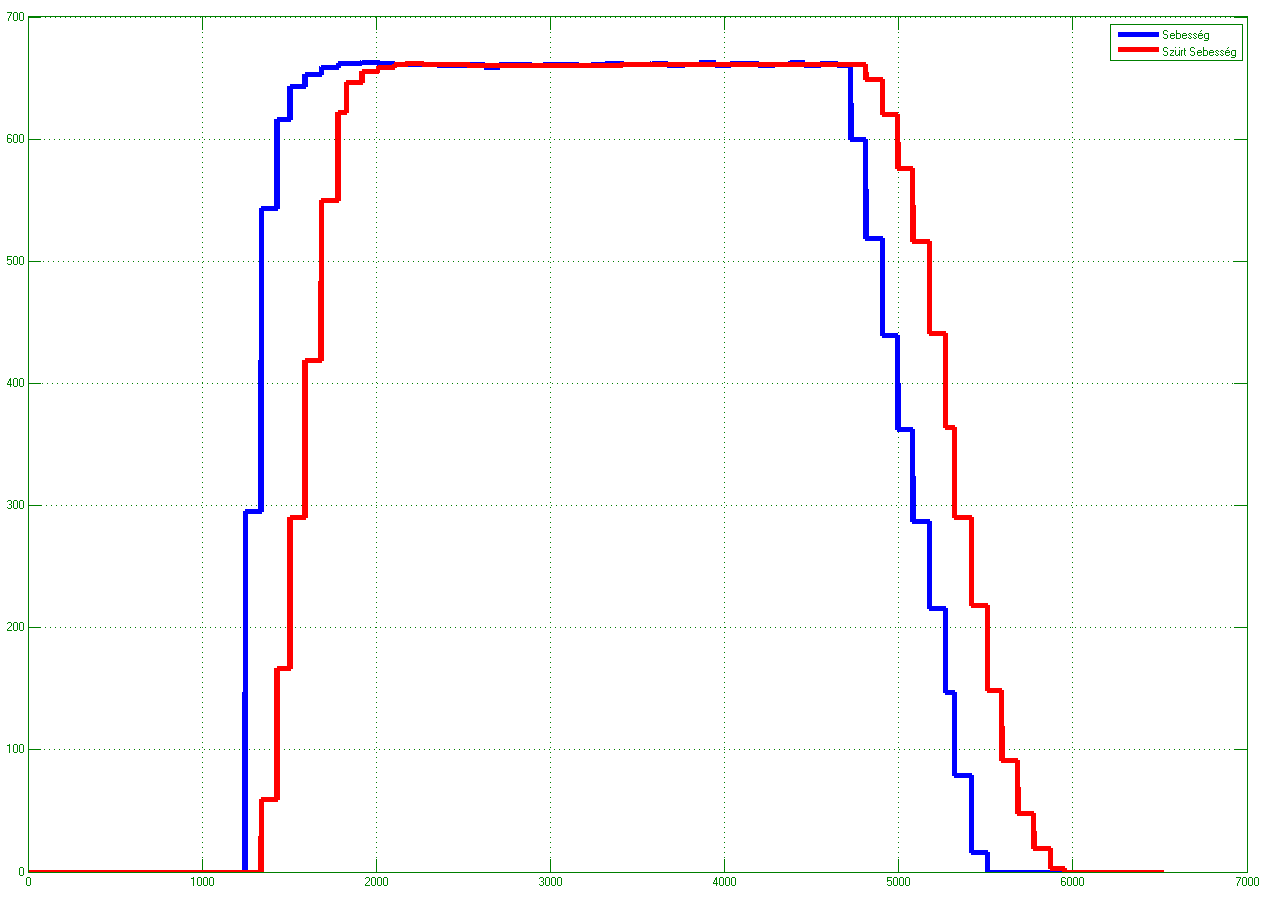
A 2.17 képen látható az újonnan generált modul bemenő adatainak a megadása illetve a mért értékek ábrázolása. A Ct3 nevű konstansba megadhatjuk a mintavételi időt msban.

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 sebességetRPMben átkel alakítani.



Kép. 3.31.a8ms



Kép. 3.31.c4ms

Kép. 3.31.b80ms

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 kommunikációs protokollon keresztül kommunikálhatunk. Az eszköz rendelkezik egy saját címmel, amelyet mi választottunk 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

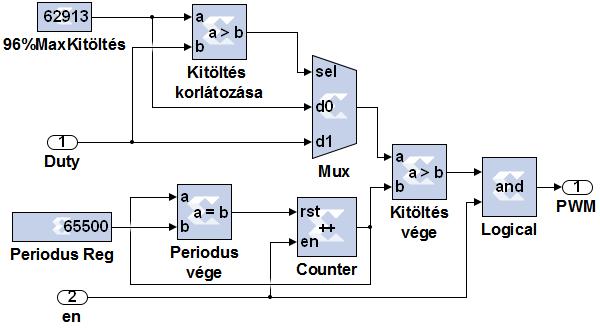
Egy hardveres számláló segítségével, amely az FPGA órajelére számol, egy komparátor segítségével összehasonlítjuk a számláló értékét az általunk megadott kitöltési tényezőértékével, mindaddig,amíg a számláló értéke kisseb a kitöltési tényezőértékénél a PWM kimenetén logikai 1-es kimenet lesz, amikor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

A PWM jel frekvenciáját egy „*PeriodusReg*” 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 kitöltési tényező értékének szüksége van egy skálázási eljárásra, amely segítségével illesszük a frekvenciához.

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 kitöltési tényező maximálisan 0 és 16biten felírható maximális érték között lehet (65535). Az alábbi egyenlet segítségével kiszámíthatjuk a kitöltési tényező regiszterének az értékét.

Kép. 3.32 A PWM generátor System Generátorban megvalósított szerkezete

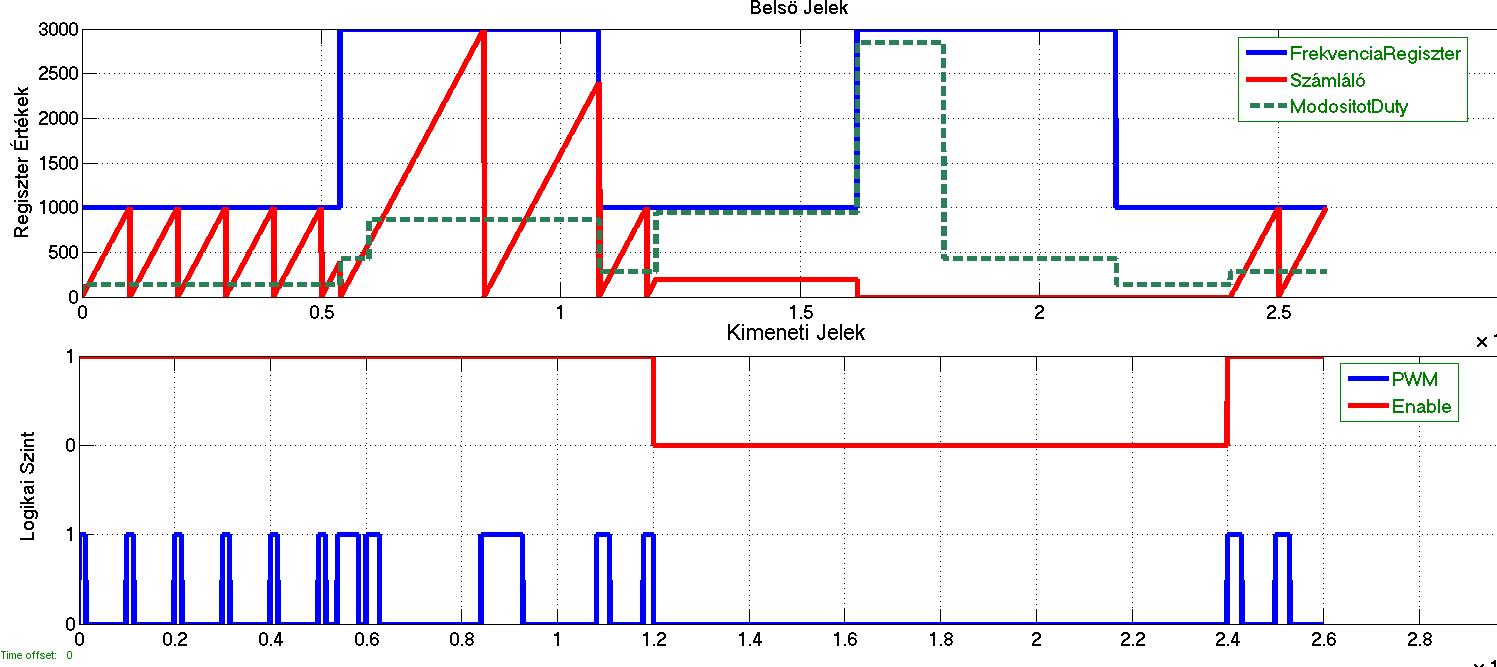


A PWM generátor kimenetét és a számláló(Counter) működését letilthatjuk az „*en*” bemenetre adott logikai 0 értékkel, máskülönben engedélyezve vannak.

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

A értéke ugyanakkor meghatározza a felbontást is, vagyis egy teljes periódust a PWM jelben hány részre tudunk felbontani. Látható hogyfordított 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 jelölt PWM jel, illetve papirossal jelölt Enable jel.



Kép. 3.33 a PWM generátor bemenő, kimenő illetve néhány belső jele (Scope1)

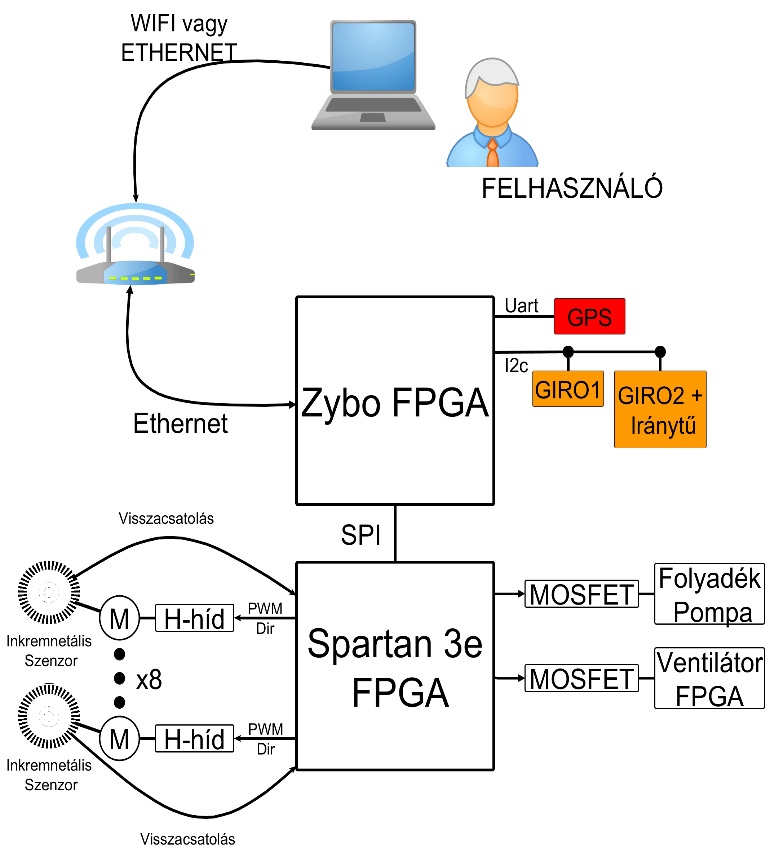
## Elektronika

### Digitális Elektronika

Az szenzorok adatainak a gyűjtésére és a beavatkozó jelek számítására FPGA rendszert használtam a flexibilitásuk miatt amely megkönnyíti a fejlesztést. 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. 3.34 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 sín rendszeren keresztül tudnak kommunikálni.

Az SpartanFPGA-ba kialakítunk egy 32 bitesMicroBlaze processzort, és a hozzá szükséges PLB sínrendszert, a sínrendszere illesztünk egy SPI kommunikációs egységet melynek feladata a ZYBO fejlesztőlappal való fizikai kommunikációs réteg kialakítása. A PLB buszra illesztünk még négy darab SebességÉsPozició szabályozó IPmagot, amelyeket a System Generátorban készítünk el és generálunk ki.

A szabályozókat tartalmazó IPmagparamétereit osztót regisztereken keresztül állíthatjuk be vagy olvashatunk ki értékeket, a regiszterek a PLB sínre vannak illesztve. A Zybo lapon található Eternet modulon keresztül kapcsolódunk egy Wifirouterhez, amely Access pontként működik. A routerhez még csatlakoztathatunk három más vezetékes eszközt, amelyek lokális hálózatba lesznek kötve a Zyboval.

#### Zybo FPGA fejlesztőlap

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

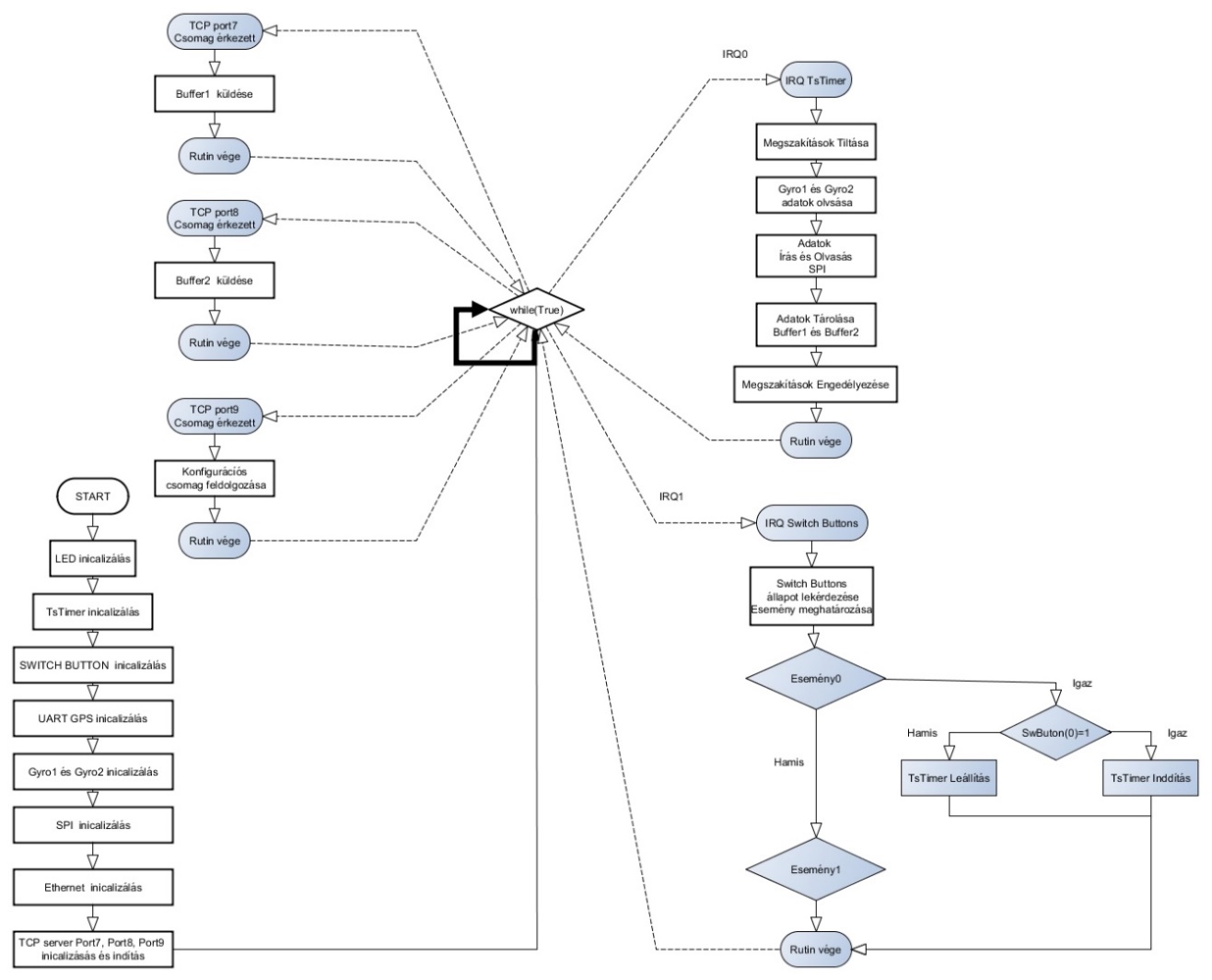
A Core0 processzor feladatai között szerepel a megszakítások lekezelése, a legfontosabb megszakítása a mintavételi periódust generáló számlálótól érkező megszakítás, amelynek bekövetkeztekor a processzor begyűjti, az adatokat a szenzoroktól (Giroszkóp 1 és 2). Lekezeli a megszakítást, amelyek az UART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

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 előkészítését és a beállításaikat, majd egy végtelen ciklusba lép. 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.

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

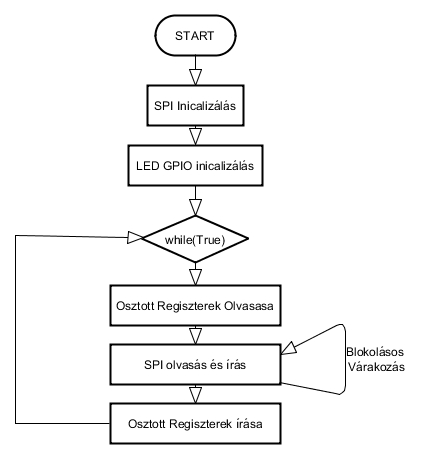
A program tartalmaz három TCP szervert is, amelyek a megadott port számon várják a kéréseket, minden kérés beérkezte után elküldik a pufferekben tárolt adatokat a kérést küldő kliensnek, és kiürítik a puffereket.



Kép. 3.36 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. 3.37 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 Zybomester 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.

s32addresReg; - cél regiszter szoftveres címe

s32dataReg; - cél regiszter adata

**outMag0**

s32USebesseg0;

s32UPozicio0;

s32SebessegPozicio0;

s32AktPozicio0;

s32SzurtSebessegPozicio0;

s32AktSebesseg0;

s32SzurtSebesseg0;

s32eSebeseg0;

s32n20;

s32n30;

s32n40;

s32n50;

s32n60;

**inMag0**

s32Config0;

s32RefPos0;

s32RefSeb0;

s32Q0\_0;

s32Q1\_0;

s32Q2\_0;

s32Ts0;

s32TsL0;

s32Egyeb0;

s32PwmFrekREgH0;

s32PwmFrekRegL0;

s32sugarakAranya0;

s32qSav0;

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 zybo rendser megtalálható egy Ethernet csatlakozó 1GBit/s sebességre képes. A programban, beagyazva az lwip140 modult, amelyet a xilinx készített. A [9] adatlap alapján három TCP portot hozunk létre.

A kommunikáció három TCP serveren 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.

A csomagok szerkezete a Kép. 3.35 látható a „*TCP csomagok*” feliratnál.

**A program:**

Első lépésben létrehozzuk a hálózat kialakításához szükséges IP4 címeket: az Zybo statikus IP címmel rendelkezik:

*IP4\_ADDR(&ipaddr, 192, 168, 0, 10);*

Az alhálózati maszk:

*IP4\_ADDR(&netmask, 255, 255, 255, 0);*

A routrer is rendelkezik egy statikus IP címmel a 192.168.0.1 amelyet beálltunk a zybo rendszeren, mint átjáró címet:

*IP4\_ADDR(&gw, 192, 168, 0, 1);*

Második lépésként létrehozunk egy fizikai címet, amelyek fog rendelkezni az eszköz:

***unsigned******char*** *mac\_ethernet\_address[] ={ 0x00, 0x0a, 0x35, 0x00, 0x01, 0x02 };*

Következő lépésben aktualizáljuk a beálltásokat a hardveren:

*init\_platform();*

*lwip\_init();*

*xemac\_add(echo\_netif, &ipaddr, &netmask,&gw, mac\_ethernet\_address,PLATFORM\_EMAC\_BASEADDR)*

*netif\_set\_default(echo\_netif);*

*platform\_enable\_interrupts();*

*netif\_set\_up(echo\_netif);*

**Szerver létrehozás:**

A Program. 3.6.1 kódrészletben létrehozunk egy TCP servert, amelyen majd fogadja a kapcsolatokat. A harmadik sorba definiálunk egy változót amelyben majd tároljuk a függvények által visszatérített értéket és ha az érték nem egyenlő 0 val akkor hiba történt a végrehajtáskor. A 4. sorban definiáljuk a port számot amelyen fog majd halgatozni a szerver. 5. létrehozunk egy uj TCP protokollt. 9. lépésben társítjuk az IP címet és a pontszámot a TCP protokollal.18. sorban társítjuk a kapcsolat kérésekor végrehajtandó eljárást.

A **accept\_callbackSV1** eljárás társítjaa csomagok érkezésekor meghívódó rutint amelyben majd történik az adatok visszaküldése a feladónak. A **recv\_callbackSV1** történik az adatok kiolvasása.

### Feladatok Elosztása

#### Zybo fejlesztőlap

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.

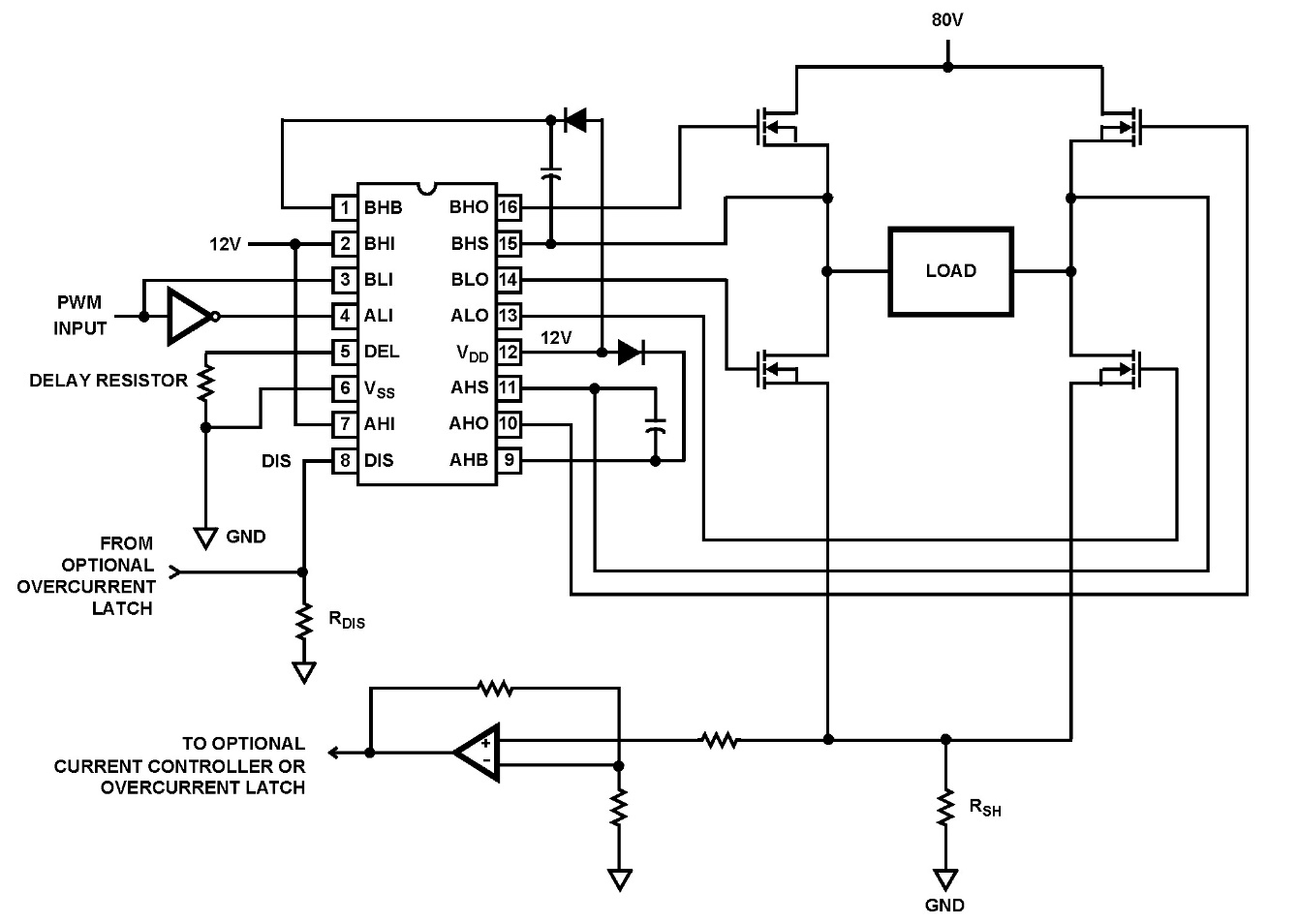
#### Spartan fejlesztőlap

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.

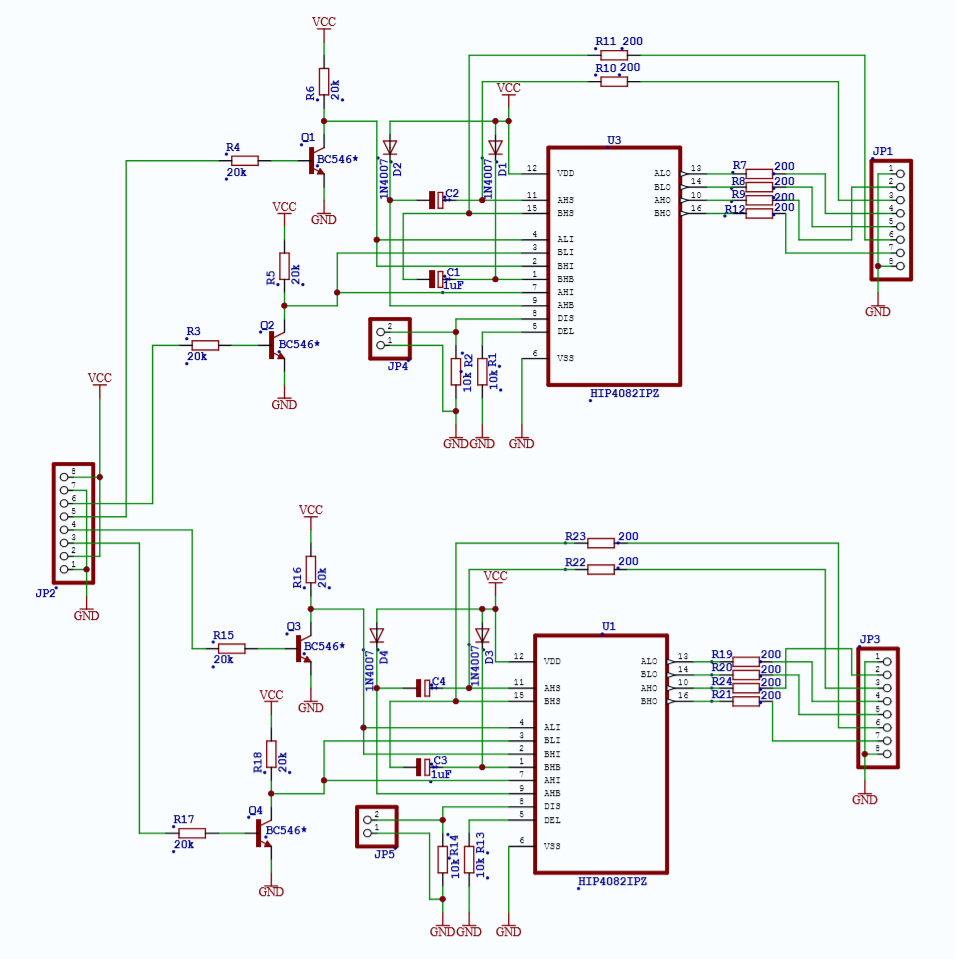
## GUI felépítése

## Teljesítmény elektronika

A DC motorok betáplálására 12V-16V egyenfeszültséget alkalmazunk a motorokat feszültségben pwm beavatkozó jel segítségével vezéreljük H-hídon keresztül. A H-hidakat N-csatornás MOSFET tranzisztor segítségével valósítottam meg, mert az N csatornás MOSFET tranzisztornak kisebb az DS ellenállása kinyitott állapotban. 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. 3.38Hip4082 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 típusa IRFB7437, adatlapi adatok szerint az 195A áramot tud vezetni maximálisan, 40V feszültséget bír el, valamit a teljesen kinyitott állapotban az ellenállása kisebb, mint 2mΩ.

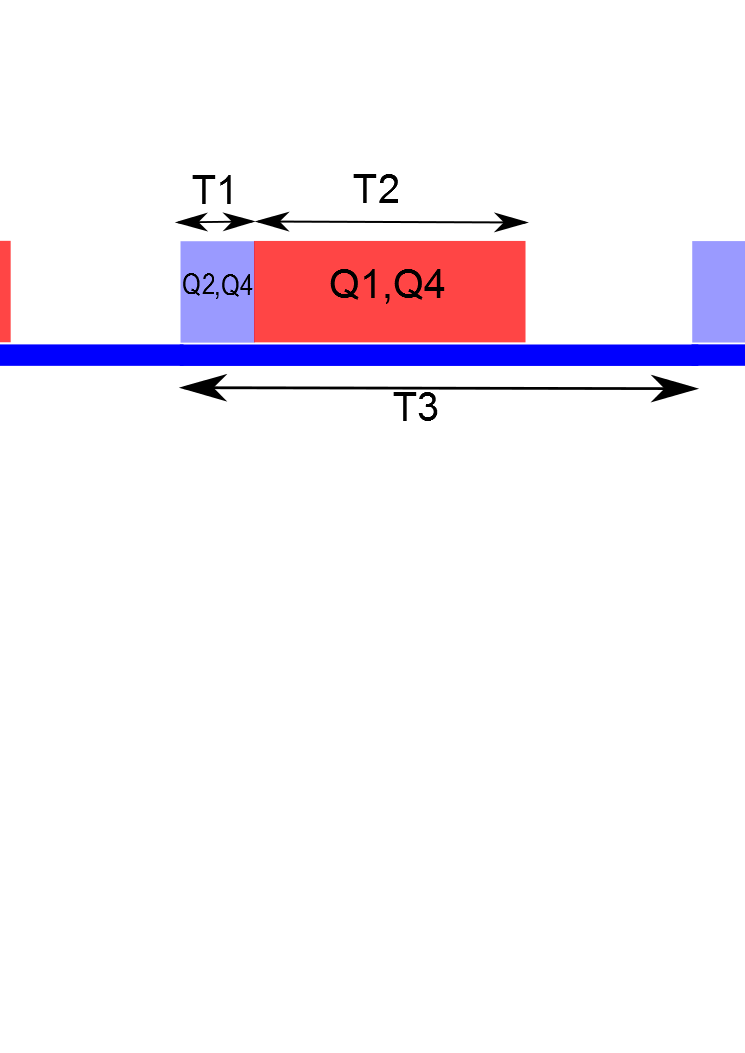


Kép. 3.39 Két hídvezérlő árak kör kapcsolási rajza HIP4082 integrált áramkörrel megvalósítva

Az N csatornás MOS tranzisztorokat pozitív feszültséggel tudjuk bekapcsolni a S (Source) kivezetéséhez képest 10V feszültséggel. A felso két tranzisztor S kivezetésének a feszültség szintjét nagyon befolyásolja a terhelés, ezért szinteltolást alkalmazunk a S kivezetéstől Boost megoldással amely egy diódát és egy kondenzátort tartalmaz, és feltételezi a szaggatást így oldva, meg hogy a kondenzátor töltődjön fel és majd a HIP4082 integrált áramkörön keresztül tudjuk rákapcsolni a felső tranzisztorok Gate bemenetére. A [9] adatlap alapján AHB, BHB az integrált áramkör azon bemenetei, amelyeknek feszültsége elvan tolva az AHS, BHS közös pontokhoz képest, a H hídban 10V feszültséggel feltöltött kondenzátor feszültségével. Az AHO, BHO azok a kimenetek, amelyek a felső tranzisztorokat vezérlik, az ALO, BLO az alsó tranzisztorok vezérlő kivezetései. A HIP4082 áramkörben 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 fél 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. 3.42 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. 3.42 PWM és a tranzisztorok kapcsolása

A kívánt kapcsolási sorendet úgy tudjuk elérni hogy a

A DIS bemeneten áramkorlát vagy más védelmeket iktathatunk be a rendszerbe, a [9] adatlapban ajánlott megoldás szerint méri az áramot az ellenálláson keresztül, 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. 3.40 látható a nem invertáló erősítő kapcsolás az Ube bemeneti feszültség az árammérő ellenállástól érkező feszültség. A kapcsolás erősítése:



Kép. 3.40 Nem invertáló erősítő forrás [13]

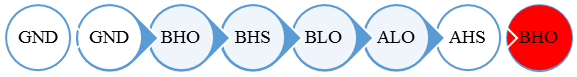
Ha , 10A szeretnénk az áramot korlátozni akkor a , ahol az a H hídon átfolyó áram.

Az erősítés: . A műveleti erősítős áramkorlát nem jelenik meg a megvalósításban, 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 meg tudjuk 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.

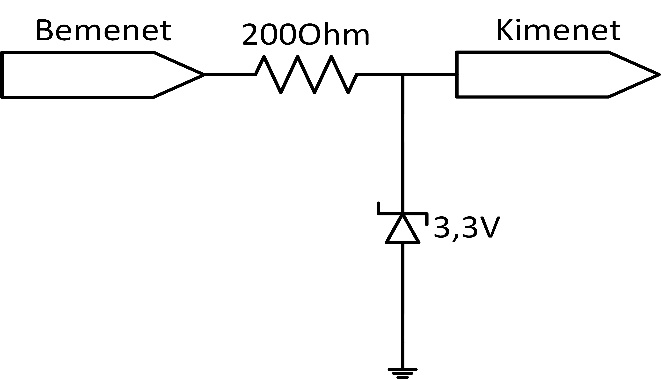
A pirossal megjelölt vezeték az 1 számú. A PWM4 egy 3,3V pwm jel, amely egy NPN (Q2) tranzisztoron keresztül kapcsolja az BLI, AHI bemeneteket, a tranzisztor a jelet megtagadja, ezért majd a FPGA pwm moduljába illesztünk egy tagadó, kaput, hogy semlegesítse ki egymást a két kapu. A PWM3 hasonlok épen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



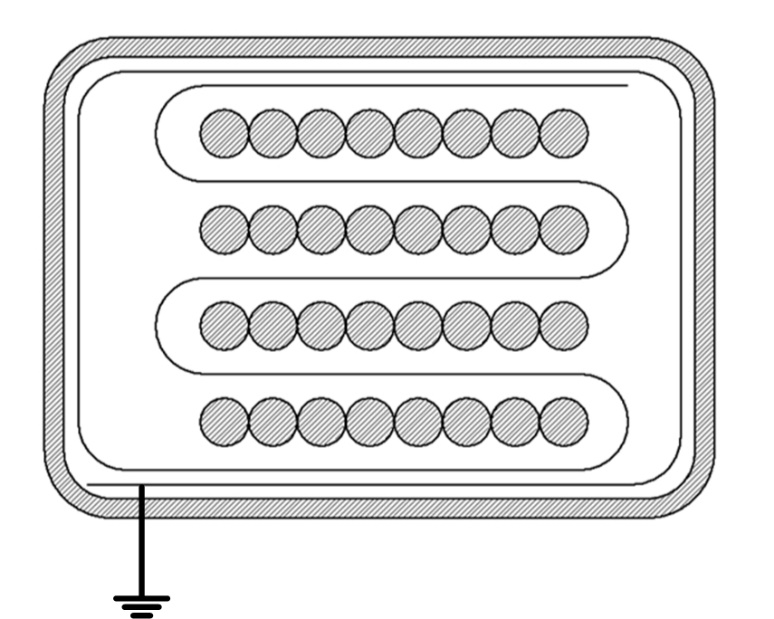
Kép. 3.42 H híd tranzisztorainak a Gate vezetékei

A Kép. 3.44látható 4H-híd B és A dobozok tartalmaznak négy H hidat, a hidak kettesével rögzítve vannak egy rézlemezre, amelyeken keresztül tudunk 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.

A négy híd vezérlésére négyKép. 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 külső műanyag szigetelés véd a fizikai behatásoktól.



Kép. 3.44FPGA kimentének a védelme



Műanyag szigetelő

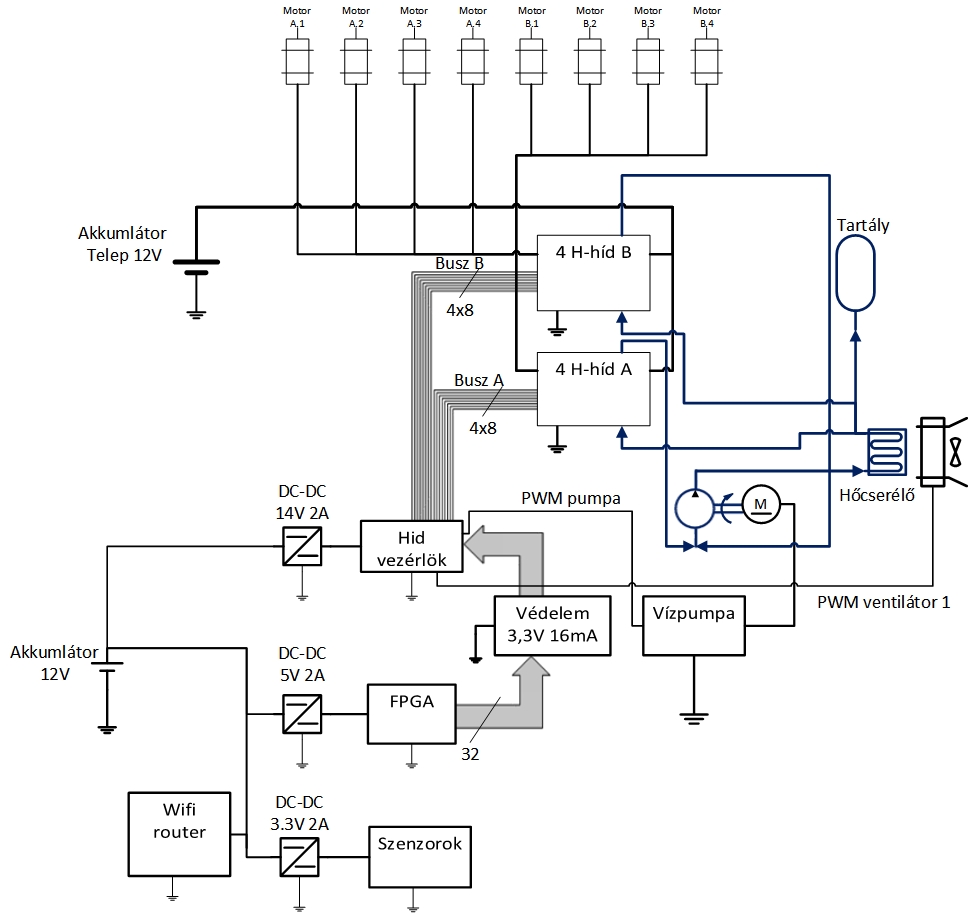
Alumínium fólia árnyékolás

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. A Kép. 3.44 látható a robot energiaellátásának a terve. Az energia ellátás akkumulátorokkal fog történi, rendeltetésük szerint két csoportba oszthatók: egy 12V akkumulátor gondoskodik a rendszer digitális áramköreinek az ellátásáról. A digitális elemeket DC-DC konverteren keresztül táplálom be melyeknek a feszültsége állítható. A minimális feszültség ami szükséges a konvertereknek 3V, és a kimeneti feszültséget állíthatjuk 3-30V-ig. A konverterek maximálisan 2A tudnak leadni.

A wifirouter modulban megtalálható a beépített konverter.

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

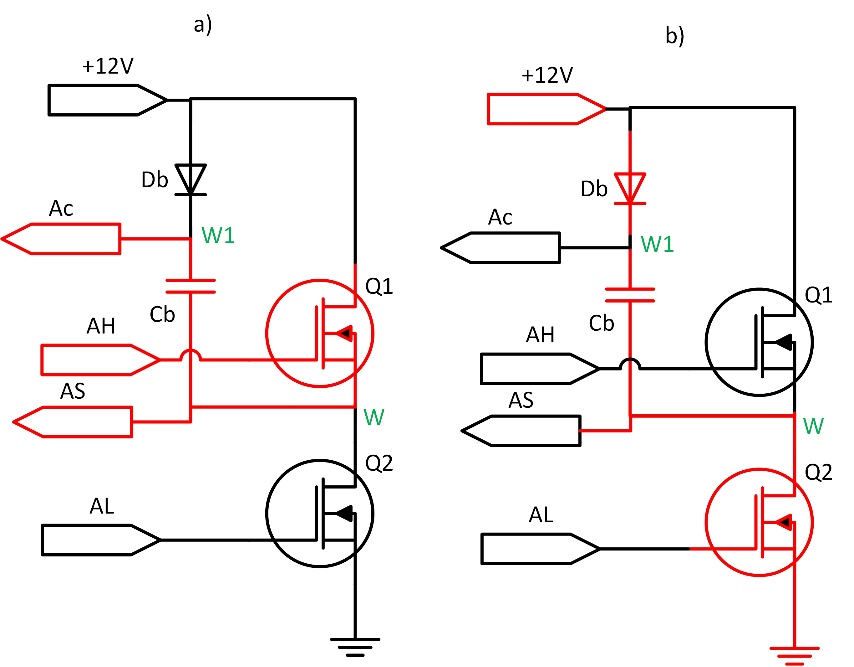


Kép. 3.45A robot energia ellátása valamint a hűtő rendszer elvi felépítése

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

### Bootstrampmüködése

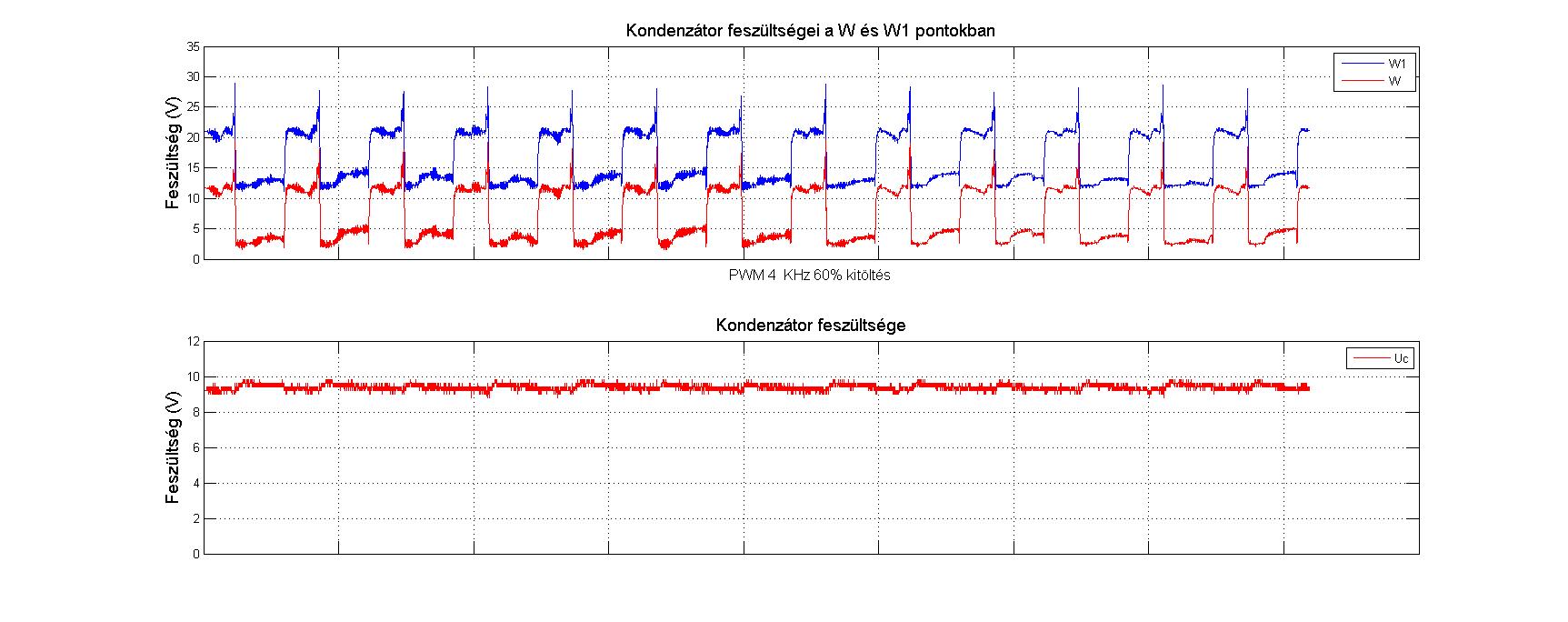
A [10] dokumentum alapján a bootstrap megoldás a Kép. 3.46 képen látható módon történik. A Cb kondenzátort töltjük fel a Db dióda segítségével azokban a pillanatokban, amikor a Q2 tranzisztor kinyitott állapotban van és a w potenciál elég alacsony, ahhoz hogy a Db dióda kinyisson és így feltöltve majdnem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



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

A Kép. 3.47 látható b) ábrán látható amint a Q2 tranzisztor tölti a Cb kondenzátort, a) képen a AS és Ac feszültségek be vanak vezetve a HIP4082 integrált áramkörbe amely majd az a vezérlő jel hatására rákapcsolja a Ac bemenet feszültségét a AH kimenetre.

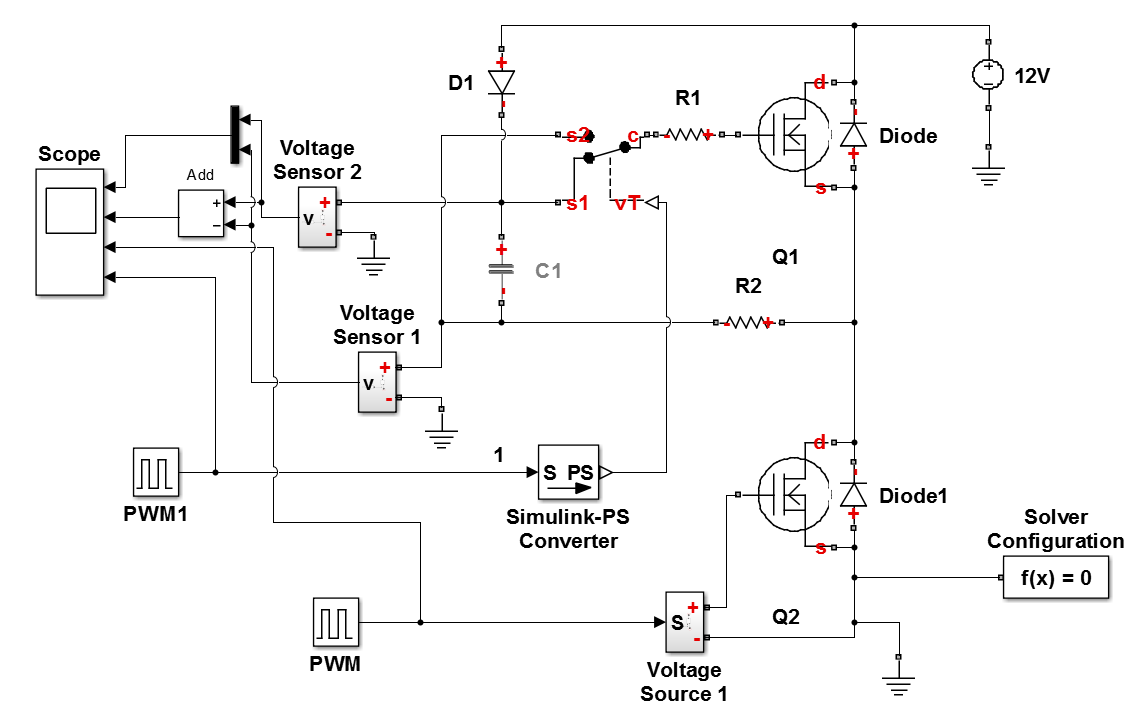
A Kép. 3.47 látható a mérése a Cb kondenzátor feszültségének, a méréseket oszcilloszkóp segítségével végeztem el és mentettem ki az adatokat, amelyeket majd ábrázoltam Matlab programmal.



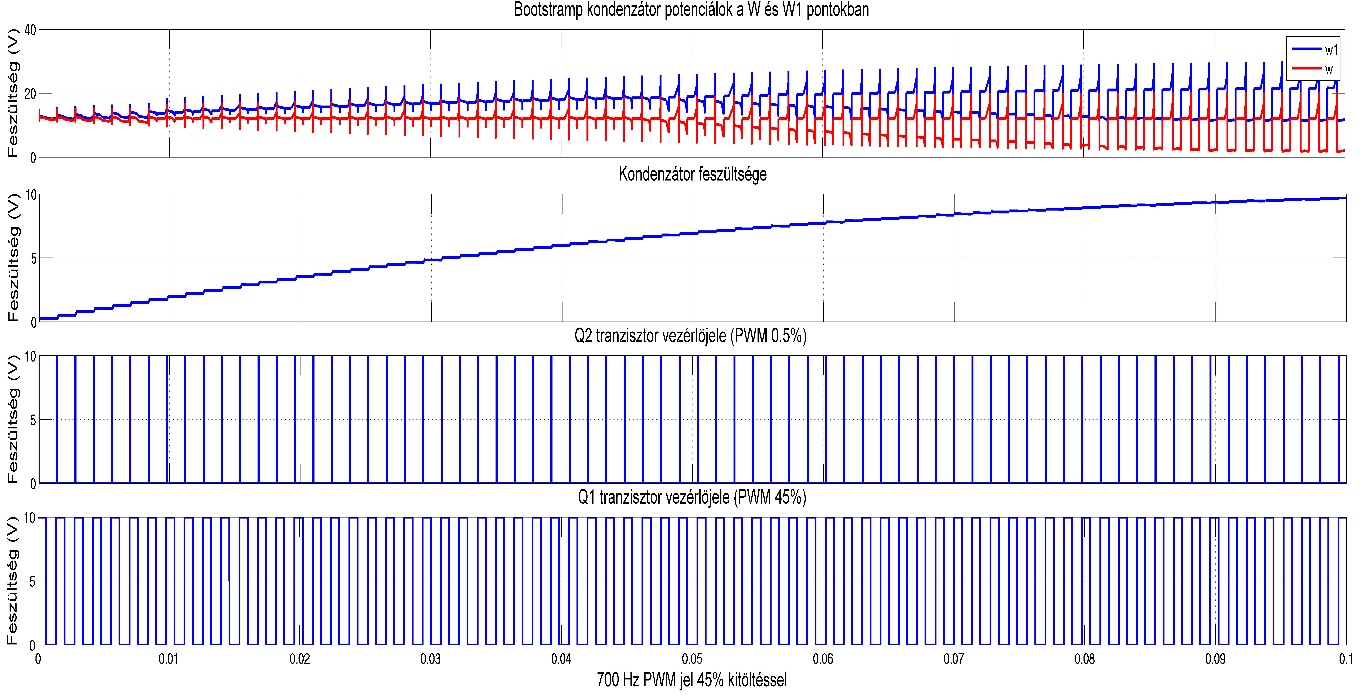
Kép. 3.48Bootstramp kondenzátor feszültsége a W és W11 pontokban

#### 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.



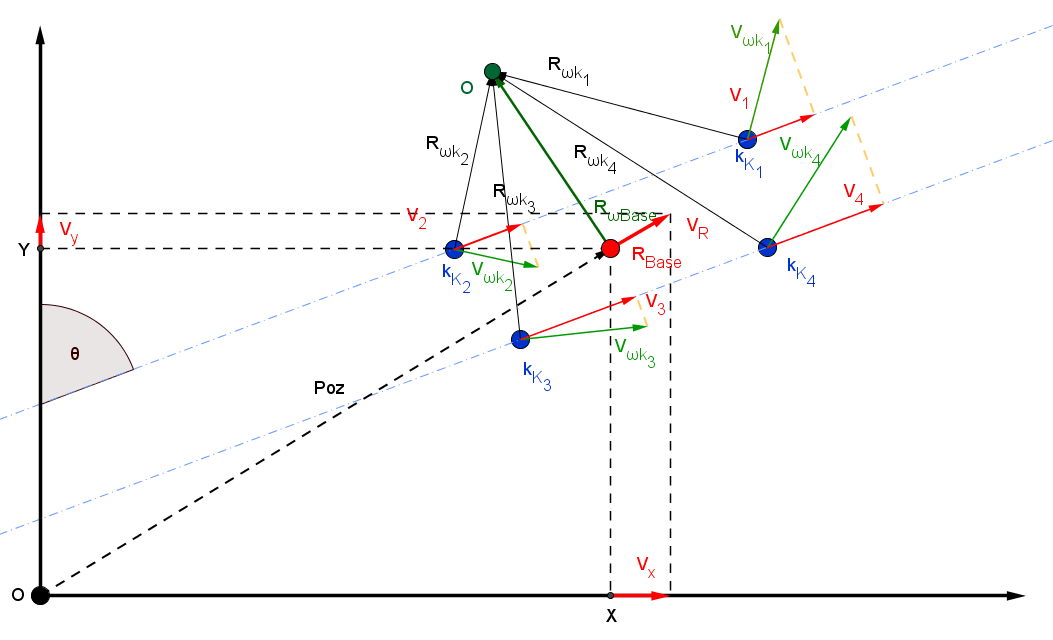
Kép. 3.49 Bootstramp működése, szimulációs modell MATLAB/SIMULINK környezetben



Kép. 3.50 Szimulációs eredmények Bootstramp

## Robot Modell

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



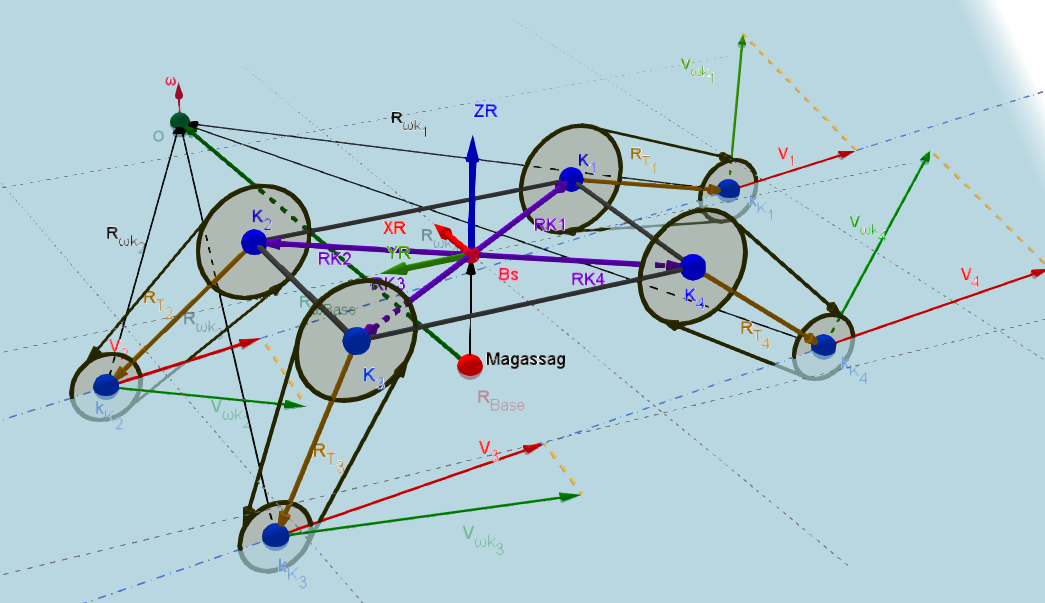
Kép. 3.49Robot kerekek sebsége és a robot mozgásának viszonya

**Jelölések**:

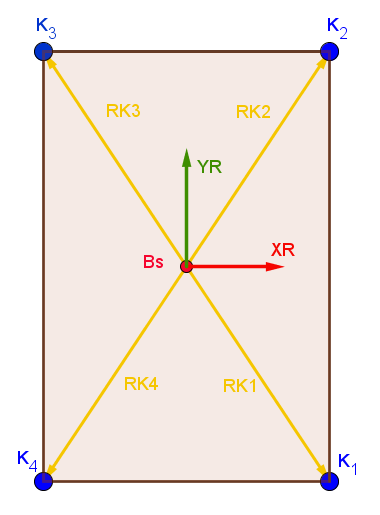
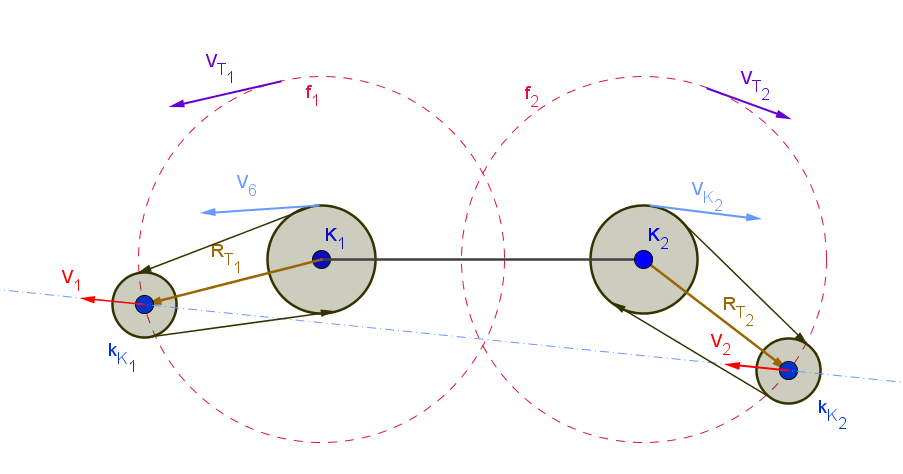
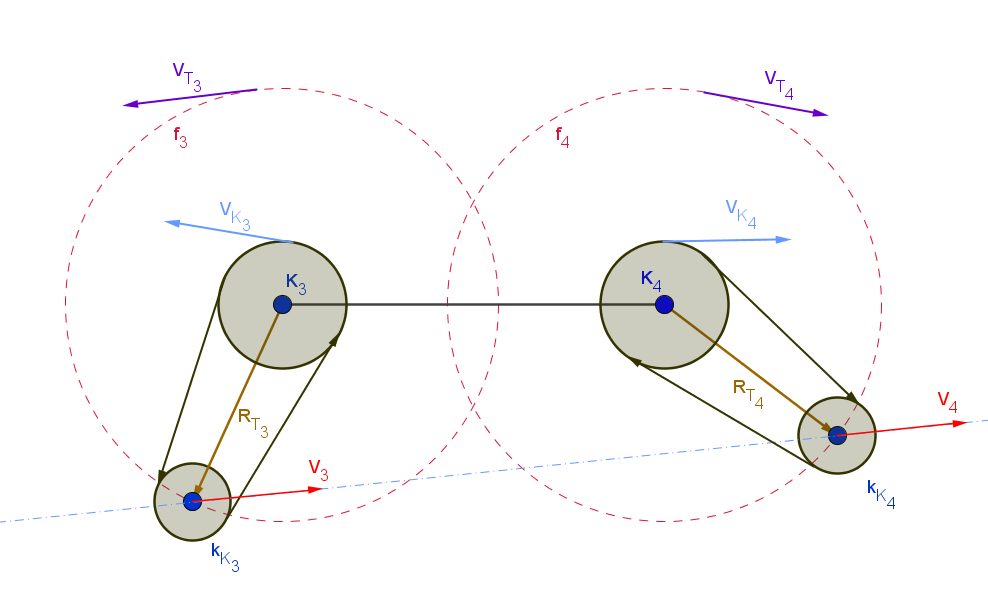
Szeretnénk, ha a robotunk egy adott körpályát írna le egy pont körül egy adott sebességgel. Jelen esetben O pont körül és, szögsebességgel.

Az [12] 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:



Kép. 3.50 Robot 3D vektorábrája



Kép. 3.51 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 AutodeskInventorban elkészített terve.

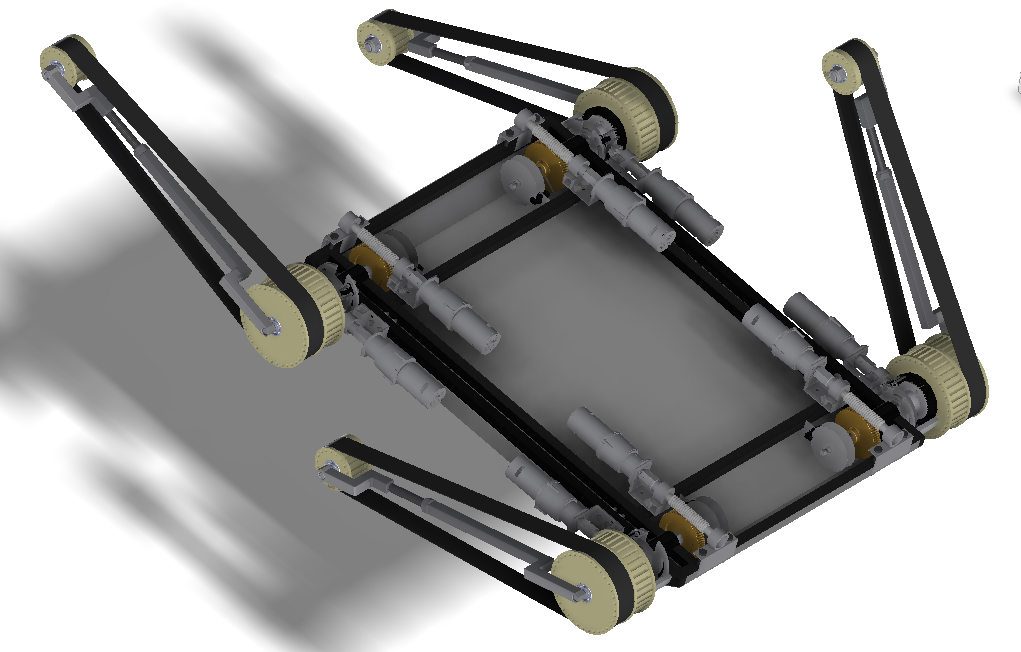
DC motrok betáplálási feszültsége: 12V, maximális terhelés alatt 10A áramot is felvehet.

A 7.1 képen látható Kup kerékáttételen keresztül hajtjuk meg a lánctalpat, a talpak mozgatására orsósáttételt használtam több okból is:

* Nagy nyomaték kifejtésére képesek, jelen esetben 40:1 az áttétel arány. Egy motor által kifejtett nyomaték névlegesen 3-4Nm között van így a karok végén tengelyre kifejtett nyomaték elérheti a 60Nm-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 nyomaték 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

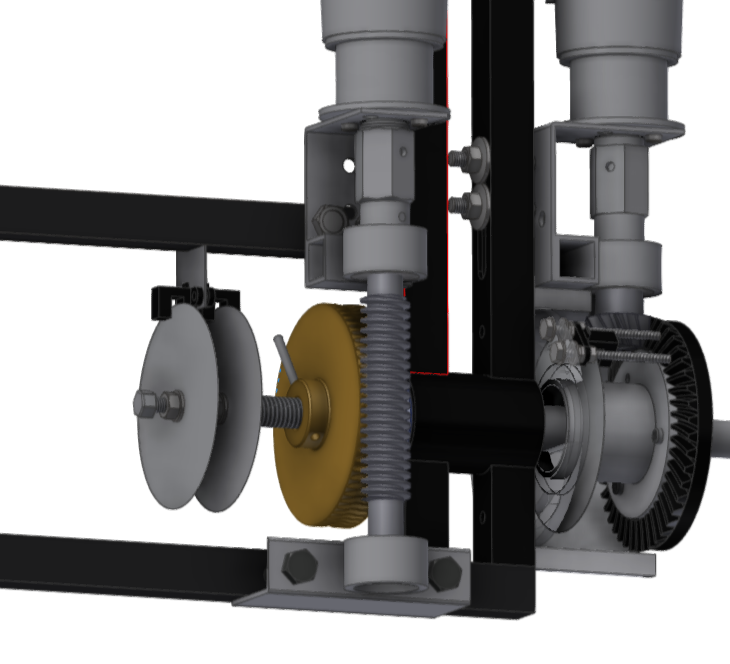
ForgóTalp4

ForgóTalp3

ForgóTalp2

KisKerék

NagyKerék



Null átmenti érzékelő tárcsája

Inkrementális szenzor tárcsa

# Elért eredmények, magvalósítások:

* AutodeskInvnetor 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
* H-hídak megépítése
* Grafikus vezérlőfelület elkészítése
* Nyáktervezés Altium tervező programban

# Következtetések:

A kivitelezés során sok ojan apró hibára rábukkantam, amelyek jó alapot nyújtanának egy a jövőben továbbfejlesztési lehetőségre, elsősorban a mechanikai rendszert kellene átalakítani. A rendszeren kívül levő motorokat be kellene vinni a vázon belűre. A lánctalpakat is átkellene alakítani, mert nem fognak megfejleni a kültéri követelményeknek csak sajnos a keret csak ennyire volt elég. A rendszer vezérlő magja az FPGA rendszer az szerintem jó választás volt, mert nagyon jo alapot nyújt mind a szoftveres mind a hardveres továbbfejlesztési lehetőségekre. A sebesség és a pozíció szabályozok meglátásom szerint beváltak, egyedüli gond a sebesség mérésével van. A sebesség mérő modult még ki kellene egészíteni, hogy kis sebességekre is jól mérjen, jelenleg alacsony sebességen a mérés elég zajos.

Ami a rendszer működése szempontjából sok előnyt jelentene DC motorok áramának a mérése, amely segítene a szabályzásban és a védelemben is.

A rendszer energia ellátására mindenféleképen minimum két független akumlátor lenne szükség amiatt, hogy a teljesítmény elektronika es a digitális elektronika külön tápforrásról kapja az ellátást olyan megfontolásból, hogy a digitális áramkörök prioritást élve zenek más elemekkel szemben. Ha a rendszert hosszabb időre szeretnénk működtetni folytonosan, akkor meg integrálni kellene egy energiaforrást például egy napelem cellát, amely biztosítana energia utánpótlást adott időn belül.

Kommunikációs összekötetés is bevált a TCP protokollal történő adatcsere, a router elősegíti a további elemek integrálását a rendszerbe például egy robotkar, amellyel tudnánk a kapcsolatot tartani Rotteren keresztül protokollon keresztül és az FPGA rendszer is elérné.

Az inkrementális szenzorok, amelyeket én késztetem beváltak, amelyek nagyon olcsón tudtam előállítani a piaci árhoz viszonyítva, fejlesztés szempontjából a tárcsák felbontása is növelhető lenne a tárcsák átmérőjének megnövelésével, és a jobb minőségű lézeres nyomtató használatával.

Összességében tekintve a rendszer nagyon jó szoftver és az elektronika fejlesztésére, a piaci ár alatt van jóval a rendszer előalításához szükséges pénz, ahhoz képest hogy ha egy kész rendszert kelet volna megvásárolni, de olcsó dolgoknak is ára van nem állná meg olyan jól a terepen a helyét.

# 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] | xilinx, „http://www.xilinx.com,” [Online]. Available: http://www.xilinx.com/support/documentation/application\_notes/xapp1026.pdf. [Hozzáférés dátuma: 10 6 2015]. |
| [10] | intersil, „http://www.intersil.com/,” [Online]. Available: http://www.intersil.com/content/dam/Intersil/documents/hip4/hip4082.pdf. [Hozzáférés dátuma: 11 6 2015]. |
| [11] | S. labs, „http://www.silabs.com/,” [Online]. Available: http://www.silabs.com/Support%20Documents/TechnicalDocs/AN486.pdf. [Hozzáférés dátuma: 11 6 2015]. |
| [12] | D. P. I. J. K. Kozłowski, „Modeling and control of a 4-wheel skid-steering”. |
| [13] | 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. |
| [14] | 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