**Sapientia EMTE**

**Műszaki és Humántudományok Kar, Marosvásárhely**

**Villamosmérnöki Tanszék**

iRoobo

Témavezető tanár: Szerző:

Dr. Brassai Sándor Tihamér Gábor Szabolcs-László

adjunktus Automatizálás IV. év

XVI. TDK Temesvár,

2015

Tartalom jegyzék

[2 Robot mechanikai felépítése 5](#_Toc419222373)

[3 Hardever 6](#_Toc419222374)

[3.1 FPGA Rendszer Felépítése 6](#_Toc419222375)

[3.1.1 Zybo FPGA fejlesztőlap 8](#_Toc419222376)

[3.1.2 Spartan3e FPGA fejlesztőlap 9](#_Toc419222377)

[3.1.3 Kommunikációs protokollok 10](#_Toc419222378)

[3.1.3.1 SPI kommunikációs protokoll: 10](#_Toc419222379)

[3.1.3.2 Ethernet 11](#_Toc419222380)

[3.2 Szenzorok 11](#_Toc419222546)

[3.2.1 Inkrementális Szenzor 11](#_Toc419222547)

[3.2.2 Relatív Pozíció mérése Inkrementális adó segítségével 12](#_Toc419222548)

[3.2.3 Szögsebesség mérése Inkrementális adó segítségével 13](#_Toc419222549)

[3.3 SZABÁLYOZÓK 14](#_Toc419222550)

[3.3.1 Diszkrét Hardveres PID szabályozó 14](#_Toc419222551)

[3.3.1.1 Megvalósítás System Generátorban 15](#_Toc419222552)

[3.3.1.2 Szoftveres Szimulációs eredmények 16](#_Toc419222553)

[3.3.2 Pozíció Szabályozása 17](#_Toc419222554)

[3.3.2.1 A szabályozó felépítése: 17](#_Toc419222555)

[3.4 Sebesség és pozíció szabályozót tartalmazó IP mag generálása System Generator-ban 18](#_Toc419222556)

[4 Robot Modell 20](#_Toc419222557)

[5 Következtetések: 23](#_Toc419222559)

[6 Szakirodalom : 24](#_Toc419222560)

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.

# 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.

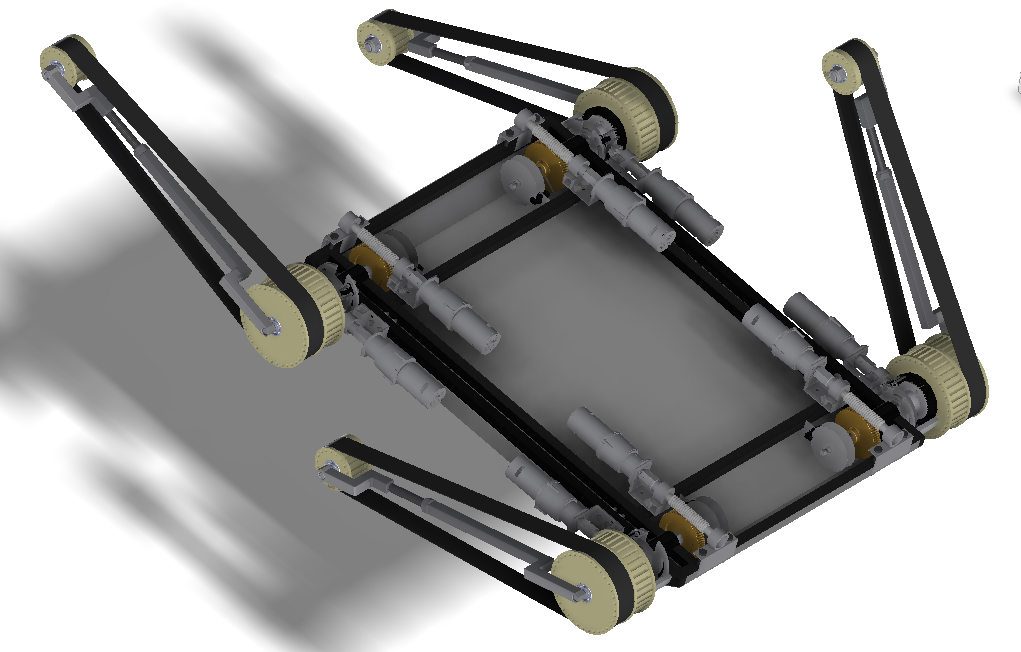
DC motorok betáplálási feszültsége: 12V, maximális terhelés alatt 15A áramot is felvehet.

A 7.1 képen látható Kup kerék áttételen keresztül hajtjuk meg a lánctalpat, a talpak szögének változtatására 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.



Kép. 2.2 Robot vázának Inventoros 3D Képe

ForgóTalp1

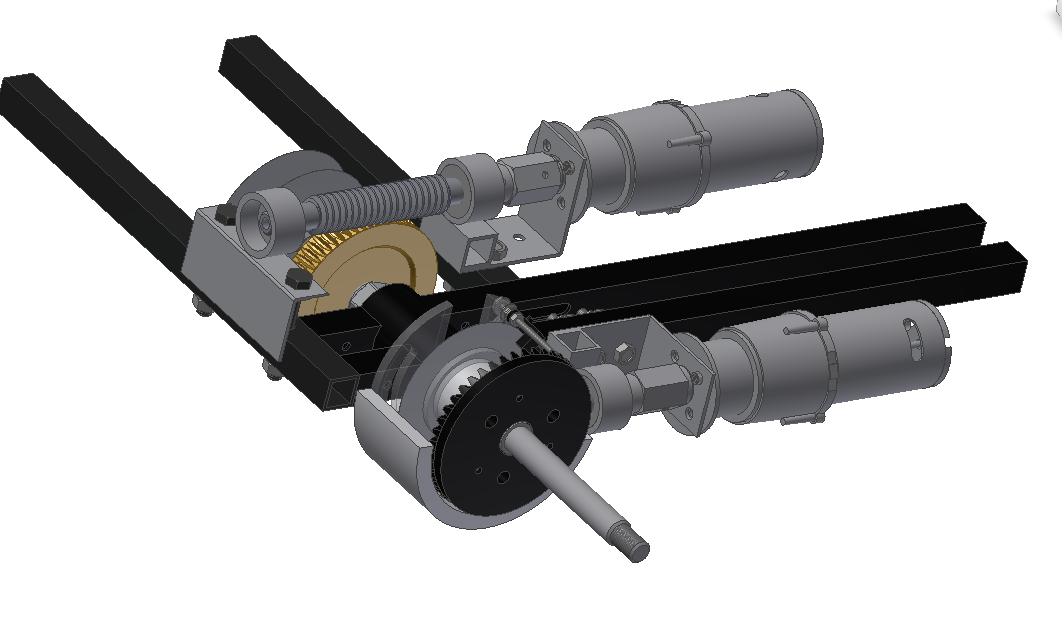
ForgóTalp4

ForgóTalp3

ForgóTalp2

KisKerék

NagyKerék



Orsó áttétel 40:1 arányban

Kupkerép áttétel 6:1 arányban

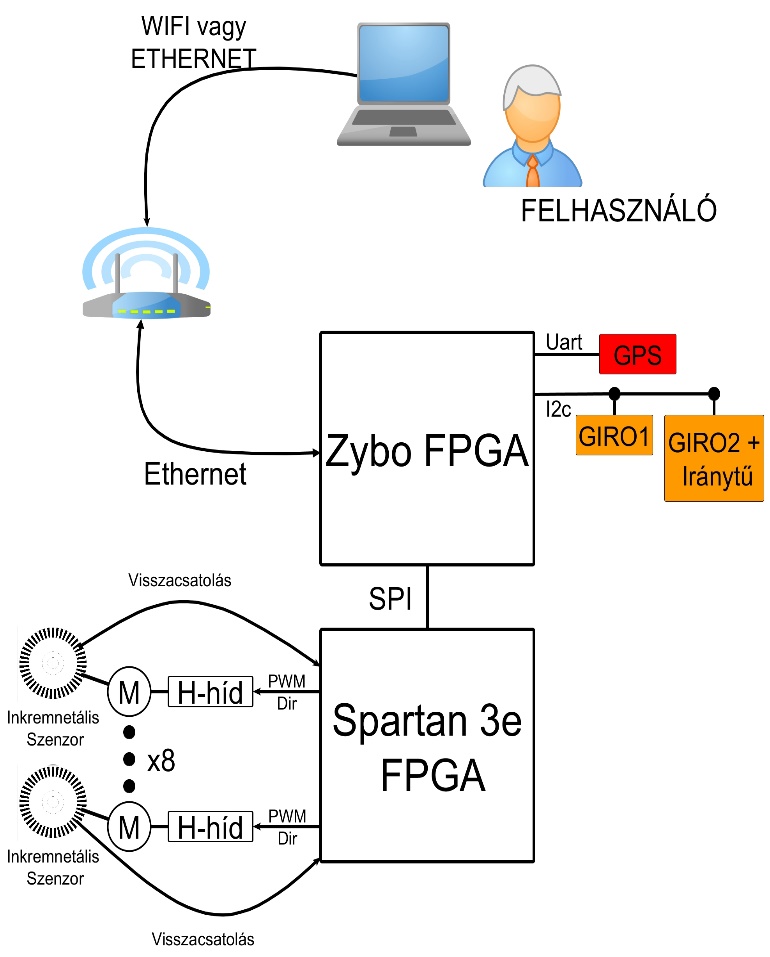
Sebesség mérő inkrementális tárcsa és szenzor

Kép. 2.2 Az alváz negyede

# Hardever

## 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.



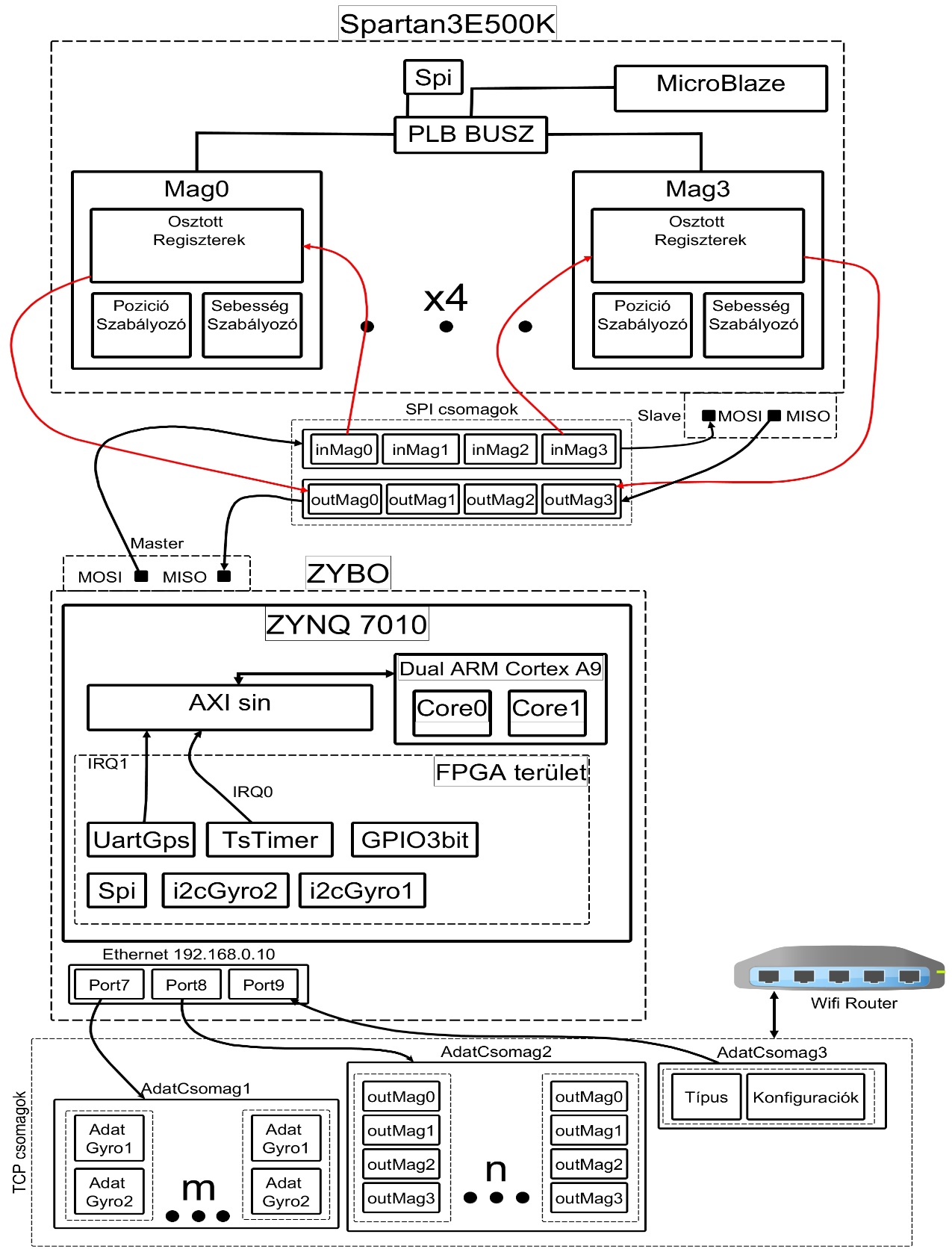
Kép. 3.1 rendszer elvi felépítése

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 Zybo fejlesztőlappal.



Kép. 3.2 Kommunikációs csomagok és az FPGA áramkörökbe programozott modulok elvi felépítése

### 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.

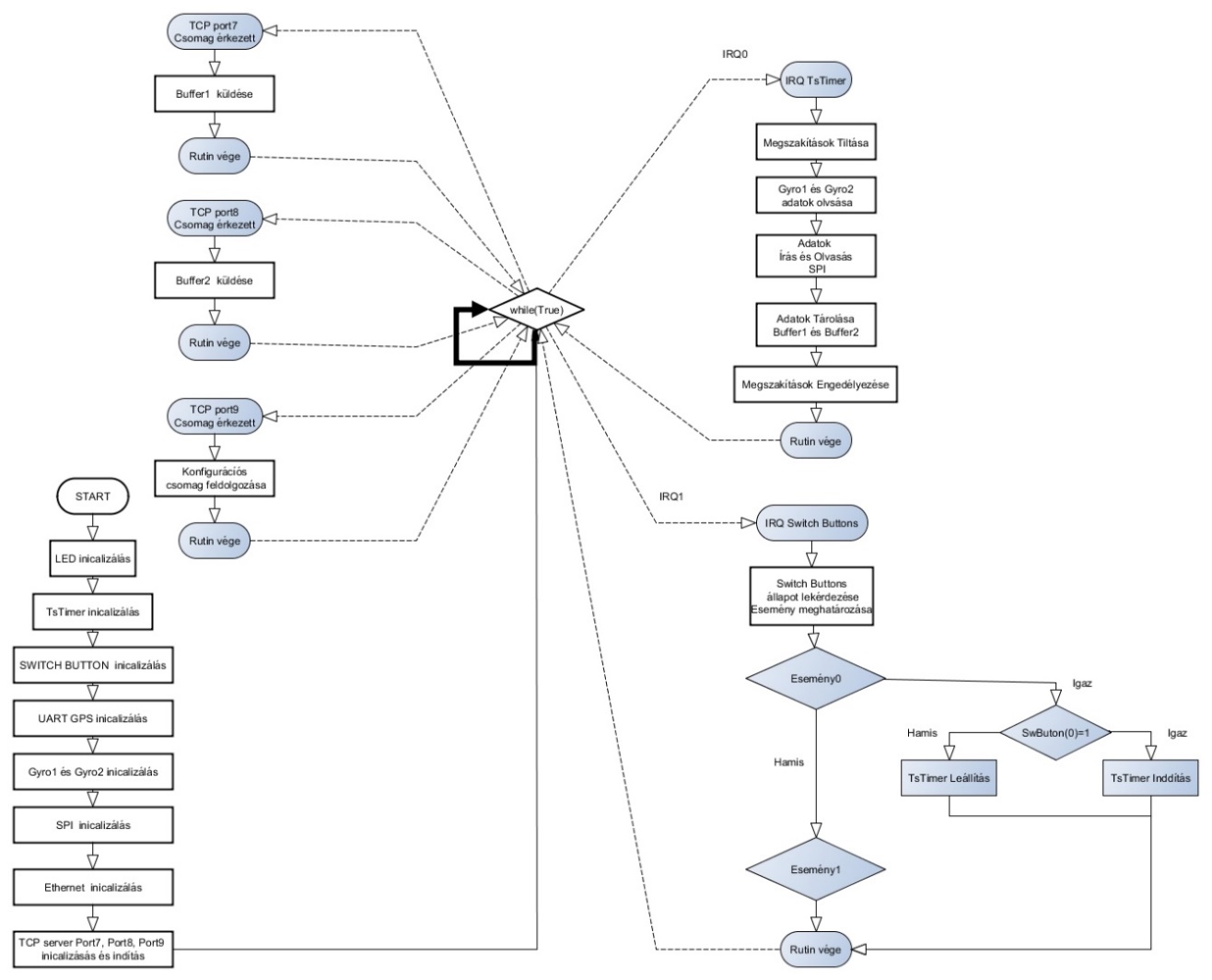
Miután végzett a Core1 a matematikai számításokkal az SPI kommunikáción keresztül elküldi a szabályozók referencia értékeit a Spartan fejlesztőlapnak.

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.

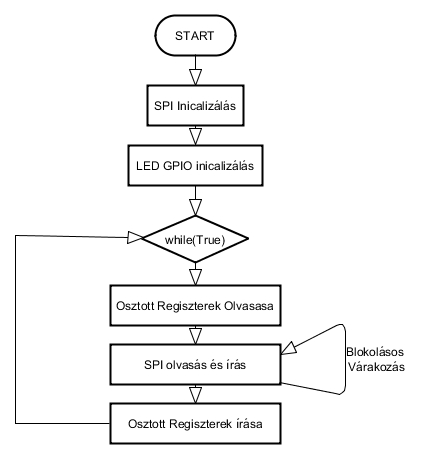
A program tartalmaz három TCP szervert is, amelyek a megadott port számon várják a kéréseket, minden kérés beérkezte után elküldik a pufferekben tárolt adatokat a kérést küldő kliensnek, és kiürítik a puffereket.

### Spartan3e FPGA fejlesztőlap



Kép. 3.3 ZYBO Core0 program folyamat árbája

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



Kép. 3.4 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. 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 spi kommunikáció a 3.2.képen látható SPI csomagszerkezeten keresztül történik.

**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 TsH0;

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.

**Config regiszter**:

|  |  |  |
| --- | --- | --- |
| Bit | =1 | =0 |
| 0 | Letiltja a pozíció szabályozójának a PWM kimenetét. | Engedélyezi a pozíció szabályozójának a PWM kimenetét. |
| 1 | Null átmenet engedélyezése a pozíció mérésénél | Null átmenet tiltása a pozíció mérésénél |
| 2 | Letiltja a sebesség szabályozójának a PWM kimenetét. | Engedélyezi a sebesség szabályozójának a PWM kimenetét |
| 3 | Megfordítja a pozíciómérő számolási irányát. | Megfordítja a pozíciómérő számolási irányát |
| 4 | Megfordítja a sebességmérő számolási irányát. | Megfordítja a sebességmérő számolási irányát. |

Tábla. 1 A konfig regiszter bitjeinek a funkciója

#### 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.



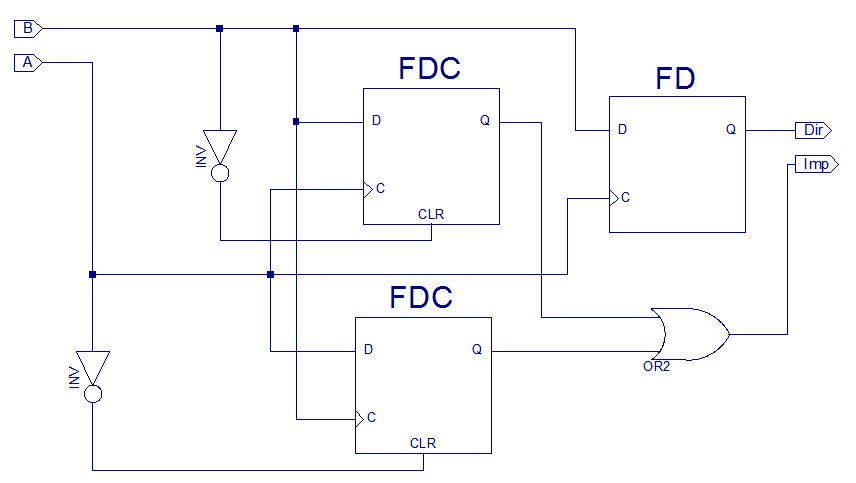




## Szenzorok

### Inkrementális Szenzor

A rendszeren megtalálható 8 inkrementális szenzor, amelyeknek adatait FPGA rendszer dolgozza fel. Az elkészített modul VHDL programnyelven készült a 3.5 képen látható kialakítás szerint, BLACK BOX modul segítségével integráljuk a Simulink környezetbe.

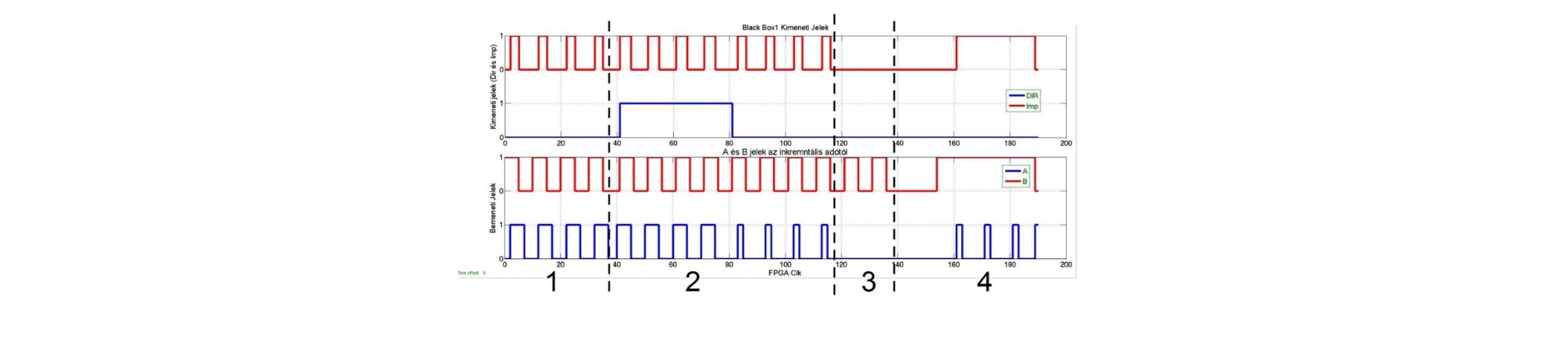


Kép. 3.5 Black Box1 modul belső felépítése

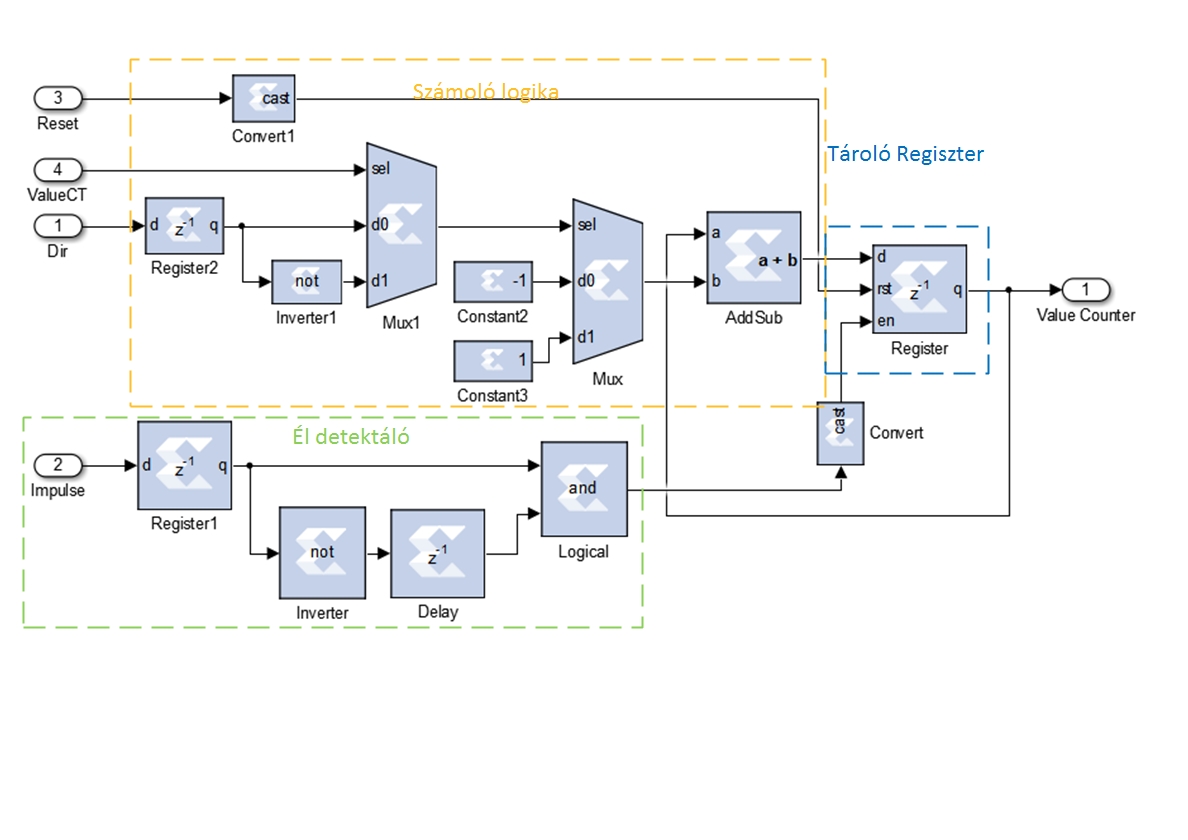
A bementi jelek négy kategóriába sorolhatók:

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 generálás 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. 3.6 Szimulációs eredmények a lehetséges bemenetekről az Black Box1 modulba



### Relatív Pozíció mérése Inkrementális adó segítségével



Kép. 3.7. CounterPosition Belső Felépítése System Generátorban

A 3.7.képen látható modul, feldolgozza a 3.5.képen megvalósított modul kimeneti jeleit. 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 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 Null bemeneten impulzus érkezik. Azt mondhatjuk, hogy a pozíciót a 0 állapothoz mérjük, amely a Null átmeneti tárcsa határozza meg.

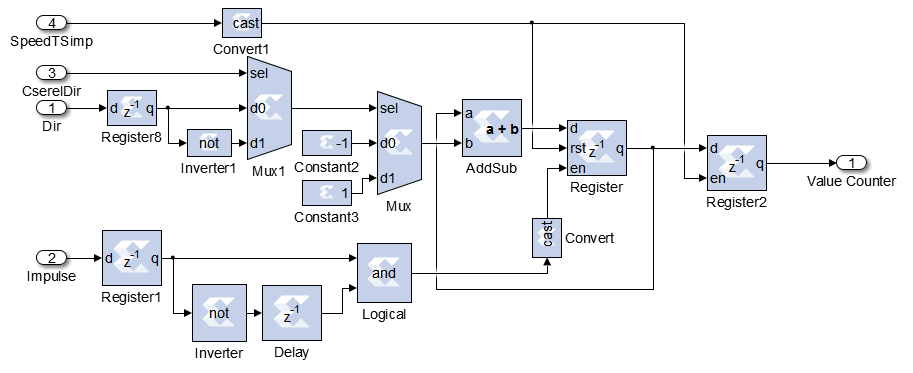
A számláló logika feladata, hogy növelje vagy csökkentse eggyel a regiszter tartalmát, annak függvényében, hogy a Dir jel milyen értékű. A ValueCT bemenet segítségével meg tudjuk fordítani a számolás irányát.

### 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. A SpeedTsimp, bementen időközönként impulzusok érkeznek. A rendszer megszámolja a két impulzus között elhaladt inkrementális tárcsától érkező impulzusokat. A sebességet a következő összefüggések alakíthatjuk át RPM be:

Kép. 3.17 Sebesség mérő modul felépítése



## SZABÁLYOZÓK

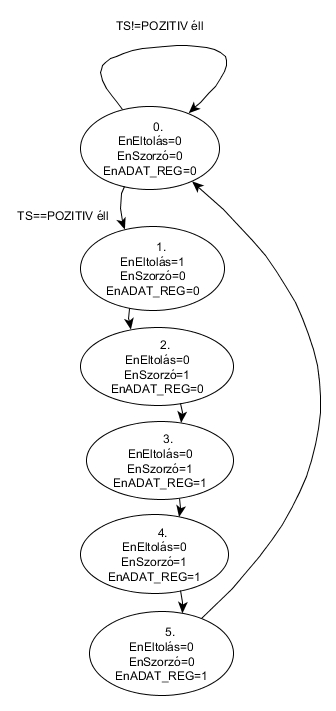
### Diszkrét Hardveres PID szabályozó

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

Forrás: [1]

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.



Kép. 3.9 Állapot automata, amely leírja a Diszkrét PID szabály ózót

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 várakozik a 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 .
* 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. 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 1.3 képen a SZORZÓ modul a két szelekciós multiplexertől kapott értéket összeszorozza, aztán hozzáadja az ADAT\_REG regiszter értékéhez.

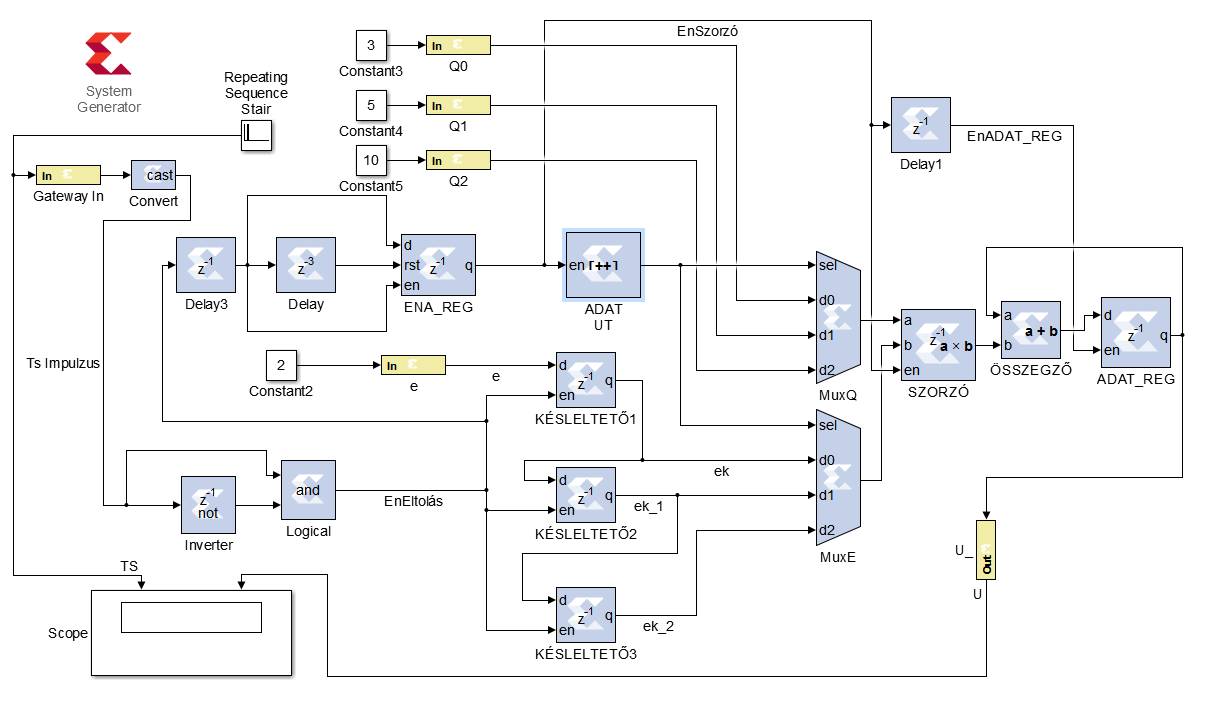
Minden modulértéke szaturálódik abban az esetben, ha túlcsordulna akár negatív vagy pozitív irányba, így elkerülhetjük azt is, hogy az integráló tag változatlan hiba bemenete esetén túlcsorduljon és felborítaná a rendszer működését.

A 3.10 képen a KÉSLELTETŐ regiszterek állítják elő múltbeli hiba értékeket, 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 utat kiválasztó számláló csak a regiszterek elcsúsztatása után indul el, amelyet a fent látható késleltető elemek segítségével valósítunk meg.

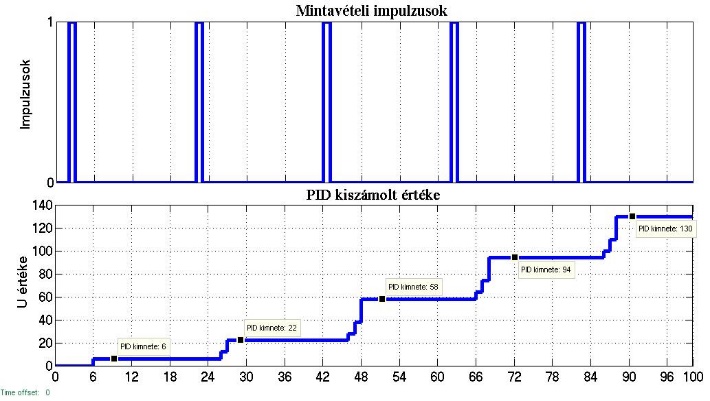
Az él detektáló elemet egy tagadó kapó (inverter) valamint egy és kapú (logical) biztosítják, mégpedig úgy, hogy figyeljük egyazon jel előbbi periódusban az értéket és összehasonlítva a két értéket tudjuk detektálni az élet.

Kép. 3.10 A PID felépítése System Generatorban



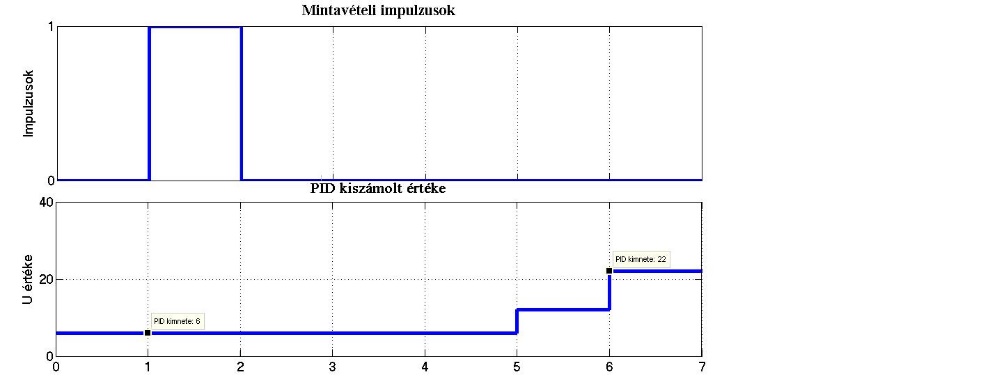
#### Szoftveres Szimulációs eredmények

A szimulációk során a számítások eredményét ellenőriztem le, amelyeket SYSTEM GENERATOR-ban végeztem el, az eredményeket majd összehasonlítottam a manuálisan számolt értékekkel. Bemeneti paraméterek: , a bemenet konstans: A fenti 3.11 képen látszik a szükséges 6 órajel a számítások elvégzésére. Megjegyzésként a szabályozó negatív bemeneti értékekre is működőképes.



Kép. 3.11 Szimulációs eredmény amely tükrözi a konstans bementre a számolási lépéseket

Kép. 3.12 PID minimális periódusa



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 szabályozó minimális mintavételi periodusa 120ns

### Pozíció Szabályozása

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:

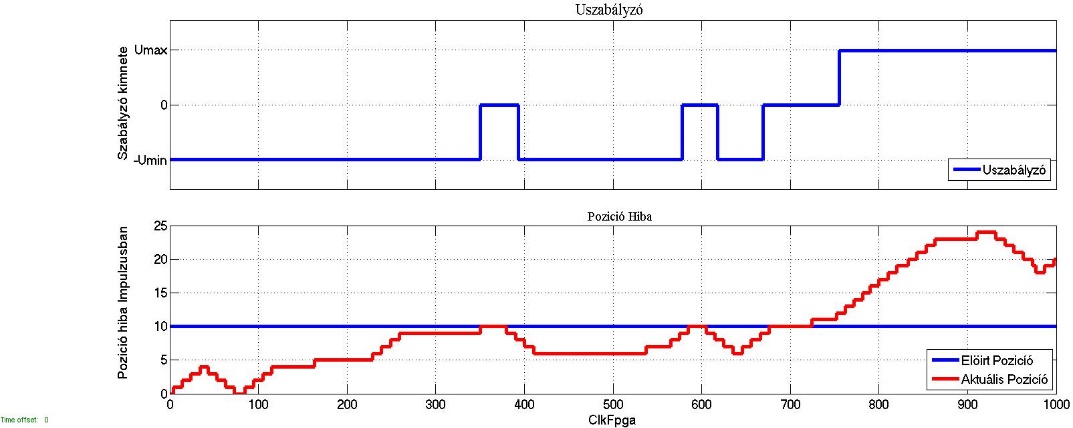
Elmondható a kimeneti szabályozó jel függ a hiba értékétől. A mechanikai rendszerben kotyogás van, és az ebből származó zajokat szeretnénk kiszűrni úgy, hogy ha a mechanizmus a megfelelő pozícióban van, akkor egy tartományban a szabályozót érzéketlené tesszük a bemenetre mindaddig, amíg a hiba ki nem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

#### A szabályozó felépítése:

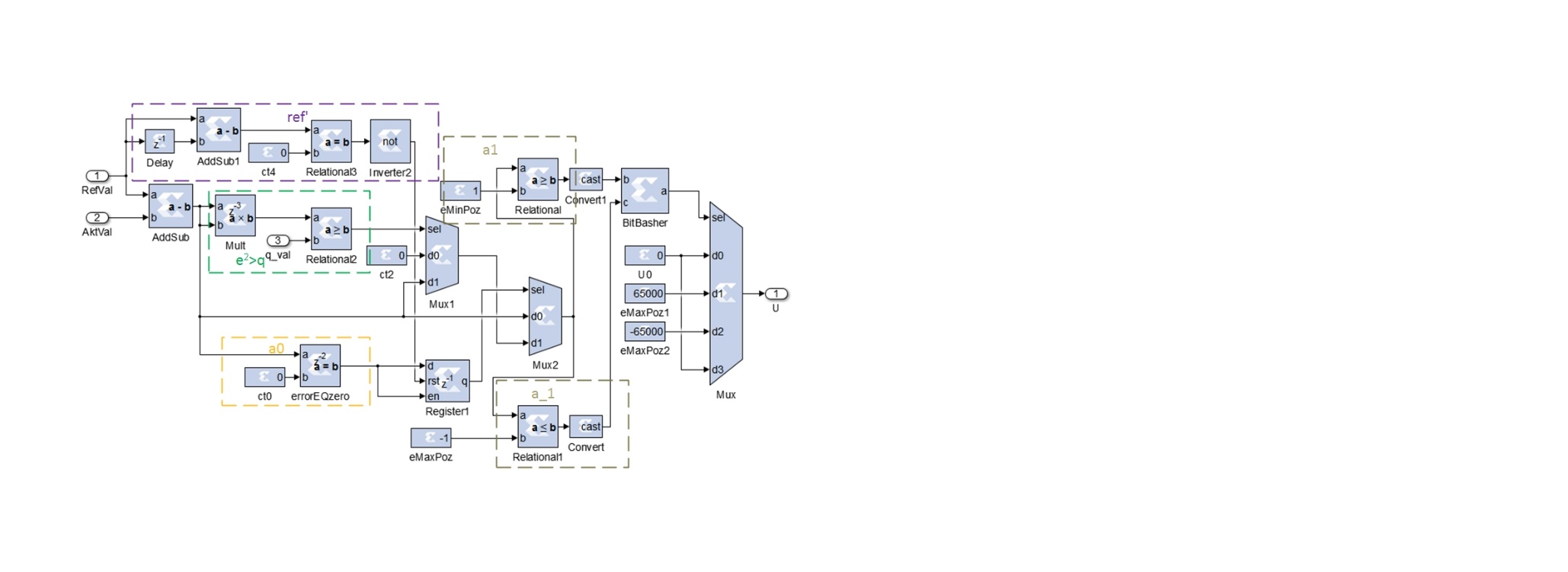
Funkcionalitás szerint öt csoportba sorolhatjuk az alábbi 1.8 képen látható elemeket:

* ref’-a referencia értéket figyeli és minden órajel periódusban megvizsgálja, hogy van váltózás az értékben.
* – négyzetre emeli a hibát és összehasonlítja az általunk megadott q értékével.
* MUX – az U kimeneti jel állapot multiplexere.
* MUX1 –az manipulált hiba multiplexere
* MUX2 –hiba multiplexere

A 3.14 képen 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. 3.14 a pozíció szabályozó bemenete (pozíció hiba), valamint a Szabályozó kimenti jele.



Kép. 3.13 A Pozíció szabályozó System generátoros felépítése

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

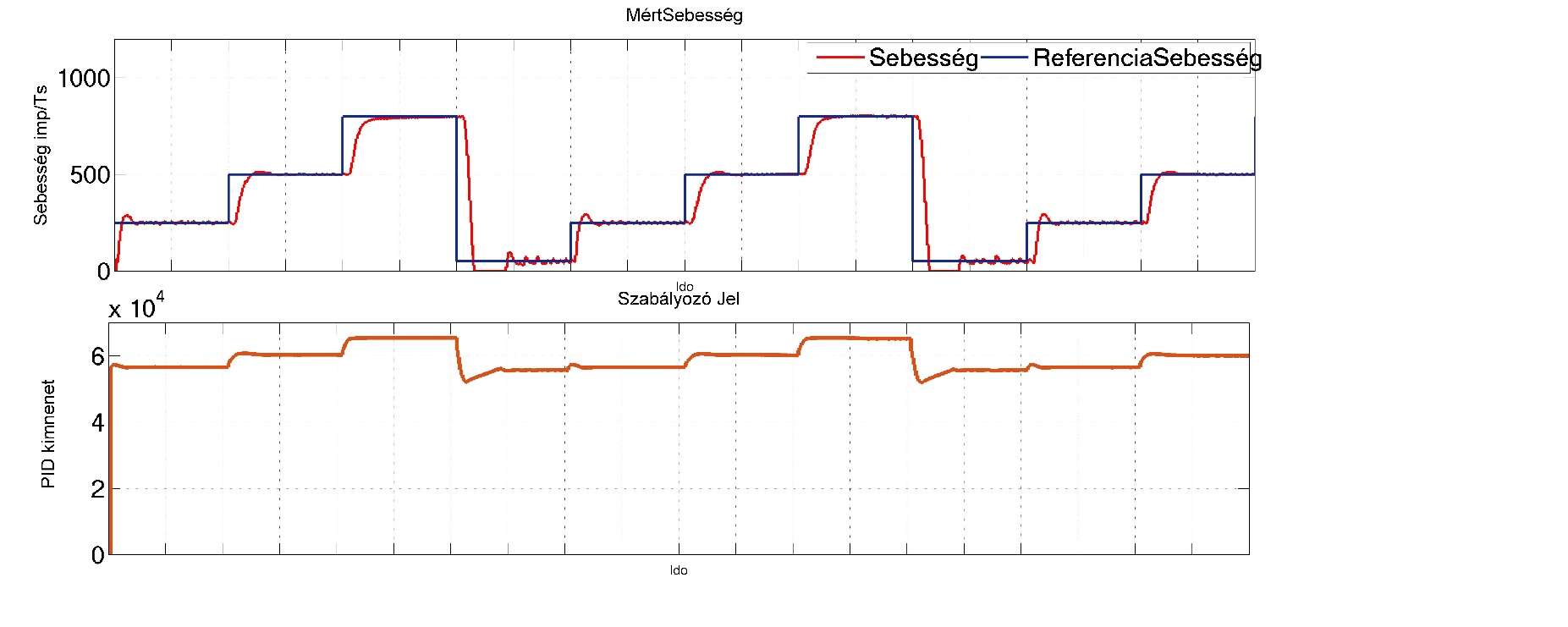
Az eddig megépítet, modulokat összekötve alkotunk egy nagy modult, amely tartalmazza a sebesség és a pozíció szabályzási hurkokhoz szükséges elemeket. A szabályozók referencia bemeneteit osztott regiszterekkel írjuk elő microBaze processzor segítségével. Mindkét szabályozó tartalmazza a PWM generátort, és a sebesség mérő modult (counterSebesseg). A két inkrementális érzékelő jeleit egyetlen modul segítségével dolgoztam fel (Inkremntális\_ Jelfeldolgozó\_2\_érzékelő), amely megfelel két BlackBox1 nevű modulnak az inkrementális szenzort tárgyaló fejezetből.

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

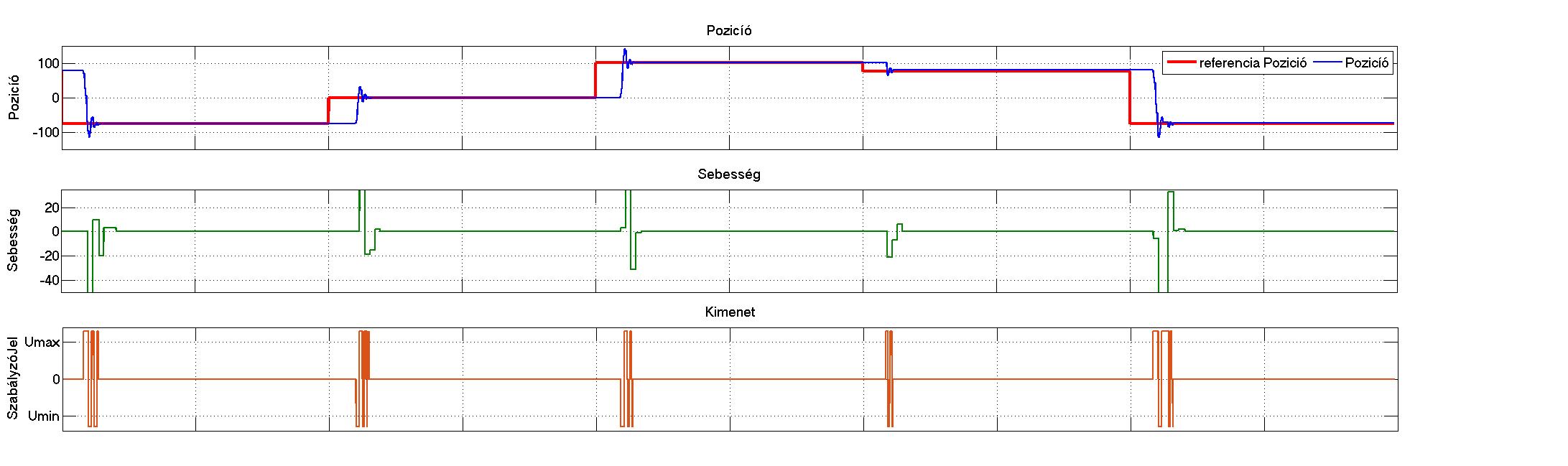
**Mérések:**

A méréseket egy 2700 RPM maximális fordulatszámmal rendelkező Dc motor segítségével végeztem el. A mintavételi idő 0.02s. A 3.7 képen látható a PID szabályozó, és a kimeneti értéke. A kimneti értéke -65535 és 65535 közöti egész szám, amélyet a PWM generátor tud értelmezni. Megfigzelhető hogy a segesség mérése alacsony fordulaton elég zajos illetve a PID kimeneteében is tulővések vanak



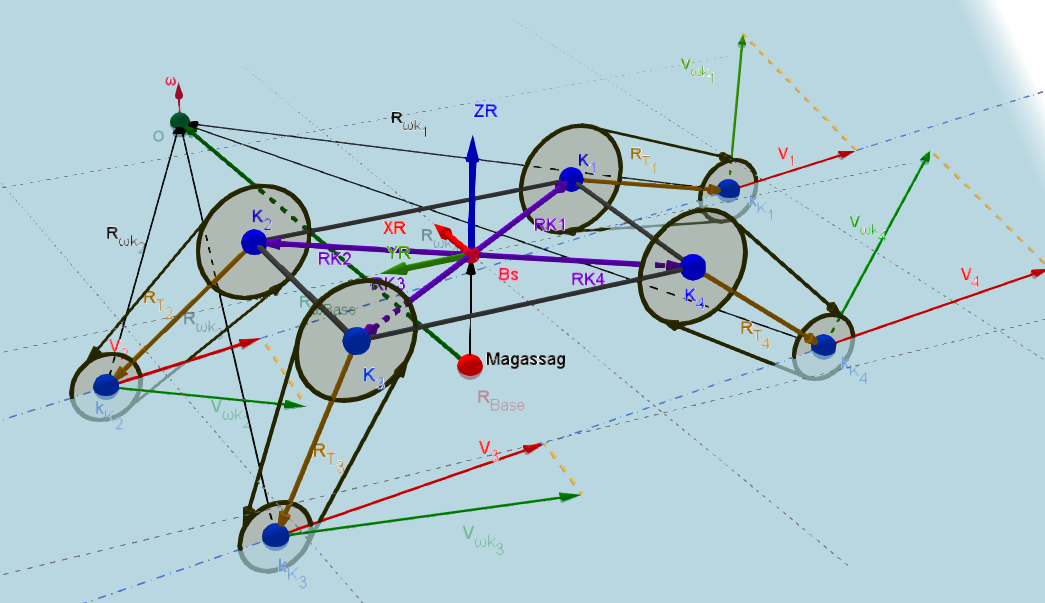
Kép. 3.15 Sebesség Szabályozás PID szabályzóval

A pozicíó mérésénél megfigyelhető, tulővés illetve oszcilácíó, az amilyat van hogy mérőstandon levő motor nem tartalmazza az orsos átételt.

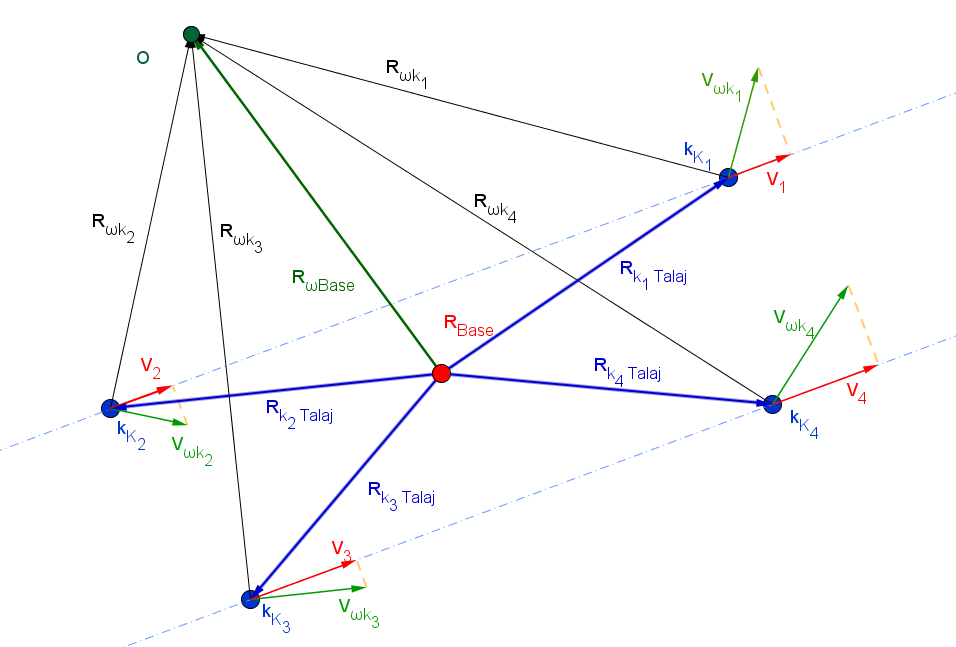


Kép. 3.16 Pozíció szabályzó

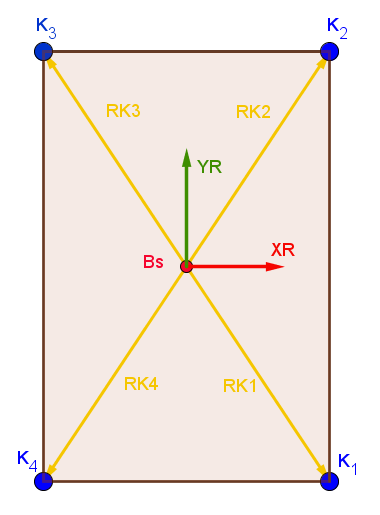
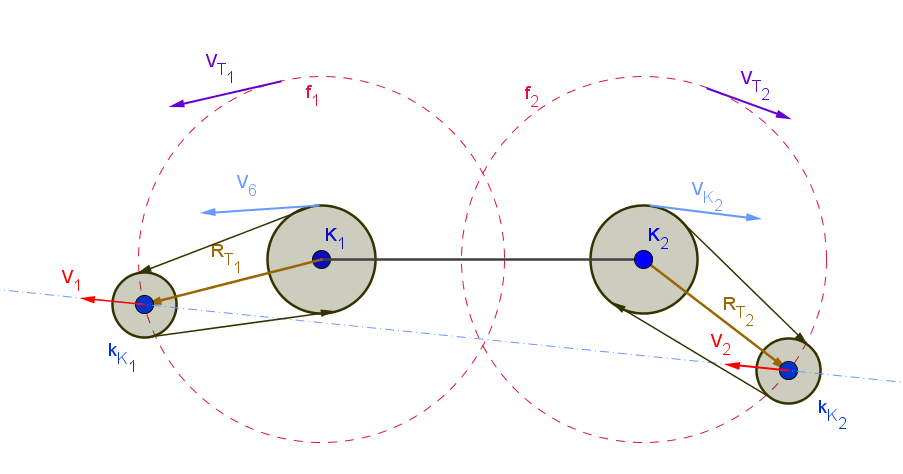
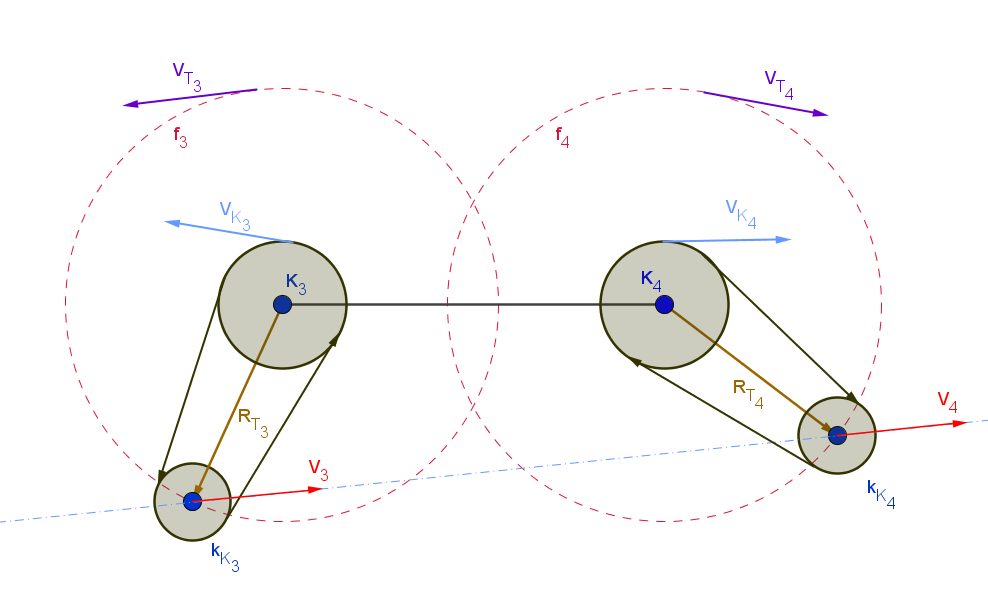
# Robot Modell



Kép. 4.1 Robot 3D vektorábrája



Kép. 4.2 a talaj sikjábban nézve a robot kerekeinek a sebességét, amikor O pont körüli forgást végez, Forrás: [2]



Kép. 4.3 Oldalnézetek és Felülnéztet, jelölések szemléltetése

**Jelölések**:

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

Az 6.2 kép 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:

# Következtetések:

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

# Szakirodalom :

1. Márton Lőrinc (2009): PID típusú szabályozók In Irányítástechnika. Scientia Kiadó, Kolozsvár.
2. Maciej Torjancki (2015):Dynamic Model of a Four-Wheeld Mobile Robot for Control Applications- A Three-Case Study In Intelligent Systems’2014 . Springer Kiadó,Switzerland.
3. <https://wiki.geogebra.org/en/Manual>
4. http://help.autodesk.com/view/INVNTOR/2014/ENU/
5. <http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf>
6. <https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1198&Prod=ZYBO>
7. <http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_5/sysgen_gs.pdf>
8. <http://www.xilinx.com/support/documentation/ip_documentation/xps_spi.pdf>
9. http://japan.xilinx.com/support/documentation/ip\_documentation/axi\_iic/v2\_0/pg090-axi-iic.pdf