**Sapientia EMTE**

**Műszaki és Humántudományok Kar, Marosvásárhely**

**Alkalmazott Társadalomtudományok Tanszék**

<SZAKDOLGOZAT CÍME>

szakdolgozat

Szakirányító tanár:

Dr. Brassai Sándor Tihamér

<egyetemi tanár/egyetemi előadótanár/adjunktus>

Végzős hallgató:

Gábor Szabolcs-László

Automatizálás IV. év

**2015**

[*1* SZABÁLYZÓK*:* 4](#_Toc417007778)

[***1.1*** ***Diszkrét Hardveres PID szabályozó*** 4](#_Toc417007779)

[1.1.1 Megvalósítás System Generátorban 6](#_Toc417007780)

[1.1.2 Szoftveres Szimulációs eredmények 7](#_Toc417007781)

[1.1.3 Q paraméterek számolása alapján. 9](#_Toc417007782)

[1.2 Pozíció Szabályzása 9](#_Toc417007783)

[1.2.1 A szabályzó felépítése: 9](#_Toc417007784)

[1.2.2 Szabályzó szimulálása 10](#_Toc417007785)

[1.3 Hardveres mérések 11](#_Toc417007786)

[1.3.1 Dc motor sebesség szabályzása mérőstandon 11](#_Toc417007787)

[1.4 Sebesség és pozíció szabályzót tartalmazó IP mag generálása System Generatorban 12](#_Toc417007788)

[2 Szenzorok 15](#_Toc417007789)

[2.1 Inkrementális szenzor 15](#_Toc417007790)

[2.1.1 Optikai inkrementális vevő felépítése 16](#_Toc417007791)

[2.2 Inkrementális érzékelő jeleinek a feldolgozása FPGA áramkör segítségével 18](#_Toc417007792)

[2.2.1 Szimuláció System Generatorban 19](#_Toc417007793)

[2.2.2 Pozíció mérése Inkrementális adó segítségével 19](#_Toc417007794)

[2.2.2.1 Hardveres mérések 21](#_Toc417007795)

[2.2.3 Szögsebesség mérése Inkrementális adó segítségével 22](#_Toc417007796)

[2.2.3.1 Szimuláció: 23](#_Toc417007797)

[2.2.3.2 Hardveres mérések 24](#_Toc417007798)

[3 Beavatkozó elemek: 27](#_Toc417007799)

[3.1 Pwm Generátor megvalósítása FPGA áramkörön System Generator környezetben. 27](#_Toc417007800)

[3.1.1 Megvalosítás 27](#_Toc417007801)

[4 Elektronika 29](#_Toc417007802)

[4.1 Digitális Elektronika 29](#_Toc417007803)

[4.1.1 FPGA Rendszer Felépítése 29](#_Toc417007804)

[4.1.1.1 Tömbvázlata 29](#_Toc417007805)

[4.2 Feladatok Elosztása 31](#_Toc417007806)

[4.2.1 Zybo fejlesztőlap 31](#_Toc417007807)

[4.2.2 Spartan fejlesztőlap 32](#_Toc417007808)

[5 Robot Modell 33](#_Toc417007809)

[6 Robot Mechanikai Felépítése 36](#_Toc417007810)

Bevevezető

A dolgozat keretein belül leirom egy mobilis kültéri terepen is használható robot megépitéséhez szükséges lépéseket és modulokat.

A tervezést a mechanikai rendszerel keztem Autodesk Inventor segitségével több változatot is megterveztem amedig eljutottam a dilgozatba tárgyalt mechanikai strukturához. A tervezés után nekilátam a mechanikai rendszer kivitelezésének, beszerztem a szükséges eszközöket a vaskereskedésből (pl: fémidomok, csapágyak), párhuzamosan terveztem a rendszer többi részét is, letiztászam a feladatokat amik szükségesek a rendszerhez.

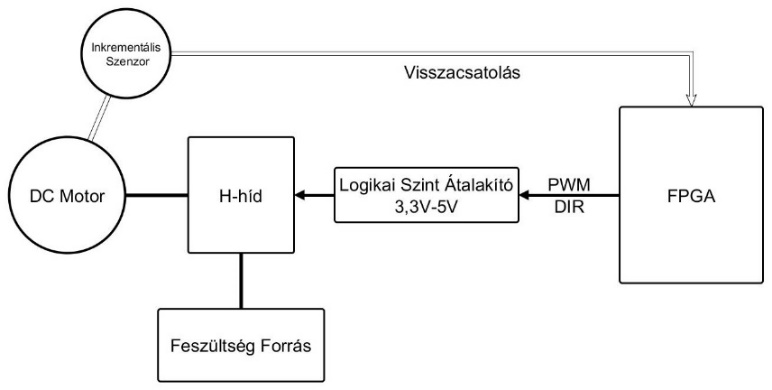
A szoftver és digitális hardver fejlesztésére FPGA rendszert választottam mert könnyen fejleszhető a szoftver és a hardver közösen.

A dolgozatban láthatjuk PWM generátor, PID szabályzó,Pozíció Szabálzó,Inkrementális érzékelő adatainak a feldolgozó modul, megvalósításást System Generátor környezetben.

Inkrementális tárcsát hogyan tervezünk meg és vitelezünk ki.

# SZABÁLYZÓK:

A DC motor és a hozzá csatolt mechanizmus pozíció és a sebesség szabályzási struktúrájánál az egyhurkú feszültségben vezérelt, PWM beavatkozó jellel és inkrementális adóval megvalósított szabályzási hurkokat valósítottam meg.



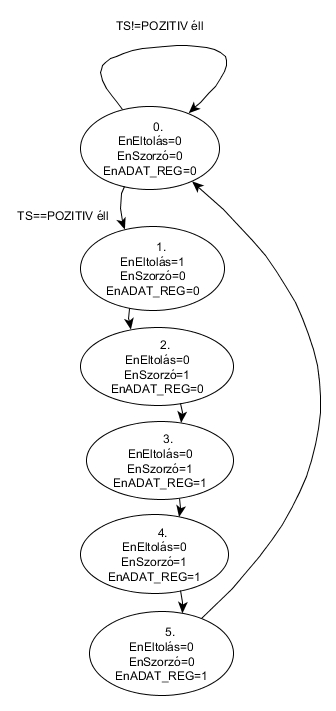
Kép. 1.1 Látható a Pozíció és a sebesség szabályzási hurok elvi strukturális felépítése

## Diszkrét Hardveres PID szabályozó

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

Forrás: http://www.ms.sapientia.ro/~martonl/Docs/Lectures/PID\_Szabalyozo.pdf

Az általam elkészített PID szabályzó hardveresen van megvalósítva FPGA áramkörben, a minél kisebb mintavételezési periódus elérése céljából. A fent látható összefüggések segítségével egy adat utas automatát írtam fel, 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, mintavételi periódus, valamint proporcionális erősítés. Az automata öt állapotot tartalmaz. A TS jel, amely biztosítja a mintavételi periódust, minden felfutó élre az automata végigpörög az állapotokon és majd visszatér a kiinduló állapotba.



Kép. 1.2 Állapot automata, amely leírja a Diszkrét PID szabály ózót

Az állapotokban végzet műveletet az FPGA fejlesztő lapon megtalálható órajel frekvenciájára végzi, minden állapoton egy órajel periódus alatt lép át.

Minden állapotban egy jól meghatározott(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 érezik 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 jelet segítségével a SZORZÓ modult aktívája. A szorzás elvégzésére egy órajel periódust vesz igénybe, és a kivetkező periódusban használhatjuk csak az eredményt.

Az állapotokban végzet műveletek:

* 0. állapotban várakozik a felfutó áll érkezésére
* 1. állapotban elcsúsztatja a e regiszterek értékét
* 2. állapotban elindítja a, müveletet.
* 3. állapotban elindítja , és eltárolja a 2. állapotban elindított művelet eredményét.
* 4. állapotban elindítja , és eltárolja a 3. állapotban elindított művelet eredményét.
* 5. állapotban eltárolja a 4. állapotban elindított művelet eredményét.

### Megvalósítás System Generátorban

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

Bemeneti paraméterek a 16bit előjeles egész érték, 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 képen látható 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 így túlcsordulna és felborítaná a rendszer működését.

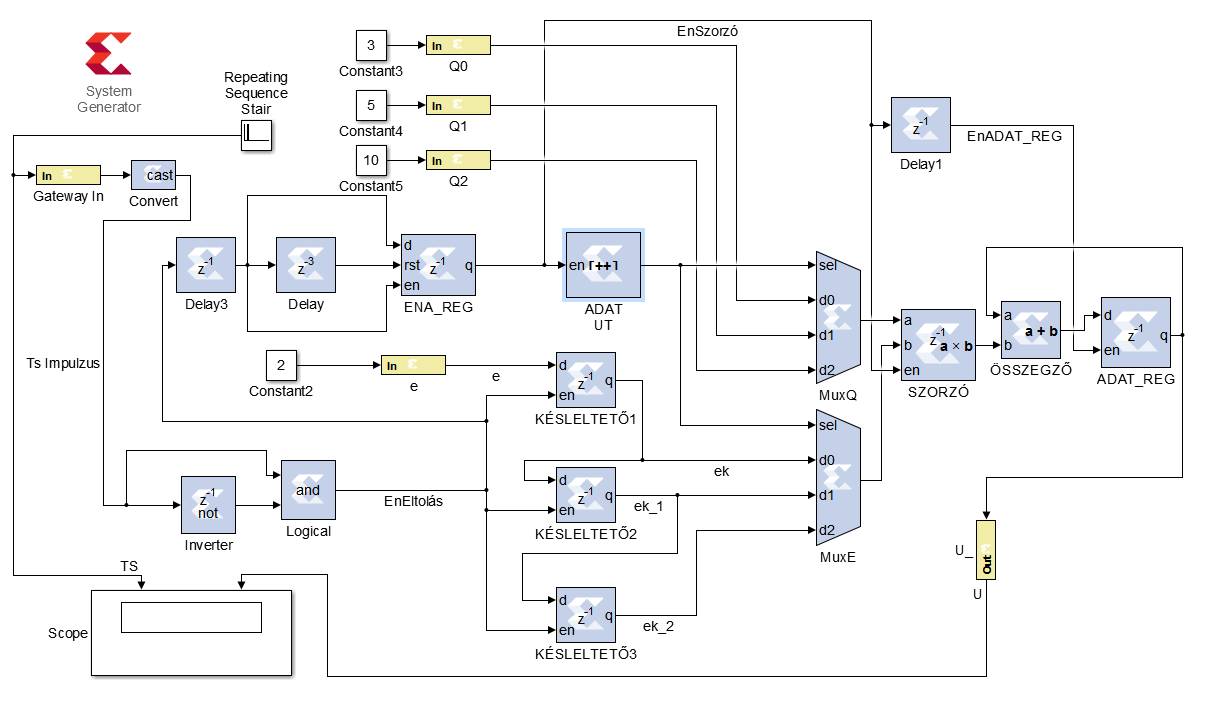
A képen látható KÉSLELTETŐ regiszterek állítják elő múltbeli hiba értékeket, úgy hogy a három regiszter egymás után van láncolva és a felfutó élére a

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

Az állapot számláló csak a regiszterek elcsúsztatása után indul el, amelyet a fent látható késleltető elemek segítségével valósítunk meg.

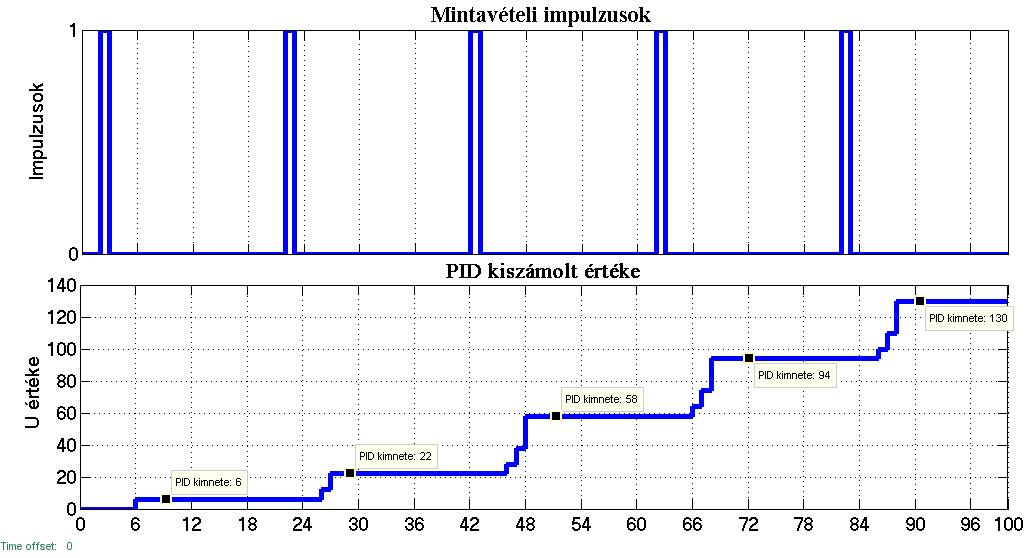
Az él detektáló elemet az Inverter valamint a Logical nevű elemek biztosítják, mégpedig úgy hogy figyeljük egyazon jel előbbi periódusban az értéket és összehasonlítva a két értéket tudjuk detektálni az élet.

Kép. 1.3 A PID felépítése System Generatorban



### Szoftveres Szimulációs eredmények

A szimulációk során a számítások eredményét ellenőriztem le, amelyeket SYSTEM GENERATORban végeztem el, az eredményeket majd összehasonlítottam a manuálisan számolt értékekkel.



Kép. 1.4 Szimulációs eredmény amely tükrözi a konstans bementre a számolási lépéseket

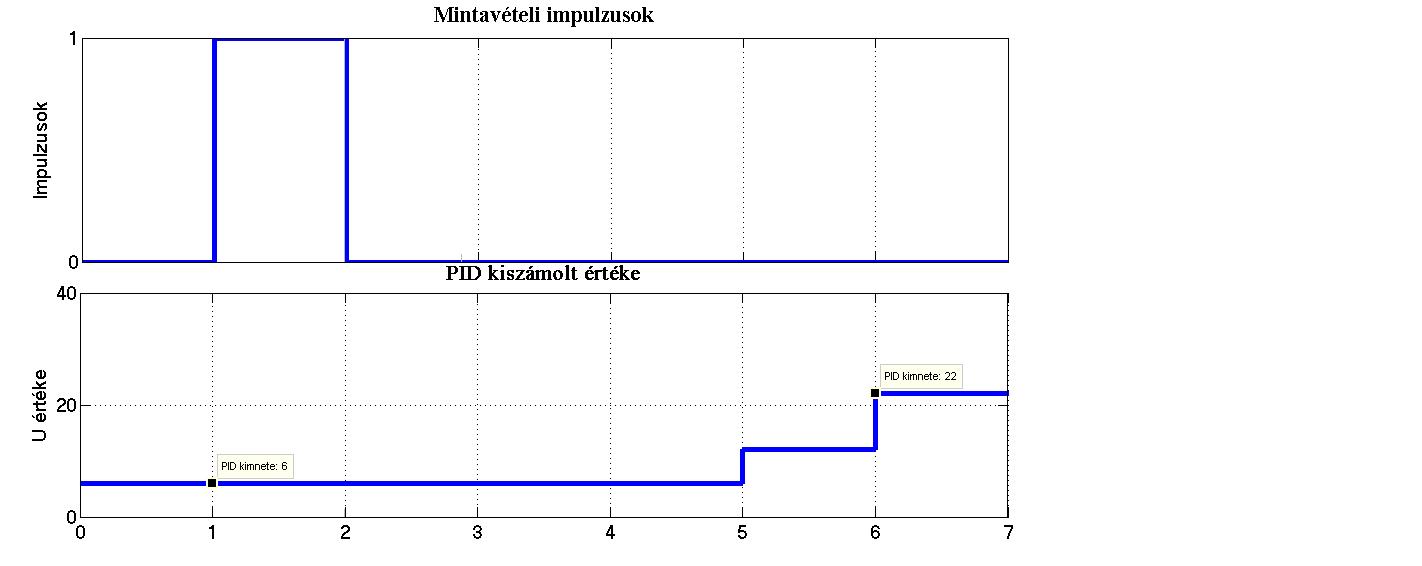
Bemeneti paraméterek: , a bemenet konstans:

A fenti képen 6 órajel látható és a PID kiszámolt értékek a bemenő paraméterekre. Megjegyzésként a szabályzó negatív bemeneti értékekre is működőképes.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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. 1.1.2‑1 Manuálisan számolt értékek a szimuláció ellenőrzése kedvéért

Kép. 1.5 PID minimális periódusa



Következtetések a Szabályzó az elvártaknak megfelelő eredményeket térít vissza. A szimulációkból többek között megfigyelhető, hogy az elindítástól a végső eredmény megjelenéséig 6 órajel periódusra 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 szabályzó minimális mintavételi periodusa 120ns, ez alat nem garantált számitott értékek helyesége.

### 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 szabályzó típusát is PI szabályzó esetén a paramétert válaszuk 0-nak, mert nem okoz számítási problémát. PD szabályzó esetén célszerű a -nek minél kisseb értéket beállítani, ami nem lehet egyenlő 0-val.

## 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 szabályzó jellel avatkozunk be a rendszerbe.

Az elkészített szabályzót a következő egyenletek írják le:

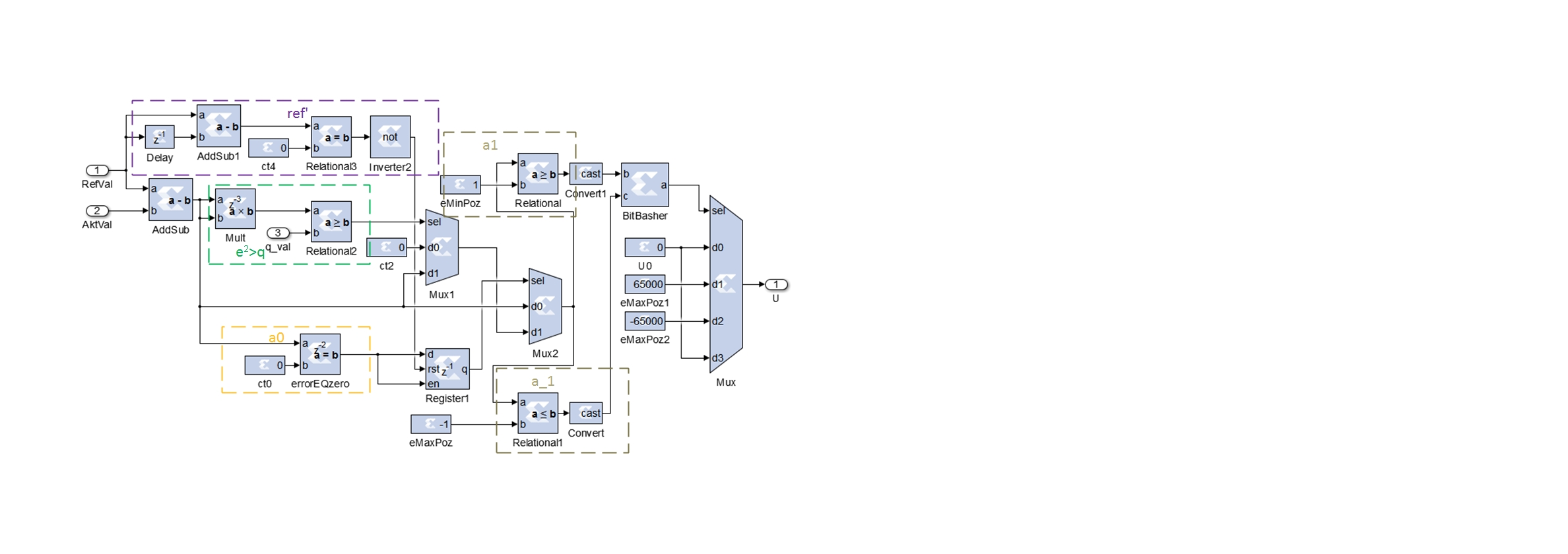
Elmondható a kimeneti szabályzó jel függ a hiba értékétől.A mechanikai rendszerben kotyogás van, és az ebből származó zajokat szeretnénk kiszűrni úgy, hogy ha a mechanizmus a megfelelő pozícióban van, akkor egy tartományban a szabályzót érzéketlené tesszük a bemenetre mindaddig, amíg a hiba ki nem lép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

### A szabályzó felépítése:

Funkcionalitás szerint öt csoportba sorolhatjuk az alábbi képen látható elemeket:

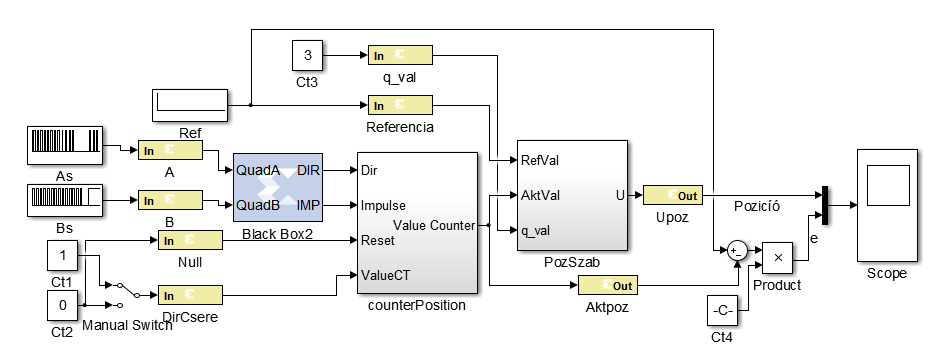
* ref’-a referencia értéket figyeli és minden órajel periódusban megvizsgálja, hogy van váltózás az értékben.
* – négyzetre emeli a hibát és összehasonlítja az általunk megadott q értékével.
* MUX – az U kimeneti jel állapot multiplexere.
* MUX1 –az manipulált hiba multiplexere
* MUX2 –hiba multiplexere

### Szabályzó szimulálása



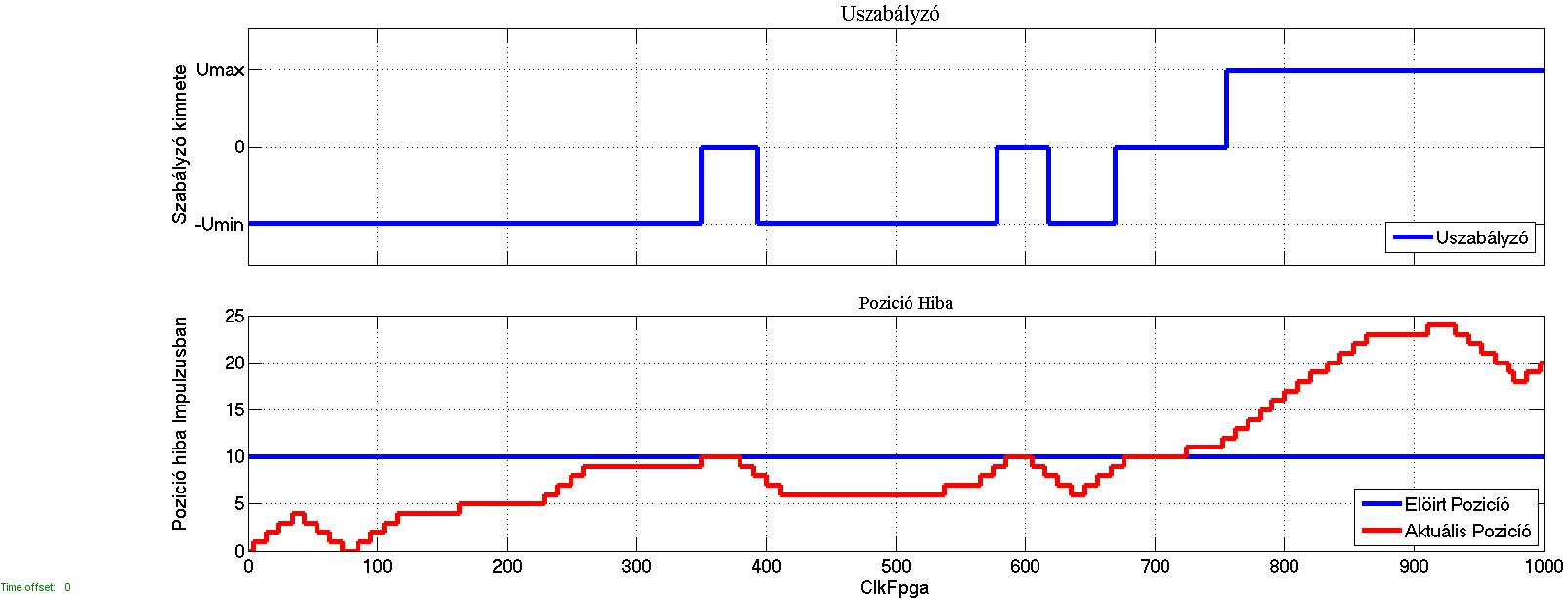
Kép. 1.6 Látható a Pozíció szabályozó System generátoros felépítése

Kép. 1.7 Látható a pozíció szabályzás moduláris felépítése System Genrator környezetben



Az alábbi képen látható a szimulációs logika, a counterPosition és a BlackBox modul részletesen megtalálható a pozíció mérése inkrementális adó segítségével. A szabályzót a PozSzab modul tartalmazza.

A grafikonról megfigyelhető hogy a kimeneti jel miként változik a hiba függvényében. Látható, ha a hiba 0 környékén van a szabályzó 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



Kép. 1.8 látható a pozíció szabályzó bemenete (pozíció hiba), valamint a Szabályzó 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 mozgatót mechanizmusra rögzített inkrementális tárcsa segítségével tudjuk mérni a motor elfordulását. A motor sebességét -ben vagyis impulzus per mintavételben mérjük, így a referencia sebességet is ebben a mértékegységben kell megadnunk, ezért átalakítást kell végezünk a következő összefüggés szerint.

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

kilépést.

## Sebesség és pozíció szabályzót tartalmazó IP mag generálása System Generatorban

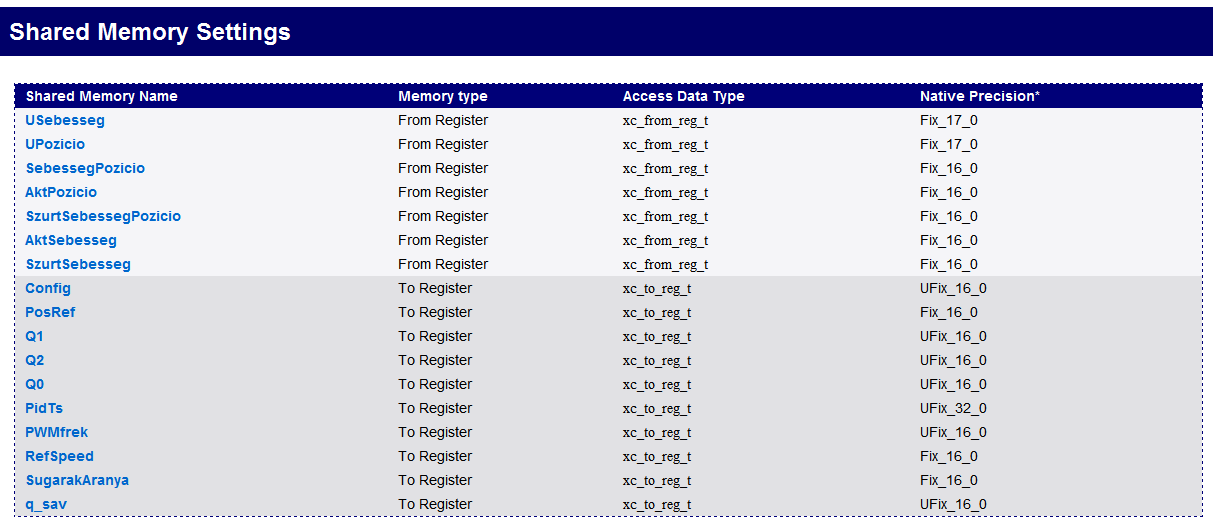
Az eddig megépítet, modulokat összekötve alkotunk egy nagy modult amely tartalmazza a sebesség és a pozíció szabályzási hurkokhoz szükséges elemeket. A szabályzók referencia bemeneteit osztott regiszterekkel írjuk elő microBaze processzor segítségével. Mindkét szabályzó tartalmazza a PWM generátort, 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 kisKerék nél, ezért megváltozik a lánctalp sebessége, ha a talajhoz viszonyítva. A sebesség szabályozó referencia bemenetéthez hozzá kell adni a pozíció változását megszorozva egy arányosáig tényezővel.

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

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

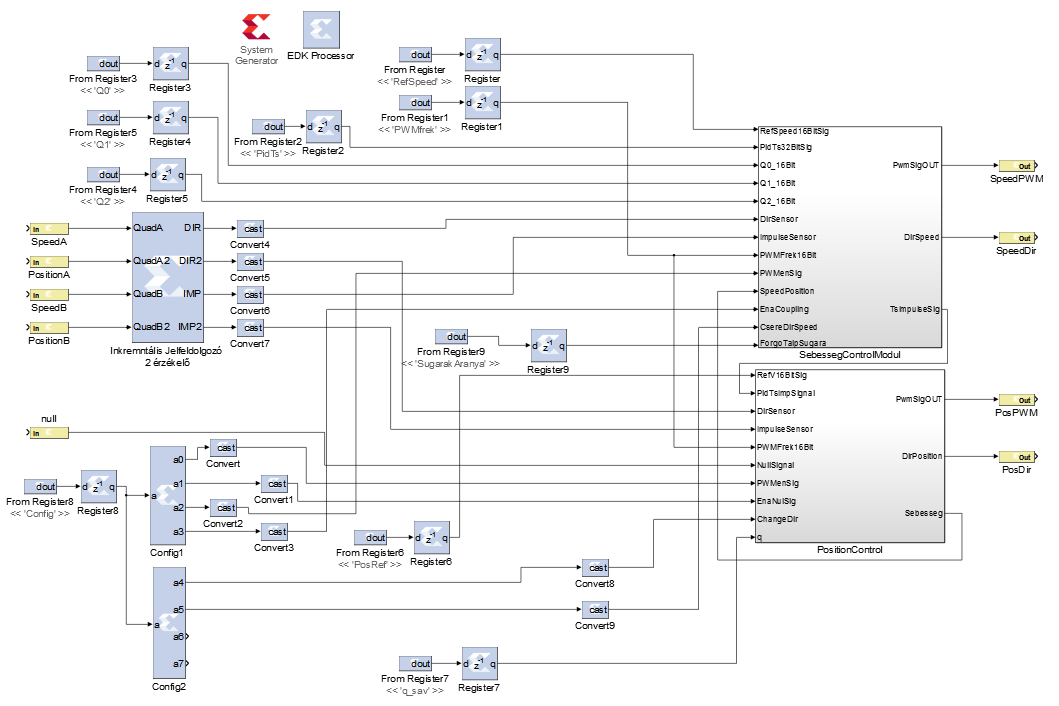
**Config regiszter**:



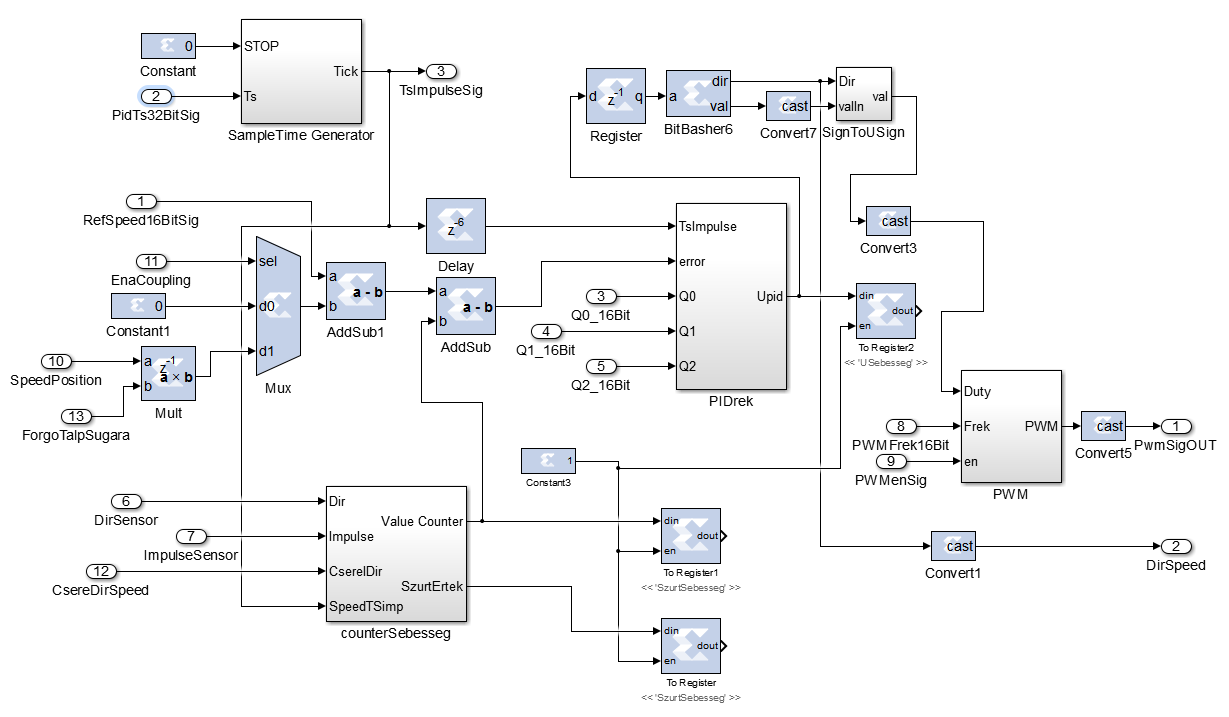
Kép. 1.9 Osztott regiszterek elnevezése és Típusa

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

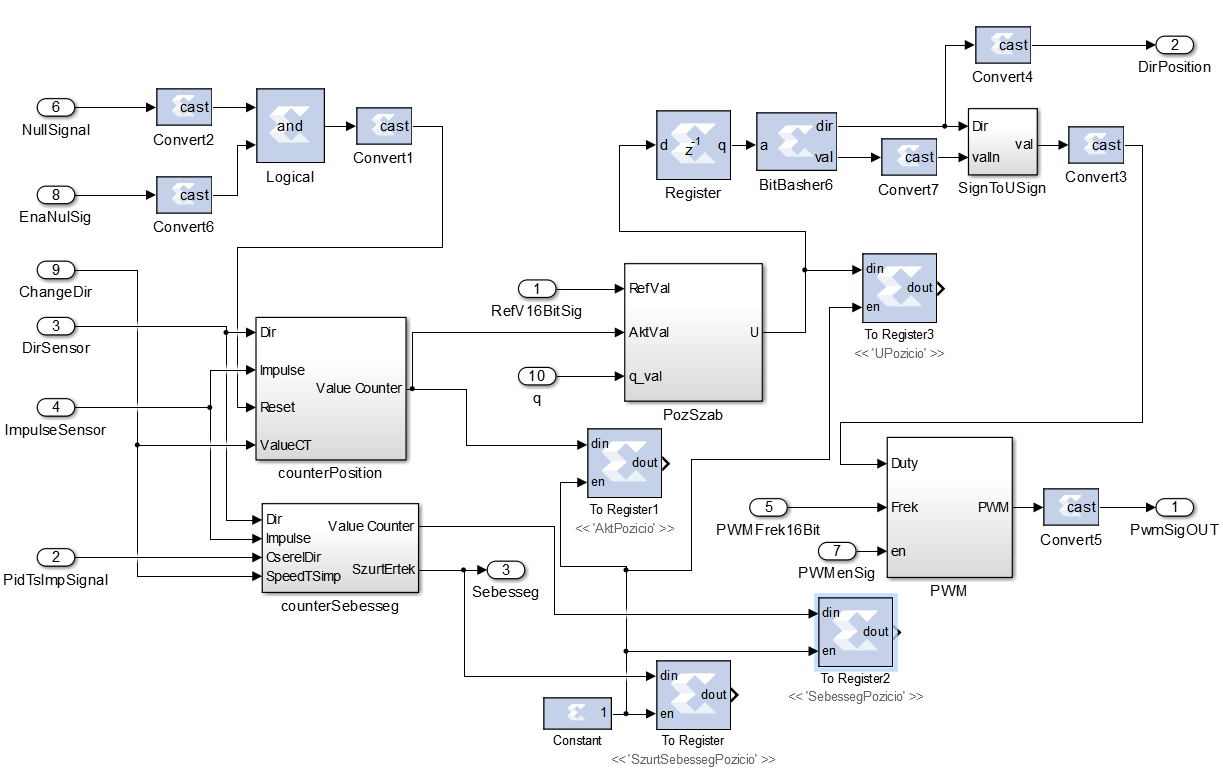
Tábla. 1 A config regiszter bitjeinek a funkcioja



Kép. 1.10 A két szabályzót tartalmazó modul System generatoros felepitesben



Kép. 1.12 Sebesség Szabályzó és beavatkozó, PositionControl belső felépítése (kép 1.10)

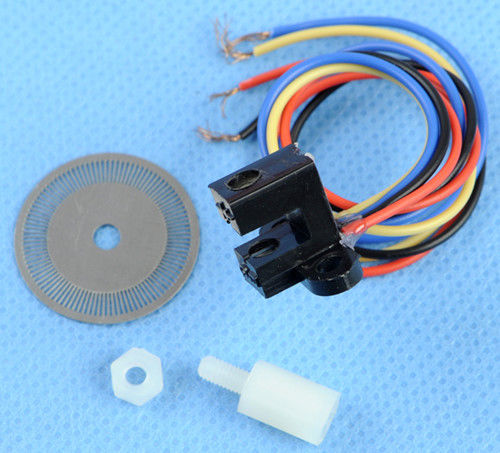


Kép. 1.11 Pozíció Szabályzó és beavatkozó SebessegControlModul belső felépítéses (kép1.10)

# Szenzorok

## Inkrementális szenzor

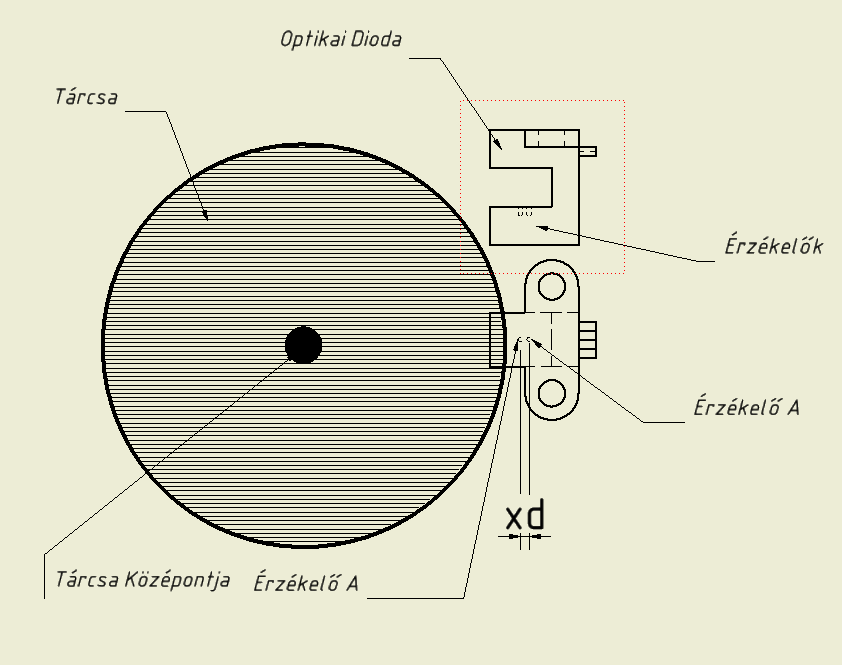
### Optikai inkrementális vevő felépítése

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

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

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

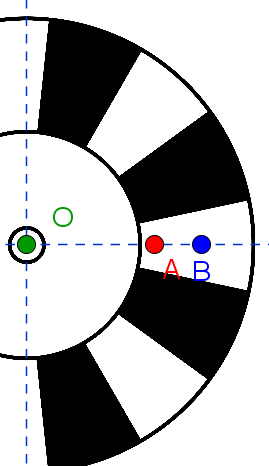
Az érzékelő számára a tárcsát a mellékelt képen látható módon kell illeszteni.



Kép. 2.1 Optikai inkrementális vevő felépítése és elhelyezése

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.

Ha a 2.2 ábrán látható modon helzezük 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 ölteni egy alfa szöggel így kialakul a késés is.

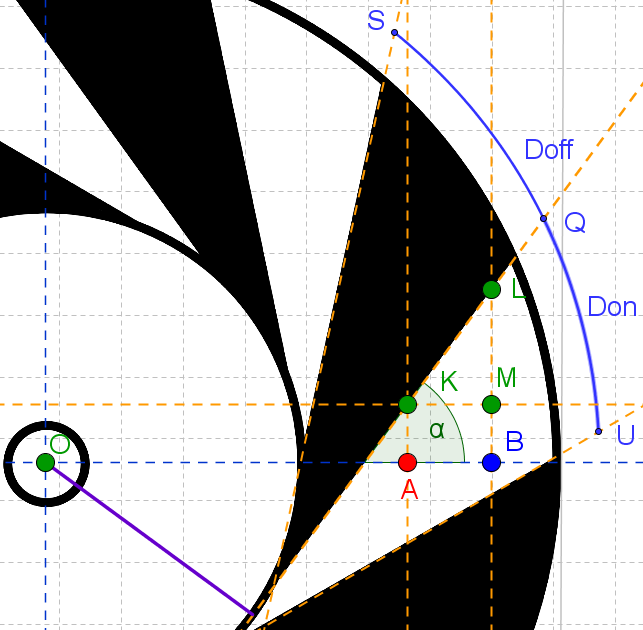


Kép. 2.2 Érzékelő tranzisztorok elhelyezése

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.

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



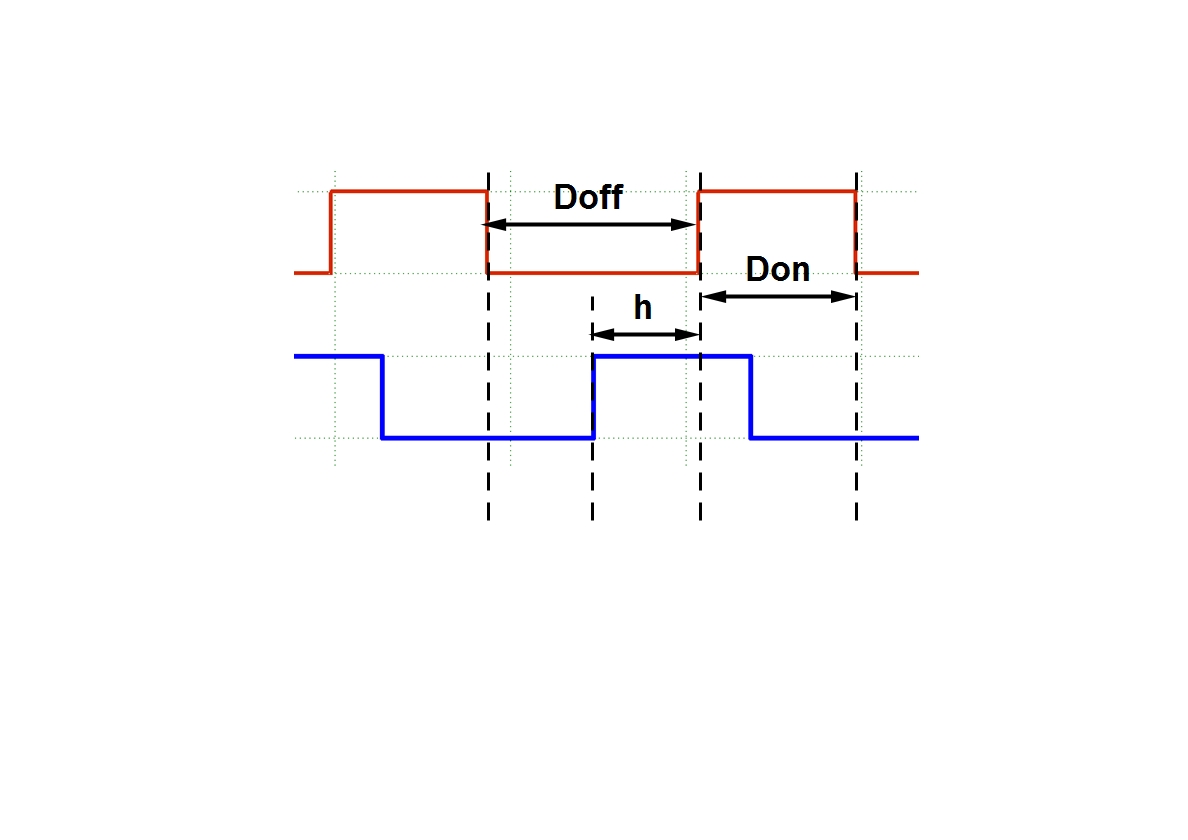
Kép. 2.3 Rések és az Érzékelők közti kapcsolat

Azokban a pontokban ahol a rések fedik az érzékelőket ott az érzékelők jele magas, míg ahol nem fedik ott alacsony.

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

**Ismertadatok**:

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 rendszerbe, felírható az összefüggés, amely meghatározza a két jel közti késést.

Kép. 2.4 Idődiagram a Tárcsa paraméterei függvénzében

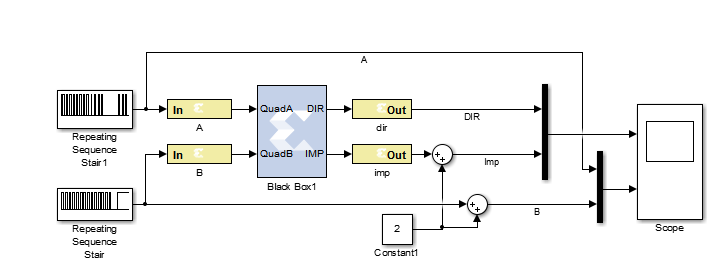
,.

A minden esetben nagyobbnak kell lennie, mint a távolságnak.

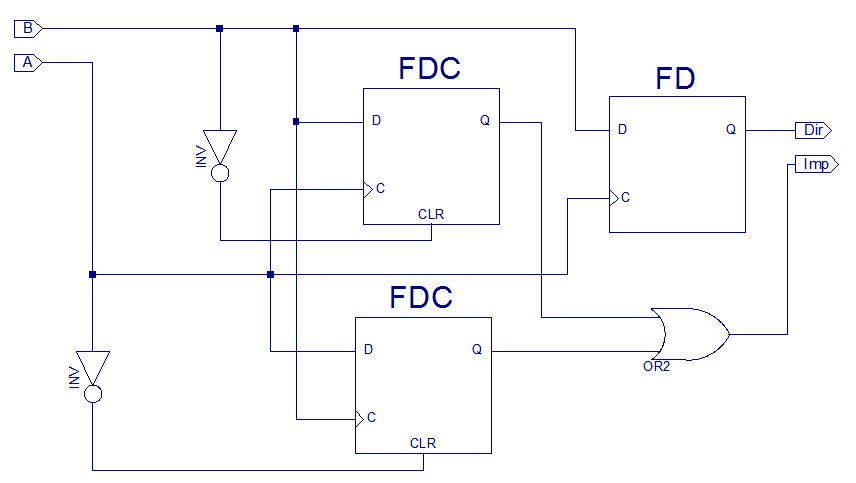
## Inkrementális érzékelő jeleinek a feldolgozása FPGA áramkör segítségével

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

A modul VHDL programnyelven készült a 2.6 képen látható kialakítás szerint, BlackBox modul segítségével integráljuk a Simulink környezetbe(BLACK BOX1). A ki és bemeneti portókat illesztjük a Simulink környezetben található elemekhez, majd létrehozzuk a szimulációs bemeneti jelelt, amelzeket az A, B sárgával jelőlt modulokon keresztül viszünk be a rendszerbe.



Kép. 2.5 Inkrementális érzékelőtől érkező jelek átdolgozása irány és impulzus jelekre



Kép. 2.6 Black Box1 modul belső felépítése

Az FPGA áramkörben megtalálható modul segítségével a jeleket feldolgozzuk és két kimenő jelet generálunk a Dir (megadja a forgás irányát), valamint a Imp (minden ablak elhaladásakor generál egy felfutó élet).

A beérkező A, és B jeleket egy tagadó kapu segítségével bekötjük a FDC tárolok CLR bemenetére, a 2.6 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.

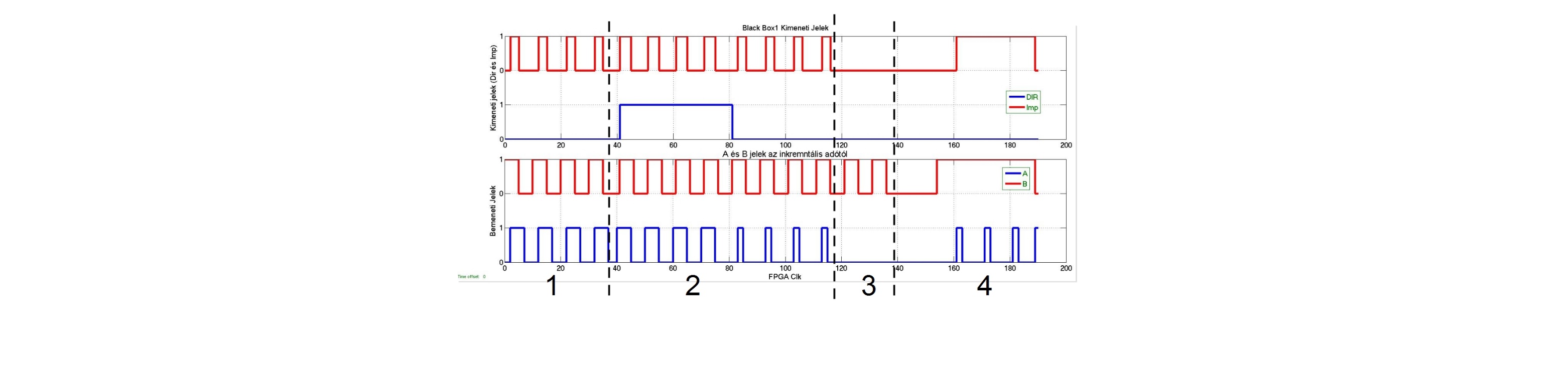
### Szimuláció System Generatorban

Az képen látható a szimulációs eredmények az A és B bementi jelek (alsó ábra), Dir és Imp kimenti jelek(felső ábra).

A bementi jelek négy kategoriara sorolhatok:

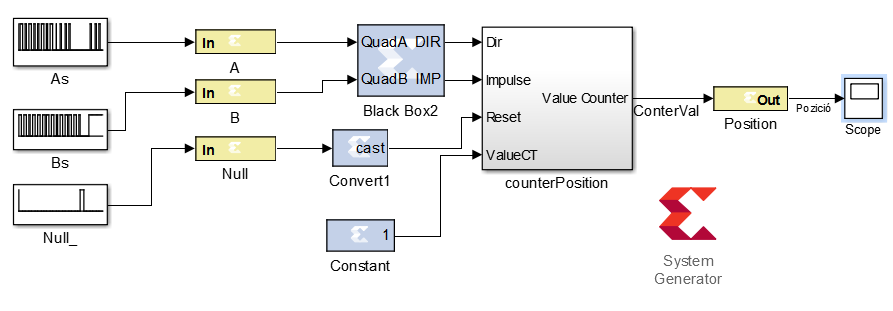
1. Az A jel késik a B jelhez képest, a kimenti jeleken látható ( érkező impulzusok és az irány).
2. Az B jel késik a A jelhez képest, látható hogy az irány megfordult.
3. Az A bemeneti jelen hibás jelek érkeznek, látható hogy ekkor nem történik impulzus generálas a kimeneten.
4. Az A bemeneti jelen ismét 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. 2.7 Szimulációs eredmények a lehetséges bemenetekről az Black Box1 modulba



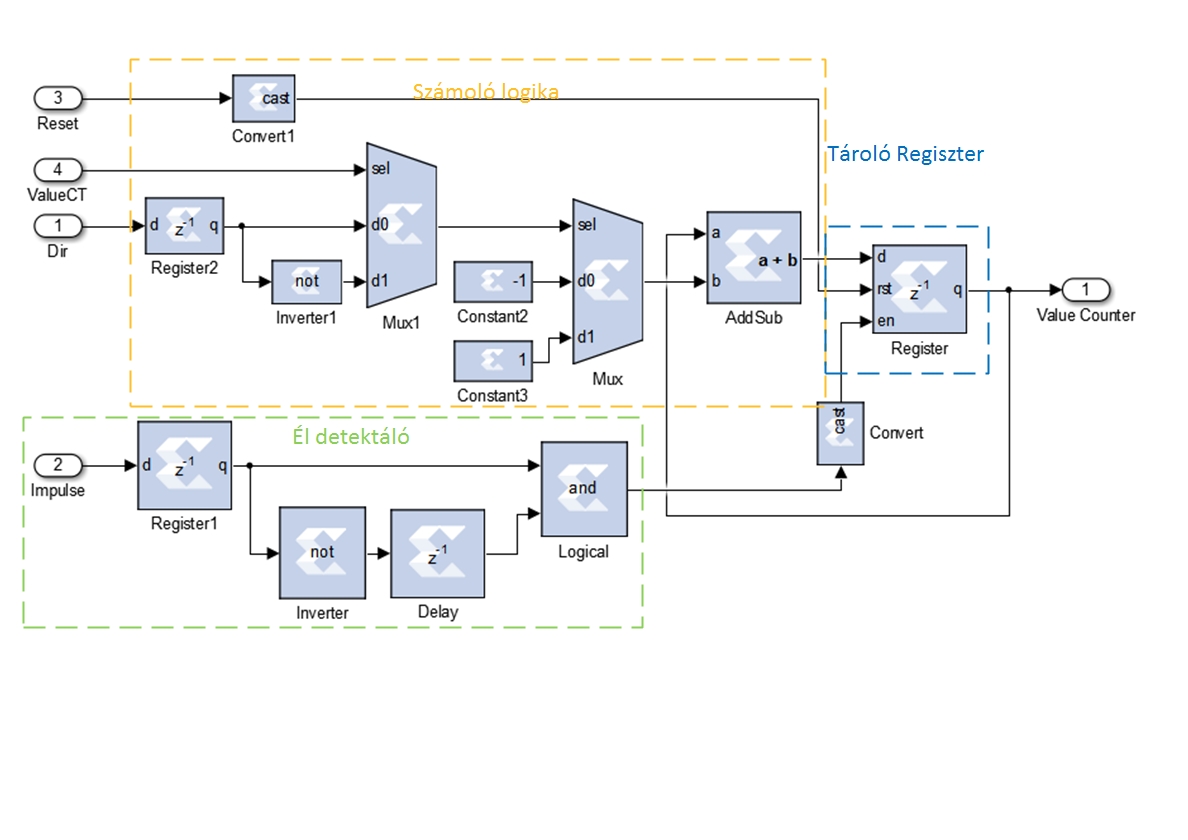
### Pozíció mérése Inkrementális adó segítségével

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



Kép. 2.8 Inkrementális adóval mért pozicó, szimulálációs modelje Sytem Generatorban

A 2.8 képen látható a fentiekben már letárgyalt Inkrementális adó jeleinek a feldolgozásáért felelős BlackBox2 (megfelel a Black Box1 modulnak) modul, valamint kiegészítve a pozíció mérésére alkalmas counterPosition modullal, amelynek belső felépítése az 2.8 képen látható.



Kép. 2.9 CounterPosition Belső Felépítése System Generátorban

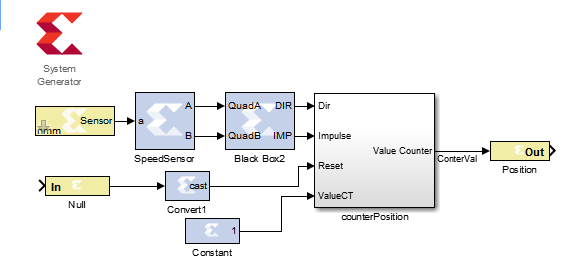
Működési logikája alapján három osztályba sorolhatok. Az első az él detektáló, melynek feladata az impulzusok felfutó éleinek a detektálása, és egyetlen órajelig tartó impulzus generálása a Tároló regiszternek így engedélyezve az adatbevitelt a regiszterbe.

Tároló regiszter, feladata az aktuális érték tárolása, a tipusa16bites 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.

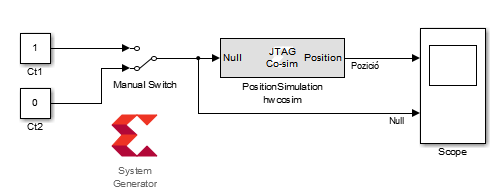
#### Hardveres mérések

A méréseket Spartan 3e500K fejlesztőlapon láthatok, egy N=180 impulzussal rendelkező, valamint a rések amelyek 90-os szöget vannak megdontve.



Kép. 2.10 Hardver Cosimulaciós modell Pozíció mérése

A Cosimulacios modell teljesen megegyezik 2.8 képen láthatóval annyi különbséggel hogy a A és B bemenetek az FPGA két kimenti I/O kivezetéséhez vannak illesztve így direkt az érzékelőtől érkeznek a jelek.

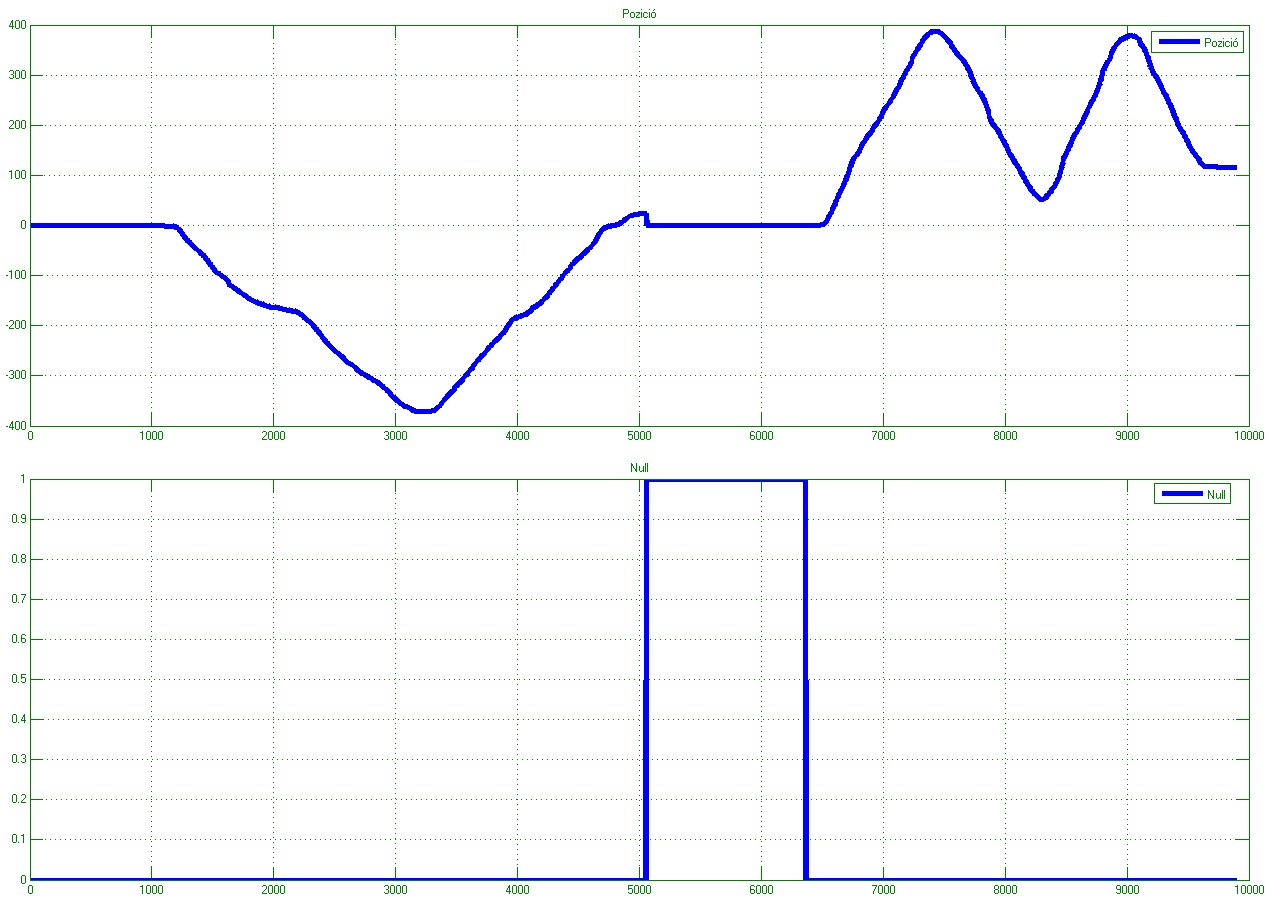


Kép. 2.12 Cosimulation Inkrementális Pozíció mérő

Kép. 2.12 Pozíció mérő felépítése

A generálás elvégzése után 2.12 képen látható modult kapjuk eredményül, amelyet az FPGA fejlesztőlapra feltöltve eltudjuk végezni a hardver Hardveres Szimulációt.

Az 2.13.képen látható a hardveres teszt, amely során kézzel forgattam a tárcsát egy tengely körül. Látható mindaddig a pozíció 0 marad ameddig a Null jell logikai 1 ben van.

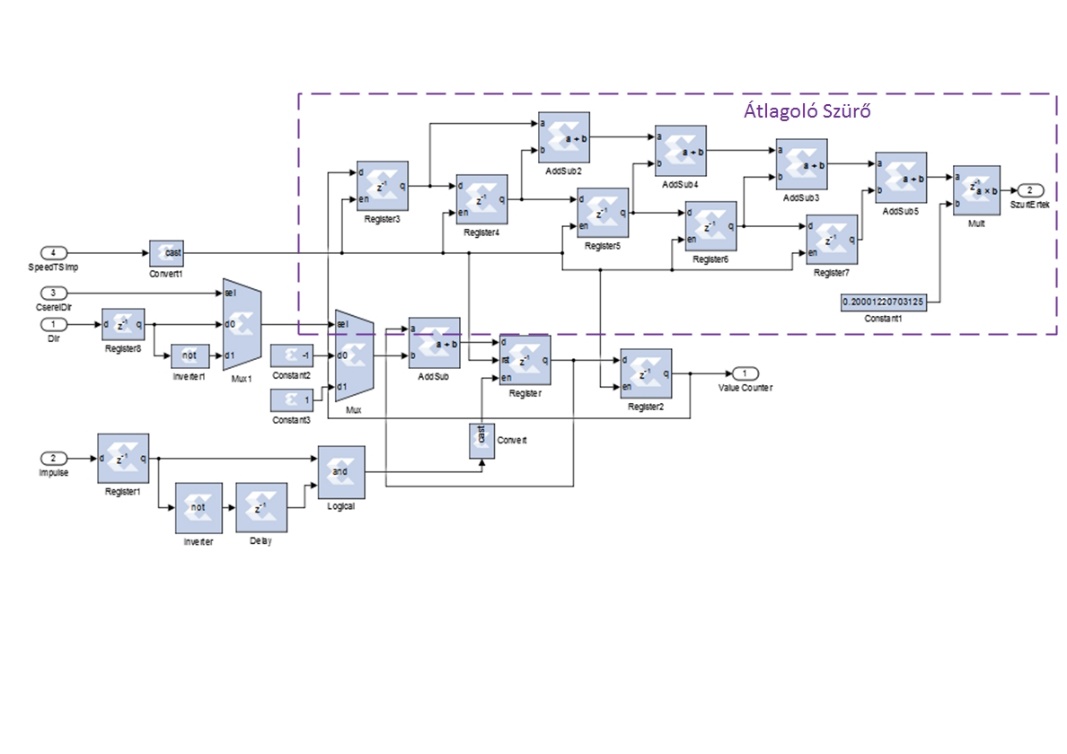


Kép. 2.13 Pozíció mérés eredményei a 2.12 képen látható kialakításban

### 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 regiszter amelyek FIFO láncot alkotnak. A regiszterek és az összeadó modulok típusa megegyezik a tároló regiszter típusával. A mért sebesség értékek bekerülnek A FIBO struktúrába, minden Ts impulzus érkezésekor a FIFO elejére illesztődik a mért érték. Minden impulzuskor összegződik a FIFO tartama.

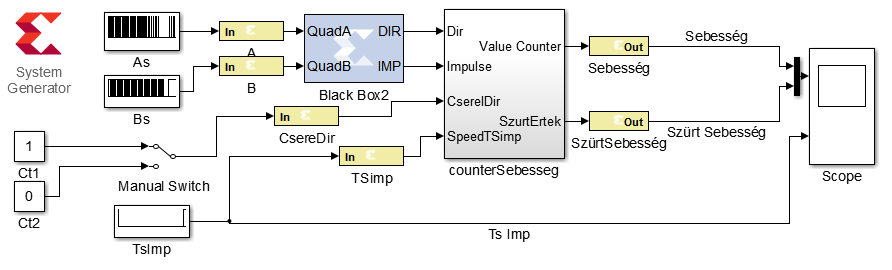


Kép. 2.14 Sebesség mérő modul felépítése

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, igy az értéke 0 lesz. A modulból kivezetjük mid a szűrt, mid a szűretlen értékeket.

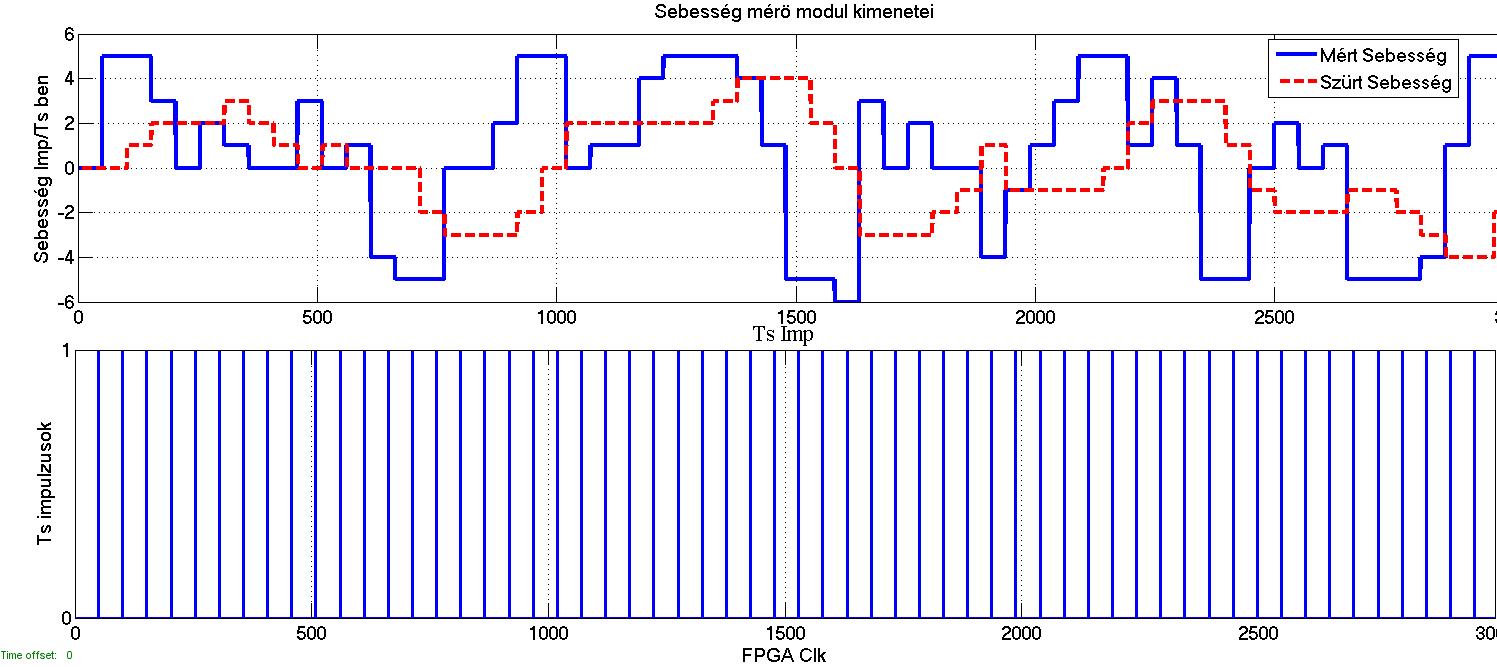
#### Szimuláció:

Az 2.15 képen látható a sebesség mérő modul szimulálására szolgáló logika kialakítása. A CounterSebesseg modul tartalmazza a 2.14 képen látható modult. A Black Box2 modul az inkrementális értékelőtől érkező jelek feldolgozásáért felelős. A Manual Switch segítségével megváltoztathatjuk a számolás irányát (lásd pozíció mérés.)



Kép. 2.15 Sebesség mérő modul szimulálására szolgáló Logikai kialakítás System generatorban

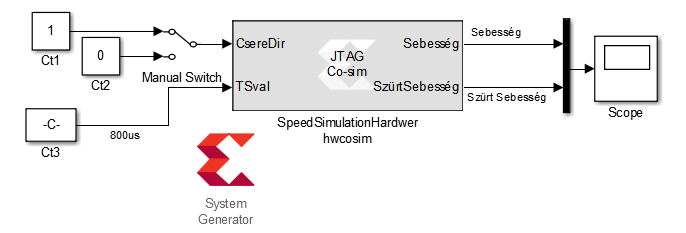
**Eredmények**:



Kép. 2.16 Sebesség mérő modul Szimulálása

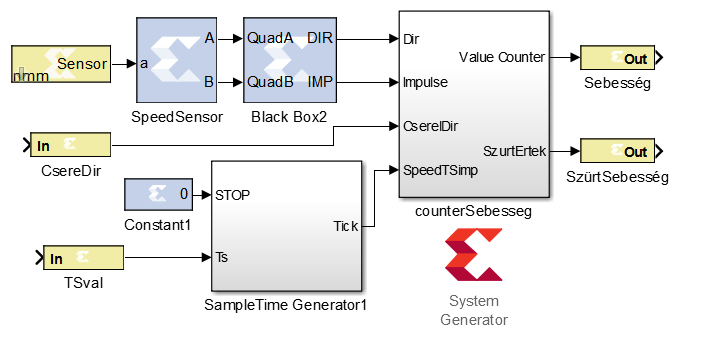
A 2.16 képen alsó részén látható a szabályos időközönként érkező mintavételi impulzusok, a fenti részen látható a mért értéket és a mért értékeknek a szűrt értéke. Megfigyelhető a szűrt érték a szűretlen értéknek egy 5 ablakos átlagolása.

#### Hardveres mérések



Kép. 2.17 Dc motor sebességének mérése Fpga lapon

A szimuláción látható egy DC motornak a tengelyén mért fordulatszáma maximális sebességen. Ahhoz hogy eltudjuk végezni a szimulációkat módosítani, kell a 2.15 képen látható logikát, az inkrementális szenzort hardveresen illesztenünk kell az FPGA kivezetéseire.



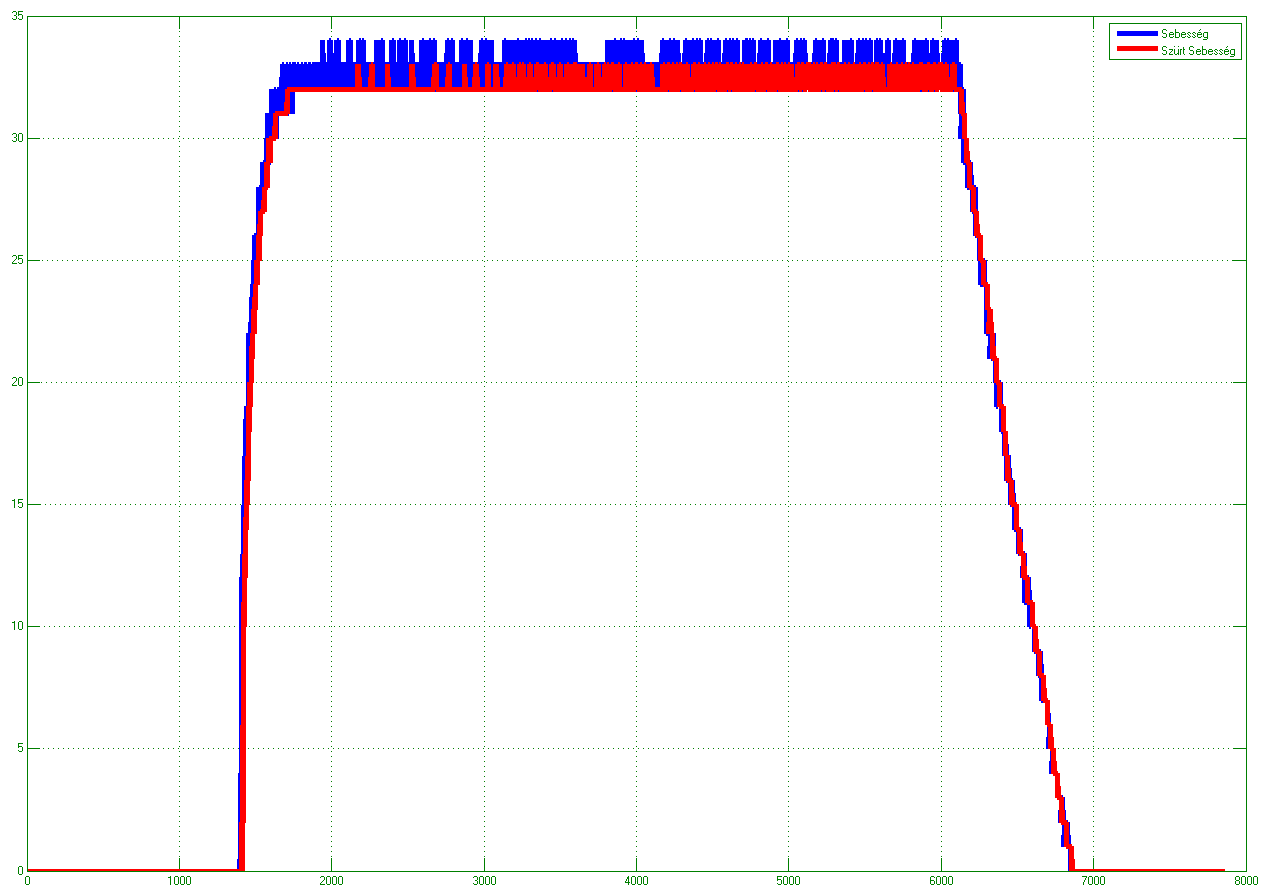
Kép. 2.18 Dc motor Sebesség mérése FPGA rendszeren, Szstem generatorban megvalósítva

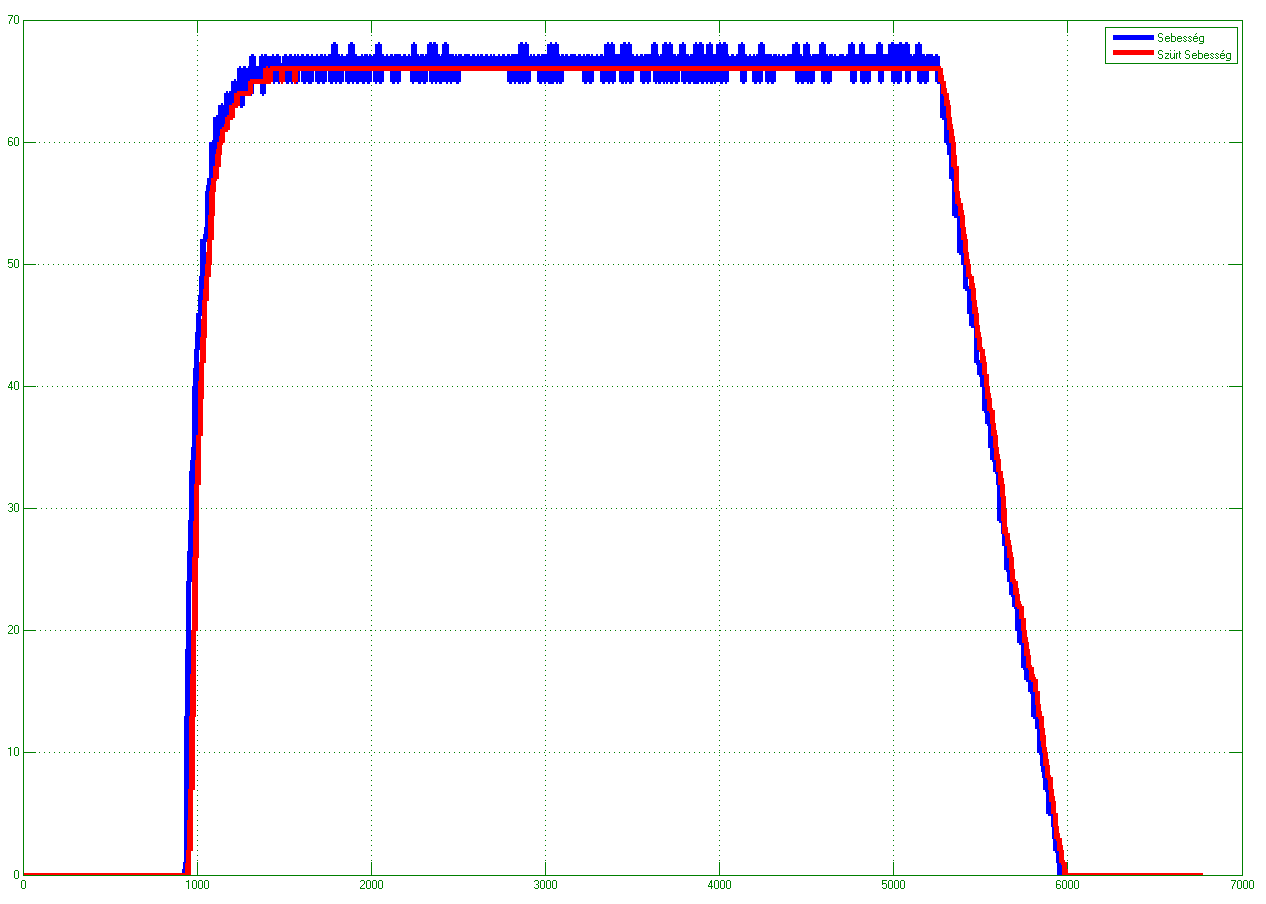
A 2.18 képen látható a Hardveres szimulációhoz szükséges logika, Sensor modul tartalmazza az FPGA azon kivezetéseit, amelyekre az érzékelő fizikailag kapcsolva van. Az érzékelő jelei egy SpeedSensor (csak a jelek bekötésében segít) nevű modulon keresztül a feldolgozó modulba érkeznek be a jelek. Ugyanakkor még megtalálható egy SampleTime Generator1 nevű modul is aminek a magyarázata megtalálható a dsfsdfdsfdsfd fejezetben.

A terv kigenerálása után kapunk egy újabb modult amely SpeedSimulationHardwer hwcosim nevet viseli.

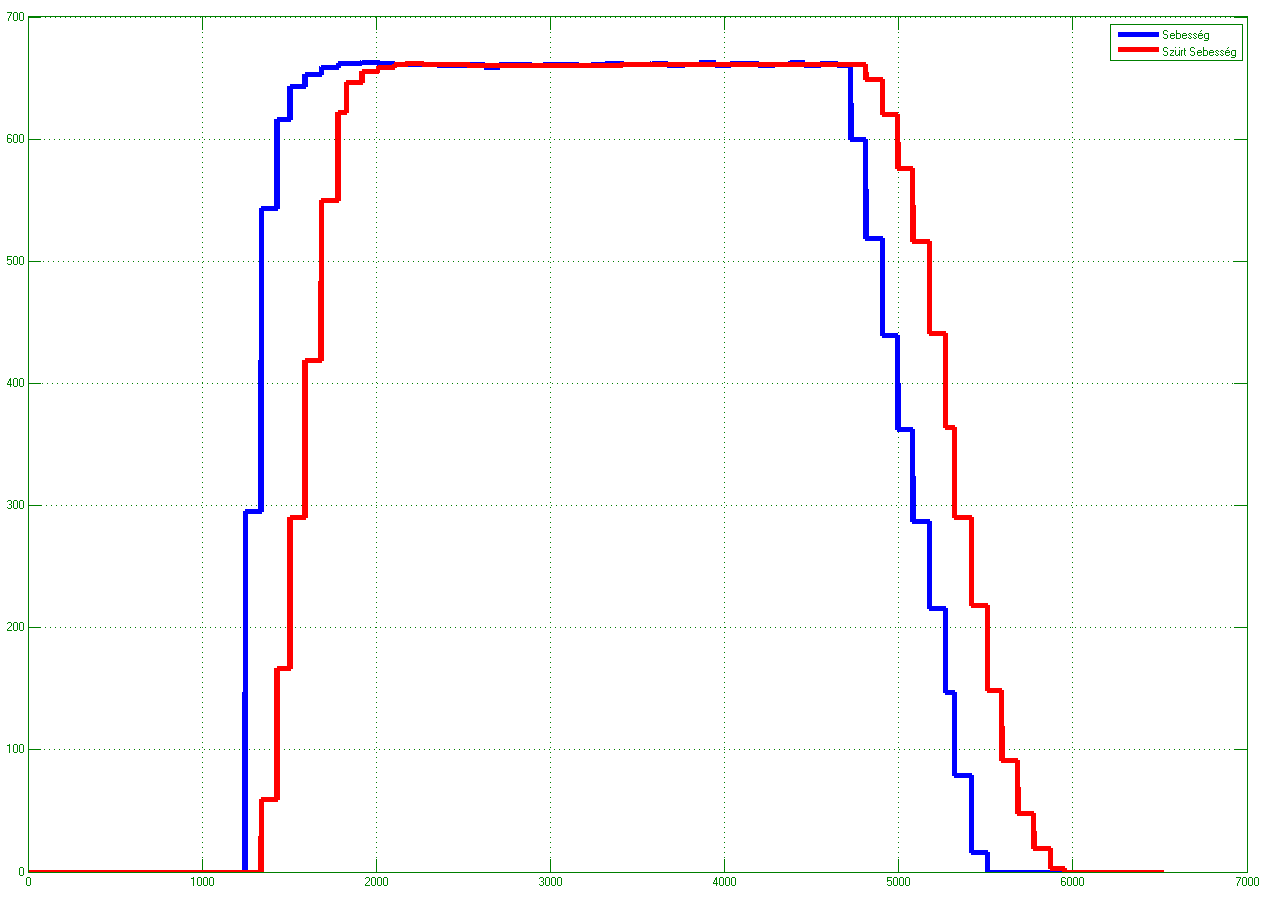
A 2.17 képen látható az újonnan generált modul bemenő adatainak a megadása illetve a mért értékek ábrázolása. A Ct3 nevű konstansba megadhatjuk a mintavételi időt ms ban.

A sebességet adott időegység alatt beérkező ipulzusok számával mérjük.

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



Kép. 2.21.a 8ms



Kép. 2.21.c 4ms

Kép. 2.21.b 80ms

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

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

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

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

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

# Beavatkozó elemek:

## Pwm Generátor megvalósítása FPGA áramkörön System Generator környezetben.

### Megvalosítás

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

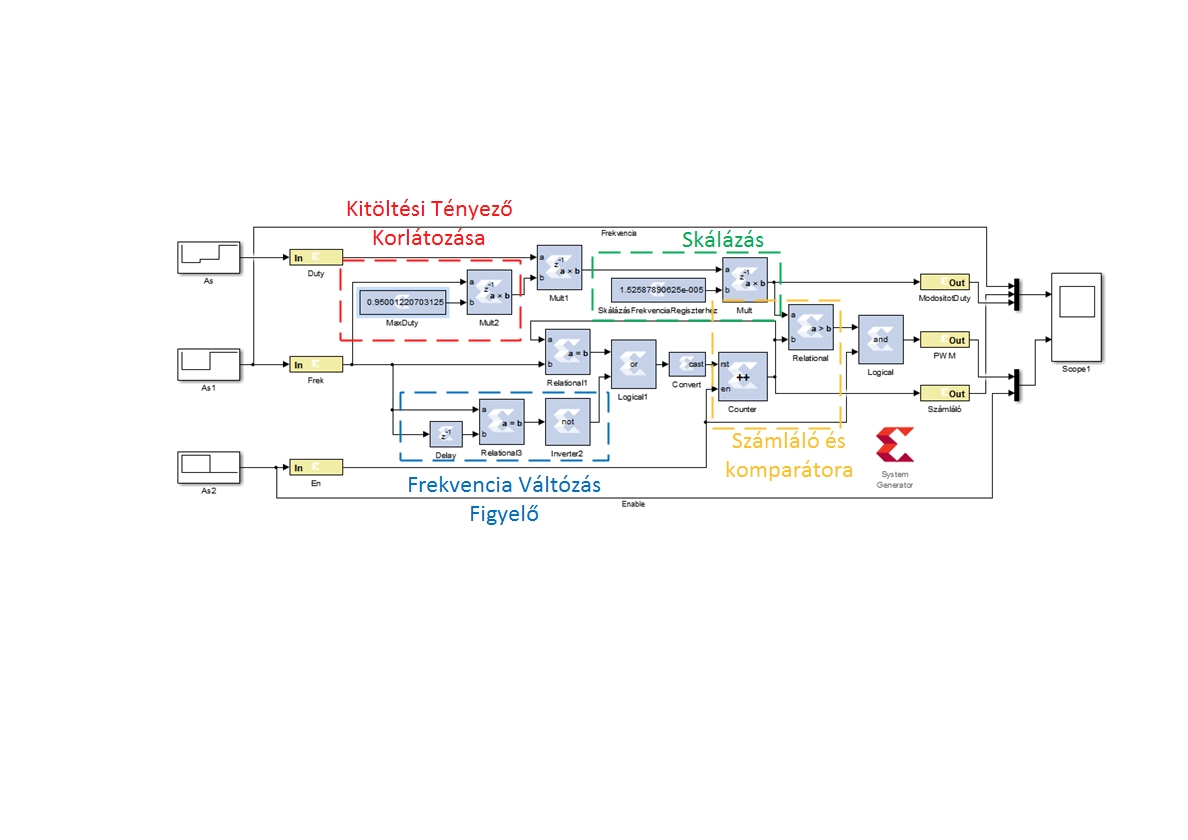
A PWM jel frekvenciáját egy Frek nevű regiszteren keresztül adhatjuk meg, a regiszter értéke összehasonlítódik a számláló értékével, és amikor a számláló értéke megegyezik a regiszter értékével a számlálót reset állapotba hozzuk.

A számlálót akkor is reset állapotba hozzuk, amikor a frekvencia regiszter értéke megváltozik, azért hogy elkerüljük azt az állapotot, amikor a számláló a túlcsordulásig számol tovább.

A PWM kitöltési tényezőjét egy Duty nevű regiszteren keresztül vihetjük be a rendszerbe, amelynek a típusa megegyezik a számláló típusával. A Duty értékének szüksége van egy skálázási eljárásra, amely segítségével illesszük a frekvenciához a Duty értékét.

Ugyanakkor a felépítésében be van iktatva egy korlátozás, amely segítéségével nem engedjük meg egy bizonyos százalék fölötti kitöltési tényezőt, MaxDuty nevű konstansba írhatjuk be 0 és 1 közötti értékben.

Kép. 3.1 Látható a PWM generátor System Generátorban megvalósított szerkezete



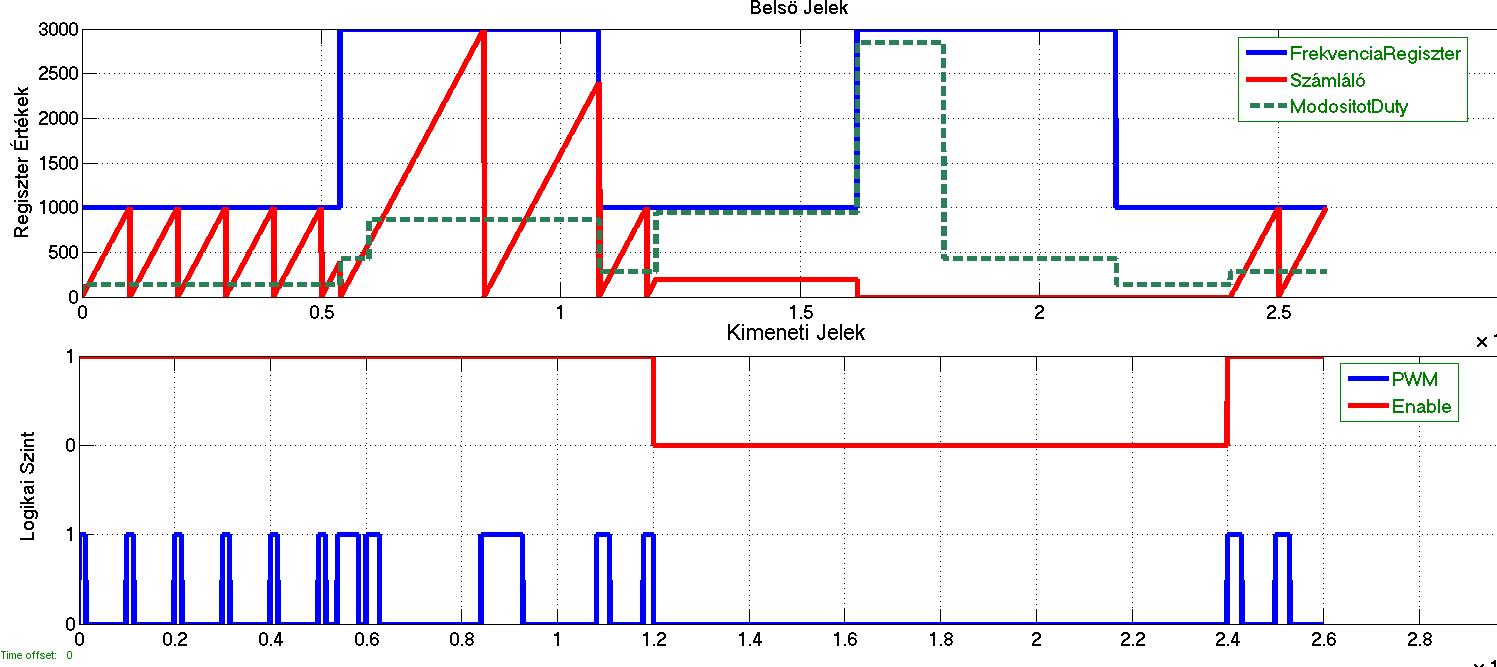
A generátor kimenetét és a számláló számlálásást letilthatjuk az (Enable) bemenetre adott logikai 0 értékkel, máskülönben engedélyezve vannak.

**Skálázás és bemeneti értékek:**

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

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

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



Kép. 3.2 a PWM generátor bemenő, kimenő illetve néhány belső jele (Scope1)

A 3.2 képen látható felső ábrán látható a kékkel jelölt frekvencia regiszter értéke, pirossal jelölt a számláló értéke, a zöld szaggatott a skálázott kiötlési tényező értéke. A kép alsó részén látható a kékkel jelelt PWM jel, illetve papirossal jelölt Enable jel.

# Elektronika

## Digitális Elektronika

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

### FPGA Rendszer Felépítése

#### Tömbvázlata

A rendszeren megtalálható két FPGA fejlesztő lap, egy ZYBO amely nagyobb érő forrássokkal rendelkezik, de kevés a kivezetéséinek a száma, és agy Spartan3e chippel rendelkező fejlesztőlap, amely kevés erőforrással bír, de számosabb kivezetéssel rendelkezik.

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.

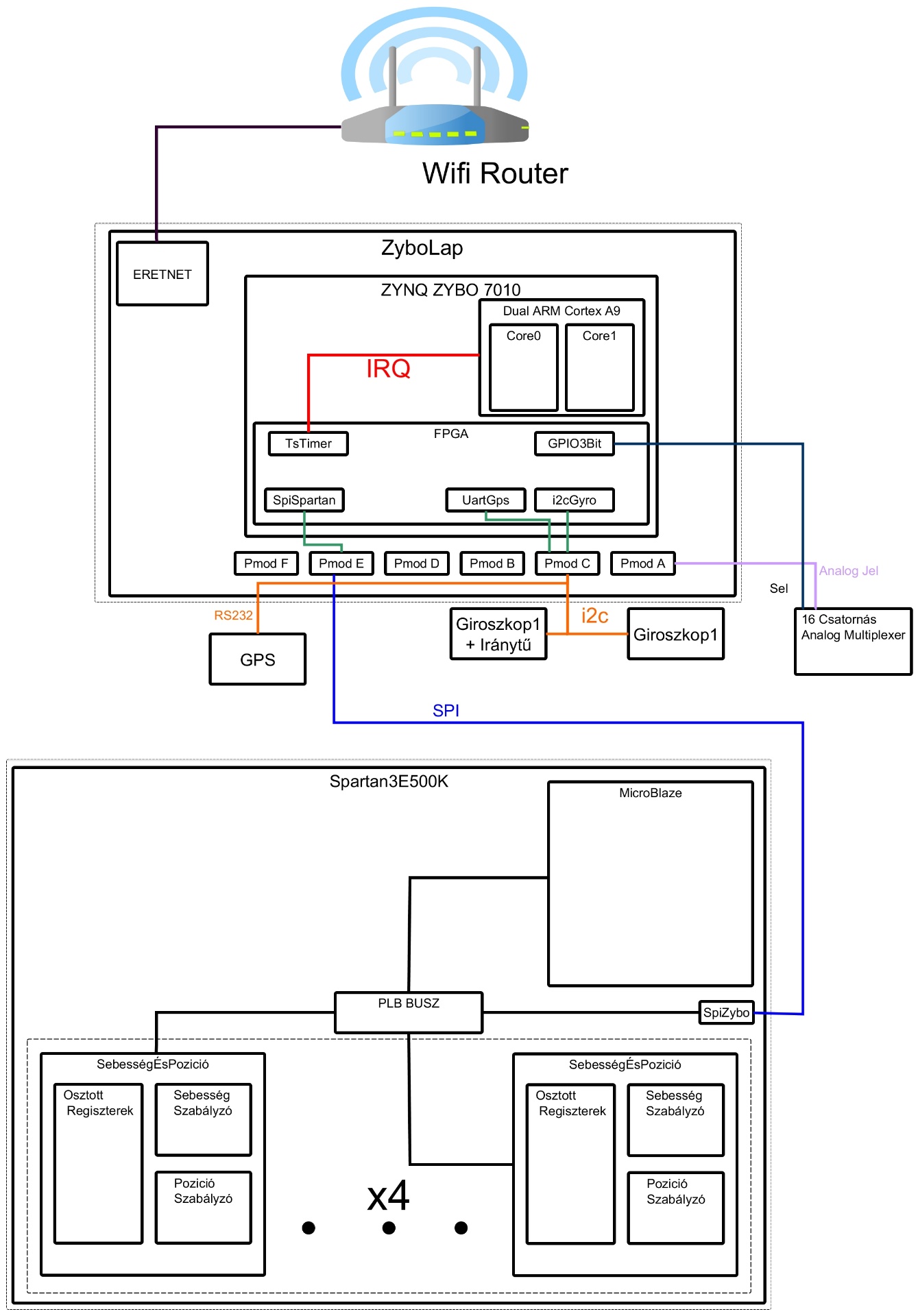
A fejlesztőlap tartalmaz 6 darab 8 bites kivezetésű PMOD csatlakozót, amelyeken keresztül tudjuk csatolni az FPGA területen levő hardveres eszközeinket vagy akár a beépített eszközök kivezetéseit is.

A Spartan lap tartalmaz több mint 120 kivezetést, és egy Spartan3e500k FPGA chipet.

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 Szabályzó ip magot, amelyeket a System Generátorban készítünk el és generálunk ki.

A szabályzókat tartalmazó IPmag osztót regisztereken keresztül állíthatjuk be a paramétereit vagy olvashatunk ki értékeket, amelyek szintén az 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 eszközt, amelyek lokális hálózatba lesznek kötve a Zybo fejlesztőlappal.



Kép. 4.1 Látható az FPGA rendszer és a fonatosabb tartozékainak kialakítási Struktúrája

**Szenzorok:**

**Zybo:**

* A Zybo laphoz illesztünk a PmodB csatlakozón keresztül egy giroszkópot, és giroszkóp+gyorsulásmérő szenzorokat, amelyek fizikailag ugyanazon a vezetéken vannak, csak az eszközök címe különbözik. A szenzorokkal i2c kommunikációs protokollon keresztül kommunikálunk.
* GPS szintén a PmodB csatlakozón keresztül illesszük a rendszerhez és RS232 protokollon keresztül kommunikálunk az eszközzel.
* A lapon megtalálható XADC modul bemenetére illesztünk egy 16 csatornás analóg multiplexert, amely segítségével nem időkritikus adatokat fogunk bevinni, mint például hőmérséklet érzékelőktől érkező analóg jeleket.

**Spartan3e:**

* A fejlesztőlapra beérkezik 8 darab inkrementális érzékelőtől érkező jelek összesen , darab.
* 4 darab null átmenet érzékelőtől bemenet, amelyek az abszolut pozíció meghatározásánál játszanak fontos szerepet.

**Kimenetek**:

**Spartan3e**:

* 4 darab kimenet a PmodB csatlakózón, amely segítségével kiválasszuk az analóg multiplexer kívánt csatornáját.
* 8 PWM kimenet és a hozzájuk tartozó Dir jel, összesen 16 darab.

## Feladatok Elosztása

### Zybo fejlesztőlap

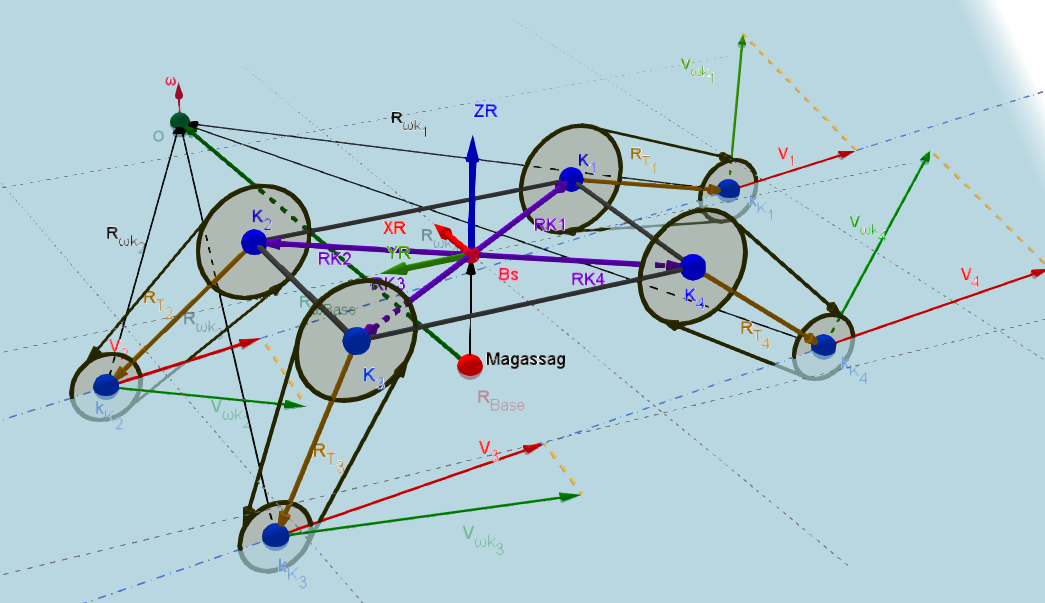
A két beépített processzorok (Core0, Core1) között munkamegosztást kell felalítani a hatékonyabb működés kedvéért.

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 szenzorok tol (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.

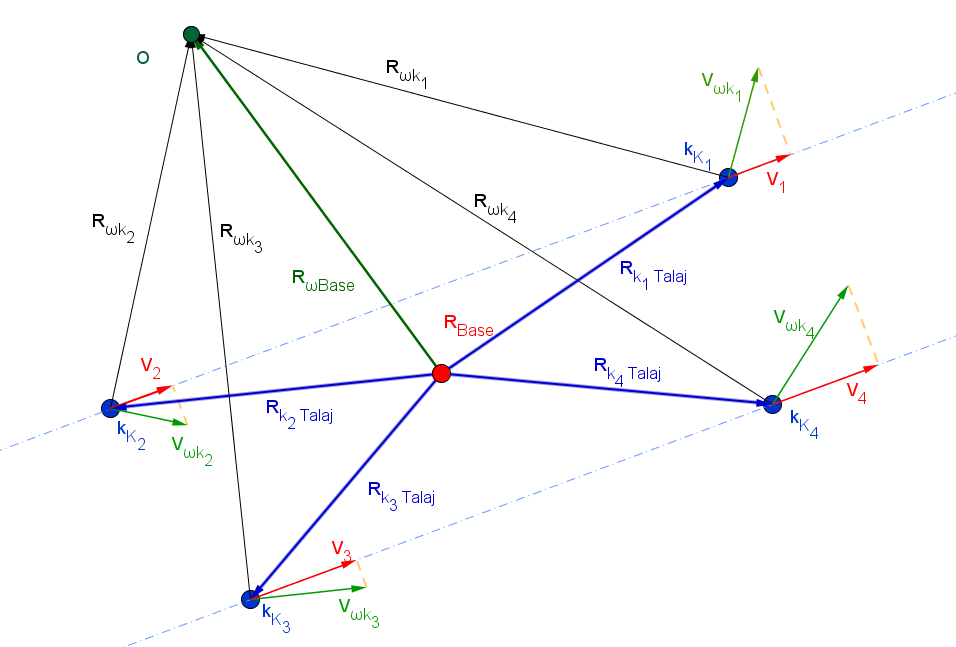
Miután végzett a Core1 a matematikai számításokkal az SPI kommunikáción keresztül elküldi a szabályzók referencia értékeit a Spartan fejlesztőlapnak.

### Spartan fejlesztőlap

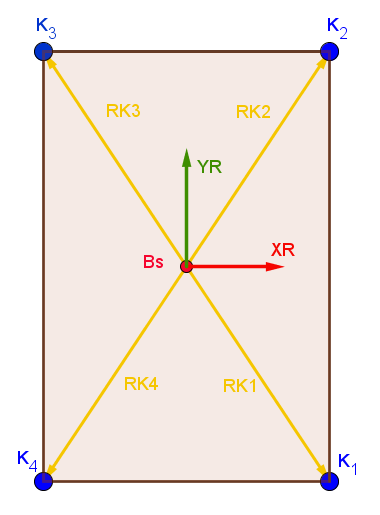
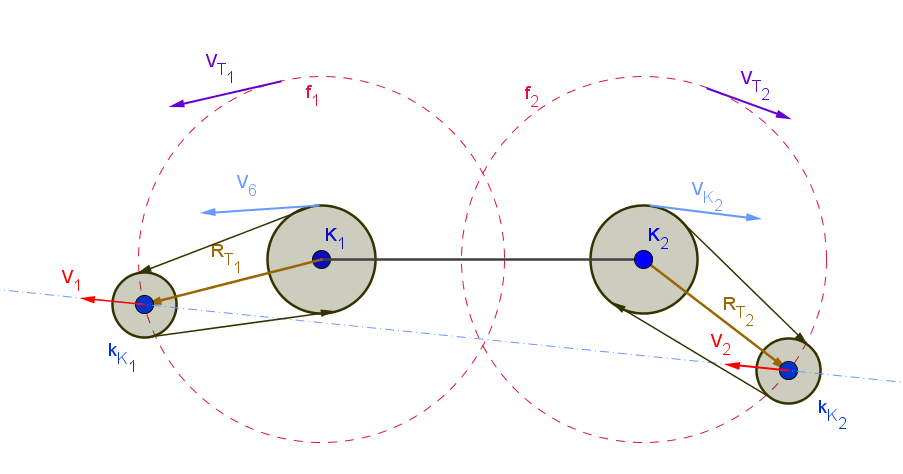
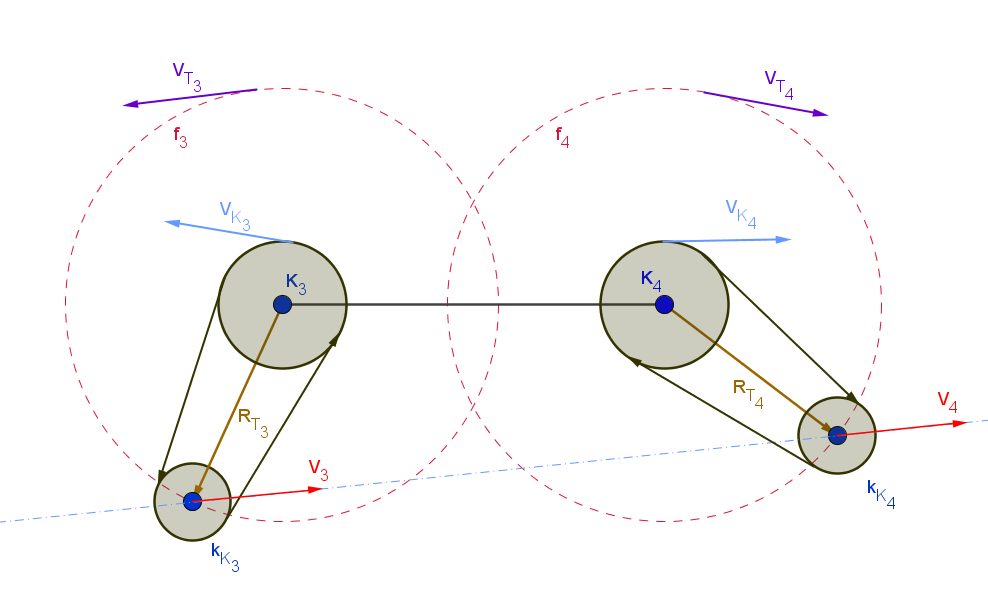
# Robot Modell



Kép. 5.1 Robot 3D vektorábrája



Kép. 6.2 Robota éstalaj sikjáv abban az esetben ha négy kiskerék a talajon van, adott pont körül adott sebességgel való körmoygás



Kép. 6.3 Oldalnézetek és Felülnéztet,jelőlések szemláltetése

**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:

# Robot Mechanikai Felépítése

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

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

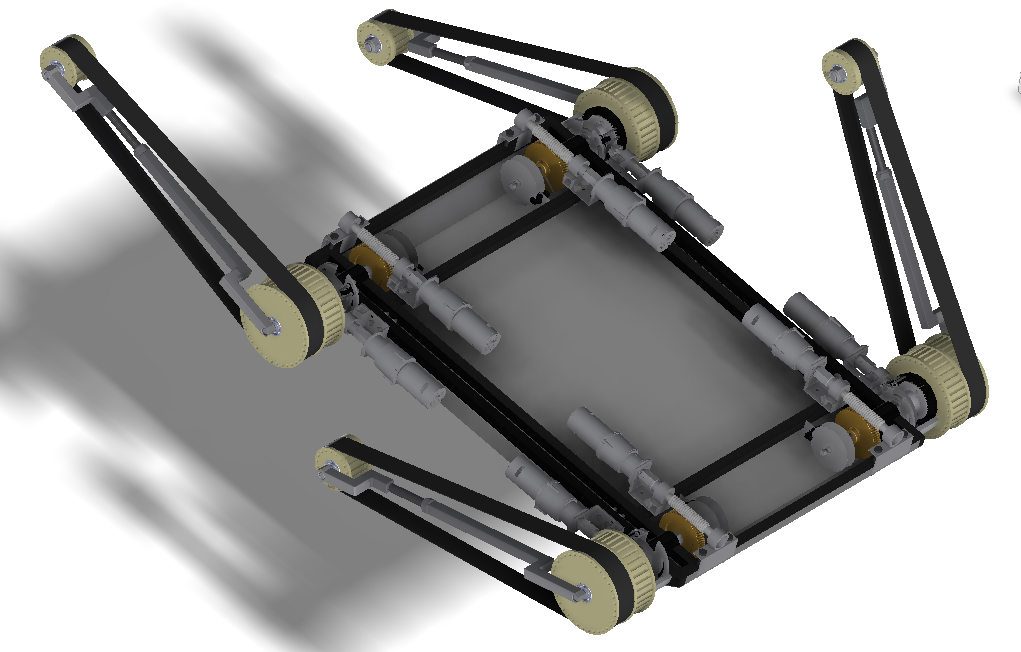
A 7.1 képen látható kupkerék átételen keresztül hajtjuk meg a lánctalpat, a talpak mozgatására orsos átételt használtam több okbol is:

* 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 nzomaték 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 pozicíoját.

A forgó talpak 360 fokban körbeforgathatok, 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.



Kép. 7.2 Robot vázának Inventoros 3D Képe

ForgóTalp1

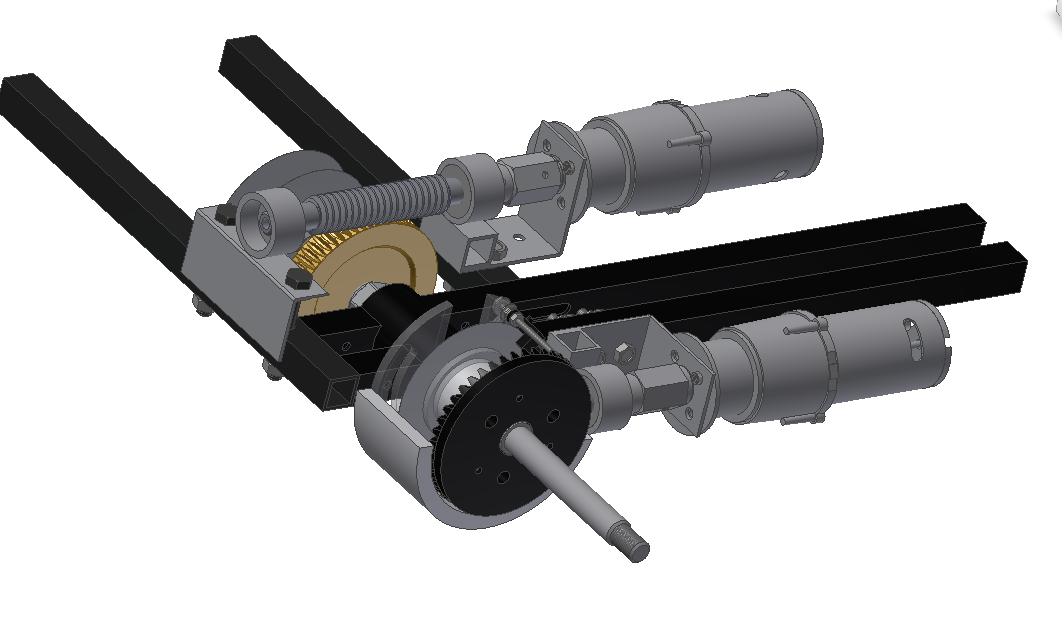
ForgóTalp4

ForgóTalp3

ForgóTalp2

KisKerék

NagyKerék

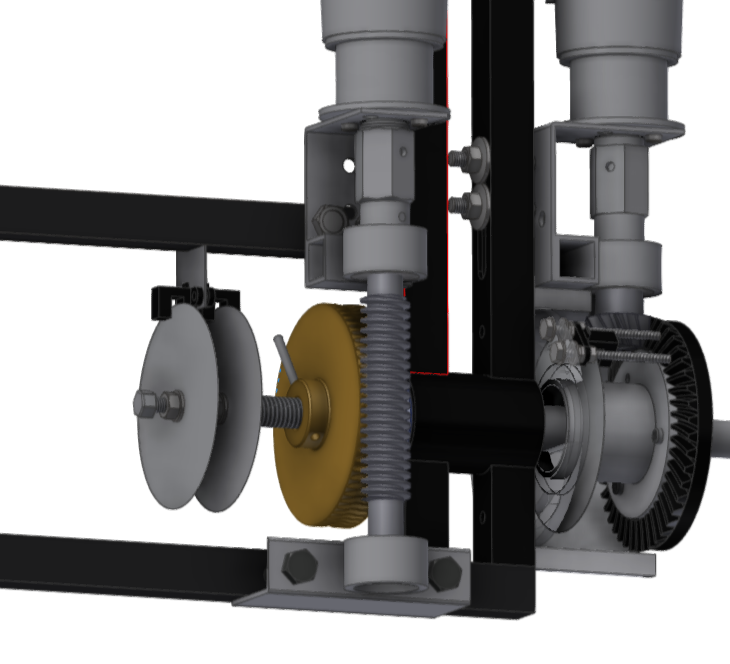


Orsó áttétel 40:1 arányban

Kupkerép áttétel 6:1 arányban

Sebesség mérő inkrementális tárcsa és szenzor

Kép. 7.1 Az alváz negyede



Null átmenti érzékelő tárcsa

Inkrementális szenzor tárcsa

Referenciák:

<http://www.ms.sapientia.ro/~martonl/MartonL_Education.htm#Control_Engineering_I>.

<http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf>

<https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1198&Prod=ZYBO>

http://www.xilinx.com/support/documentation/sw\_manuals/xilinx14\_5/sysgen\_gs.pdf