**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*:* 3](#_Toc416820356)

[***1.1*** ***Diszkrét Hardveres PID szabályozó*** 3](#_Toc416820357)

[1.1.1 Megvalósítás System Generátorban 5](#_Toc416820358)

[1.1.2 Szoftveres Szimulációs eredmények 6](#_Toc416820359)

[1.1.3 Q paraméterek számolása alapján. 7](#_Toc416820360)

[1.2 Hardveres mérések 7](#_Toc416820361)

[1.2.1 Dc motor sebesség szabályzása mérőstandon 7](#_Toc416820362)

[1.2.2 Pozíció Szabályzása csigahajtás áttételen keresztül DC motor segítségével gyors Hardveres Automatával 8](#_Toc416820363)

[1.2.3 A szabályzó felépítése: 9](#_Toc416820364)

[1.2.4 Szimulálás Szoftveresen 9](#_Toc416820365)

[2 Szenzorok 11](#_Toc416820366)

[2.1 Inkrementális szenzor 11](#_Toc416820367)

[2.1.1 Optikai inkrementális vevő felépítése 11](#_Toc416820368)

[2.1.2 Inkrementális érzékelő jeleinek a feldolgozása FPGA áramkör segítségével 13](#_Toc416820369)

[2.1.3 Szimuláció System Generatorban 13](#_Toc416820370)

[2.1.4 Pozíció mérése Inkrementális adó segítségével 14](#_Toc416820371)

[2.1.4.1 Hardveres mérések 15](#_Toc416820372)

[2.1.5 Szögsebesség mérése Inkrementális adó segítségével 16](#_Toc416820373)

[2.1.5.1 Szimuláció: 17](#_Toc416820374)

[2.1.5.2 Hardveres mérések 17](#_Toc416820375)

[3 Beavatkozó elemek: 20](#_Toc416820376)

[3.1 Pwm Generátor megvalósítása FPGA áramkörön System Generator környezetben. 20](#_Toc416820377)

[4 Elektronika 20](#_Toc416820378)

[4.1 Digitális Elektronika 20](#_Toc416820379)

[4.1.1 FPGA Rendszer Felépítése 20](#_Toc416820380)

[4.1.2 Vezetékek Elrendezése 20](#_Toc416820381)

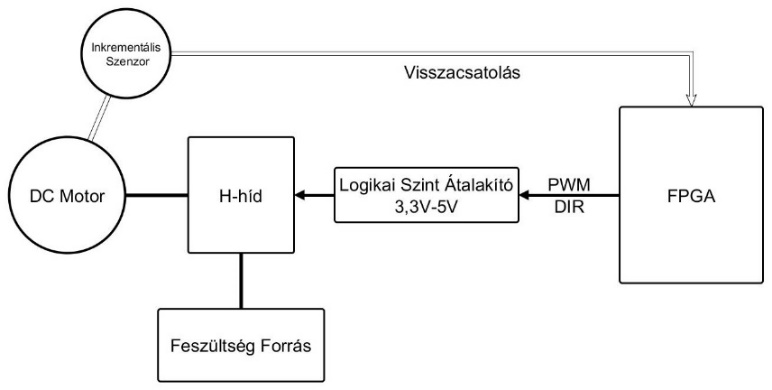
[5 Software 20](#_Toc416820382)

[6 Robot Modell 20](#_Toc416820383)

[7 Robot Mechanikai Felépítése 20](#_Toc416820384)

# 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

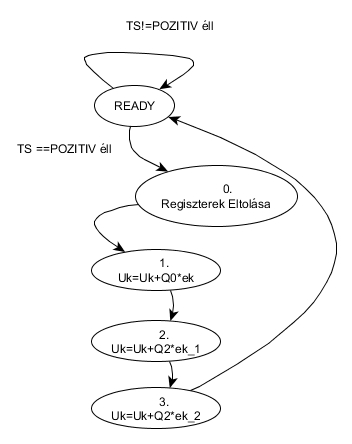
## Diszkrét Hardveres PID szabályozó

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 áramkör segítségével, a minél kisebb mintavételezési periódus elérése céljából. A fent látható összefüggések segítségével egy állapot automata irható fel, amelyet majd System Geratorban 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 állapot egy órajel periódust igényel.

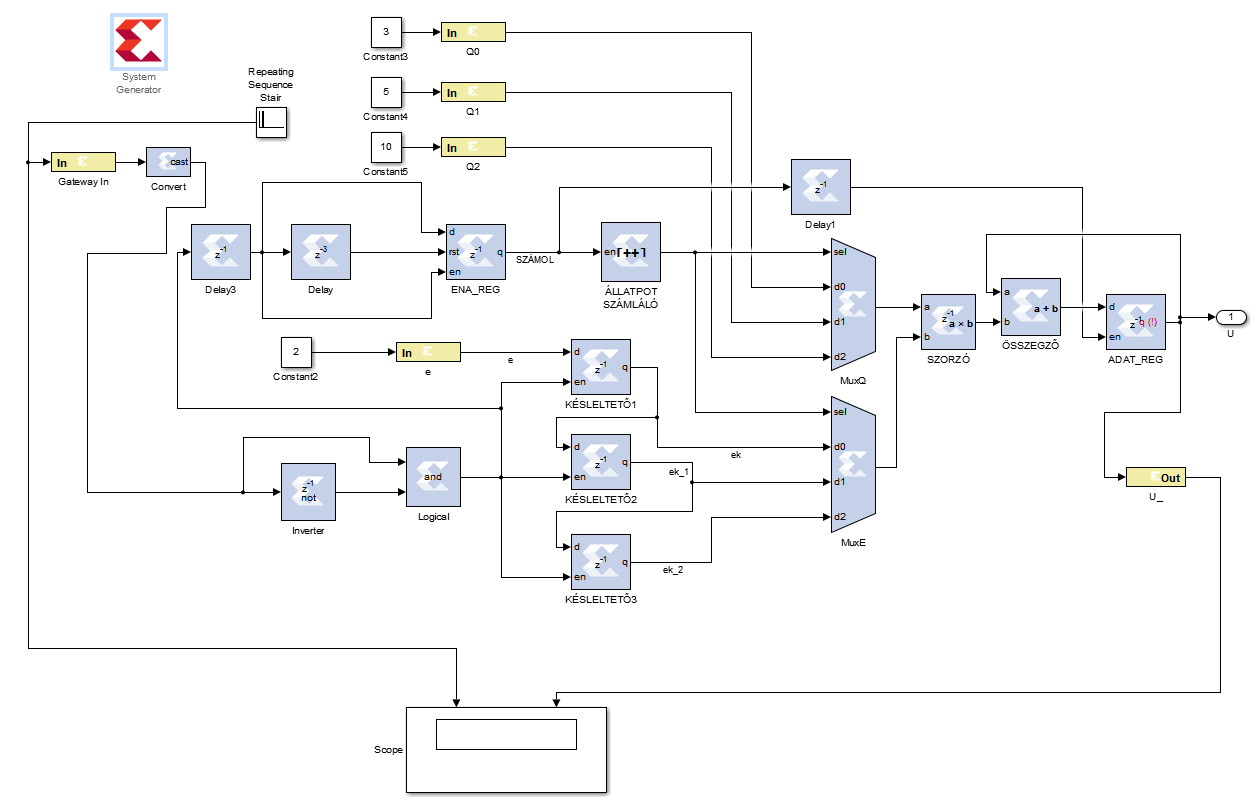
Minden állapotban egy jól meghatározott regiszterhez adjuk, hozzá a műveletek eredményét és így valósul meg a fenti rekurzív összefüggés.

Az automata mindaddig READY állapotban van, amíg a TS szignálon nem érezik egy felfutó él, ekkor a 0. állapotban végrehajtja az regiszterek eltolását, vagyis , és regiszterbe betölti az aktuális bemeneti értéket.

Az 1,2,3 állapotokban matematikai műveleteket végez és az ábrán látható összefüggések szerint és a 3. állapotból automatikusan visszatér a READY állapotba.

### Megvalósítás System Generátorban

Az állapotok tárolására és léptetésére egy 2bit-es számlálót alkalmazunk (ÁLLAPOT SZÁMLÁLÓ) amely órajelre számol, ha az enable (en) bemenetén logikai 1 érték található, 2biten a számláló maximum a 4 értéket vehet, ezért a számlálót úgy állítjuk, be hogy a maximális értéke 2 lehessen, így három állapotunk lesz, és utána a következő lépésnél ismét 0 lesz az értéke. Az állapot számláló regiszter a fenti állapot automata diagramon az 1. 2. 3. állapotok kiválasztásában játszik szerepet azáltal hogy a két MUXQ és MUXE multiplexereket megvezérli a szelekciós értékkel.



Kép. 1.3 Látható a PID strukturális felépítése System Generatorban

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 összeszorozz, 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 konstans hiba esetén végtelen nagy értéket vegyen fel.

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.

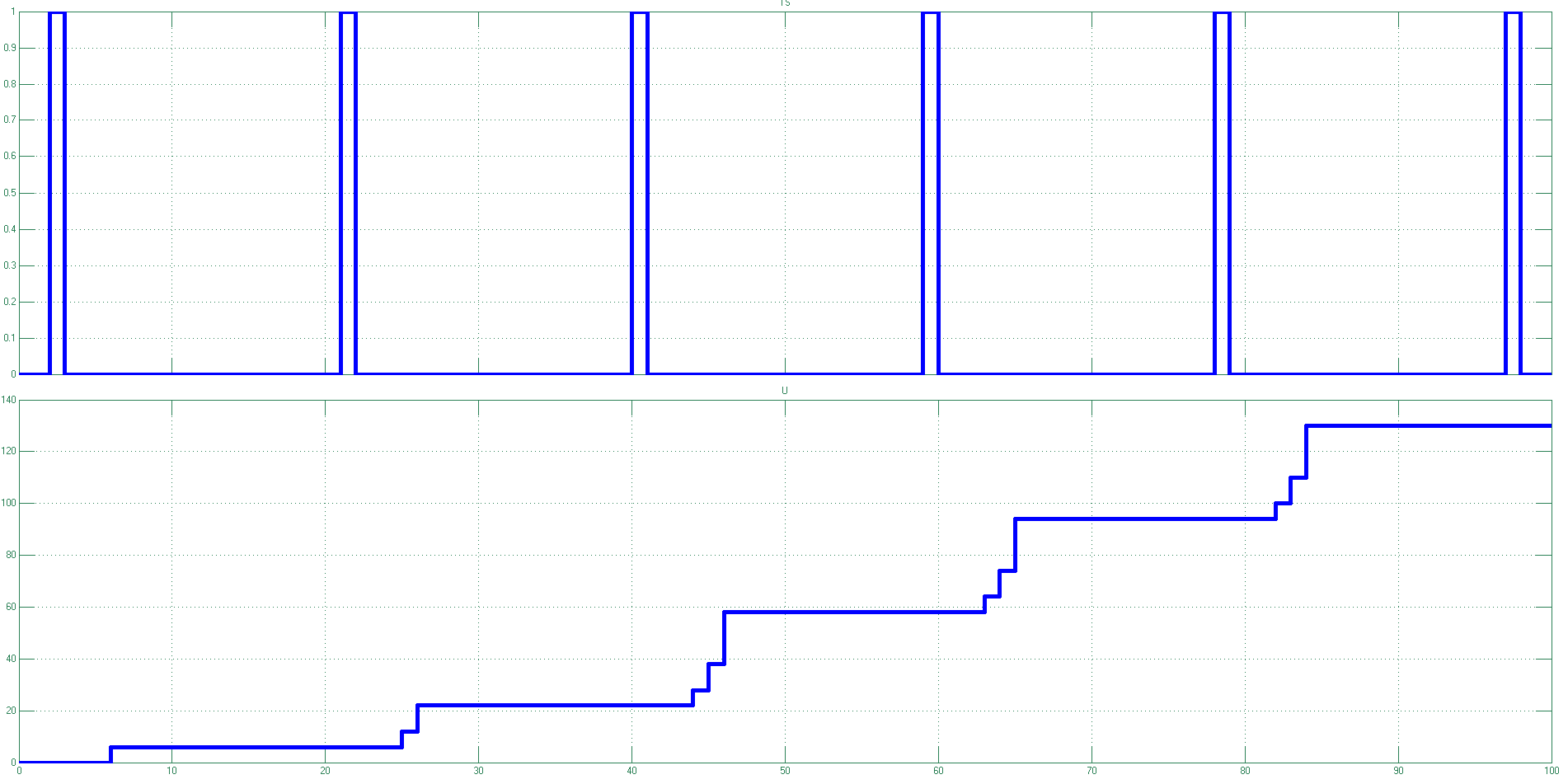
Az állapot számláló csak a regiszterek elcsúsztatása után indul el, amelyet a fent látható késleltető elemek segítségével valósítunk meg.

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.

### Szoftveres Szimulációs eredmények

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

Bemeneti paraméterek: , a bemenet konstans:



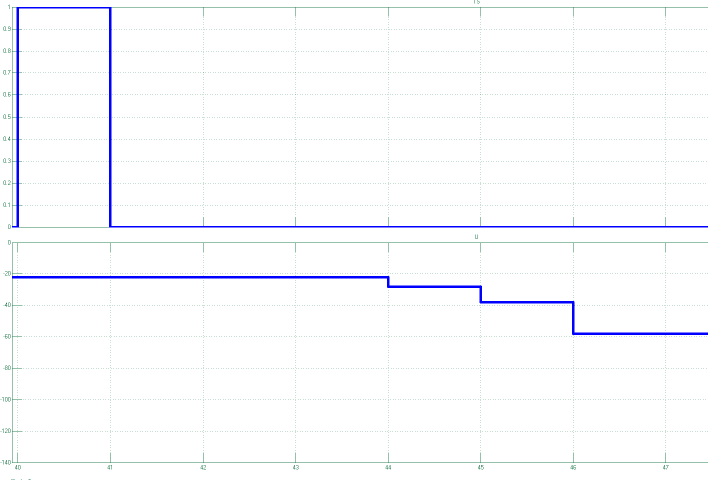
Kép. 1.4 Szimulációs eredmény amely tükrözi a konstans bementre a számolási lépéseket

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

Következtetések a Szabályzó az elvártaknak megfelelő eredményeket ad vissza. A szimulációkból többek között meg 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.



Kép. 1.5 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 és a et 0-nak, de lehet egy nullához közeli pozitív szám.

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.

## Hardveres mérések

### Dc motor sebesség szabályzása mérőstandon

A sebesség szabályzás még szükséges érzékelőt az általam megvalósított inkrementális szenzor 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.

### Pozíció Szabályzása csigahajtás áttételen keresztül DC motor segítségével gyors Hardveres Automatával

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.

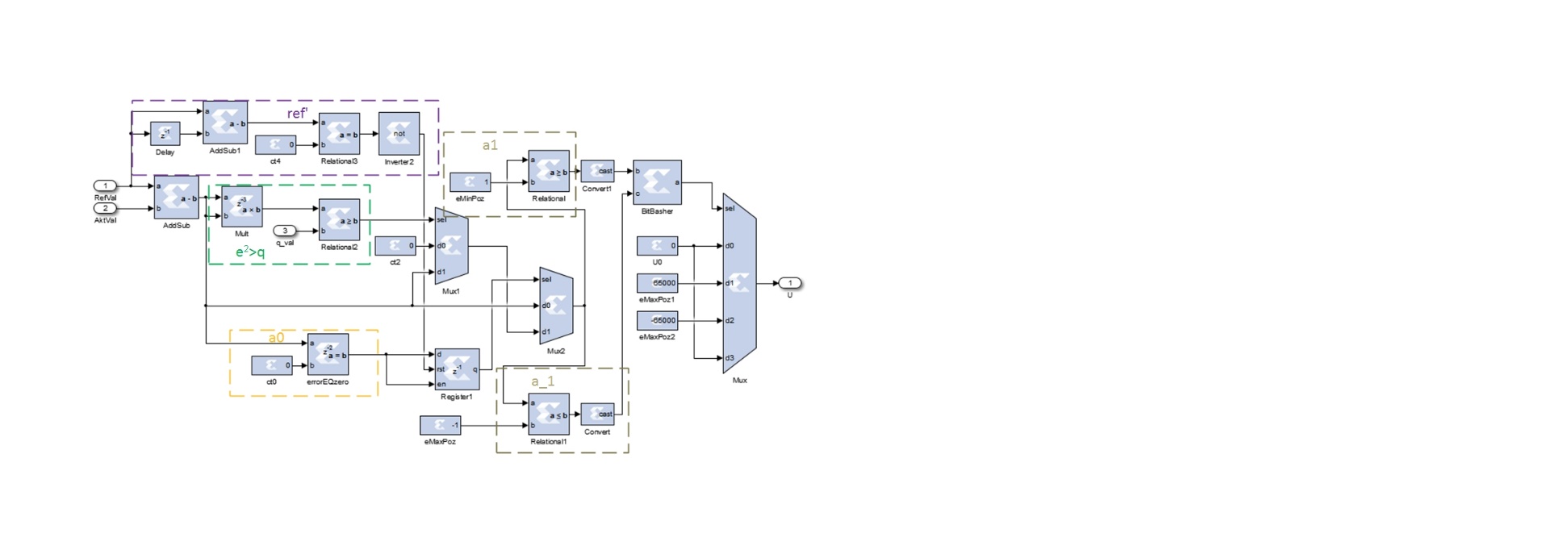
Az elkészített szabályzót a kivetkező egyenletek írják le:

Elmondható a kimeneti szabályzó jel függ a hiba értékétől, amit manipulált hibának nevezhetünk. A bementi, hiba manipulálására azért van szükségünk, mert 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:

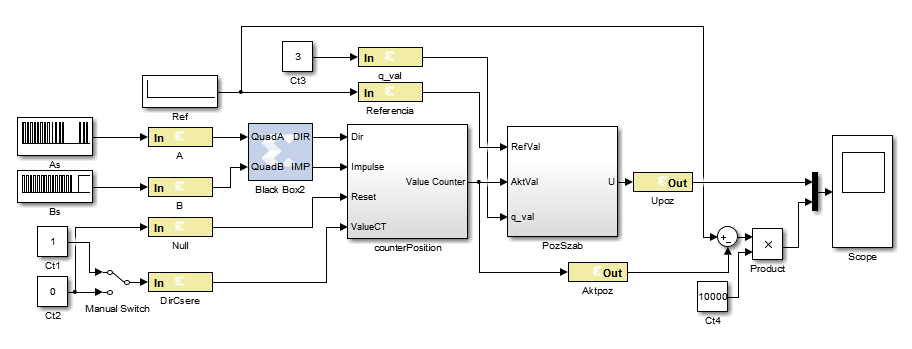
* ref’-a referencia értéket figyeli és minden órajel periódusban megvizsgálja, hogy van váltózás az értékben.
* – négyzetre emeli a hibát és összehasonlítja az általunk megadott q értékével.
* MUX – az U kimeneti jel állapot multiplexere.
* MUX1 –az manipulált hiba multiplexere
* MUX2 –hiba multiplexere



Kép. 1.6 Látható a Pozíció szabályozó System generátoros felépítése

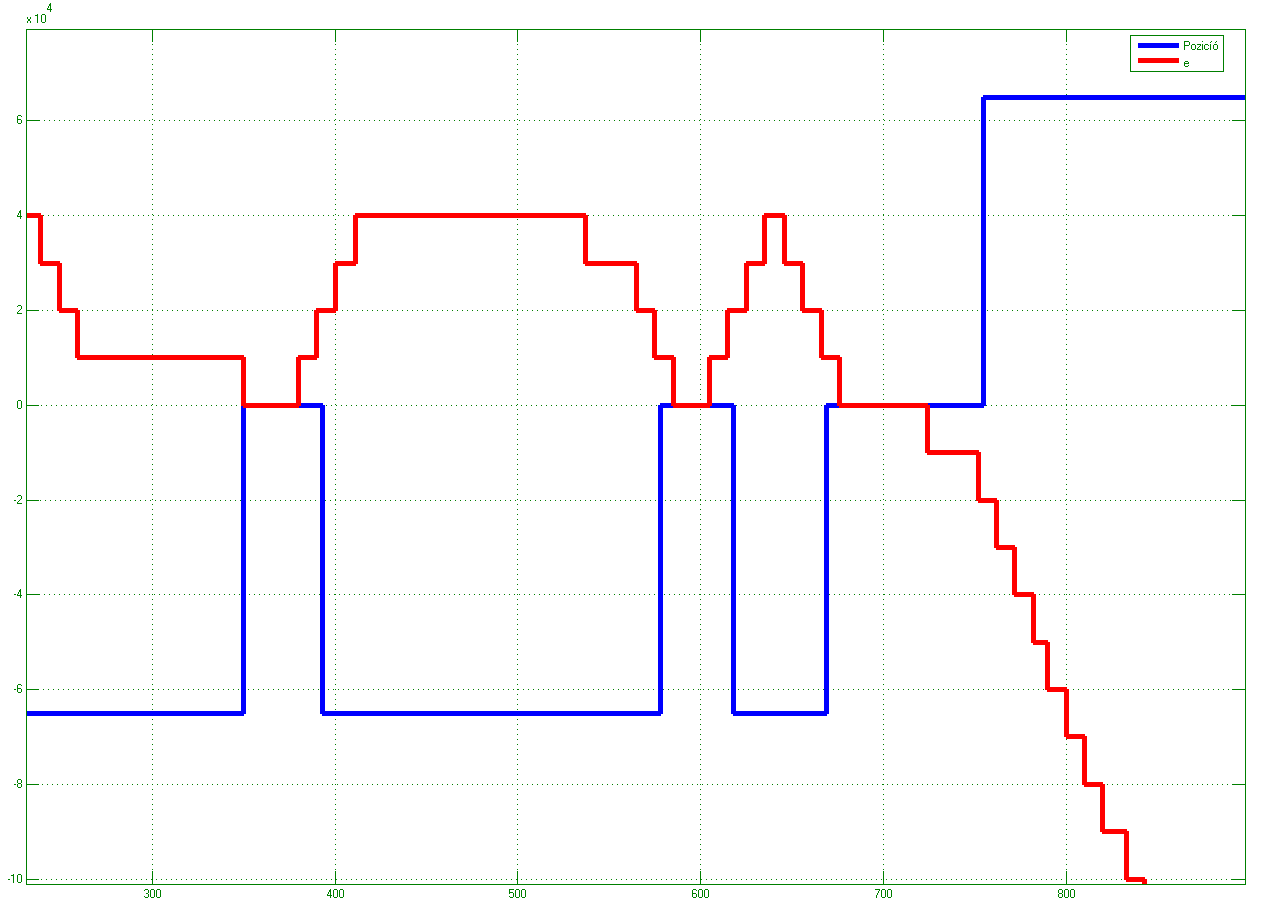
### Szimulálás Szoftveresen

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.



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

A grafikonról megfigyelhető hogy a kimeneti jel miként változik a fiba 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 kilépést.



Kép. 1.8 látható a pozíció szabályzó bemenete (pozíció hiba), valamint a Szabályzó kimenti jele.

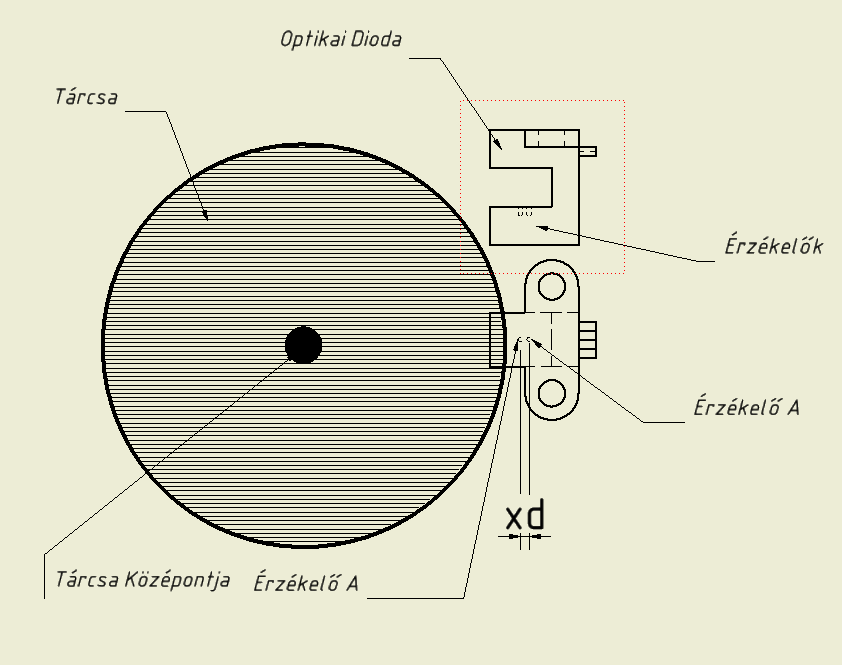
# Szenzorok

## 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.

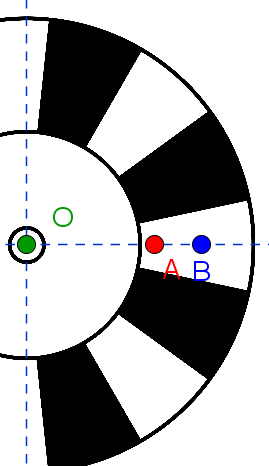


Kép. 2.1 Optikai inkrementális vevő felépítése és elhelyezése

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.

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



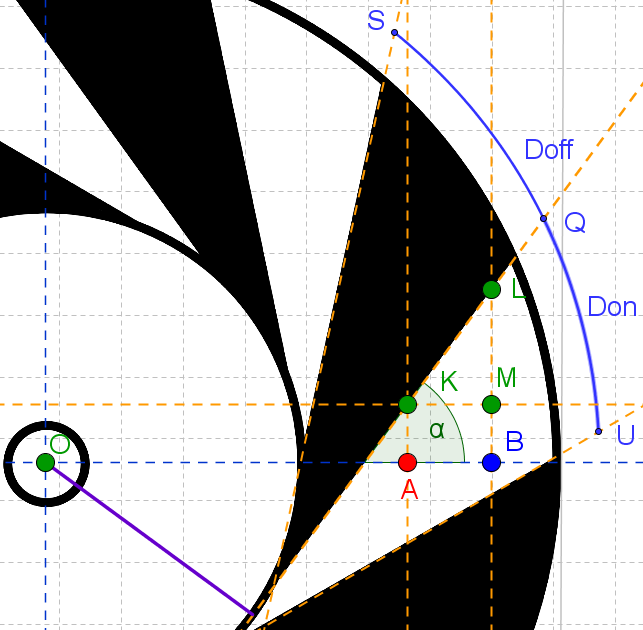
Kép. 2.2 Érzékelő tranzisztorok elhelyezése

Ha a réseket sugár irányban helyeznénk, el nem jönne létre késés a két jel között (A és B), így nem lehetne meghatározni a forgás irányát. Ezt elkerülendő a réseket meg kell dönteni 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.

A fehér mezők az inkrementális tárcsa réseit képviselik. A rések száma megadja, a tárcsa felbont ásást N.

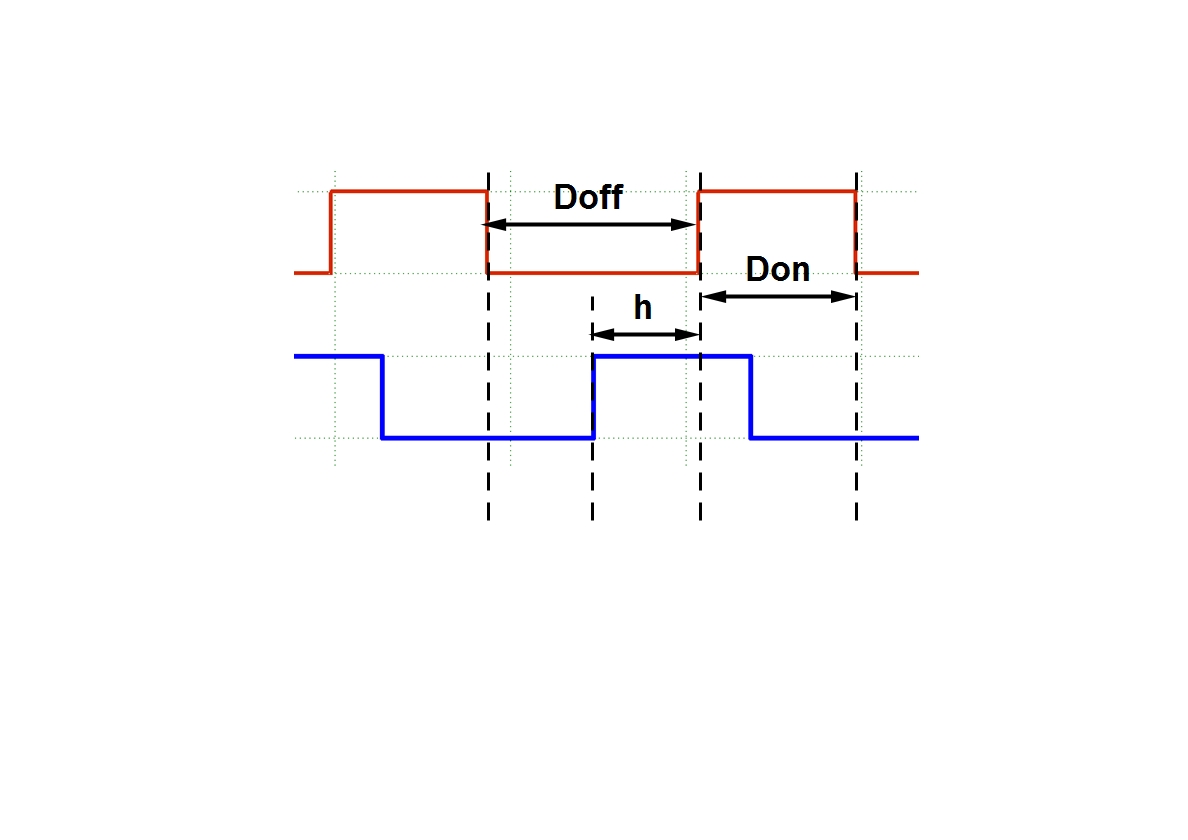


Kép. 2.3 Rések és az Érzékelők közti kapcsolat

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

Az érzékelők tartó egyengesse, amelyeket A és B pontok határoznak meg, a tárcsa sugarára merőleges egyenessel egy alfa szöget zár be. Az alfa szög változtatásával megváltozik az A, és B jelek közti időeltolás.

**Ismertadatok**:



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

Ha az érzékelő A és B pontjai a tárcsa sugarán helyezkednek el, akkor felírható az összefüggés, amely meghatározza a két jel közti késést.

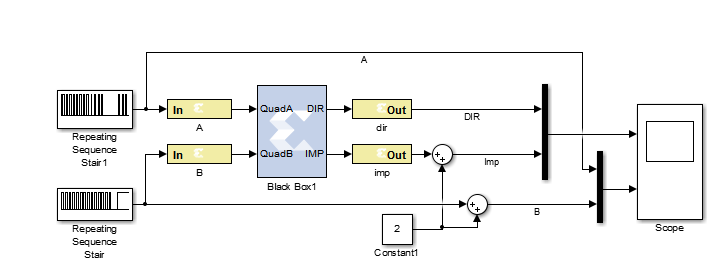
Kép. 2.4 Idődiagram a Tárcsa paraméterei függvénzében

,.

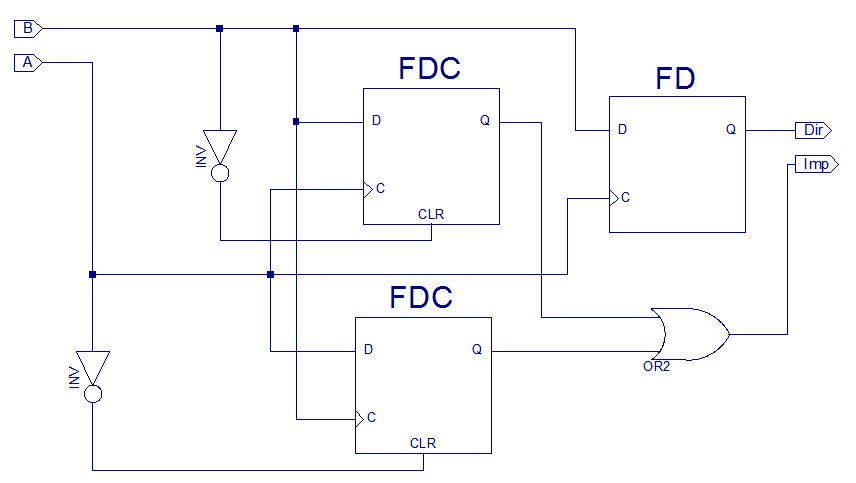
A minden esetben nagyobbnak kell lennie, mint a távolságnak.

### 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. 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).



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

 A modul VHDL nyelvben van megírva, az alábbi kép látható logika alapján:

A beérkező A, és B jeleket egy tagadó kapu segítségével bekötjük a FDC tárolok órajelére, és a bemenetére a képen látható módon. Egy harmadik tároló segítségével eldönthetjük, hogy mikor haladt el egy rés az érzékelő előtt.

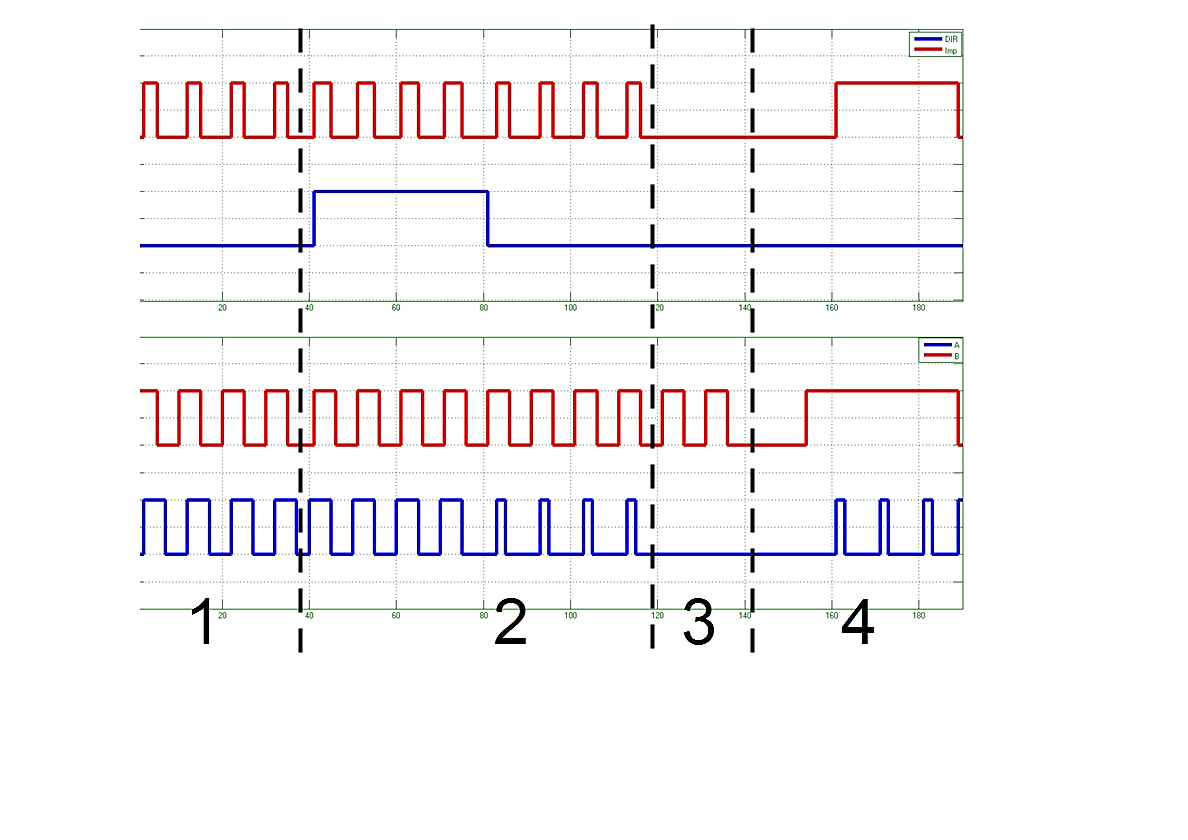
### Szimuláció System Generatorban

A VHDL fájlt egy BlackBox modul segítségével integráljuk a körszeletbe. A ki és bemeneti portókat illesztjük a Simulink környezetben található elemekhez, majd létrehozzuk a szimulációs bemeneti jelelt

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

A bementi jelek négy kategoriara sorolhatok:

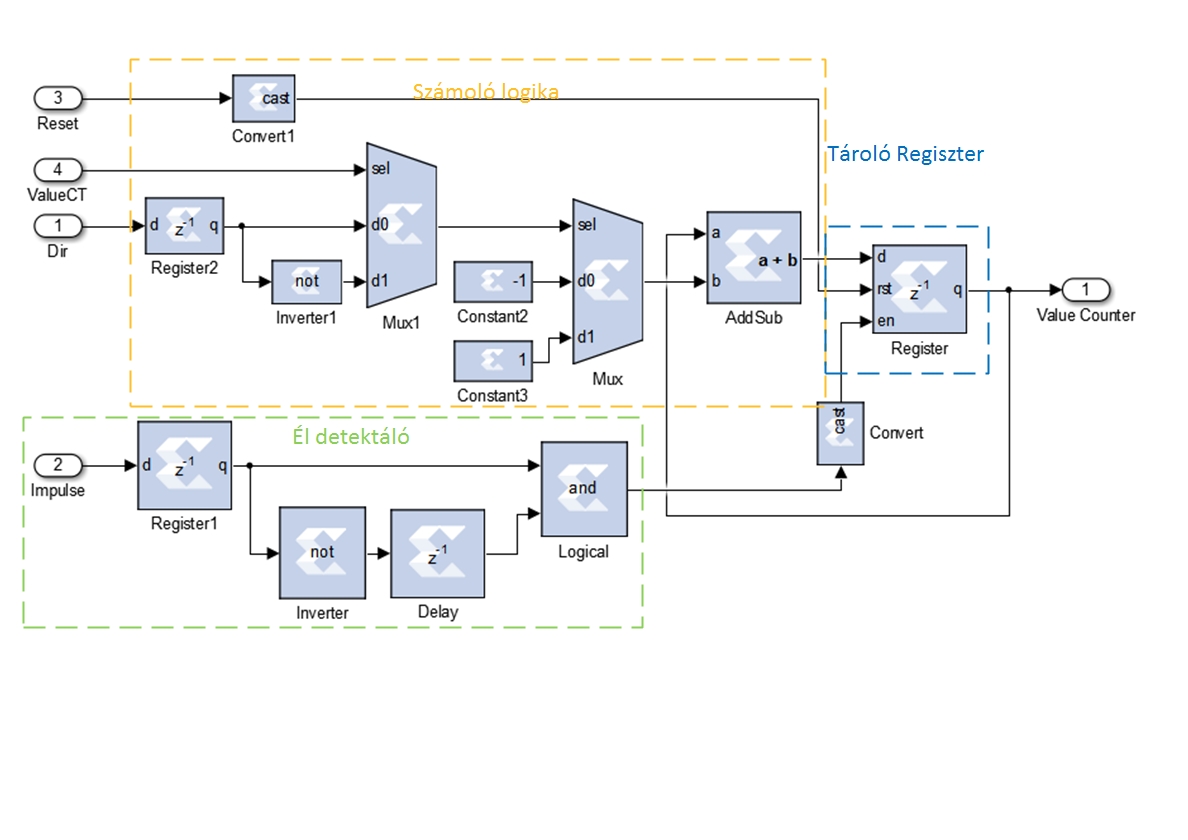
1. Az A jel késik a B jlhez képest, a kimenti jelekn látható az érkező impulyusok és az irány.
2. Az A jel késik a B jelhez képest, látható hogy az irány megfordult.
3. Az A bemeneti jelen hibás adatok érkeznek, látható hogy ekkor nem történik impulzus generálas a kimeneten.
4. Az A bemeneti jelen ismét 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. 2.6 Szimulácios eredménzek a lehetséges bemenetekről

### 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. 2.7 Pozíciót mérő modul Felépítése System Generátorban

A lenti ábrán látható a fentiekben már letárgyalt Inkrementális adó jeleinek a feldolgozásáért felelős BlackBox modul, valamint kiegészítve a pozíció mérésére alkalmas counterPosition modullal, amelynek belső felépítése az fentebi képen látható.

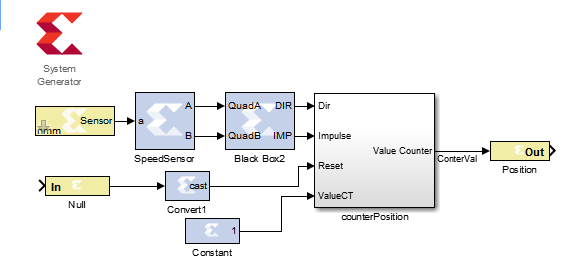
Működési logikája alapján három osztályba sorolhatok. Az első az él detektáló, melynek feladata az impulzusok felfutó éleinek a detektálása, és egyetlen órajelig tartó impulzus generálása a Tároló regiszternek így engedélyezve az adatbevitelt a regiszterbe.

Tároló regiszter, feladata az aktuális érték tárolása,a tipusa16bites előjeles egész értékű a kezdőértéke mindig 0. A rst pártján érkező jel 0 értékre hozza a regiszter tartalmát, ez akkor történik meg, ha a Null szignálon impulzus érkezik. Azt mondhatjuk, hogy a pozíciót a 0 állapothoz mérjük, amely egy meghatározott pontban van.

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

#### Hardveres mérések

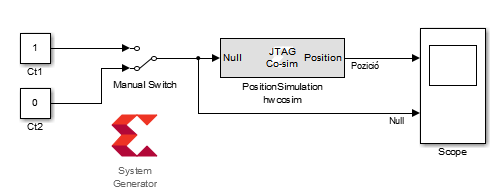
A méréseket Spartan 3e500K fejlesztőlapon láthatok, egy N=180 impulzussal rendelkező, valamint a rések amelyek 90-os szöget zárnak be a tárcsa sugarához képest.

 A Cosimulacios modell teljesen megegyezik x.képen láthatóval annyi különbséggel hogy a A és B bemenetek az FPGA két kimenti pártjára vannak illesztve így direkt az érzékelőtől érkeznek a jelek.

Kép. 2.8 Pozíció mérő felépítése

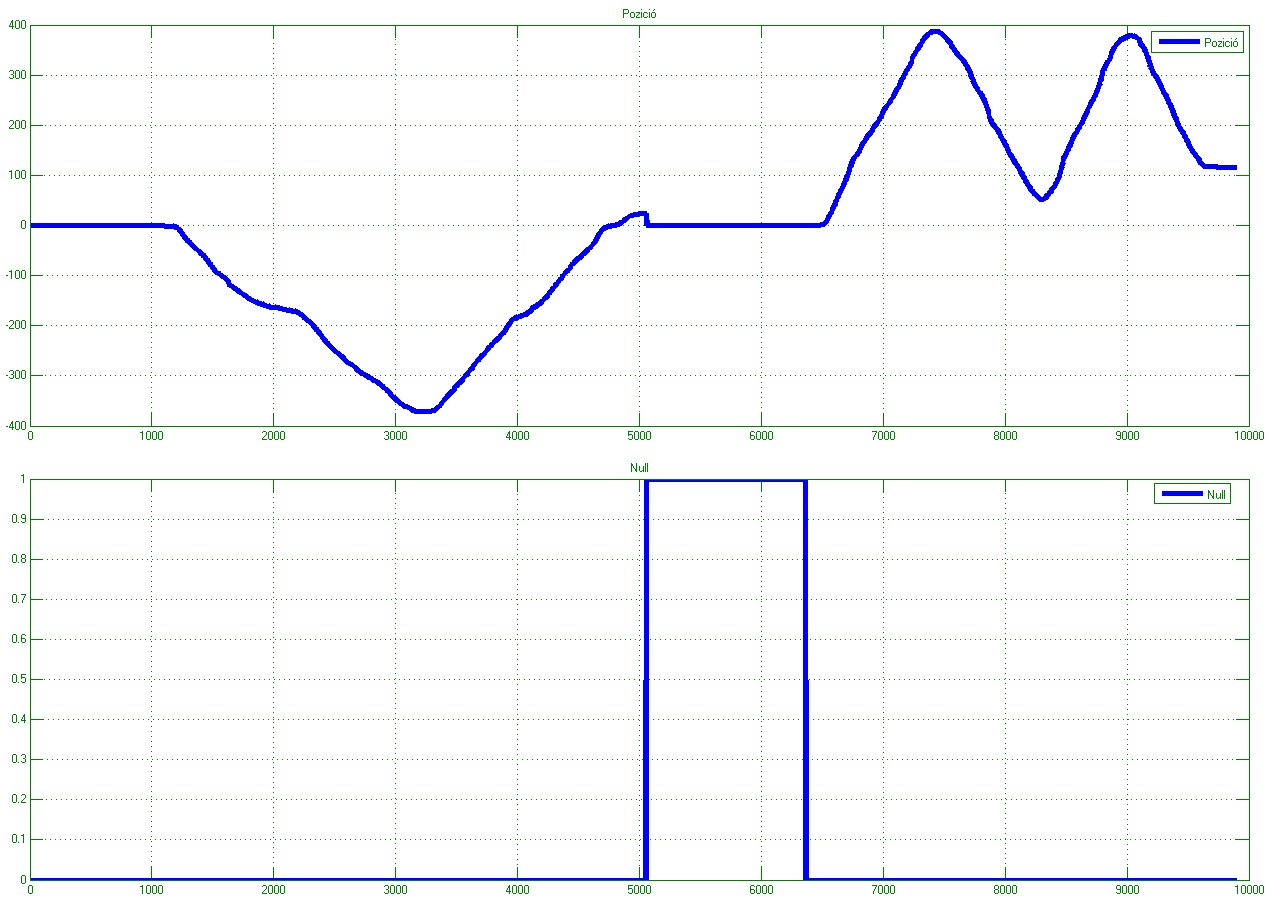
A mellékelt képen látható a szükséges modoítások.

A generálas elvégzése után 2.1.4.3.ábrán látható modult kapjuk eredményül amelyet az FPGA fejlesztőlapra feltöltve eltudjuk végezni a hardver Cosimulaciot.



Kép. 2.9 Cosimulation Inkrementális Pozíció mérő

Az 2.1.3.4.képen látható a hardveres teszt, amely során kézileg forgattam a tárcsát egy tengely körül. Látható mindaddig a pozíció 0 marad ameddig a Null szignál logikai 1 be van.

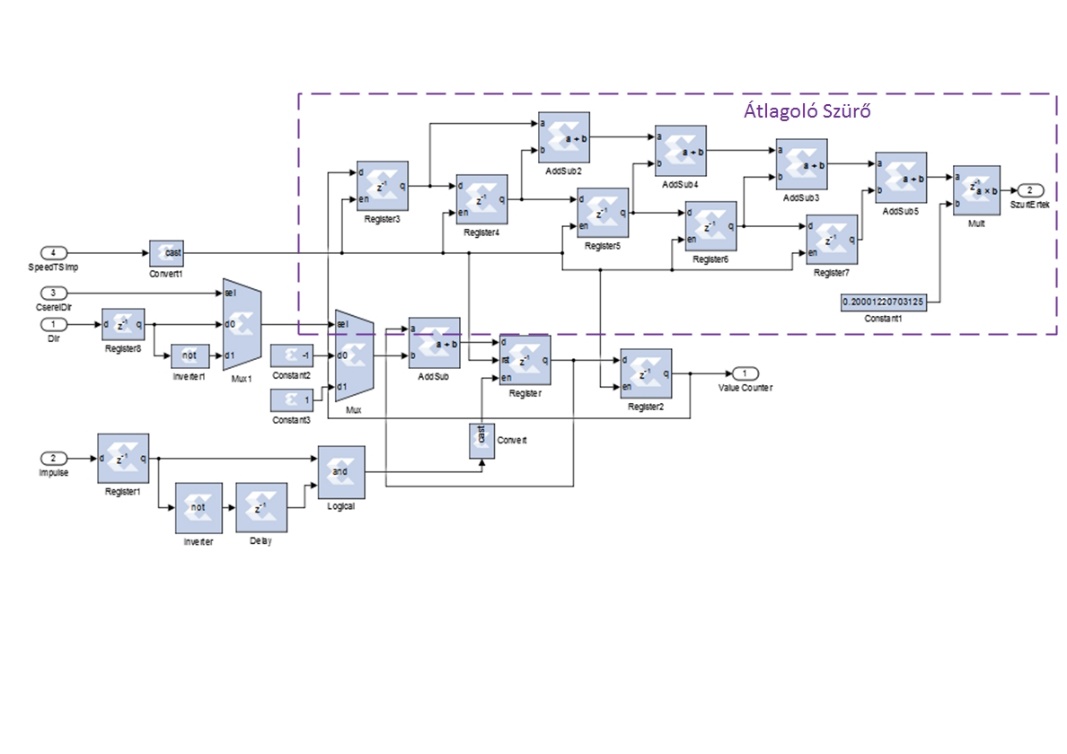


Kép. 2.10 Pozició és Null szignál mérési eredménzek

### Szögsebesség mérése Inkrementális adó segítségével

A sebesség mérésénél hasonlóképpen járunk el, mint a pozíció mérésénél. A sebességet imp/Ts, 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 szépen sorba egy szabályos mintavételi periódus taksára végighaladnak a regisztereken és minden taktusban a regiszterek értéke összegződik. Az összegzés után az értéket elosztjuk a regiszterek számával. 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 szüret len értékeket.

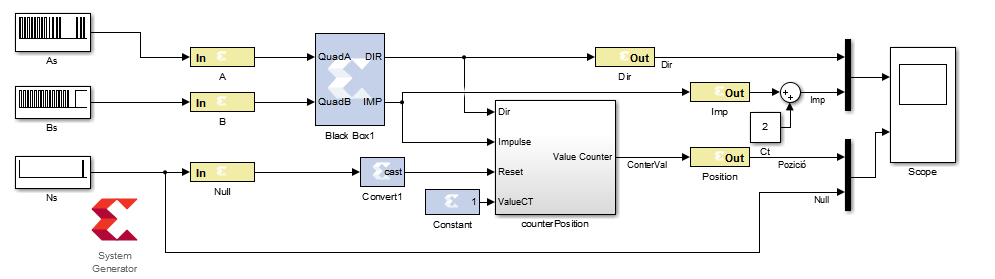


Kép. 2.11 Sebesség mérő modul felépítése

#### Szimuláció:

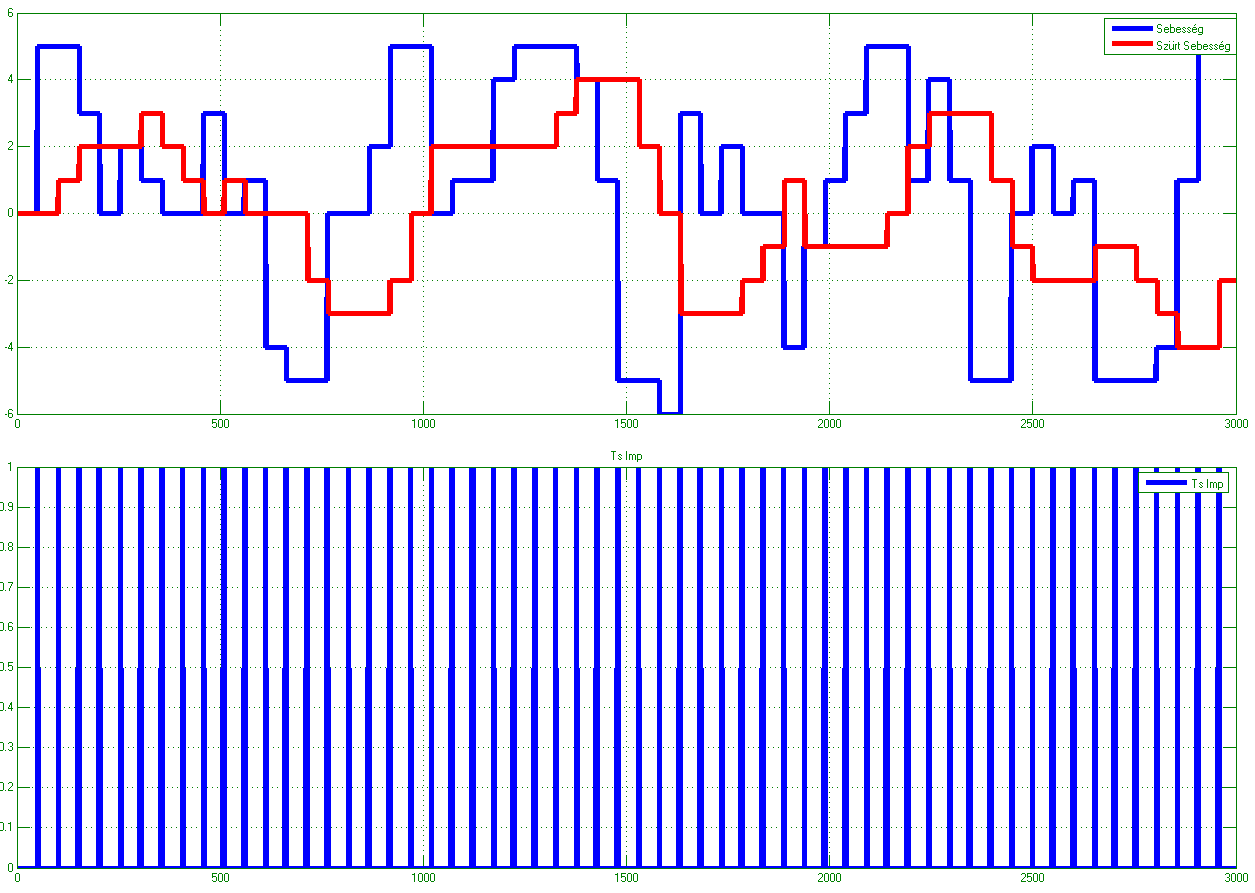
Az alábbi képen látható a sebesség mérésére szolgáló logika kialakítása. A CounterSebesseg modul tartalmazza a x.ábrán látható modult. A black box2 modul az inkrementális értékelőtől érkező jelek feldolgozásáért felelős. A Manual Switch segítségével megváltoztathatjuk a számolás irányát (lásd pozíció mérés.)

Kép. 2.12 Pozició mérő modul szimulálására szolágló Logikai kialakítás



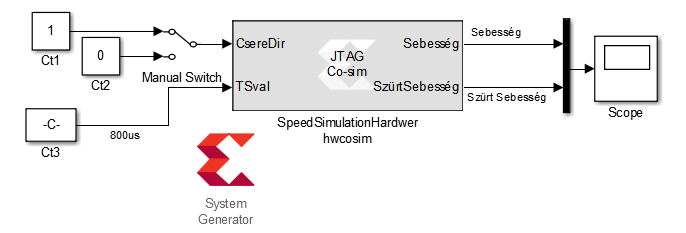
**Eredmények**:

A x.ábán alsó részén látható a szabályos időközönként érkező impulzusok, a fenti részen látható a mért és a mért értékeknek a szűrt értéke. Megfigyelhető a szűrt érték a szüret len értéknek egy 5 ablakos átlagolása.



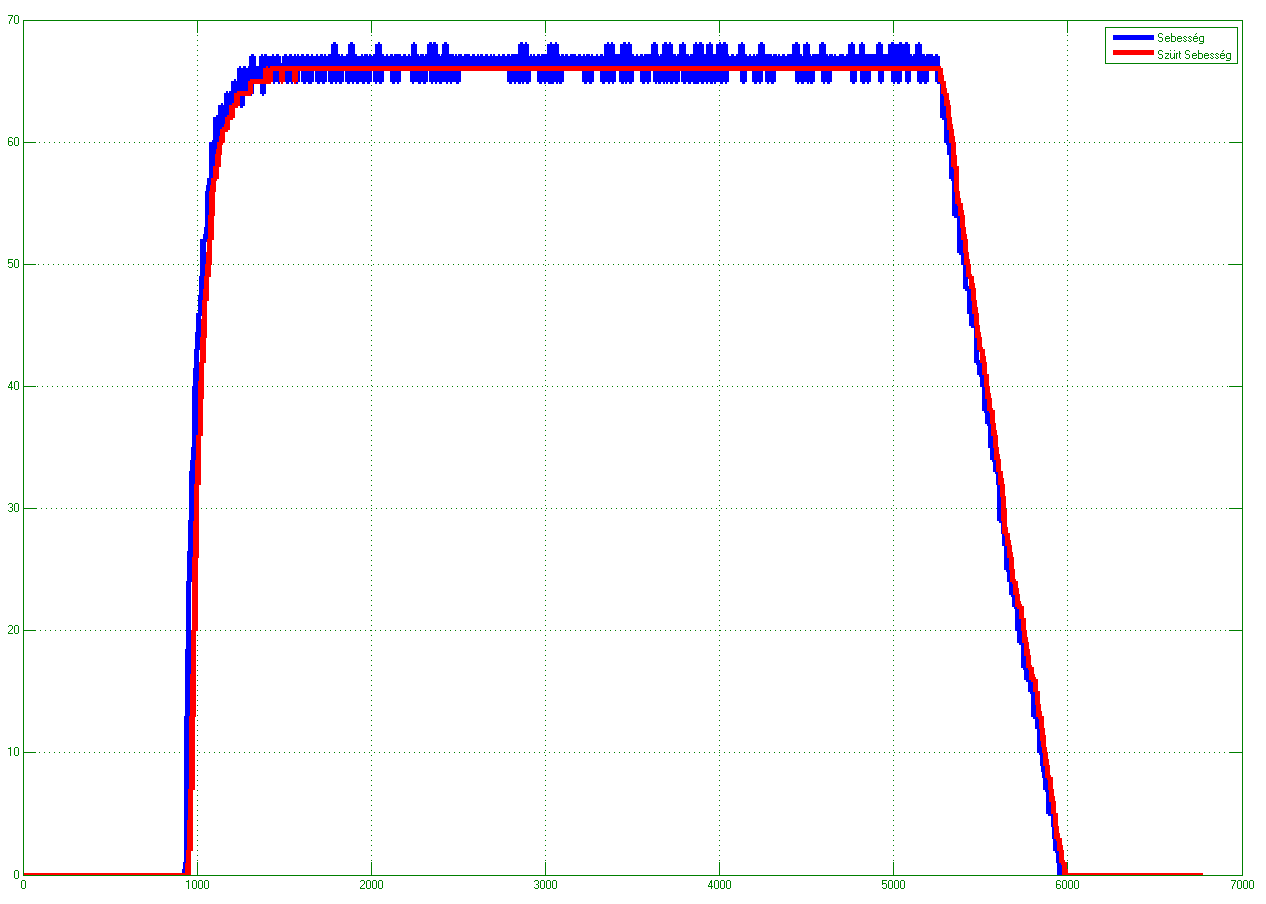
Kép. 2.13 Pozíció mérő modul Szimulálása

#### Hardveres mérések

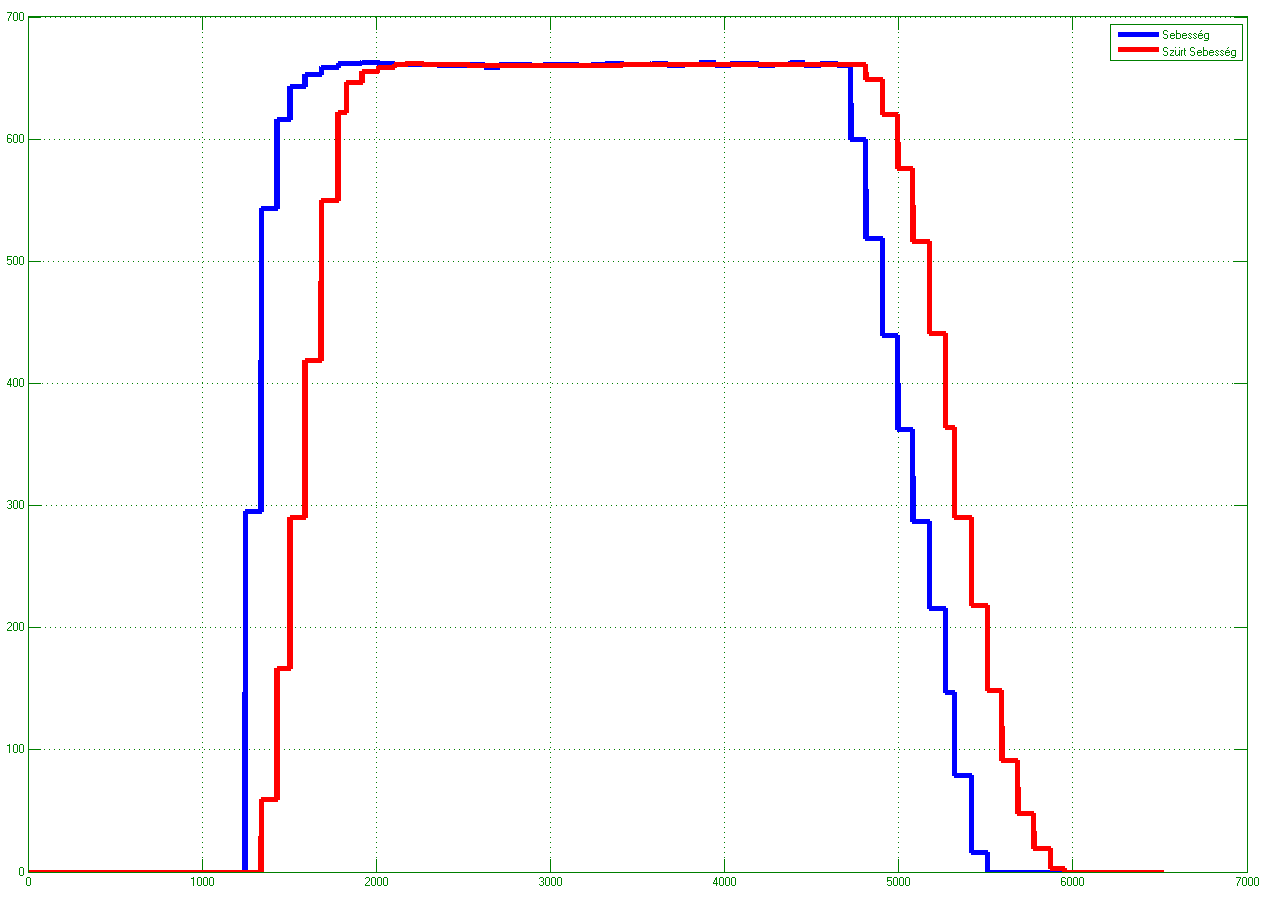
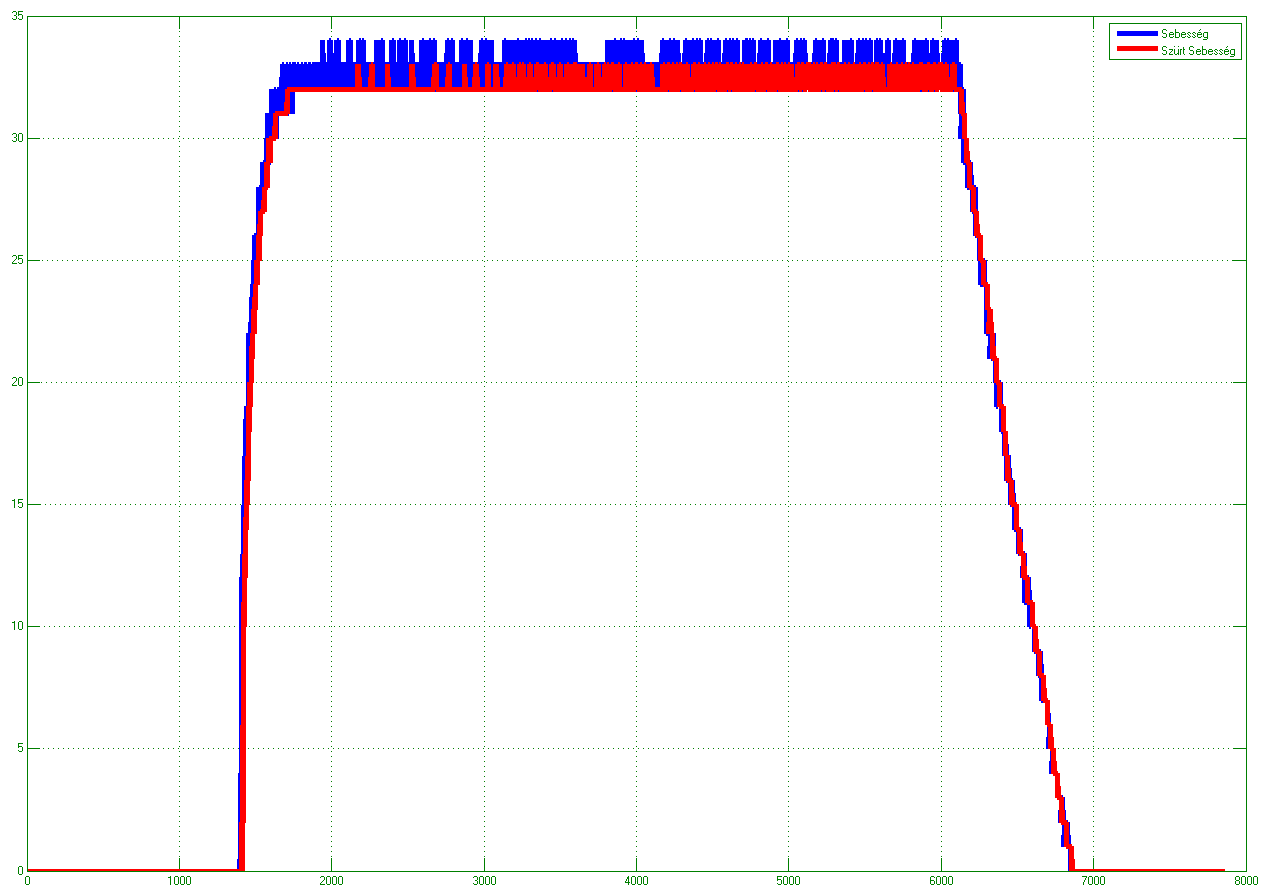
A szimuláció látható egy DC motornak a tengelyén mért fordulatszáma maximális sebességen. Ahhoz hogy eltudjuk végezni a szimulációkat módosítani kell a x.képen látható logikát, az inkrementális szenzort hardveresen illesztenünk kell az FPGA kivezetéseire.

A x.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 nevű modulon keresztül a feldolgozó modulba érkeznek be a jelek. Ugyanakkor még megtalálható egy SampleTime Generator1 nevű modul is amiznek a magyarázata megtalálható a dsfsdfdsfdsfd fejezetben. A terv kigenerálása után kapunk egy újabb modult amely SpeedSimulationHardwer hwcosim nevet viseli.

A x.képen látható az újonnan generált modul bemenő adatainak a megadása illetve a mért értékek ábrázolása. A Ct3 nevű konstansba megadhatjuk a mintavételi időt jelen esetben 800us.

 **Eredmények**: x.képen látható a motor mért sebessége illetve a szűrt sebessége imp/Ts ben. Ahhoz hogy megkapjuk a sebességet RPM ben átkel alakítani.

Kép. 2.14 8ms



Kép. 2.17 8ms

Kép. 2.17

Kép. 2.17

Következtetések: A fenti példában a az maximális fordulat, látható hogy a sebesség skálát 65 része osztottuk fel. Ha növelni szeretnénk a felosztások számát () Növelnünk kell a mintavételi időt. Az alábbi összefüggés szerint kiszámíthatjuk az optimális, mint vételi időt ismerve a maximális fordulatszámot percenként, és a tárcsa adatait.

# 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 típus egy 16bites pozitív egész értékű.

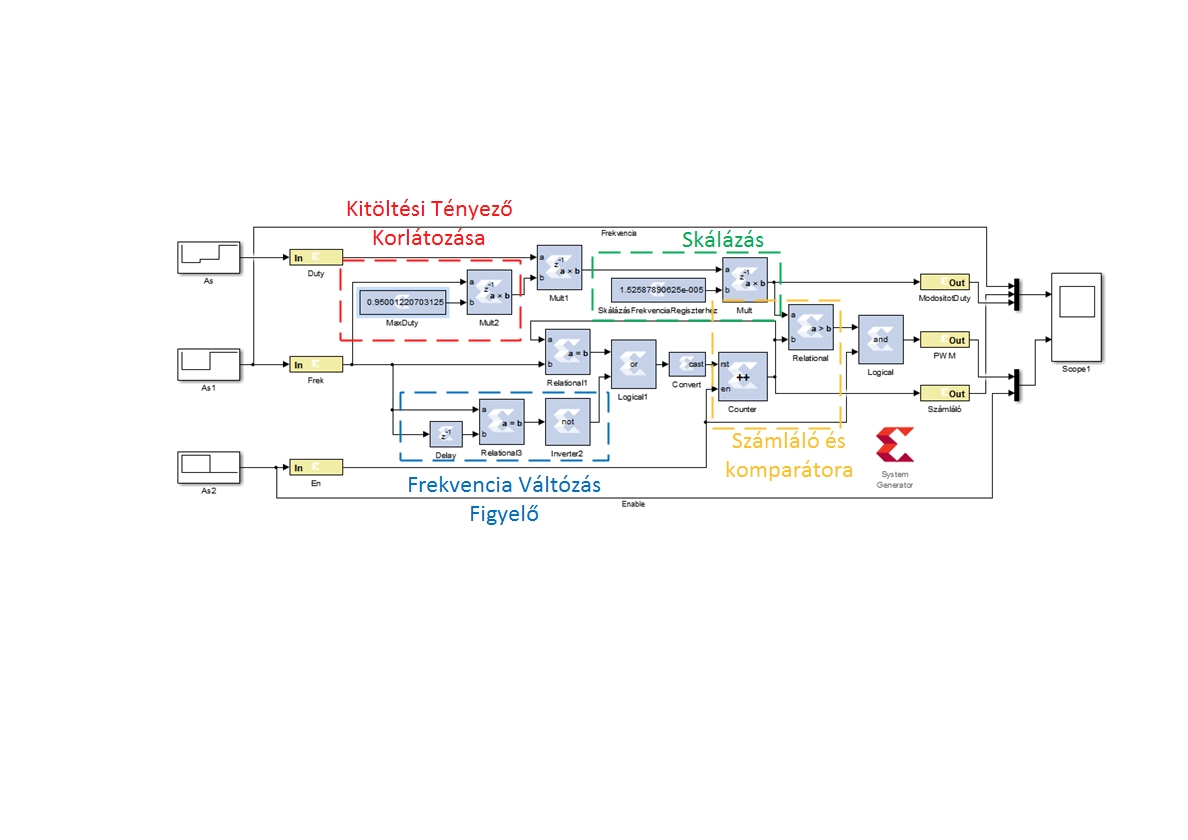
A PWM jel frekvenciáját egy Frek nevű regiszteren keresztül adhatjuk meg, a regiszter értéke összehasonlítódik a számláló értékével, és amikor a számláló értéke megegyezik a regiszter értékével a számlálót reset állapotba hozzuk.

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 meg a felépítésében meg aláható 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 számlál ásást letilthatjuk az (Enable) bemenetre adott logikai 0 értékkel, máskülönben engedélyezve vannak.

**Skálázás és bemeneti értékek:**

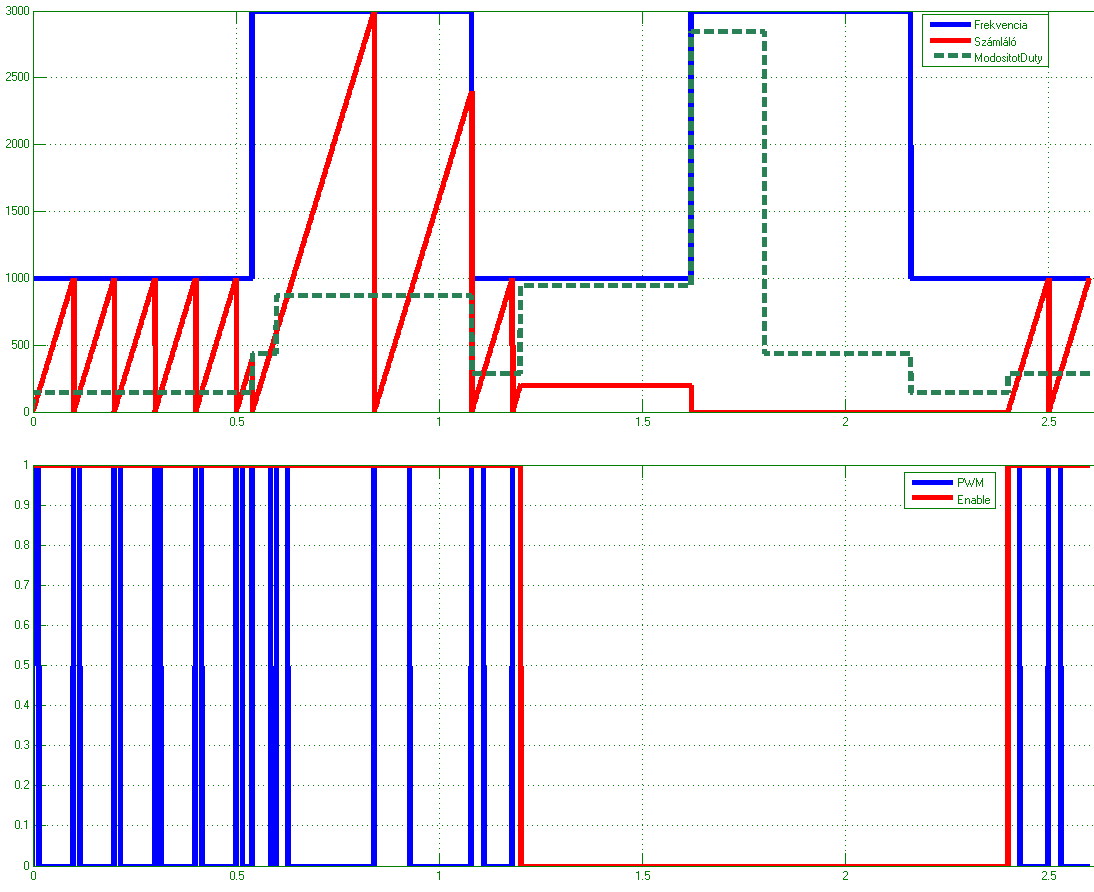
A Duty bemenet maximálisan 0 és 16biten felírható maximális érték között lehet (65535). Az alábbi egyenlet segítségével kiszámíthatjuk a kitöltési tényező regiszterének az értékét.

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 neveljük a frekvenciát csökkeni fog a rezolúció.

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



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

## Digitális Elektronika

Az szenzorok adatainak a gyűjtésére és a beavatkozó jelek számítására FPGA rendszert használtam a gyorsaságuk és a flexibilitásuk miatt. FPGA rendszeren, kényedén 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 meg megtalálható egy FPGA terület, és egy előre elkészített 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.

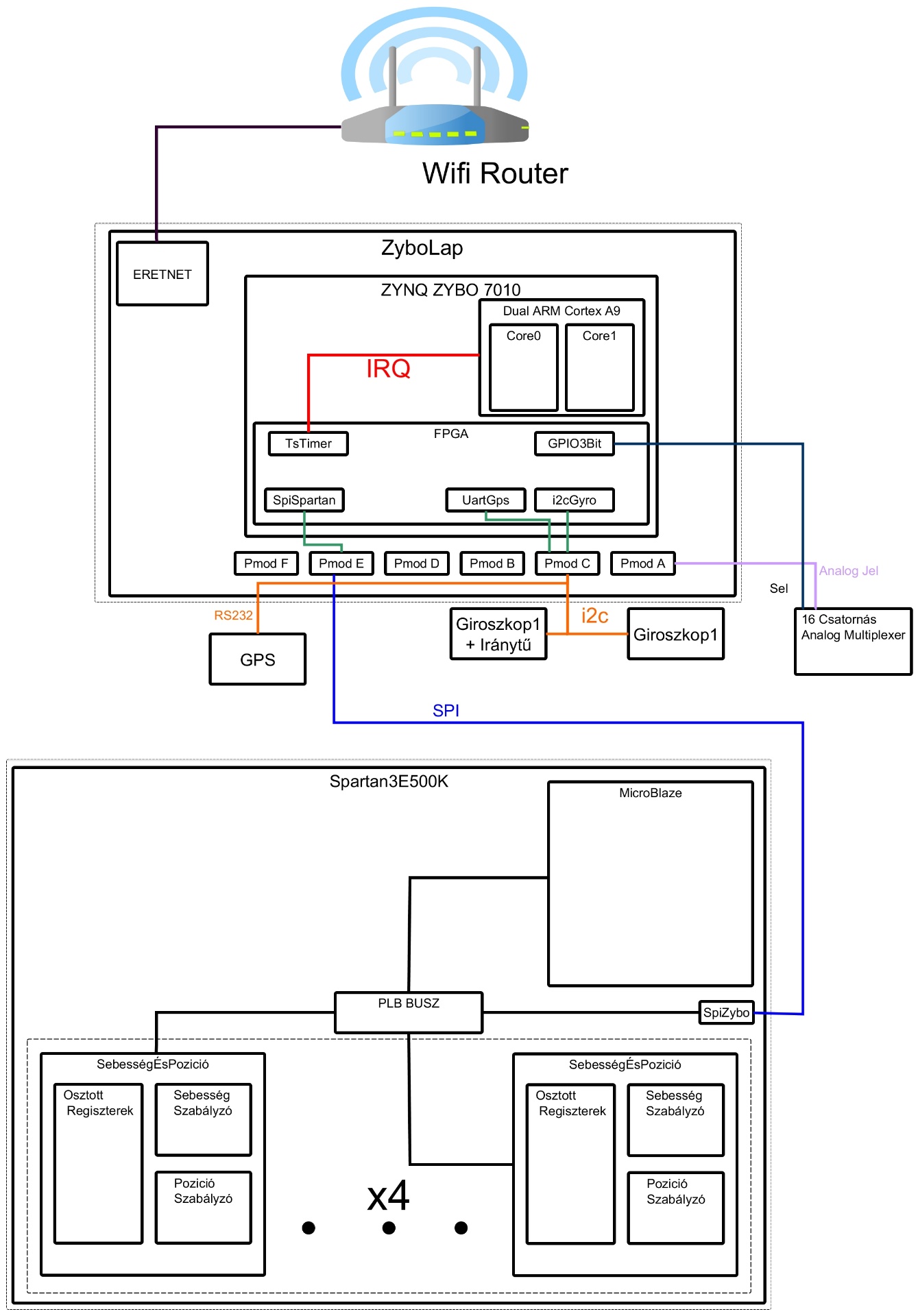
A Spártán lap tartalmaz több mint 120 kivezetést, és egy Spartan3e500k FPGA chipet.

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 ip mag osztót regisztereken keresztül állit hatjuk be a paramétereit vagy olvashatunk ki értékeket, amelyek szintén az AXI buszon találhatók.

A Zybo lapon megtalálható Eternet modulon keresztül vezetéken keresztül kapcsolódunk egy Wifi routerhez amely Access pontként működik. A routerhez meg csatlakoztathatunk három más vezetékes eszközt, amelyek lokális hálózatba lesznek kötve a Zybo fejlesztőlappal.

**A fenti képen látható a struktúra kialakítása, amely szerint a rendszer működik.**



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 egy 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 a relatív pozíció meghatározásánál játszanak fontos szerepet.

**Kimenetek**:

**Spartan3e**:

* 4 darab kimenet a PmodB csatlakózón, amely segítségével kiválasszuk az analóg multiplexer kívánt csatornáját.

**Spartan3e**:

* 8 PWM kimenet és a hozzájuk tartozó Dir jel, összesen 16 darab.

### Vezetékek Elrendezése

# Software

Ip mag generálása

## Feladatok Elosztása

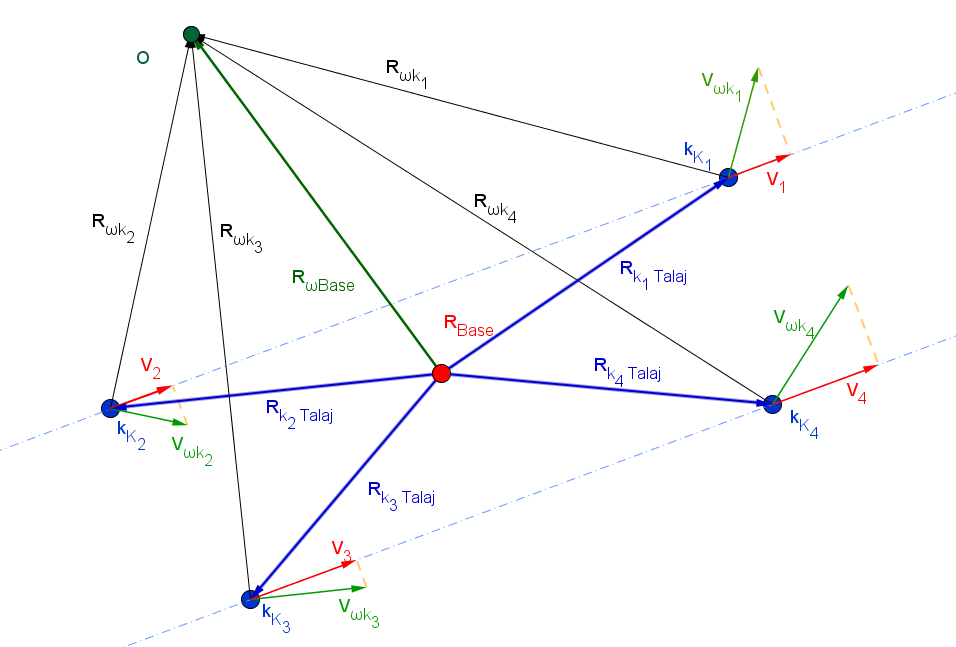
### Zybo fejlesztőlap

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

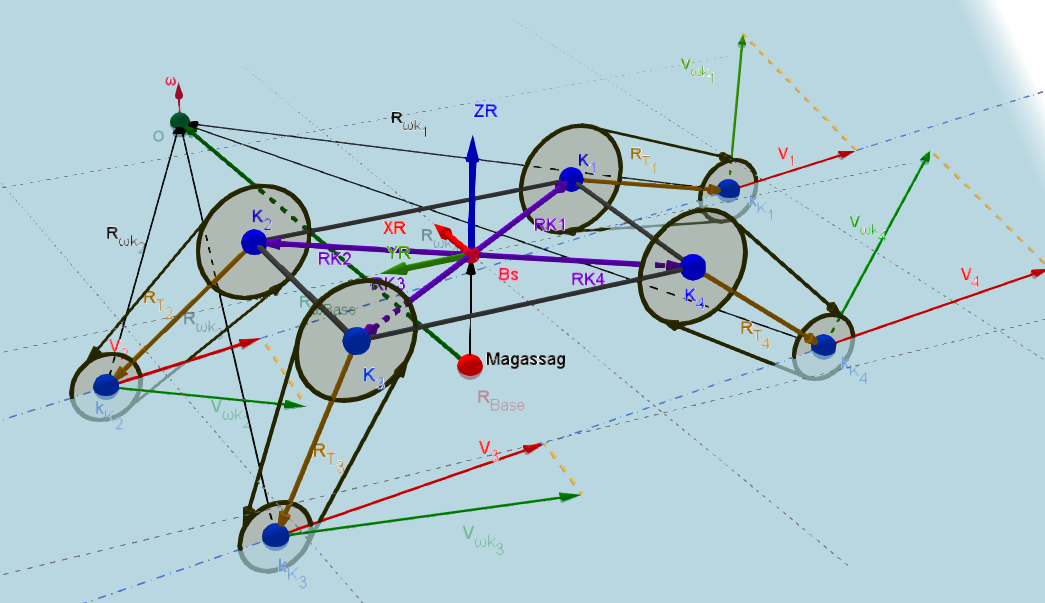
A Core0 processzor feladatai között szerepel a megszakítások lekezelése, a legfontosabb megszakítása a mintavételi periódust generáló számlálótól érkező megszakítás, amelynek bekövetkeztekor a processzor begyűjti, az adatokat az szenzorok tol (Giroszkóp 1 és 2). Lekezeli a megszakítást, amelyek az UART 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 számi SPI kommunikáción keresztül elküldi a szabályzók referencia értékeit a Spartan fejlesztőlapnak.

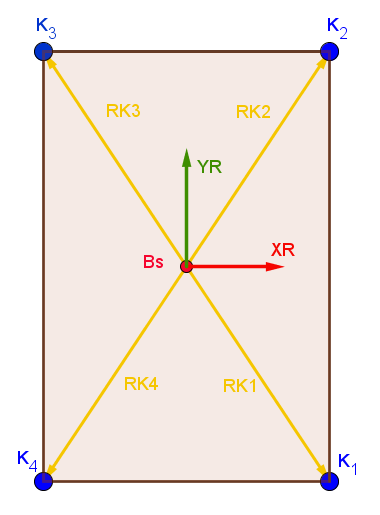
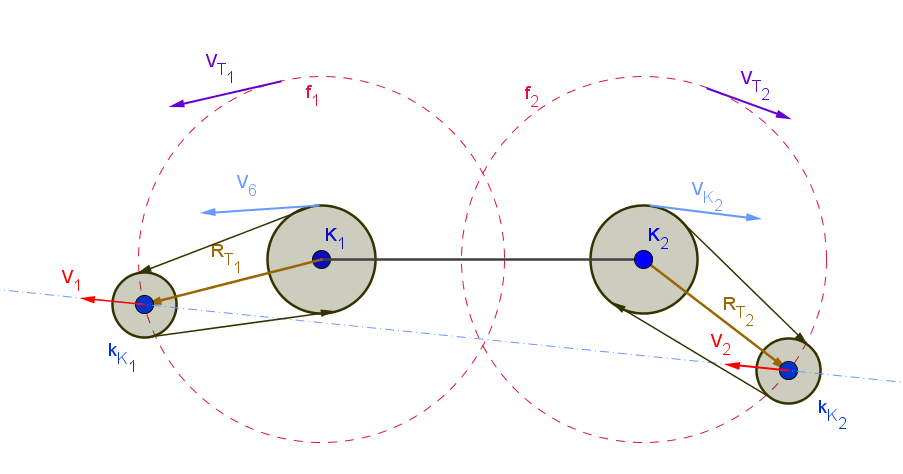
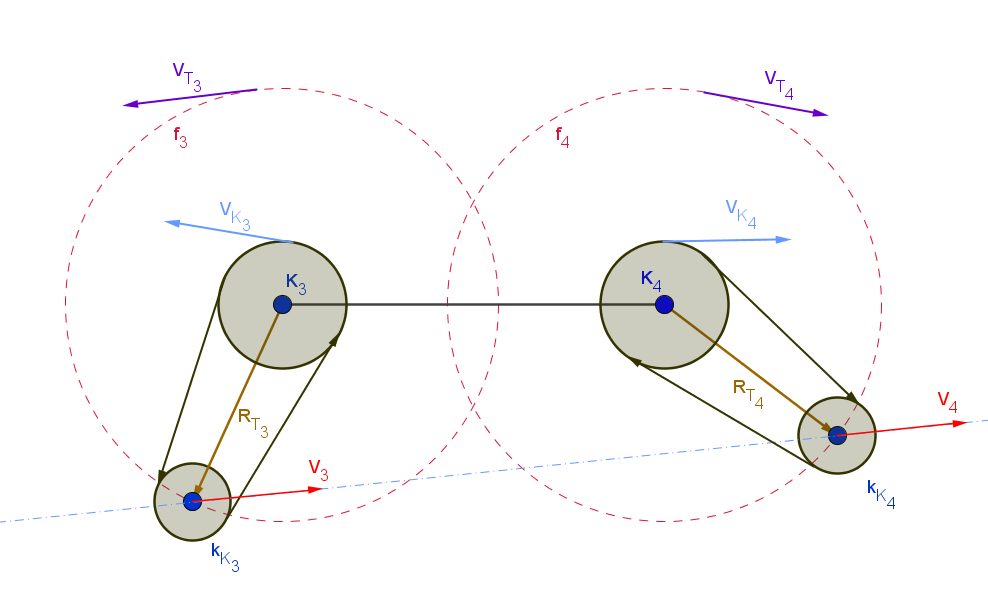
# Robot Modell



Kép. 5.1 Robot Fordulása adott pont körül adott sebességgel



Kép. 6.2 Robot 3D vektorábrája



Kép. 6.3 Oldalnézetek és Felülnéztet

**Jelölések**:

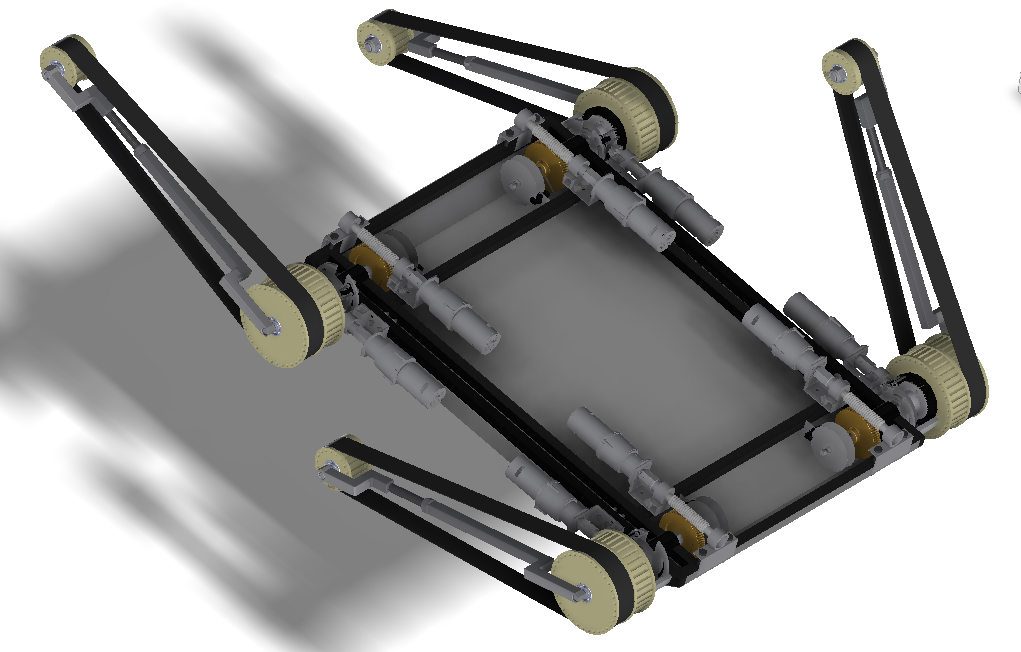
Szeretnénk, ha a robotunk egy adott körpályát írna le egy pont körül egy adott sebességgel. Jelen esetben O pont körül és, szögsebességgel.

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 amelyek könny fémprofilokból áll össze és hegesztésekkel rögzítjük egymáshoz. A váz és az egész rendszer szimmetrikus két tengelyre nézve is ezért a továbbiakban csak a rendszer negyedét részletezzük.



Kép. 7.1 Robot vázának Inventoros 3D Képe

