**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM**

**MŰSZAKI ÉS HUMÁNTUDOMÁNYOK KAR, MAROSVÁSÁRHELY**

**AUTOMATIKA ÉS ALKALMAZOTT INFORMATIKA SZAK**

**Kültéri mobilis robot tervezése és megvalósítása FPGA alapú hardver erőforrás alkalmazásával**

**DIPLOMADOLGOZAT**

**Témavezető: Végzős hallgató:**

**Dr. Brassai Sándor Tihamér Gábor Szabolcs-László**

**egyetemi adjunktus**

**2015**

**UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA**

**FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ**

**SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ**

**Proiectarea și implementarea unui robot mobil utilizând resurse hardware bazate pe circuite FPGA**

**Proiect DE DIplomă**

**Coordonator ştiinţific: Absolvent:**

**Dr.ing. Brassai Sándor Tihamér Gábor Szabolcs-László**

**2015**

|  |
| --- |
| **UNIVERSITATEA “SAPIENTIA” din CLUJ-NAPOCA LUCRARE DE DIPLOMĂ FACULTATEA DE ŞTIINŢE TEHNICE ŞI UMANISTE, Candidat (a) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Filiala Tîrgu Mureş**  **Specializarea : Automatică și informatică aplicată**  **Anul absolvirii : \_\_\_\_2015\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Coordonator ştiinţific Viza facultăţii** |
| **a) Tema lucrării de licenţă:** |
| **b) Problemele principale tratate:**  **c) Desene obligatorii:**  **d) Softuri obligatorii:**  **e) Bibliografia recomandată:** |
| **f) Termene obligatorii de consultaţii:** |
| **g) Locul şi durata practicii :** |
| **Primit tema la data de:** |
| **Termen de predare:** |
| **Semnătura Director Departament Semnătura coordonatorului** |
| **Semnătura candidatului** |

**DECLARAŢIE,**

Subsemnatul\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

absolvent al specializării \_\_\_\_\_\_ Automatică și informatică aplicată\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

la **Facultatea de Ştiinţe Tehnice şi Umaniste – Tîrgu Mureş** , **Universitatea „Sapientia” din Cluj-Napoca** certific prin prezenta că am luat la cunostinţă de cele prezentate mai jos şi că îmi asum, în acest context, originalitatea lucrării mele de licenţă/disertaţie cu:

titlul \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

coordonator \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

prezentată în sesiunea \_\_\_\_2015\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

La elaborarea lucrării de licenţă/disertaţie, se consideră plagiat una dintre următoarele acţiuni:

reproducerea exactă a cuvintelor unui alt autor, dintr-o altă lucrare, în limba română sau prin traducere dintr-o altă limbă, dacă se omit ghilimelele şi referinţa precisă;

redarea cu alte cuvinte, reformularea prin cuvinte proprii sau rezumarea ideilor din alte lucrări dacă nu se indică sursa bibliografică;

prezentarea unor date experimentale obţinute sau a unor aplicaţii realizate de alţi autori fără menţionarea corectă a acestor surse;

însusirea totală sau parţială a unei lucrări în care regulile de mai sus sunt respectate, dar care are alt autor.

Data \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Semnătura \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Notă**

Se recomandă:

- plasarea între ghilimele a citatelor directe şi indicarea referinţei într-o listă corespunzătoare la sfârsitul lucrării;

- indicarea în text a reformulării unei idei, opinii sau teorii şi corespunzător în lista de referinţe a sursei originale de la care s-a făcut preluarea;

- precizarea sursei de la care s-au preluat date experimentale, descrieri tehnice, figuri, imagini, statistici, tabele etc.;

- precizarea referinţelor poate fi omisă dacă se folosesc informaţii sau teorii arhicunoscute, a căror paternitate este unanim acceptată.

**Proiectarea și implementarea unui robot mobil utilizând resurse hardware bazate pe circuite FPGA**

Extras

Conceptul lucrării este proiectarea unui robot mobil de teren, prezentarea elementelor necesare şi implementare.

Pe un şasiu masiv sunt ataşate patru tălpi pivotante, pe care se poate roti cu 360 de grade, faţă de şasiu. Pe tălpile pivotante sunt montate şenile. Şenilele sunt puse în mişcare cu ajutorul motoarelor dc cu angrenaj de roţi dinţate conice.

Structura şasiulul masiv este format din cadru metalic, cu profile din oţel, unde componentele sunt fixate între ele prin sudare. Structura cadrului, precum şi sistemul, sunt simetrice faţă de cele două axe.

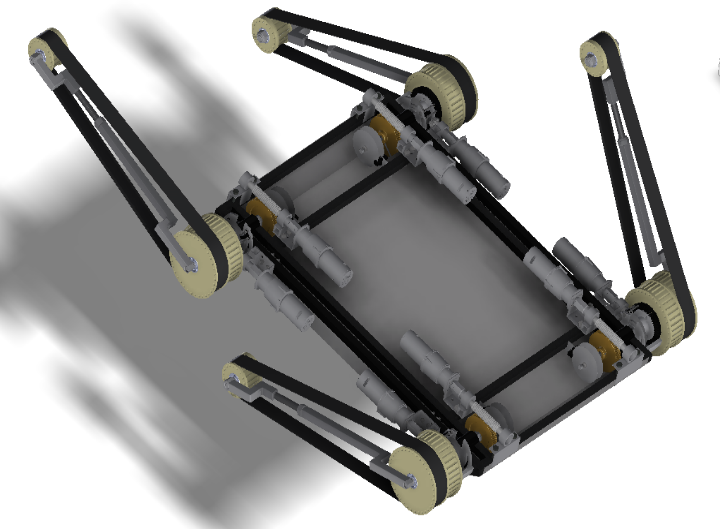


Fig. 1.31.4 Structura Robotului- Inventor 3D Foto

ForgóTalp1

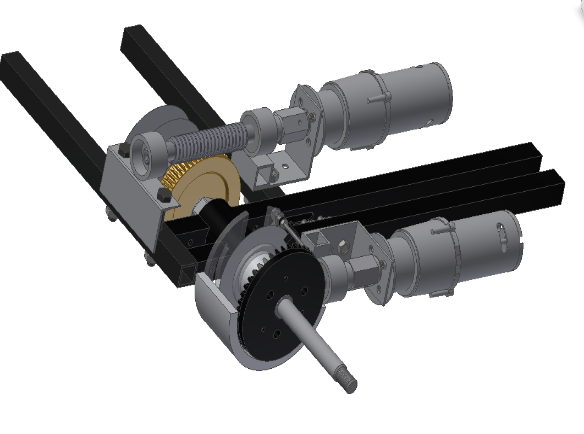
ForgóTalp4

ForgóTalp3

ForgóTalp2

KisKerék

NagyKerék







Orsó áttétel 40:1 arányban

Kupkerép áttétel 6:1 arányban

Sebesség mérő inkrementális tárcsa és szenzor

Fig. 1.31.4 Șasiu

Pe sistem se poate găsi 8 motoare DC, cu ajutorul cărora putem punem în mişcare şenilele, putem modifica unghiul tălpilor faţă de cadru. Cu sistem H-pod controlez motoarele.

## ALIMENTARE CU ENERGIE

Pe figura 3.48 se poate observa 4H-pod (cutiile A și B conțin câte 4 H-pod), podurile sunt fixate câte doi pe o tablă din cupru, și cu ajutorul unei țeavă de cupru se circulă apa, astfel răcind tranzistoarele. Tranzistoarele sunt separate galvanic de tablă, cu ajutorul unui izolator electric, cu toate acestea izolatorul are o conduită termică bună. Din sistem, temperatura este transmis cu metoda răcirii de apă, deoarece sistemul trebuie să funcționeze pe teren și praful să nu poate să pătrundă.

Pe figura 3.48 se poate vedea o conductă de 32 bit BUS, care vine din sistemul FPGA, și aici sunt semnalele de acționare pwm, care sunt necesare la funcționarea celor 8 motoare, cu 3,3 V amplitudine. . Operația Bus protejează și sistemul FPGA.

Protecția se realizează prin diode zener de 3.3 V, care are rolul de a împiedica trecerea tensiunii peste 3,3 V in sistemul FPGA. Rezistența electrică se află în linie cu diodele, care limitează curentul electric, deoarece FPGA suportă la intrare 16mA. Pe figura 3.48 este prezentat planul de alimentare cu energie a robotului. Alimentarea cu energie se va realiza cu acumulatoare, care după funcțiune pot fii împărțite în două grupe: un acumulator de 12 V, care asigură alimentarea sistemului cu circuite digitale. Elementele digitale sunt alimentate prin convertere DC-DC, cu tensiune reglabile.

A doua sursă de energie o să fie compusă din mai multe acumulatoare, care sunt conectate paralel și alimenteză H-podurile cu energie.

Pumpa de apă și motorele ventilatoarelor sunt conduse cu ajutorul unui tranzistor MOFSET cu canal N.

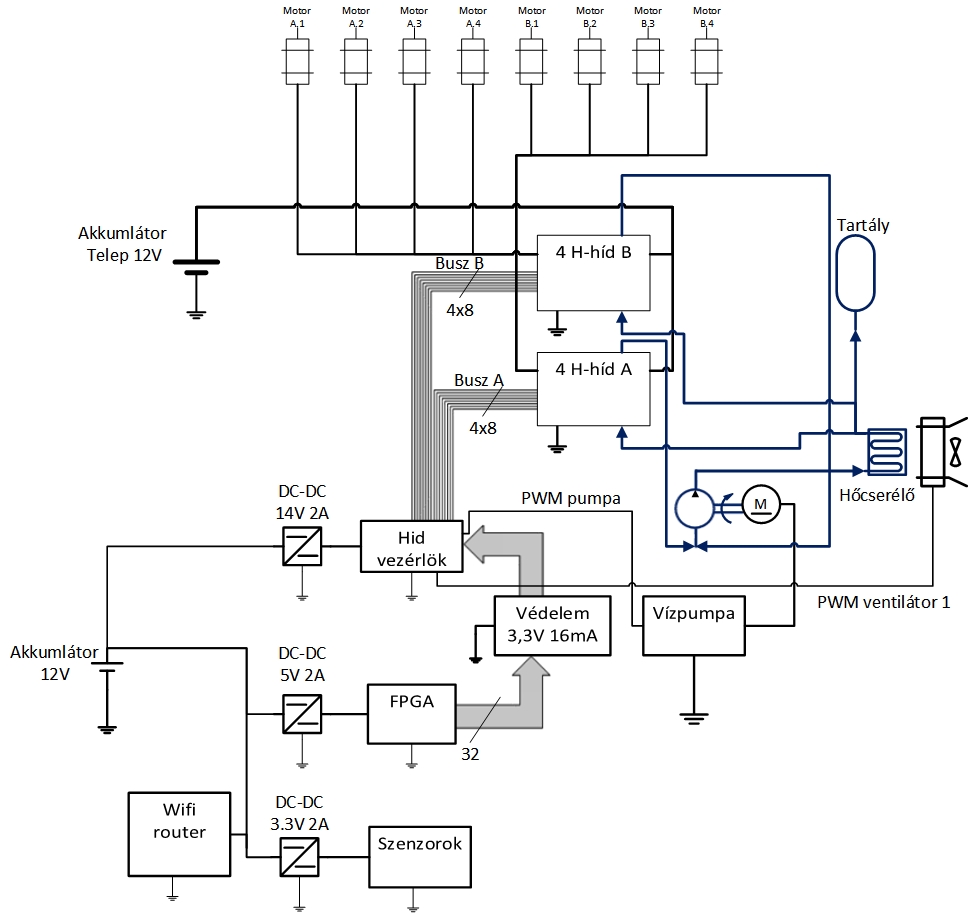


Fig. 1.5 Alimentarea cu energie electrică a robotului și structura sistemului de răcire

# STRUCTURA SISTEMULUI FPGA

Sistemul este structurat pe doua plăci de dezvoltare FPGA. Una este ZYBO la care puterea sursei este mare, dar are un număr de ieşiri limitate. Placa cealaltă conţine un chip Spartan3e, cu puţine resurse, dar are 120 de ieşiri.

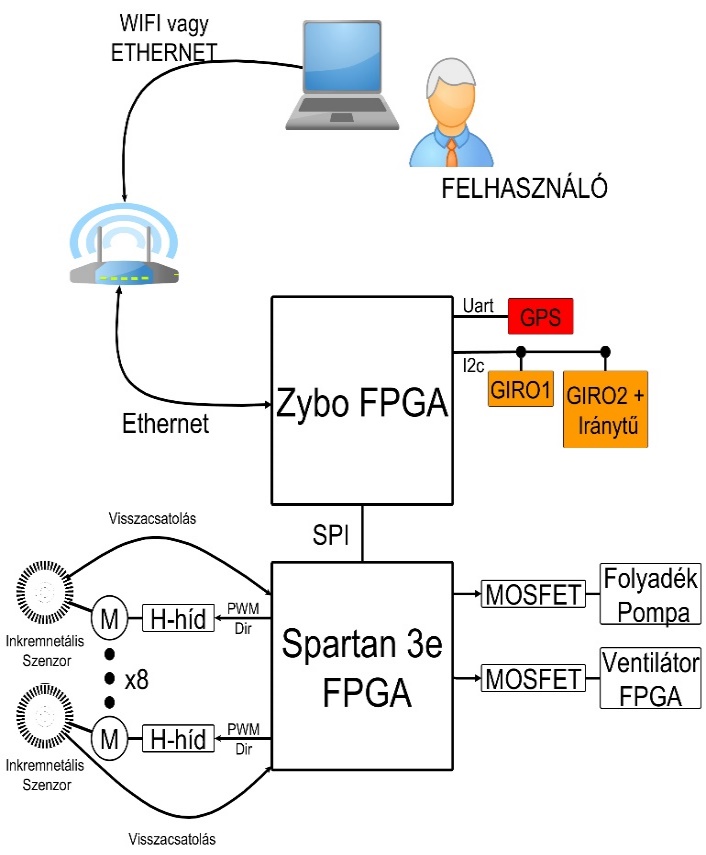


Fig. 2.1 Structura sistemului

Sistemul Spartan va controla cele 8 motoare, primeşte şi prelucrează datele de la sistemul ZYBO, apoi efectuează intervenţia corespunzătoare.

Sarcinile sistemului ZYBO sunt: colectarea datelor despre senzori complexe (giroscop, modulul GPS) şi comunicarea prin Ethernet.

Este necesar reglarea poziţiei a celor patru motoare a robotului realizat cu un hardver automat. De asemenea este necesar şi reglarea tip PID a vitezelor la patru motoare. . Am ales un sistem cu o buclă de control, iar feed-backul am realizat cu un emiţător incremental. Am folosit regulator de tip PID pentru reglarea vitezei și o automată hardware pentru reglarea poziției.

Regulatoarele au fost realizate și simulate în System Generator. Am făcut și măsurătoare pe partea de hardware, tot cu System Generator.

Regulatorele de PID-hardware și de poziție au fost așezat într-un nucleu IP. Parametrii regulatoarelor le putem definii prin registere distribuite. Conținutul registerelor sunt citite și scrise de către procesorul MicroBlaze. Acest procesor primește valorile regulatoarelor prin protocol de comunicare SPI și înscrie în registere. De asemenea, paralel trimite datele buclelor de control.

În sistemul ZYBO sunt două procesoare integrate: CORE 0, care are ca sarcină citirea senzorilor și operarea celor 3 servere de tip TPC, prin care putem prelua datele măsurate și să trimitem comenzi pentru sistem. Celălalt, CORE 1, o să aibă ca sarcină calculul modelului matematic.

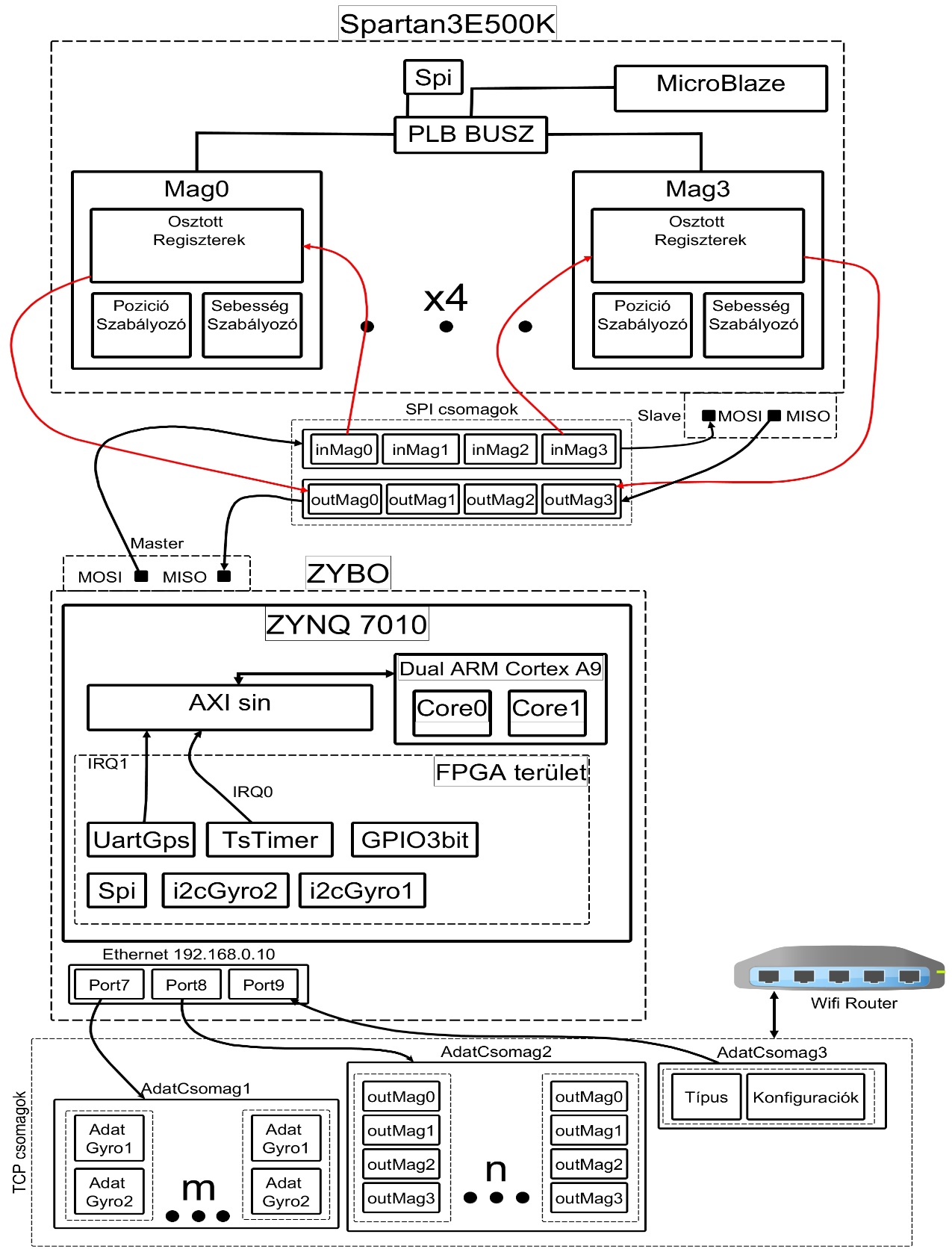


Fig. 2.2 Pachetele comunicaționale și modulele programate în circuitele electrice FPGA

# REGLAREA POZIȚIEI

Cu privire la structura sistemului mecanic, dacă se oprește motorul de antrenare și axul antrenat rămâne sub încărcare, axul antrenat nu poate să antreneze în sens invers din cauza frecărilor mecanice. Din această cauză este suficient, dacă în momentul potrivit vom opri motorul. La schimbarea polarității a motorului DC se schimbă și direcția de rotație a motorului, ajunge dacă intervenim în sistem cu regulator de semnal minim sau maxim.

Regulatorul executat este definit cu ajutorul următoarelor ecuații:





Se poate spune că semnalul regulatorului de ieșire depinde de valoarea erorii . În sistemul mecanic există un anumit joc între piesele componente, de aici rezultă un zgomot, pe care dorim să filtrăm. În astfel de situație, în care mecanismul se află într-o poziție adecvată, atunci într-un domeniu , vom pune insensibil regulatorul la intrare, până când eroarea nu iese din bandă sau nu se schimbă semnalul de referință.

Structura regulatorului:

Intrările: „RefVal”-poziție prescrisă măsurat în impuls, cu valoare 16 bit de semn, „AktVal”- măsurarea actuala a poziției, în impuls cu valoare 16 bit de semn; „U”- ieșire 17 bit de semn;

Modulul „*Calcul eroare*” efectuează calculul erorilor din poziţia actuală. Eroarea poate să fie chiar şi negativă, de aceea cu ajutorul comparatorului „*Eroarea este negativă?*”selectăm eroarea calculată şi decidem dacă este negativă sau pozitivă. După acea cu ajutorul lui „*ModuluszMux*”selectăm eroarea specificată, dacă este pozitiv, sau forma negată dacă este negativ, astfel apropiând la modulul.

Dacă eroarea ajunge la 0, atunci valoarea „Eroare=0 reg”este 1, până când poziţia prescrisă nu se schimbă.

Modulul " Eroare ieşit din bandă" analizează valoarea modulului eronat, şi dacă această valoare este mai mică decâ valoarea bandei atunci cu ajutorul „Manipulare eroare Mux” valoarea eroarei o să fie 0.

„Selector de eroare Mux” selectează eroarea manipulată sau calculată prin intermediul „Eroare=0 reg”, care merge mai departe în regulatorul cu 3 poziţii.

Cu ajutorul registerelor „Umin”, „Umax”, „U0” cu 17 bit de semn putem selecta valoarea maximă şi minimă a semnalului de pe regulator. „Eroare Pozitivă” şi „Eroare Negată”decid în care domeniu este eroarea. Putem distinge 3 domenii: negativ, pozitiv şi eroare 0. „Mux” selectează starea actuală a semnalului de control.

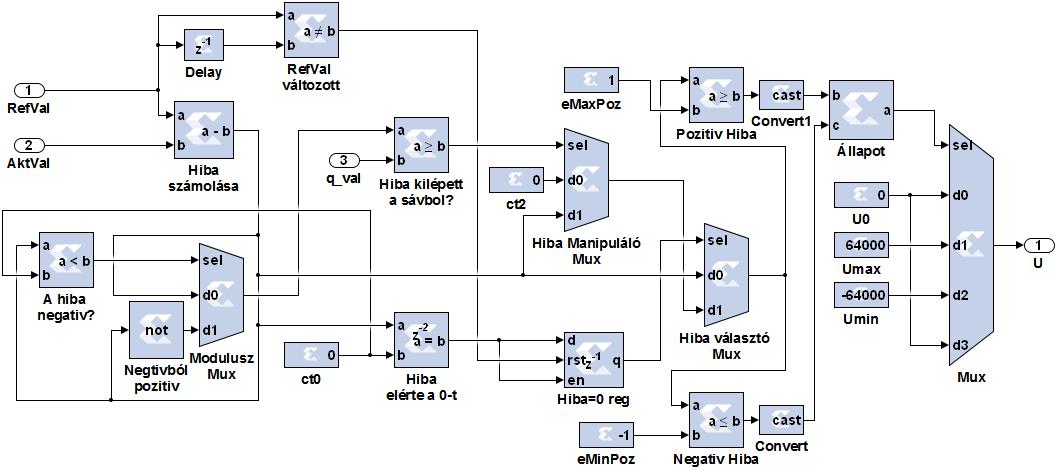


Fig. 3.1 Structura regulatorului de poziție în System generator

**Reglarea poziţiei motorului DC pe stand de măsurare:**

Standul de măsurare nu are la dispoziţie angrenaj de melc, din cauza asta regulatorul va oscila slab, dar acest lucru nu va fi semnificativ în prezenţa angrenajului.

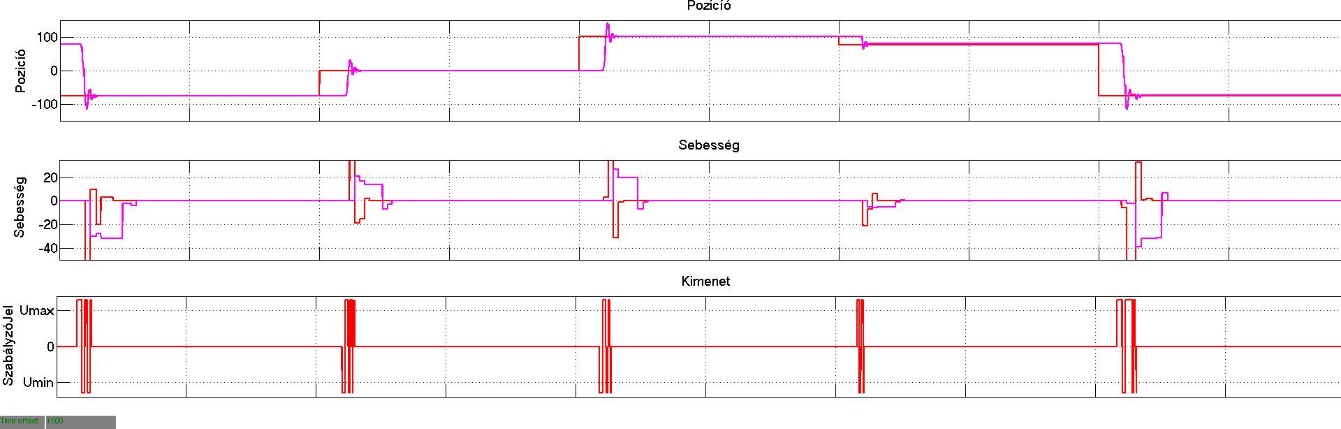


Fig. 3.2 Reglarea poziției din Fig 3.1 în structură vizibilă

# **REGULATOR PID-HARDWARE discret**

În prezent tipul de regulator PID este una dintre cele mai utilizate regulatoare, al cărei ecuaţie recursivă este următorul:



 [2]

Regulatorul PID realizat de mine este pus în aplicare cu ajutorul circuitului electric FPGA prin hardware.

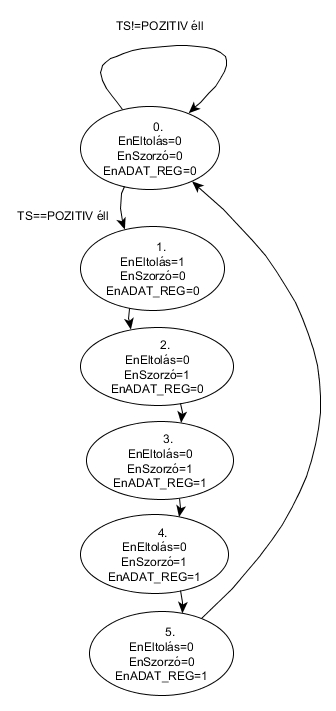


Fig. 4.1 Stare automată, care descrie regulatorul Diszkret PID

Pe baza datelor mai sus prezentate am proiectat o reţea de date automată, pe care am construit în System Generator.

Pe parametrii regulatorului PID putem defini cu ajutorul parametrilor Q, care depind de la parametrii cunoscuţi: - timp derivat, -timp integrat,  perioadă de eşantionare şi întărire proporţională.

Automata are 5 stări. La fiecare eşantion, automata rulează pe stări şi revine la starea de pornire. Operaţiile efectuate pe stări sunt executate pe placă dezvoltătoare FPGA pe frecvenţa semalului de ceas. Automata trece de la o stare la alta peste un semnal de ceas.

# REALIZARE ÎN SYSTEM GENERATOR

La alegerea reţelei de date aplicăm un numărător de 2 bit (ADAT UT) care, calculează conform semnalul ceasului, dacă la intrarea enable vom găsi valoarea logică 1. La 2 bit numărătorul poate să aibă maxim 4 valori şi din cauza asta vom regla numărătorul ca valoarea maximă să fie egală cu 2, aşa poate lua valori 0,1,2. Reţelele de date sunt selectate cu două multiplexe MUXQ şi MUXE.

Parametrii de intrare  sunt valori întregi de 16 bit de semnal.  are semnal de 16 bit , de tip -bool.

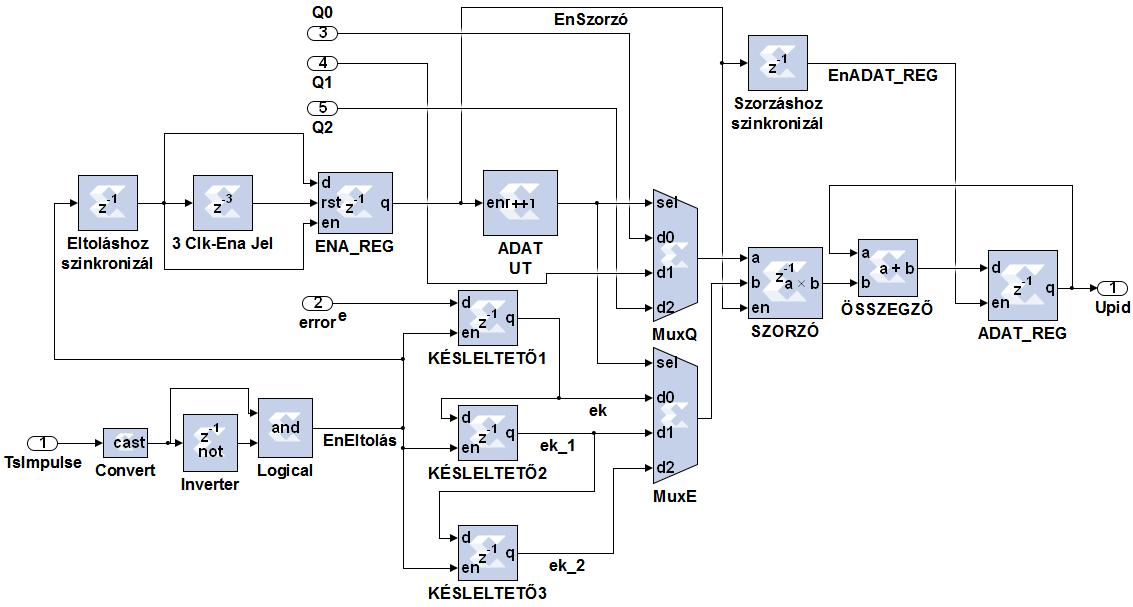
Ieşiri:  17 bit de semn, valoare întreagă. Pentru selectarea parametriilor Q es te responsabil MUXQ, iar MUXE este responsabil pentru selectarea valorile de intrare în timp întârziat. Pe Figura 3.3 modulul „Înmulţire”înmulţeşte valoarea primită de la multiplexer cu două selecţii, şi după aia adaugă la valoarea registrului „*ADAT\_REG*”.

Valoarea fiecărui modul saturează în caz că depăşeşte ori în sens negativ ori în pozitiv. Astfel putem evita depăşirea valorii modulului -în caz de componenta integrală cu intrarea erorii neschimbate-, care ar răsturna funcţionarea sistemului.

Pe figura 3.3 putem observa registere DELAY, care exploatează valorile erorilor  din trecut. Cele trei registere sunt legate între ele şi valoarea ajunge pe . Valoarea eroarei de eşantionare totdeauna ajunge în registerul DELAY1.

Numărătorul „Reţea de date” porneşte numai după împingerea registerelor, pe care realizează elementul întârzietor „*Eltoláshoz Szinkronizál”.* „ENA\_REG” şi „3 Clk-Ena Jel” relizează împreună semnalul logic până la 3 semnale de ceas, care porneşte numărătorul.

Fig. 5.1 StructuraPID în System Generator



Elementul detector este asigurat de către „*Inverter*”şi de un „AND” poartă (Logical), observăm valoarea semnalului din perioada anterioară. Prin compararea a celor două valori putem detecta scimbarea semnalului. Cu ajutorul modulului „*Sincronizează la înmulţire” putem autoriza intrarea „ ADAT\_REG*”. Pe modulul A „*Szorzó*” putem întârzia cu un semnal de ceas modul faţă de semnalul „*EnSzorzó*”.

# REGLAREA VITEZEI MOTORULUI DC PE STAND DE MĂSURARE

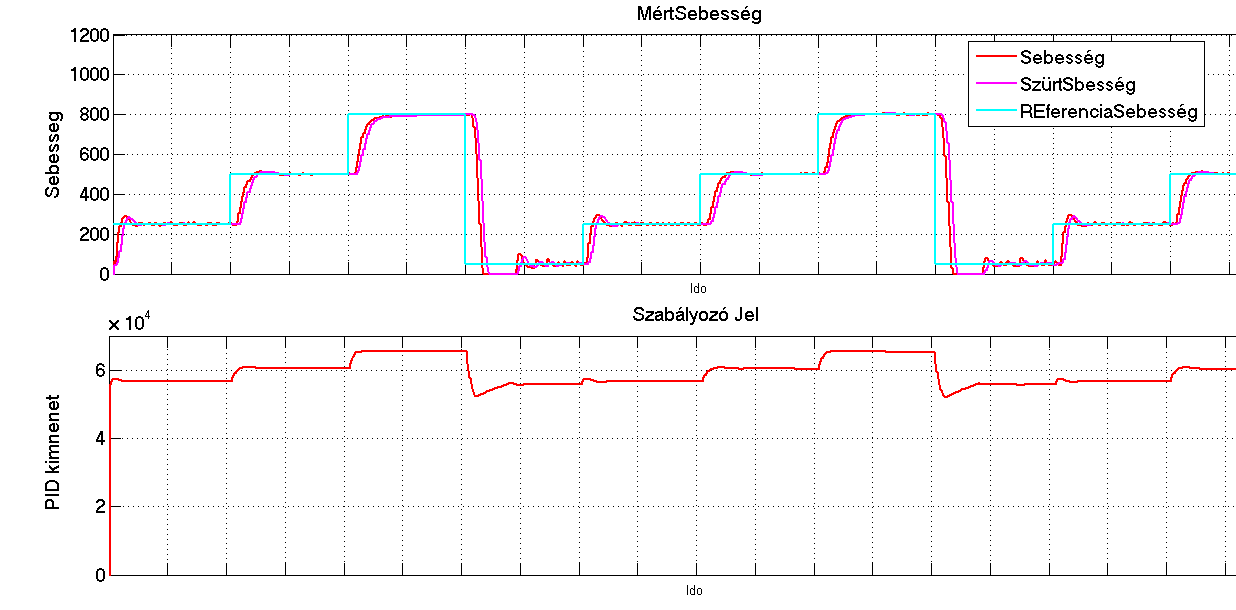


Fig. 6.1 Reglarea vitezei cu regulator

**Realizări**

* Sistem mecanic construit
* Sistem FPGA compus:
  + Comunicaţie SPI intre cele două FPGA
  + Realizarea elementelor hardware în System Generatorban
  + Sistem multi interminent szoftware şi hardware
  + Programe C pentru procesoare MicroBlaze şi ARM
* Regulator de motor 8 dc (PID, poziţie)
* Stand de măsurare construit pentru măsurarea regulatoarelor şi pentru testarea senzorilor
* Citirea datelor de senzor: gyroscop1 és gyroscop2, senzori de incrementare
* Comunicare Ethernet + client GUI (în Matlab)
* Construirea electronicii de putere: pod controleri, H-pod, răcire cu apă

Kivonat

A dolgozat célja mobilis tereprobot tervezése, valamint a megépítéséhez szükséges elemek tárgyalása. A mechanikai rendszer Autodesk Inventor-ban volt megtervezve, és az elkészített terv alapján kivitelezve.

**Vezérlő elektronika** a rendszeren megtalálható két FPGA fejlesztőlap: egy nagyobb erőforrásokkal rendelkező ZYBO Zynq™-7000 (beépített ARM processzorral), mely a matematikai számítások elvégzésére hivatott, valamint egy kisebb kapacitású FPGA lap (SPARTAN3e500), amely tartalmaz 8 hardveresen megvalósított szabályozót és egy MicroBlaze processzort. Ezek a szabályozók 12V DC motor sebességét vagy pozícióját koordinálják. A SPARTAN3e500 laphoz van illesztve 8 db. inkrementális érzékelő, amelyek a motorok pozícióját, illetve sebességét mérik.

A dolgozatban bemutatjuk a PID szabályozó, a pozíció szabályozó és az inkrementális érzékelő adatait feldolgozó modulok megvalósítását, System Generator környezetben, valamint a modulokkal végzett hardveres és szoftveres szimulációkat.

A ZYBO rendszerrel egy wifi routeren keresztül kommunikálhatunk TCP protokoll segítségével. Három TCP server fut a Zybo rendszeren:

* szenzorok adatait kérhetjük le (giroszkóp)
* motor vezérlők adatait olvashatjuk vissza (sebesség, pozíció, beavatkozó jelek stb.)
* paramétereket, illetve utasításokat adhatunk a rendszernek

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

***Kulcsszavak***: FPGA, PID, motor, inkrementális szenzor

Abstract

The purpose of this paper is the projecting of a terrain mobile robot and discussion about the necessary elements of the construction. The mechanical system was designed in Autodesk Inventor, and executed according to the plan. The next subtask was the planning of the electronic parts of the controller as well as the integration of the sensors in the system. A variety of sensors can be found on the system, of which the most important should be the incremental encoders, used to measure speed and position.

On the system two FPGA development boards can be found: a Zybo, with larger resources (integrated ARM processor) it is used to perform mathematical calculations and a small-capacity FPGA (SPARTAN3e500), with 8 hardware-implemented controller and with a MicroBlaze processor. These controllers control the 12V DC motor speed or the position of it. The MicroBlaze processor is responsible for receiving data and after a simple processing for writing to the appropriate register of the hardware-based controller. To the SPARTAN3e500 board 8 pieces of incremental sensors are mounted which measure the position and speed of the motors. The sensor’s data is sent by the Spartan board via a fast SPI communication to the Zybo board. The fast hardware, the PID controller was implemented on an FPGA development board, with the Xilinx System Generator design tool, the simulation was carried out with hardware co-simulation.

We can communicate with the ZYBO system using a Wi-Fi router via TCP protocol. Three TCP server is running on the Zybo system:

* We can request data from sensors (gyroscopes).
* Motor Controllers data can be read back (speed, position, intervening signal...).
* Parameters and instructions may be added to the system.

The planning started with the mechanical system. With the help of Autodesk Inventor, several variants were designed until arriving to this mechanical structure of the discussed paper. The mechanical system was made based on the plans, and tests were made, in which the gear ratios were carried out. The results showed that the modification of the system for fixing the motors is needed. The designing and rebuilding of the mechanical system took two months.

To the software and digital hardware development the FPGA system was chosen, because software and hardware are easy to develop together. The sensors were chosen so as to be easily fitted to the FPGA system, all sensors are working on 3.3V voltage level.

The incremental sensors signal’s processing module is realized in System Generator. Once the position and speed could be measured, the controllers for the system operation have been designed, at first the PID control. The PID control was used for speed and position controlling, but the results showed that the PID is not effective to perform the position controlling.

Due to the backlash in the referrals system, the position controlling hasn’t been correct, so a different regulatory concept was developed, which has proved to be viable.

In the dissertation we present the implementation of the PID controller, position controller and the data processing modules of the incremental sensor in System Generator environment, as well as the simulations with hardware and software modules.

More massive complementary accessories such as robot arm, lawn mowers can be fixed to the robot chassis.

Applicability is possible even in agriculture, for example a weeding machine, or even in safety engineering, as an actuating device.

***Keywords:*** FPGA, PID,motor, incremental sensor

[Ábrák, táblázatok jegyzéke 10](#_Toc422599283)

[1 Bevezető 12](#_Toc422599284)

[2 BIBLIOGRÁFIAI TANULMÁNY 14](#_Toc422599285)

[2.1 Hasonló FPGA fejlesztőrendszeren megvalósított PID szabályzók 14](#_Toc422599286)

[2.2 Inkrementális érzékelő 14](#_Toc422599287)

[2.3 Szögsebesség mérése FPGA segítségével 15](#_Toc422599288)

[2.4 Egyenáramú motorok 16](#_Toc422599289)

[2.5 PID szabályozó hangolása Ziegler-Nichols módszerrel 17](#_Toc422599290)

[3 RENDSZER TERVEZÉSE 19](#_Toc422599291)

[3.1 SZABÁLYOZÓK: 19](#_Toc422599292)

[3.1.1 Diszkrét Hardveres PID szabályozó 19](#_Toc422599293)

[3.1.1.1 Megvalósítás System Generátorban 20](#_Toc422599294)

[3.1.1.2 Simulink szimulációs eredmények 22](#_Toc422599295)

[Táblázat. 3‑1 Manuálisan számolt értékek a szimuláció ellenőrzésére 23](#_Toc422599296)

[3.1.1.3 Q paraméterek számolása alapján. 23](#_Toc422599297)

[3.1.2 Mintavételezési periódus jelének generálása 24](#_Toc422599298)

[3.1.3 Pozíció Szabályozása 25](#_Toc422599299)

[3.1.3.1 A szabályozó felépítése: 26](#_Toc422599300)

[3.1.3.2 Szabályozó szimulálása 27](#_Toc422599301)

[3.1.4 Hardveres mérések 28](#_Toc422599302)

[3.1.4.1 DC motor sebesség szabályozása mérőstandon 28](#_Toc422599303)

[3.1.4.2 DC motor pozíció szabályzása mérőstandon 29](#_Toc422599304)

[3.2 Sebesség és pozíció szabályozót tartalmazó IP mag generálása System Generátor-bAn 30](#_Toc422599305)

[3.3 Szenzorok 36](#_Toc422599306)

[3.3.1 Inkrementális Érzékelő 36](#_Toc422599307)

[3.3.1.1 Optikai inkrementális vevő felépítése 36](#_Toc422599308)

[3.3.2 Inkrementális érzékelő jeleinek a feldolgozása FPGA áramkör segítségével 38](#_Toc422599309)

[3.3.2.1 Szimuláció System Generátorban 39](#_Toc422599310)

[3.3.2.2 Pozíció mérése Inkrementális adó segítségével 40](#_Toc422599311)

[3.3.2.3 Szögsebesség mérése Inkrementális adó segítségével 41](#_Toc422599312)

[3.4 MPU-6050 giroszkóp és gyorsulásmérő 44](#_Toc422599313)

[3.5 Beavatkozó elemek: 46](#_Toc422599314)

[3.5.1 Pwm Generátor megvalósítása FPGA áramkörön System Generátor környezetben. 46](#_Toc422599315)

[3.5.1.1 Megvalósítás 46](#_Toc422599316)

[3.6 Elektronika 48](#_Toc422599317)

[3.6.1 Digitális Elektronika 48](#_Toc422599318)

[3.6.2 FPGA Rendszer Felépítése 49](#_Toc422599319)

[3.6.2.1 Zybo FPGA fejlesztőlap 51](#_Toc422599320)

[3.6.2.2 Spartan3e FPGA fejlesztőlap 52](#_Toc422599321)

[3.6.2.3 Kommunikációs protokollok 53](#_Toc422599322)

[3.6.3 Feladatok Elosztása 56](#_Toc422599323)

[3.6.3.1 Zybo fejlesztőlap 56](#_Toc422599324)

[3.6.3.2 Spartan fejlesztőlap 56](#_Toc422599325)

[3.7 Teljesítmény elektronika 57](#_Toc422599326)

[3.7.1 Bootstramp működése 64](#_Toc422599327)

[3.7.1.1 Szimuláció simulink környezetben 66](#_Toc422599328)

[3.8 Robot Modell 68](#_Toc422599329)

[4 Robot Mechanikai Felépítése 71](#_Toc422599330)

[5 Elért eredmények, magvalósítások: 74](#_Toc422599331)

[6 Következtetések: 74](#_Toc422599333)

[7 Bibliográfia 76](#_Toc422599334)

[8 FÜGGELÉK 77](#_Toc422599335)

Ábrák, táblázatok jegyzéke

[Kép. 2.1 Inkrementális érzékelő jelek 15](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422454)

[Kép. 2.2 A rendszer egységugrásra adott válasza és megközelítése egyenesekkel. 18](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422455)

[Kép. 3.1 A Pozíció és a sebesség szabályzási hurok elvi strukturális felépítése 19](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422456)

[Kép. 3.2 Állapot automata, amely leírja a Diszkrét PID szabályózót 20](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422457)

[Kép. 3.3 A PID felépítése System Generatorban 21](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422458)

[Kép. 3.4 PID Simulink szimulációs model 21](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422459)

[Kép. 3.5 Szimulációs eredmény amely tükrözi a konstans bementre a számolási lépéseket 22](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422460)

[Kép. 3.6 PID minimális periódusa 23](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422461)

[Kép. 3.7 Mintavételi taktust generáló modul 23](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422462)

[Kép. 3.8 Szimulációs eredmények, mintavételi jelgenerátor. 24](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422463)

[Kép. 3.9 A pozíció szabályozó System generátoros felépítése 26](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422464)

[Kép. 3.10 A pozíció szabályozás moduláris felépítése System Genrator környezetben 26](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422465)

[Kép. 3.11 Sebesség szabályozás PID szabályzóval. 28](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422466)

[Kép. 3.12 Pozíció szabályozása a Kép. 3.9 látható kialakításban. 28](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422467)

[Kép. 3.13 Sebesség és pozíció szabályózást tartalazó Ipmag System generátoros felépítése 31](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422468)

[Kép. 3.14 Pozíció Szabályozó modul belső felépítése 32](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422469)

[Kép. 3.15 Sebesség Szabályozó modul felépítése 33](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422470)

[Kép. 3.16 Optikai inkrementális vevő felépítése és elhelyezése 34](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422471)

[Kép. 3.17 Érzékelő tranzisztorok elhelyezése 34](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422472)

[Kép. 3.18 Idődiagram a Tárcsa paramétereinek függvényében 35](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422473)

[Kép. 3.19 Rések és az Érzékelők közti kapcsolat 35](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422474)

[Kép. 3.20 Inkrementális jelfeldolgozó modul1 érzékelő modul belső felépítése 36](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422475)

[Kép. 3.21 Inkrementális érzékelőtől érkező jelek átalakító iránya és impulzusa jelekre 36](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422476)

[Kép. 3.22 Szimulációs eredmények a lehetséges bemenetekről az Black Box1 modulba 37](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422477)

[Kép. 3.23 Inkrementális adóval mért pozíció, szimulációs modellje SytemGeneratorban 38](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422478)

[Kép. 3.24 Sebesség mérő modul felépítése 38](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422479)

[Kép. 3.25Dc motor sebességének mérése FPGA lapon 39](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422480)

[Kép. 3.26Dc motor Sebesség mérése FPGA rendszeren, System generatorban megvalósítva 39](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422481)

[Kép. 3.29.a8ms 40](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422482)

[Kép. 3.29.c4ms 40](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422483)

[Kép. 3.29.b80ms 40](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422484)

[Kép. 3.30 Giroszkóp mért adatainak az ábrázolása a GUI program segítségével 42](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422485)

[Kép. 3.31 A PWM generátor System Generátorban megvalósított szerkezete 43](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422486)

[Kép. 3.32 A PWM generátor bemenő, kimenő illetve néhány belső jele (Scope1) 44](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422487)

[Kép. 3.33 A rendszer elvi felépítése 44](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422488)

[Kép. 3.34 Kommunikációs csomagok és az FPGA áramkörökbe programozott modulok elvi felépítése 46](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422489)

[Kép. 3.35 ZYBO Core0 program folyamat árbája 47](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422490)

[Kép. 3.36 Spartan3e500, microblaze szoftver Folyamat ábrája 47](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422491)

Kép. 3.37 Hip4082 alkalmazása H híd kapcsolásban. Forrás: http://www.intersil.com/en/products/space-and-harsh-environment/harsh-environment/half--full-bridge-and-three-phase-drivers/HIP4082.html 52

[Kép. 3.38 Két hídvezérlő áramkör kapcsolási rajza HIP4082 integrált áramkörrel megvalósítva 53](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422493)

[Kép. 3.39 PWM és a tranzisztorok kapcsolása 54](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422494)

[Kép. 3.40 Nem invertáló erősítő forrás [14] 54](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422495)

[Kép. 3.41 Dupla hídvezérlő áramkör vezérlő jelei JP2 csatlakozó a Kép. 3.39-n. 55](#_Toc422422496)

[Kép. 3.42 H híd tranzisztorainak a Gate vezetékei 55](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422497)

[Kép. 3.43 FPGA kimentének a védelme 56](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422498)

[Kép. 3.44 A négy Kép. 3.45 látható szalagvezeték jelenik meg a Buszvezetékben. 56](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422499)

[Kép. 3.45 A robot energia ellátása valamint a hűtő rendszer elvi felépítése 57](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422500)

[Kép. 3.46 Vízpumpa és a ventilátor motorjának vezérlő teljesítmény elektronikai kapcsolása 58](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422501)

[Kép. 3.47 Bootstramp megoldás a felső tranzisztor Gate bemenetének a meghajtására 58](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422502)

[Kép. 3.48 Bootstramp kondenzátor feszültsége a W és W11 pontokban 59](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422503)

[Kép. 3.49 Bootstramp működése, szimulációs modell MATLAB/SIMULINK környezetben 60](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422504)

[Kép. 3.50 Szimulációs eredmények Bootstramp 60](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422505)

[Kép. 3.51 Robot kerekek sebessége és a robot mozgásának viszonya 61](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422506)

[Kép. 3.52 Robot 3D vektorábrája 62](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422507)

[Kép. 3.53 Oldalnézetek és Felülnéztek, jelölések szemléltetése 63](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422508)

[Kép. 4.1 Robot vázának Inventoros 3D Képe 64](file:///C:\Users\laca\Desktop\Allamvizsga\Dolgozat\Allamvizsgadolgozat\GaborSzabolcsLaszlo_v4.docx#_Toc422422509)

[Kép. 8.1 A mechanikai rendszer műszaki rajza 70](#_Toc422422510)

[Táblázat. 3‑1 Manuálisan számolt értékek a szimuláció ellenőrzésére 15](#_Toc422126992)

# **Bevezető**

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

A tervezést a mechanikai rendszerrel kezdtem. AutodeskInventor segítségével több változatot is megterveztem ameddig eljutottam a dolgozatban tárgyalt mechanikai struktúrához. A mechanikai rendszert, saját magam viteleztem ki a tervek alapján. A kivitelezés után tesztet végeztem, amely során a fogaskerék áttételeket teszteltem. A szoftver és digitális hardver fejlesztésére FPGA rendszert választottam, mert könnyen fejleszthető a szoftver és a hardver közösen. A szenzorokat úgy választottam, meg hogy könnyen illeszthető legyen az FPGA rendszerhez, minden szenzor 3,3V feszültségszinten dolgozik.

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

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

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

# **BIBLIOGRÁFIAI TANULMÁNY**

## Hasonló FPGA fejlesztőrendszeren megvalósított PID szabályzók

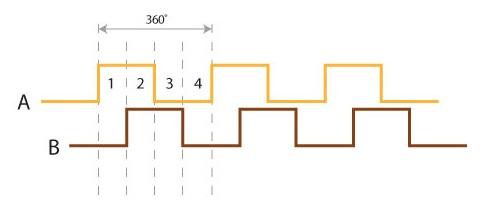
A [1]cikkben tárgyalt FPGA erőforráson kivitelezett PID szabályozó, amelyet a nagyobb működési sebesség kedvéért FPGA alapon valósított meg. A szabályozó paraméterei fordításkor vannak meghatározva, a hardverben kívülről nem lehet megadni, ami a hangolás szemszögéből nem előnyös. A [2] PID szabályozó folytonos átviteli függvényéből indul ki, és levezeti a diszkrét átviteli függvényt, amelyből majd a rekurzív mintavételes szabályozót kapja meg.

[2]

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

## Inkrementális érzékelő

Az inkrementális érzékelőknek két kimenete van. Jelölésük általában A és B, a két jel időbeni viszonya alapján tudjuk megállapítani a forgás irányát. A jelek generálódása egy dióda és egy optikai tranzisztor segítségével történik. Közben egy tárcsa, amelyen ablakok találhatók, mozgáskor elhalad a dióda és a tranzisztor között. A két jel időben 90 fokos késésben van egymáshoz viszonyítva. Az érzékelők alkalmasak szögsebesség, szögelfordulás mérésére.



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

## Szögsebesség mérése FPGA segítségével

A sebesség szabályozásához mérni kell a sebességet, [6] dolgozat két sebességmérő módszert említ meg, amelyeket ötvözve használ.

Az első az inkrementális adó segítségével mért időalapú sebességmérés, amely abból áll, hogy méri a két impulzus között eltelt időt egy számláló segítségével, amely az FPGA órajelére számol. A sebességet a következő összefüggéssel határozza meg:

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

A második módszer megszámolja, az inkrementális adótól érkező éleket, idő alatt. Ahol a másodpercben, kifejezett idő.

E módszer előnye, hogy minél nagyobb a fordulat annál kisebb a hiba.

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

## Egyenáramú motorok

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

Az elektromos egyenlet mellé még felírja a mechanikai egyenleteket is:

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

A motor dinamikus moteljéhez egyesíti a két egyenletet:

Ahol az a motor szög pizicíója.

A motor állapotteres motelje, a választott állapotok:

Ahol - állapotok, és a - bemenetek.

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

## PID szabályozó hangolása Ziegler-Nichols módszerrel

A [8] irodalom a PID szabályozóra kidolgozott hangolási módszert írja le. A módszer csak olyan folyamatoknál alkalmazható, ahol a rendszerre nézve nem jelent kockázatot, ha a stabilitásának a határára visszük. A módszer, első lépésben kiiktatjuk a szabályozóból az integráló és deriváló tagokat, így marad csak egy erősítő tag. A folyamat az állandósult állapotban szinuszos lengést fog mutatni az alapjel körül. A lengések periódusát () megmérve és ismerve a *Kp* erősítést, amelyen a lengések jelentkeznek, kiválaszthatjuk a megfelelő PID paramétereket egy táblázat alapján. Mintavételezett megvalósítás esetén, a mintavételezési periódust körüli értékre kell tenni.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Kp |  | Ti | Td |
| P |  |  | - | - |
| PI |  |  | 0.85 |  |
| PID |  |  |  |  |

Táblázat. 2‑1 Ziegler-Nichols módszerrel történő PID hangolás

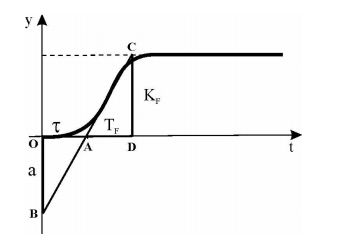
**Oppelt módszer**

A rendszer egységugrásra adott válaszából következtet a szabályozó paramétereire. A módszer feltételezi, hogy az irányított folyamat elsőfokú holtidős, és stabil.

A rendszert három paraméterrel lehet jellemezni:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Kp | Ti | Td |
| P |  | - | - |
| Pi | 0.8/a | 3 | - |
| PID | 1.2/a | 2 | 0.24 |

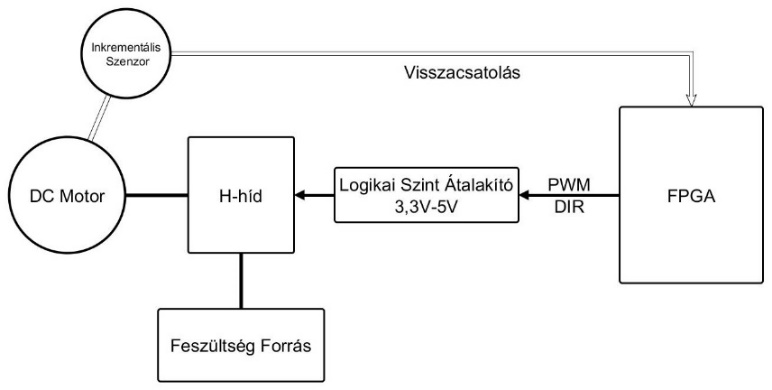
Táblázat. 2‑2 Oppelt módszer hangolás

 Mintavételes megvalósításnál a rendszer mintavételezési periódusát a értékre kell megválasztani. A módszer előnye, hogy nem kell a rendszert a stabilitás határára vinni, így biztonságosabb, valamint könnyű a bemenet előállítása.

# RENDSZER TERVEZÉSE

## **SZABÁLYOZÓK:**

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



Kép. 3.1 A pozíció és a sebesség szabályozási hurok elvi strukturális felépítése

### **Diszkrét Hardveres PID szabályozó**

Napjainkban az egyik leghasználtabb szabályozótípus a PID, amelynek rekurzív egyenlete a következő:

(1)

(2) [2]

Az általam elkészített PID szabályozó hardveresen van megvalósítva FPGA áramkörben, a minél kisebb mintavételezési periódus elérése céljából. A fent látható összefüggése alapján egy adat utas automatát terveztem, amelyet majd System Generátorban építtettem meg. A PID szabályozó paramétereit, a Q paraméterek segítségével adhatjuk meg, amelyek függenek az ismert paraméterektől: - deriválási idő, -integrálási idő, mintavételezési periódus, valamintproporcionális erősítés.

Az automata öt állapotot tartalmaz. Minden mintavételre, az automata végigpörög az állapotokon és majd visszatér a kiinduló állapotba. Az állapotokban végzet műveletet az FPGA fejlesztő lap órajelének a frekvenciáján hajtjuk végre. Az automata minden állapoton egy órajel periódus alatt lép át.

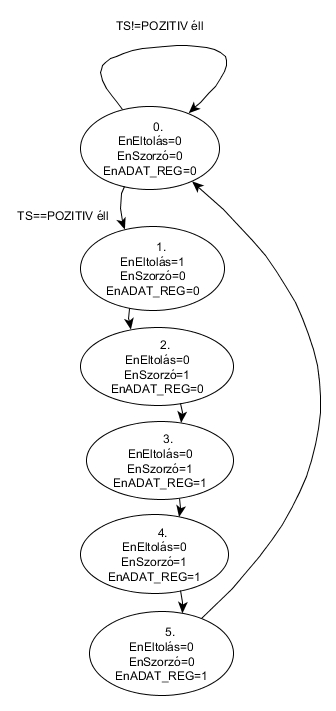
Minden állapotban egy (ÖSSZEGZŐ) regiszterhez adjuk hozzá a műveletek eredményét és így valósul meg a fenti rekurzív összefüggés.

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

Az 2,3,4 állapotokban matematikai műveleteket végez, azáltal hogy En Szorzó 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.



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

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

#### Megvalósítás System Generátorban

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

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

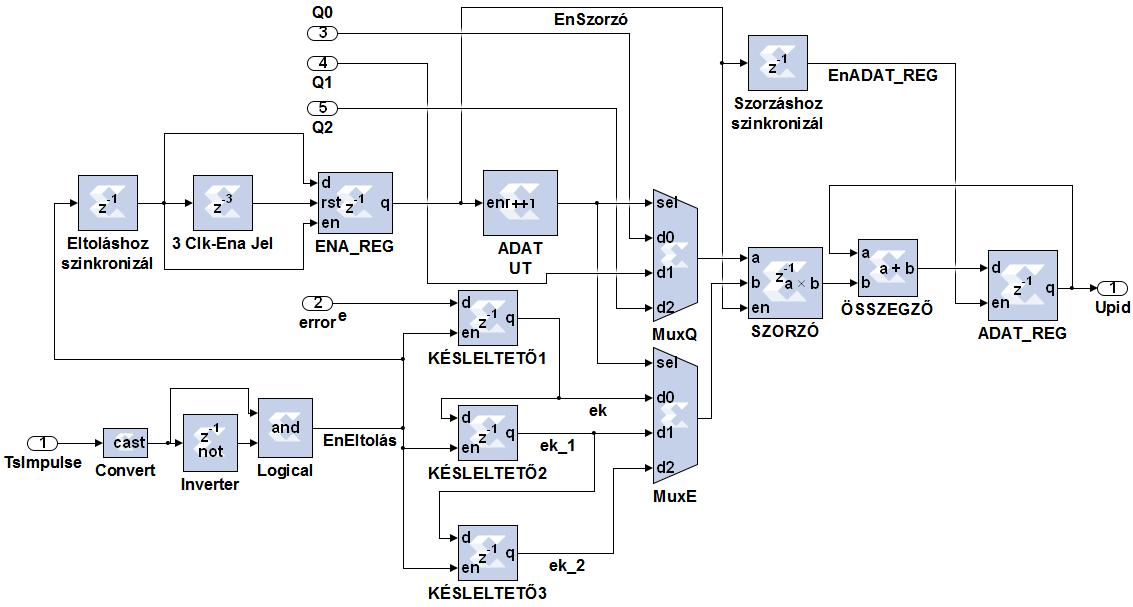
Kimenetek: 17bit előjeles egész érték. A MUXQ, a Q paraméterek kiválasztásáért valamint a MUXE az időben késleltetett bemeneti értékek kiválasztásáért felelős. A Kép. 3.3 a „*SZORZÓ*” modul a két szelekciós multiplexertől kapott értéket összeszorozza, aztán hozzáadja az „*ADAT\_REG*” regiszter értékéhez.

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

A Kép. 3.3 látható KÉSLELTETŐ regiszterek állítják elő múltbeli hiba 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 UT”* számláló csak a regiszterek elcsúsztatása után indul el, amelyet „*Eltoláshoz Szinkronizál*” késleltető elem valósít meg. Az „ENA\_REG” és a „3 Clk-Ena Jel” egyűt valósítják meg a három órajelig tartó logikai engedélyező jelet, amely a számlálót indítja el.

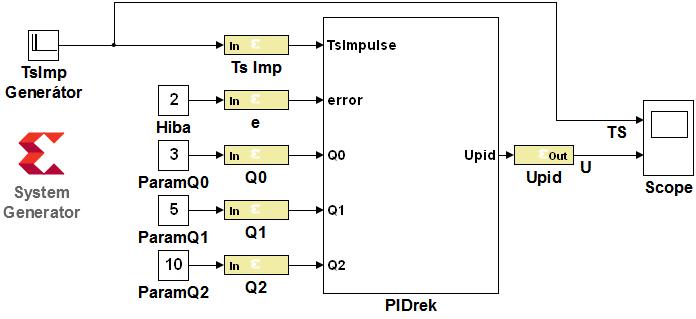
Kép. 3.3 A PID felépítése System Generátorban



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

#### Simulink szimulációs eredmények

A „PIDrek” modul tartalmazza a Kép. 3.3 képen látható modult, a bemenetekre és a kimentre illesztünk egy-egy konvertáló elemet, mely segítségével adatokat közölhetünk, vagy nyerhetünk a megtervezett XilinxSytem Generátoros hardverrel, amely az FPGA-n fut, és az adatokat Matlaban tudjuk kezelni.

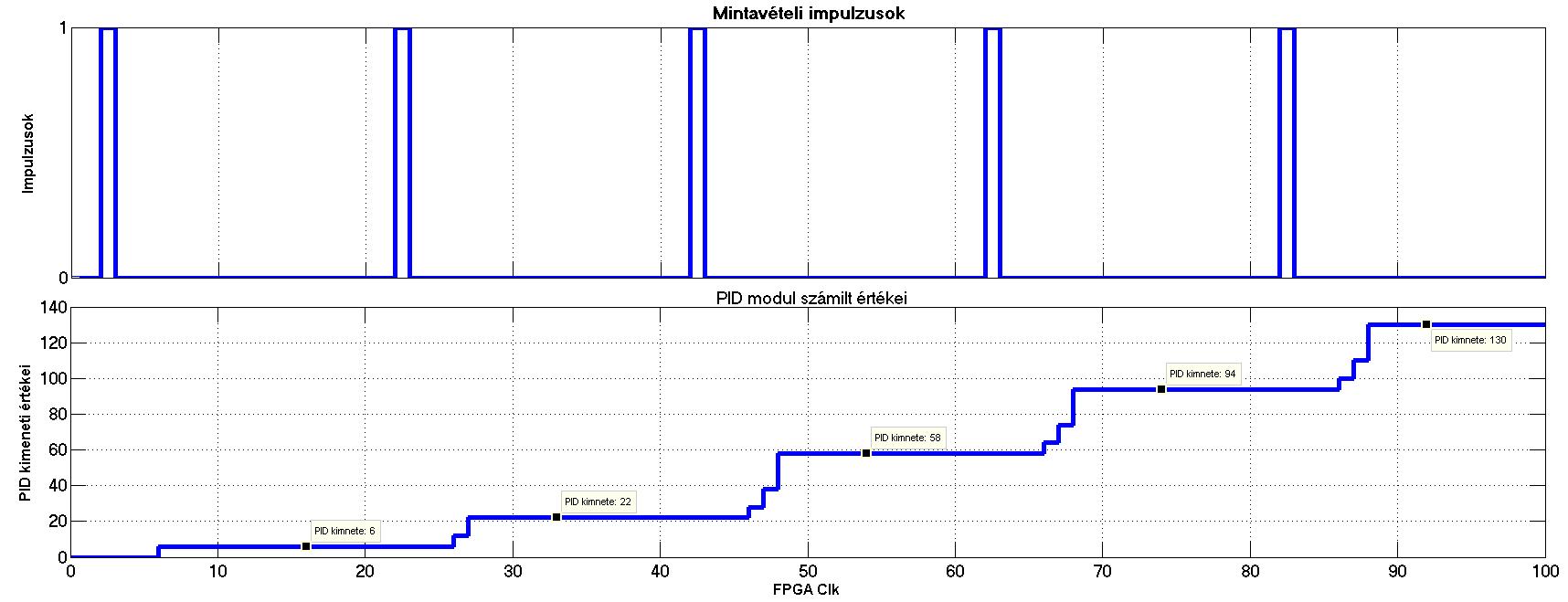


Kép. 3.4 PID Simulink szimulációs modell

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

Bemeneti paraméterek:, a bemenet konstans:

A Kép. 3.6 látható a 6 szükséges órajel a számítások elvégzésére. A szimulációk igazolják, ezt a 6 órajelet az elindítástól a végső eredmény megjelenéséig. Megjegyezném, hogy a szabályozó negatív bemeneti értékekre is működőképes.



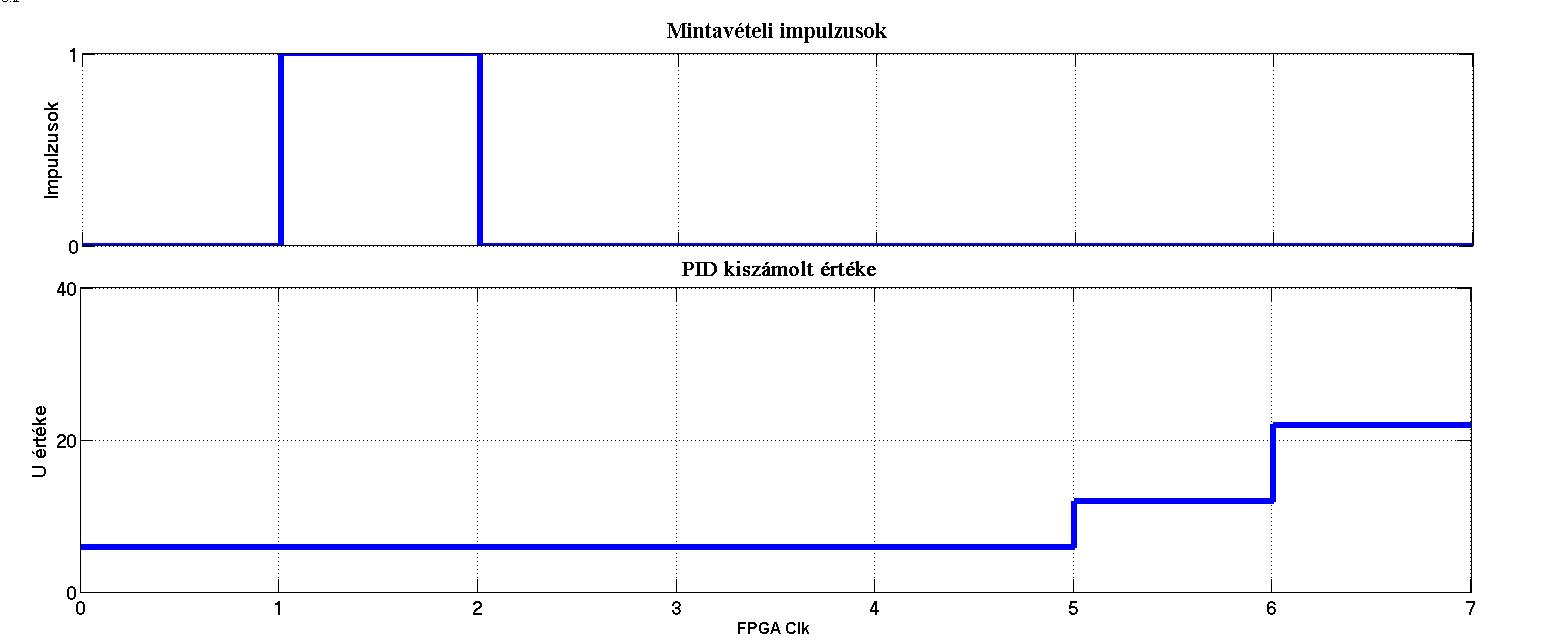
Kép. 3.5 Szimulációs eredmény, amely tükrözi a konstans bementre a számolási lépéseket

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

Táblázat. 3‑1 Manuálisan számolt értékek a szimuláció ellenőrzésére

A Szabályozó az elvártaknak megfelelő eredményeket térít vissza. 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 . Következésképpen a PID szabályozó maximális mintavételezési periódusa 120ns.

Kép. 3.6 PID minimális periódusa



#### Q paraméterek számolása alapján.

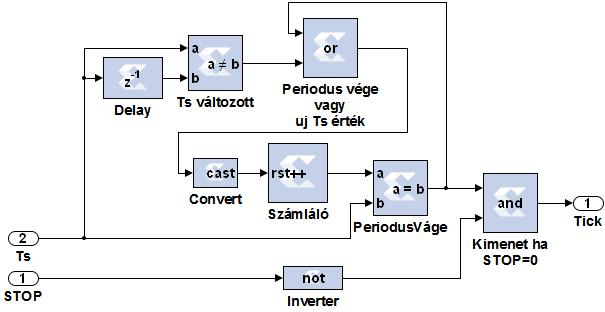
Adottak az összefüggések, látható hogy megjelennek a nevezőben, így fennáll annak a veszélye, hogy 0-val való osztás történik. Ezért soha ne válasszuk a –t 0-nak, de lehet egy nullához közeli pozitív szám. A mindig nagyobb, mint nulla.

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

### Mintavételezési periódus jelének generálása

Az Kép. 3.7 Mintavételezési taktust generáló modul belső felépítése.

Kép. 3.7 Mintavételi taktust generáló modul



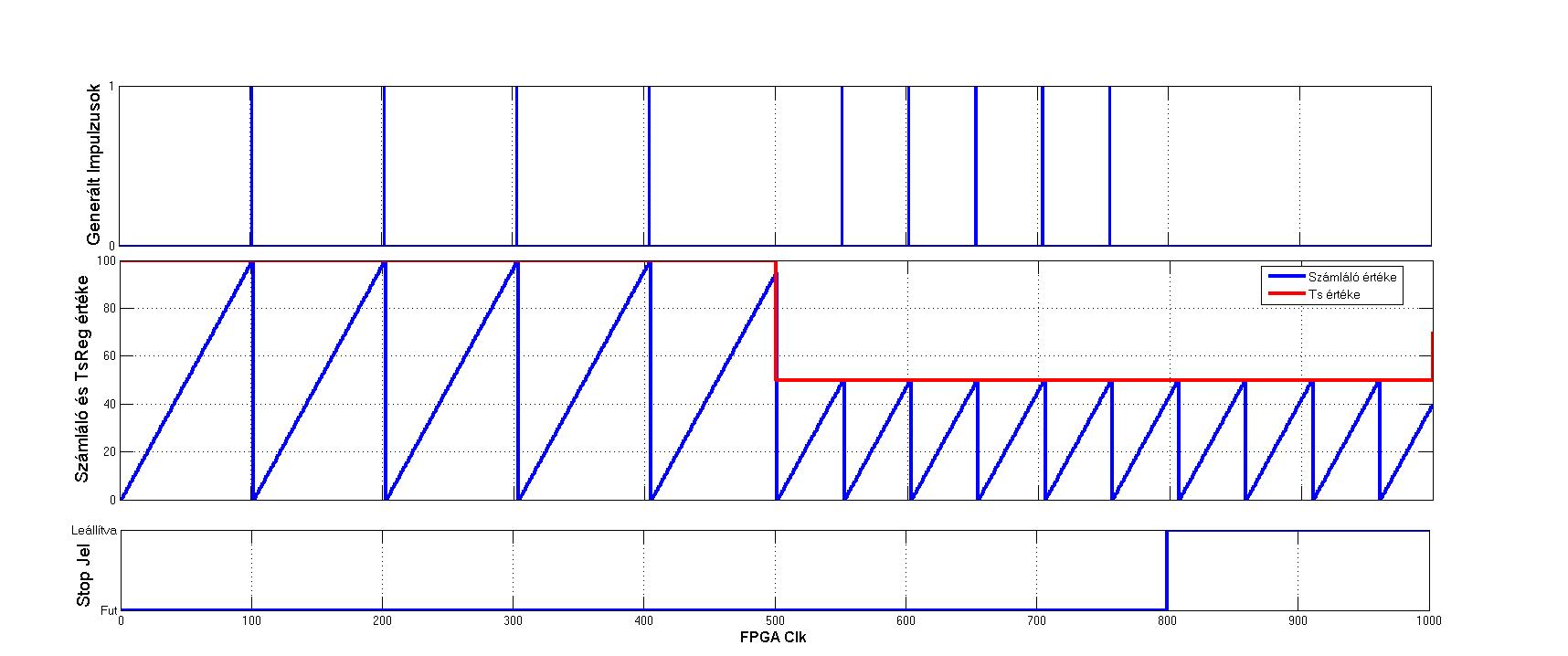
A modulban a „*Számláló*” 32 bites számláló az FPGA órajelére számol, és az értékét összehasonlítjuk a „*Ts*” bemenet értékével. Ha az érték megegyezik, akkor generálódik egy impulzus, amely lenullázza a számlálót. A számláló akkor is nullázódik, ha megváltozott a „*Ts*” értéke, amelyet a „*Delay*” késleltető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

Az Kép. 3.8 látható a pirossal jelölt „*Ts*” két különböző értékére hogyan történik az impulzusok generálása. A „*Tick*” kimenten az impulzusok 1 órajel periódusig tartanak, vagyis 20ns-ig 50MHz órajelen.

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

Impulzusok frekvenciája:

Ahol a Ts a frekvenciát beállító regiszter, (0, 4,294,967,295) közötti egész értékeket vehet fel. A generátor kimeneti impulzusainak a frekvenciája () 25MHz (FPGA órajele osztva 2- vel) től 0.01Hz –ig lehet változtatni.



Kép. 3.8 Szimulációs eredmények mintavételi jelgenerátor.

A szimulációkat:

, végeztem el, és a következő eredményeket kaptam frekvenciának.

### Pozíció Szabályozása

A mechanikai rendszer kialakításából adódóan, ha a hajtómotor leáll és a hajtott tengely terhelés alatt marad, a meghajtott tengely a súrlódások miatt nem tud visszafele hajtani. Ezért elegendő, ha a megfelelő időpillanatban a hajtómotort leállítjuk. 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 nem lép ki a sávból vagy a referencia jel meg nem változik.

#### A szabályozó felépítése:

A bemenetek: „RefVal” –előírt pozíció impulzusban mérve, 16 bites előjeles érték; „AktVal” – aktuálisan mért pozíció impulzusban mérve, 16 bites előjeles; „U” –kimenetel 17 bites előjeles;

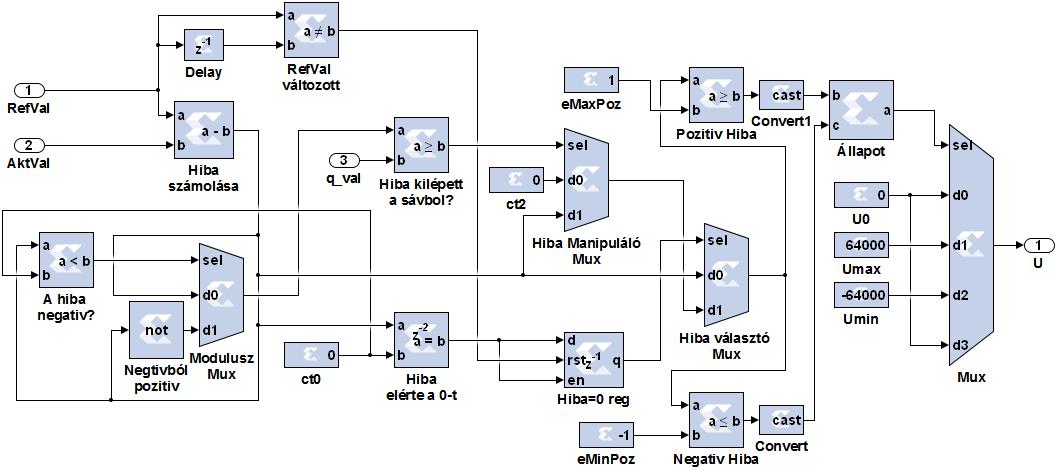
A „*Hiba Számolása*” modul végzi a hiba kiszámolását az aktuális és az előirt pozícióból. A hiba lehet negatív is ezért „*A hiba negatív?”* Komparátor segítségével eldöntjük, hogy negatív vagy pozitív a hiba. Azután a „*ModuluszMux*” segítségével kiválasztjuk a magát a számolt hibát, ha az pozitív, vagy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

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

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

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

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



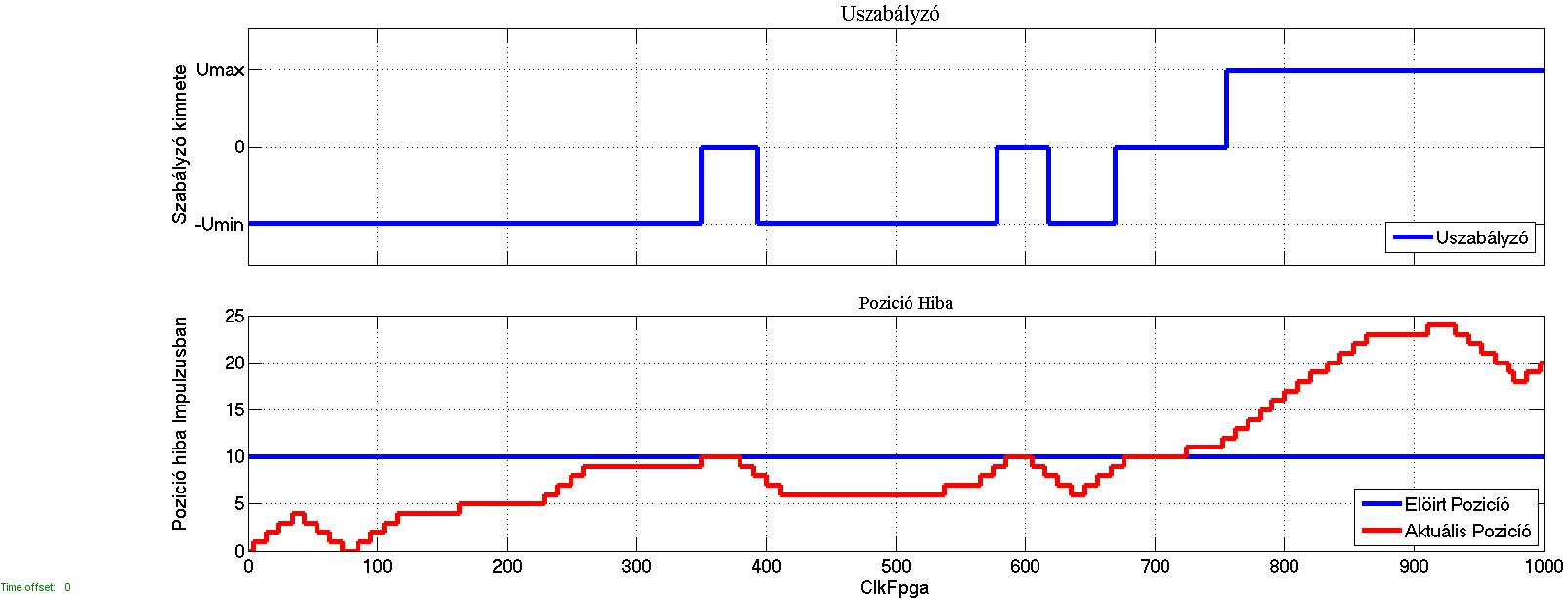
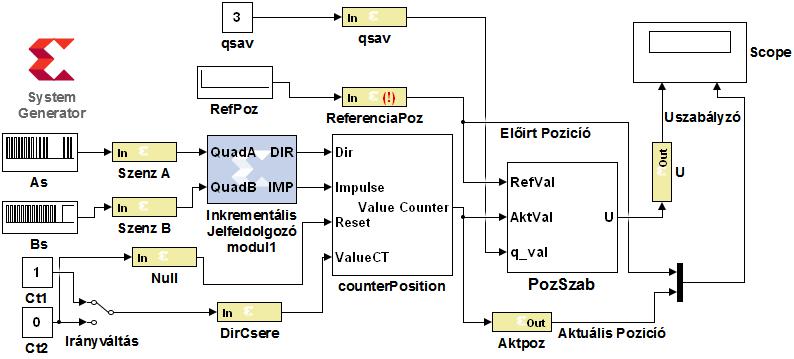
Kép. 3.9 A pozíció szabályozó System generátoros felépítése

#### Szabályozó szimulálása

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

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

Kép. 3.10 A pozíció szabályozás moduláris felépítése System Generátor környezetben



### Hardveres mérések

#### DC motor sebesség szabályozása mérőstandon

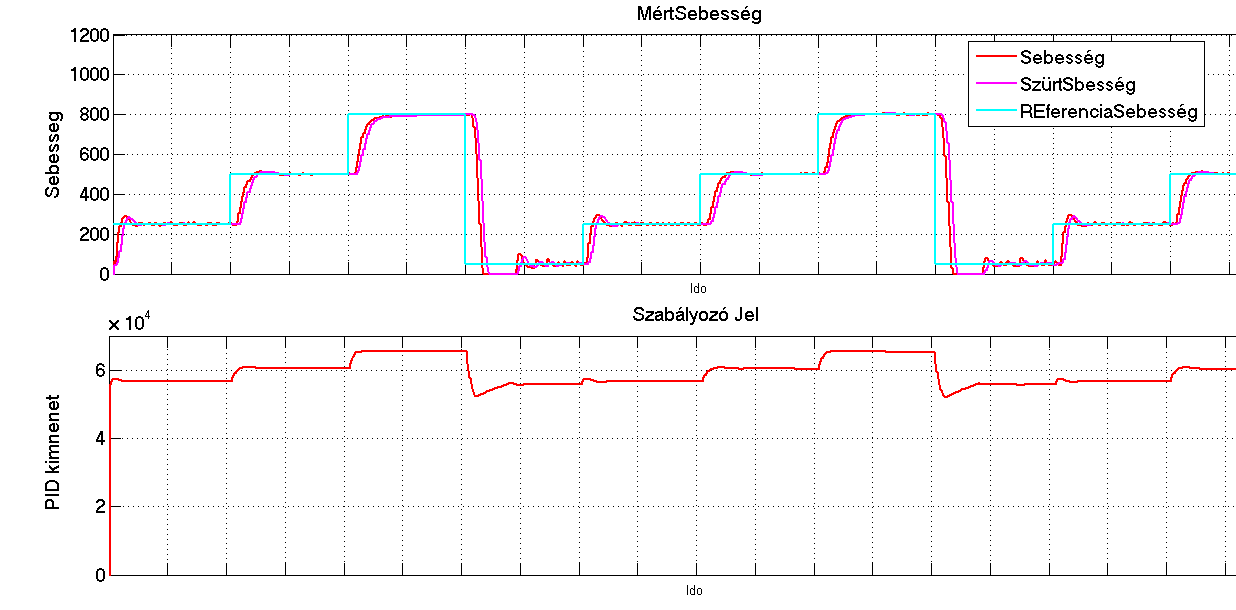
A sebesség szabályozáshoz szükséges érzékelőt az általam megvalósított inkrementális jeladó segítségével oldjuk meg. A motor egy H hídba van kötve négy N csatornás MOSFET tranzisztor segítségével. A híd bemenetei 5V logikai szintű PWM (kitöltési tényezője maximálisan 99% lehet) és DIR (irány- segítségével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

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

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

a PID szabályzó mintavételi periódusa

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

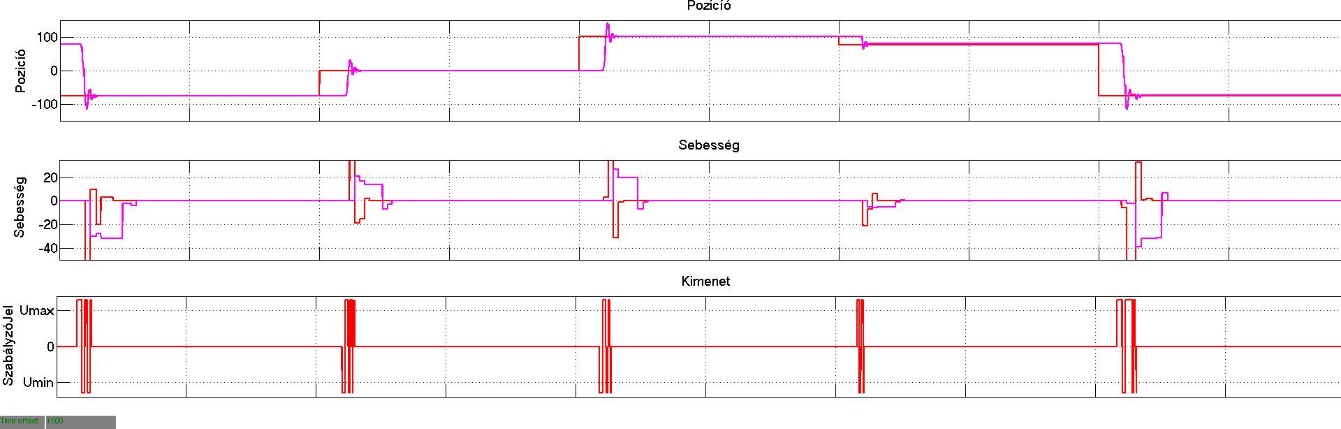


Kép. 3.11 Sebesség szabályozás PID szabályozóval.

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

#### DC motor pozíció szabályzása mérőstandon

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



Kép. 3.12 Pozíció szabályzása a Kép. 3.9 látható kialakításban.

## Sebesség és pozíció szabályozót tartalmazó IP mag generálása System Generátor-bAn

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

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

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

A Kép. 3.15 „*Sebesség Szab*” modult tartalmazza, Kép. 3.17 látható modulokat és a sebesség szabályzására hivatott, míg a „*Pozicíó Szab*” tartalmazza a Kép. 3.16 látható modulokat és a pozíció szabályzózására hivatott.

A szabályozók referencia bemeneteit osztott regiszterekkel írjuk elő a MicroBlaze processzorból. Mindkét szabályozó tartalmazza a PWM generátort, és a sebesség mérő modult (*counter Sebesseg*). A két inkrementális érzékelő jeleit egyetlen modul segítségével dolgoztam fel „*Inkrementális Jelfeldolgozó 2*”, amely látható a. Kép. 3.22 .

A „*Config*” osztott regiszter bitjeivel be- vagy kikapcsolhatunk funkciókat:

- PWM generátorok kimentének engedélyező jele

- sebesség mérésének valamint a pozíció mérésének előjel változtatása

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

Tábla. 1Konfig regiszter funkciói

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

A Kép. 3.51 látható az 1 forgó talp sebességet generál az kör mentén, a robot lánctalpának az aktuális sebessége

A sebességek összeadását (Kép. 3.17 látható) *Mult, Mux, Viszacsatolás* modulok végzik el. Az összeadást ki vagy be kapcsolhatjuk a *Config* regiszter negyedik bitjével.

Abban az esetben, ha az összekapcsolást létre szeretnénk hozni a *MUX* elnevezésű modul szelekciós bementére 1t adunk így kiválasztva a szorzó modultól érkező 16 bites egész számot. Az egész számot úgy generáljuk, hogy a mért szögsebességét beszorozzuk a Forgótalpak hosszával, majd elosztva a nagykerék kerületével.

Jelen esetben:

Következésképpen, ha a Forgótalpak pozícióját változtatjuk, miközben a rajta levő lánctalp konstans sebességgel halad a földhöz képest, a lánctalp sebességét nem tudjuk konstanson tartani csak a sebesség szabályzóval. Eezért van szükség a sebesség szabályozó referencia értékének a módosítására.

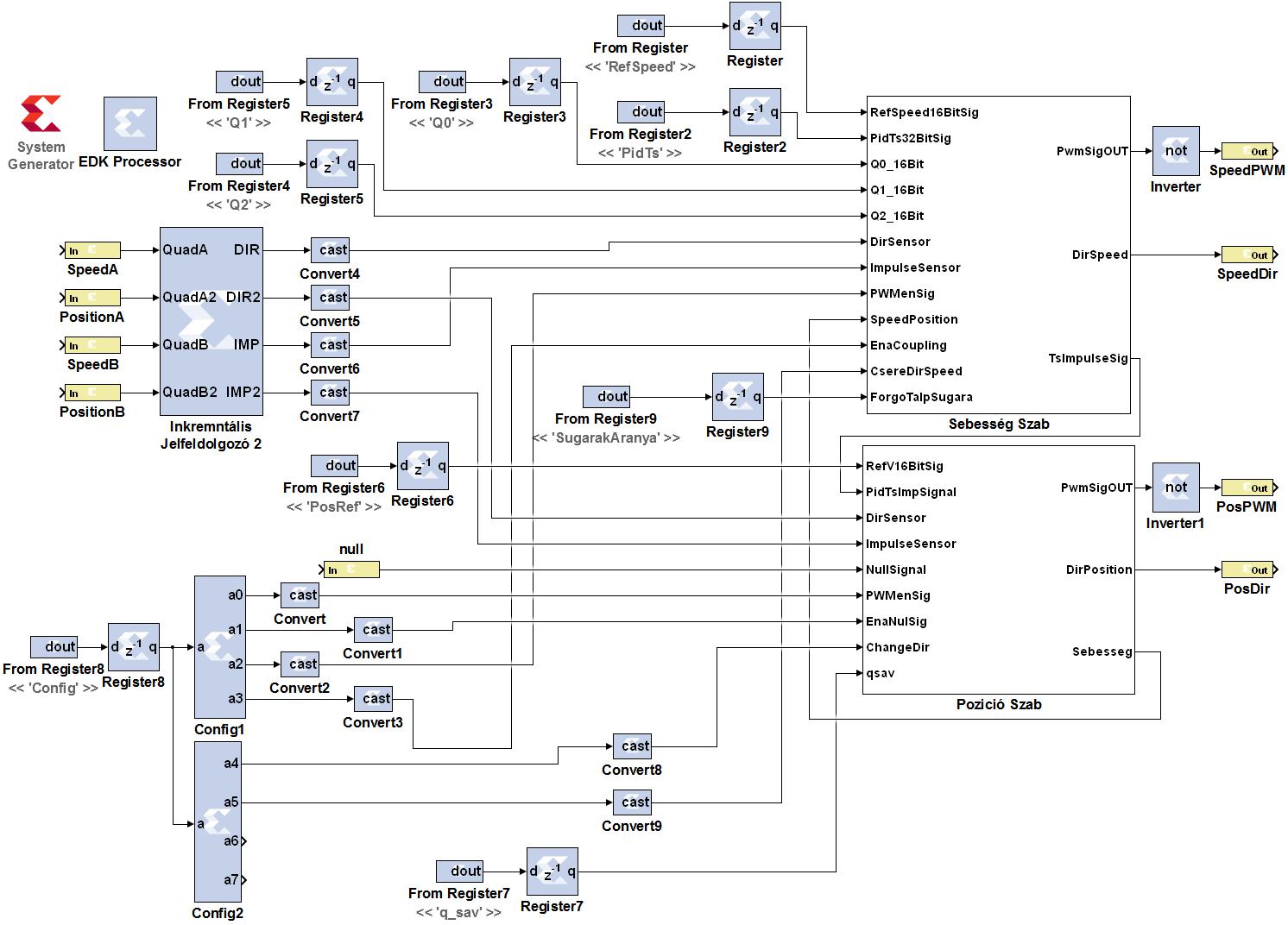
Az osztott regiszterek olyan hardveres elemek, amelyeket az FPGA-ban hozunk létre, szoftveresen a memóriába illesztett címekkel rendelkeznek, amelyeket írhatunk vagy olvashatunk. A System generátorban a *ToRegister* illetve *FromRegister* elemekkel tudjuk kivitelezni. A *ToRegister* típusú regiszterek csak írhatóak, a szoftvertől tudunk adatokat leküldeni a hardvernek. A *FromRegister* típusúak pedig csak olvashatók, vagyis adatokat tudunk felvinni a szoftvernek, ha a hardver oldalról nézzük. A szoftver általában egy MicroBlaze processzoron fut vagy egy beépített mikroprocesszoron. A regisztereknek betudunk állítani típusokat. A rendszerben, a könnyebb kezelhetőség érdekében, az osztott regiszterek típusa 16-bit vagy 32-bit nagyságúak, előjeles vagy előjel nélküliek. Szoftveresen pedig egy memória művelettel tudjuk kinyerni vagy beírni az adatokat.

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

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

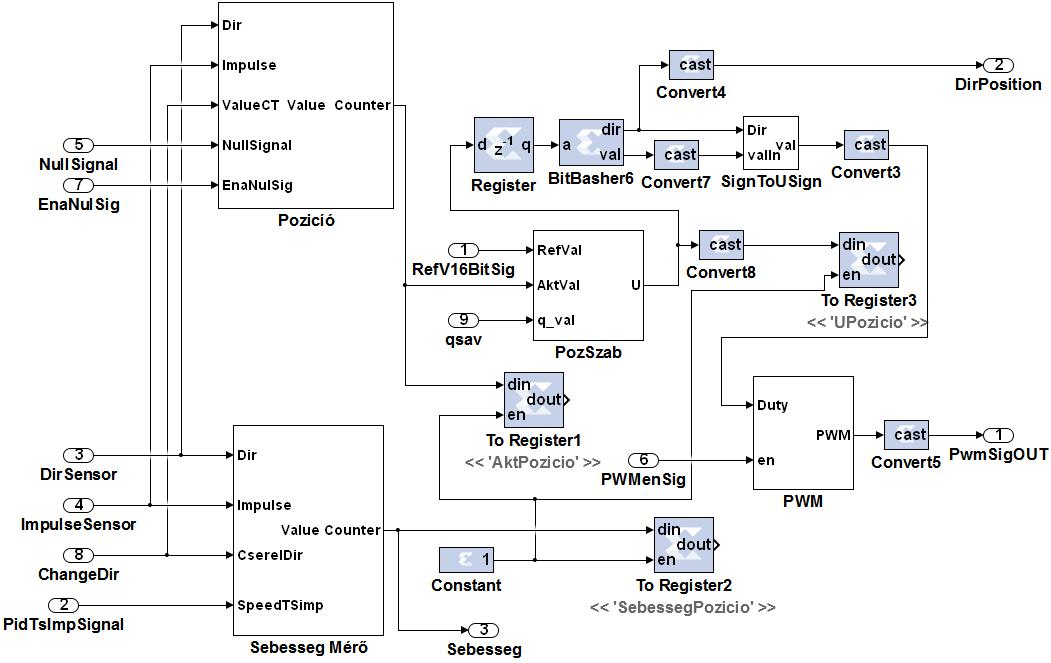
Az IPmag modul erőforrás igénye az FPGA rendszerben: Flip-Flops=760, LUTs=579 rendelkezésre álló erőforrások : Flip-Flops=9,312, LUTs=9,312.

Az IPmagok az FPGA-ba hardveresen vannak összealítva logikai kapuk és egyéb digitális elemek segítségével, ezért a benne található modulok mind az FPGA 50Mhz órajelére működnek.



Kép. 3.13 Sebesség és pozíció szabályozást tartalmazó Ipmag System generátoros felépítése

**Pozíció szabályozó modulok**:

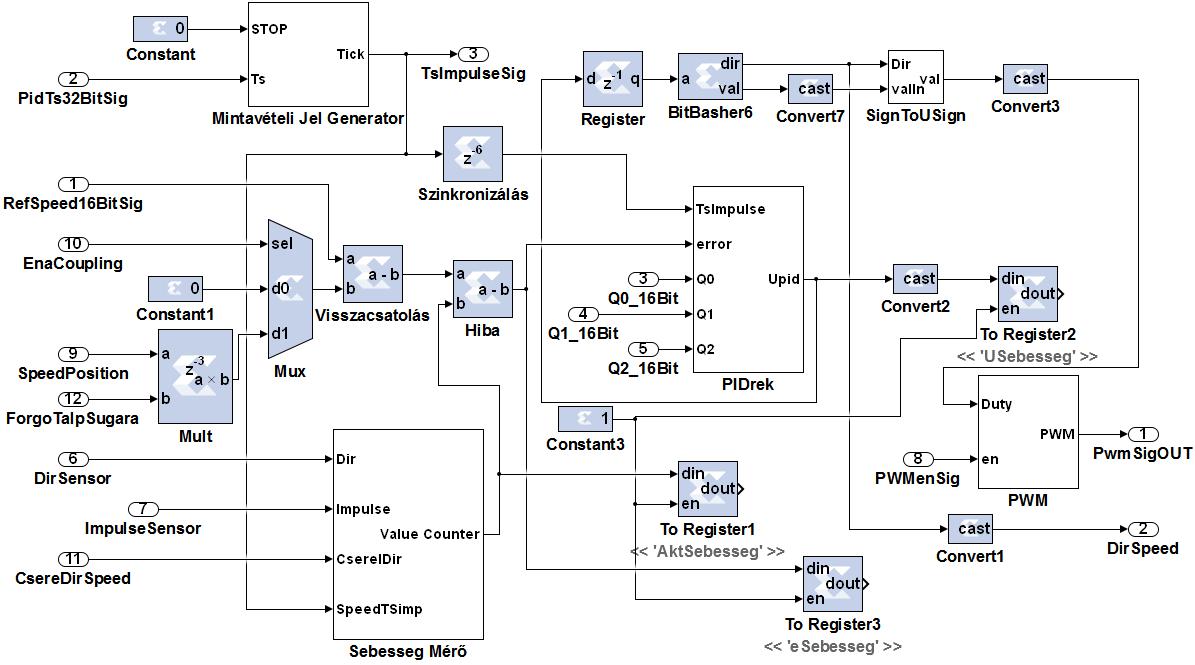


Kép. 3.14 Pozíció Szabályzó modul belső felépítése

* A „*Pozíció*” nevű modul (Kép. 3.25 moduljai) tartalmazza a képen látható modulokat, feladata a relatív pozíció mérése impulzusban, kimenete egy 16 bites előjeles szám.
* A „*Sebesség Mérő*” modul (Kép. 3.26 moduljai) feladata a sebesség mérése impulzus per mintavételben, a kimenete egy 16 bites előjeles szám.
* A „*Pozíció Szab*” nevű modul (Kép. 3.9 moduljai) feladata a pozíció szabályozása.
* A „*PWM*” nevű modul (Kép. 3.32 moduljai) feladata a pwm jel előállítása.
* A „*BitBasher6*”, és a „*SignToUsign*” nevű modulok átalakítják a szabályozótól érkező 17 bites előjeles számot egy 16 bites előjel nélküli számmá és egy 1 bites jellé, amely tartalmazza az a17 bites szám előjelét.

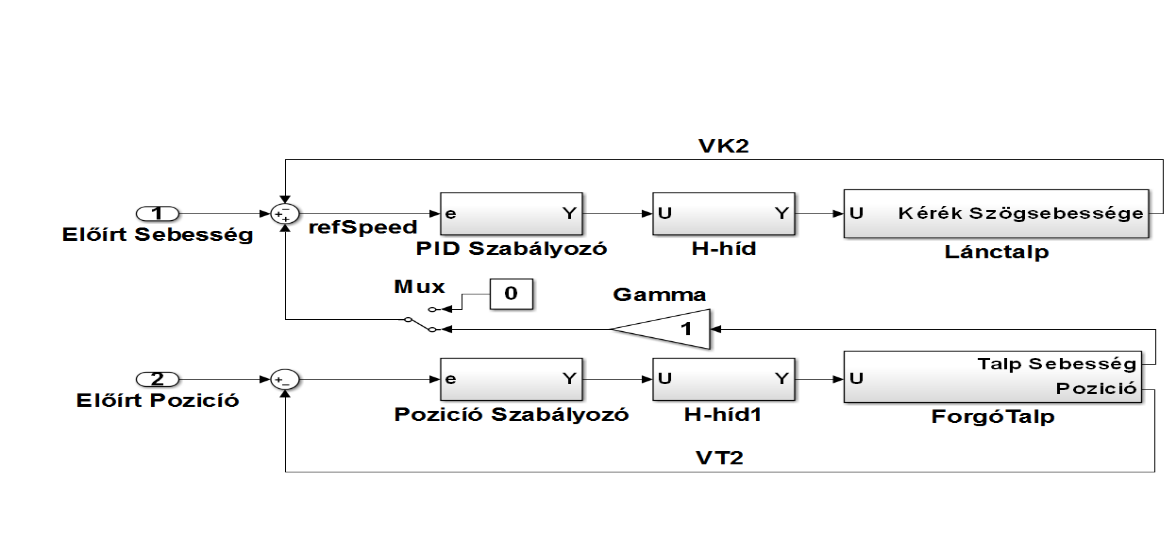
A mért paraméterek osztott regiszterekbe kerülnek: „*UPozicio*”,-a beavatkozó jel, „*SebessegPozicio*”- pozíció deriváltja, „*AktPozicio*”- pozíció. A szabályozó körök az FPGA hardveres szabályzóin keresztül zárodnak, a kikerülő mért adatok csak kirajzoláshoz szükségesek. A „*Pozició*” modul segítségével megkapjuk a szöget, amelyben a Forgótalp pillanatnyilag áll. A *Sebesség Mérő* visszatéríti a pozíció deriváltját, a szögsebességet impulzus/mintavételben. A szögsebesség mérése szinkronizálva van „*PidTsImpSignal*” bemeneten keresztül a PID szabályozó mintavételi periódusával, amelyet a Kép. 3.17 látható „*Mintavételi jel Generátor*” állit elő.

A „*Sebesség Szab*” modul felépítése hasonló a pozíció szabályzó felépítésére, annyi eltérés van, hogy itt nem jelenik meg pozíció mérés csak sebesség. Viszont megjelenik a „Mintavételi Jel Generátor” (lásd Kép. 3.7), melynek feladata, hogy biztosítja a mintavételi periódust a sebesség mérő modulok PID szabályzók számára.



Kép. 3.15 Sebesség Szabályzó modul felépítése

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

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

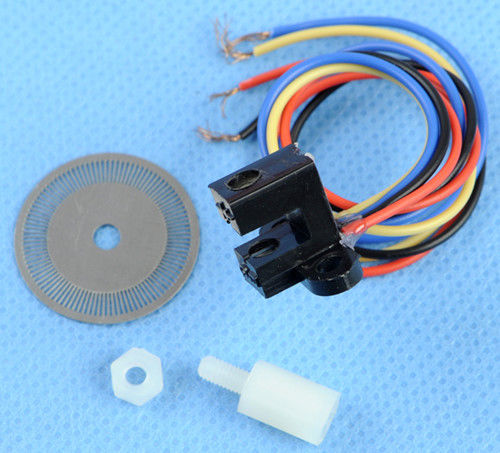
A Kép. 3.51 látható alsó ábrán a pozíció változása sebességet generál a kis keréknek, ha mi a sebességet szeretnénk tartani, akkor a következő a teendő: előírjuk a sebesség szabályzónak hogy ne változón meg a sebesség: , ahol az egy arányos ági tényező.

A PWM generátorok, mind a sebesség mind a pozíció szabályzóknál, függetlenül működnek a szabályzóktól, a szabályzó csak a kitöltést tudja befolyásolni. A szabályzó kimente hardveresen összekapcsolódik a PWM generátor bemenetével. A sebesség mérő modul a „*Mintavételi Jel Generátor*” által előállított periodikus impulzusok között méri meg az inkrementális tárcsa elfordulását. A sebességmérő közvetlenül csatlakozik, a szabályozóhoz egy 16 bites fizikai összeköttetés segítségével.

## Szenzorok

### Inkrementális Érzékelő

#### Optikai inkrementális vevő felépítése

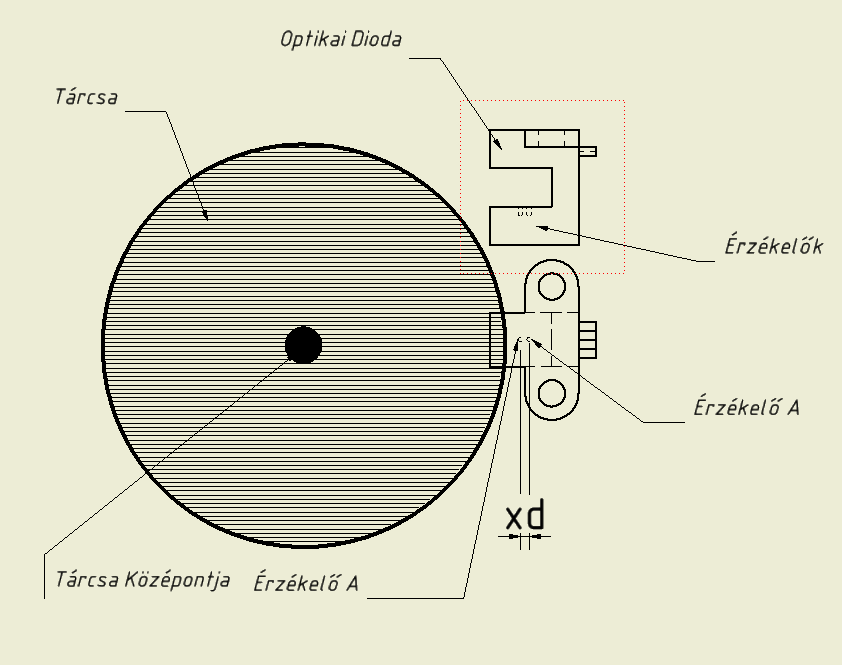
 Az optikai érzékelő két részből áll, egy optikai forrásból és egy vevő részből. Két optikai kapcsoló eszközt tartalmaz egymástól távolságra.

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

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

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

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

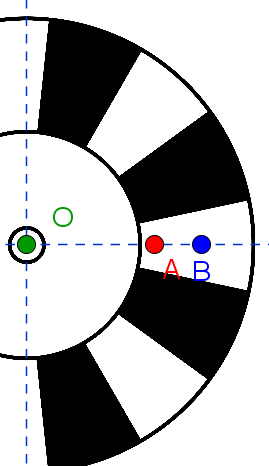


Ha a Kép. 3.18 látható módon helyezzük el, sugár irányban nem jön létre késés a két jel között (A és B), így nem lehetne meghatározni a forgás irányát. Ez elkerülendő, a réseket meg kell dönteni egy alfa szöggel így kialakul a késés is.

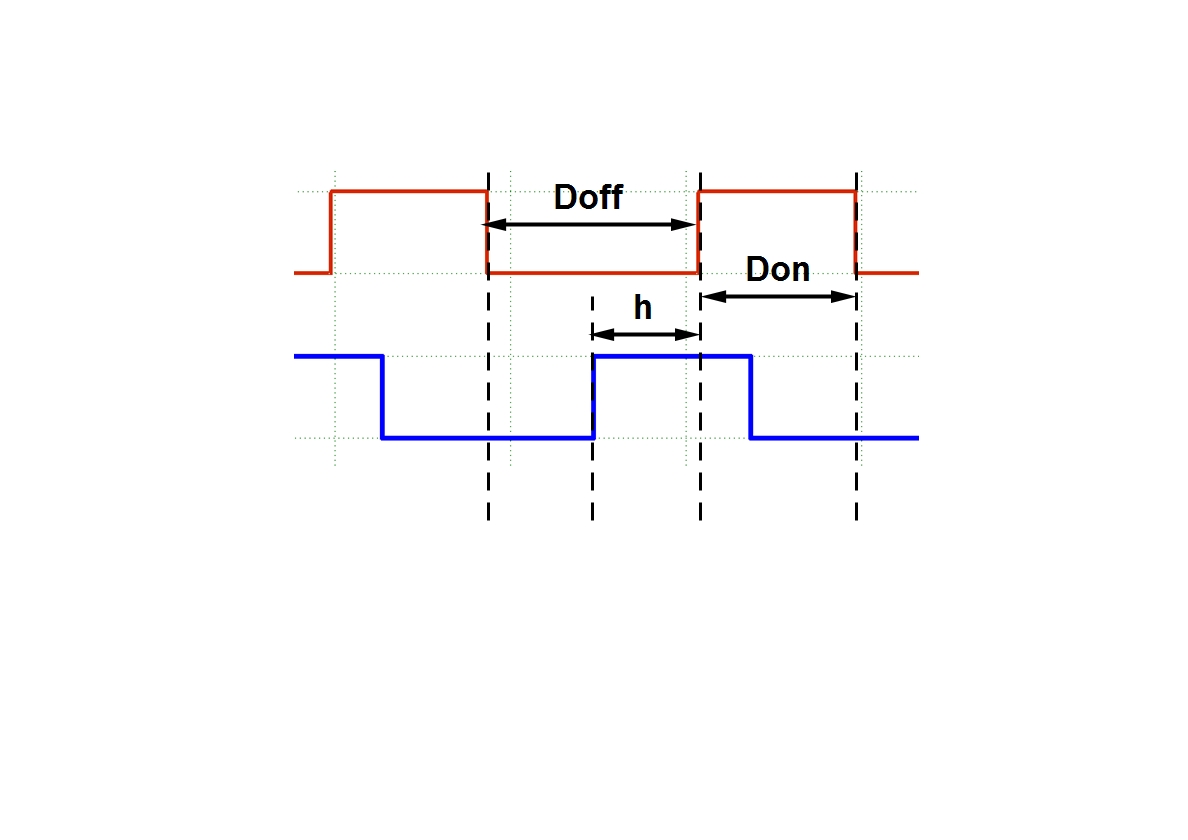
Tekintsük az A és B pontokat az Érzékelő A és Érzékelő B pontjainak. Az AB szakasz hossza ismert, amely megadja az érzékelők közti távolságot.

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

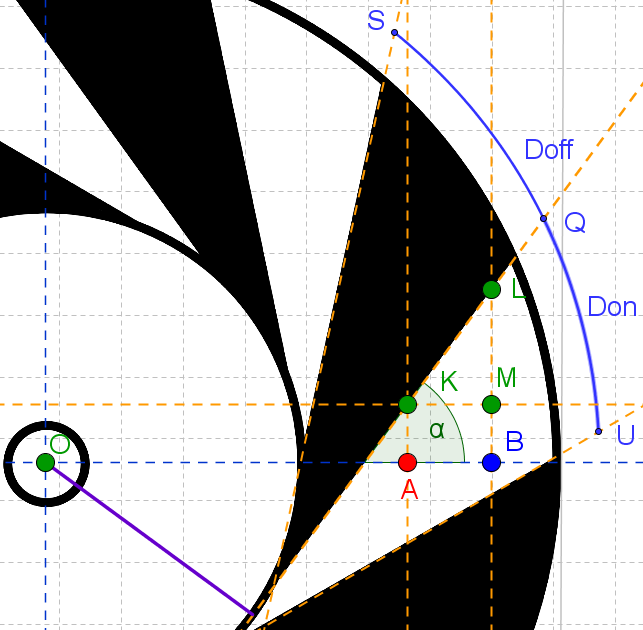
Az Kép. 3.19 a fehér mezők az inkrementális tárcsa réseit képviselik. A rések száma megadja, a tárcsa felbontását N.



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

 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 (Kép. 3.21).

**Ismertadatok**:



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

Ha az érzékelő Kép. 3.21 módon van illesztve a tárcsához, akkor felírható az összefüggés, amely meghatározza a két jel közötti késést.

,.

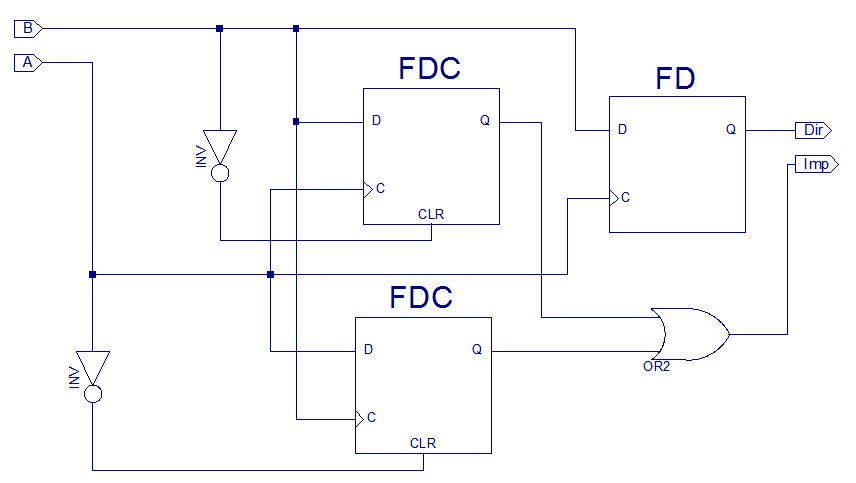
A minden esetben nagyobbnak kell lennie, mint a távolságnak.

### Inkrementális érzékelő jeleinek a feldolgozása FPGA áramkör segítségével

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

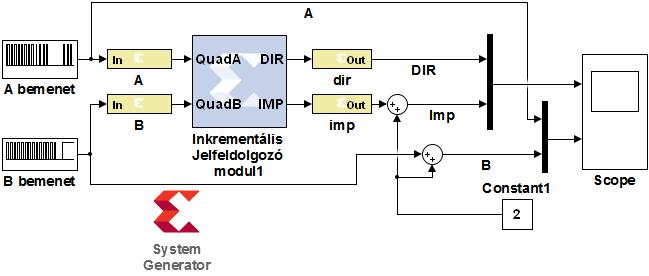
A modul VHDL programnyelven készült a Kép. 3.22 látható kialakítás szerint. BlackBox modul segítségével integráljuk a System Generátor környezetben (BLACK BOX1). A ki és bemeneti portókat illesztjük a System Generátor környezetben található elemekhez, majd létrehozzuk a szimulációs bemeneti jelelt, amelyeket az A, B sárgával jelölt modulokon keresztül viszünk be a rendszerbe. Az FPGA áramkörben megtalálható modul segítségével a jeleket feldolgozzuk és két kimenő jelet generálunk a Dir (megadja a forgás irányát), valamint az Imp (minden ablak elhaladásakor generál egy felfutó jelet).

A beérkező A és B jeleket egy tagadó kapu segítségével bekötjük a FDC tárolok CLR bemenetére, a Kép. 3.22 látható módon. Egy harmadik tároló segítségével meghatározhatjuk a forgás irányát. Az „*imp*” impulzus kimenete akkor lesz logikai 1, ha valamely FDC tároló Q kimenete is Logikai 1 lesz.



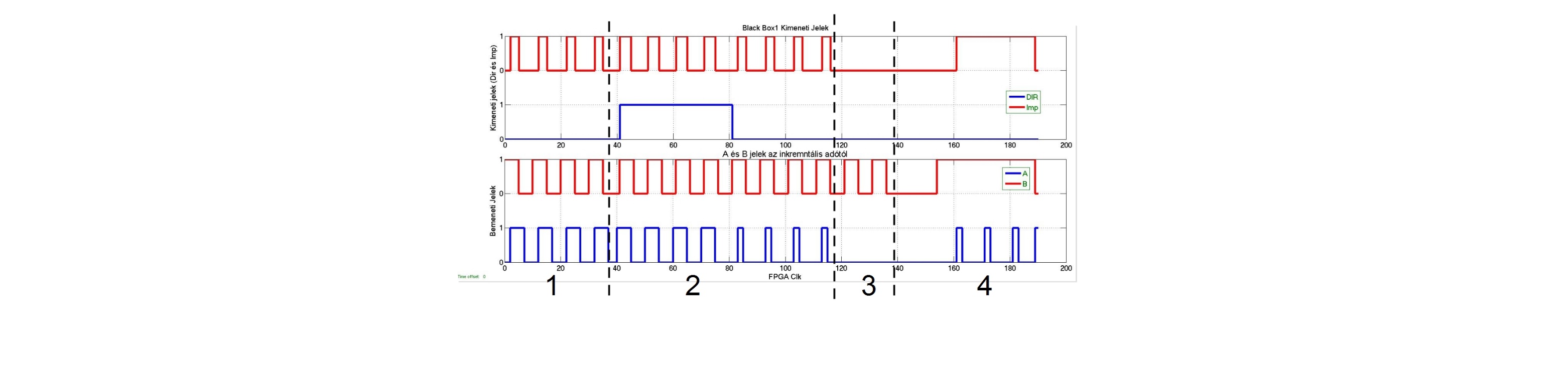
#### Szimuláció System Generátorban

A Kép. 3.24 látható a szimulációs eredmények az *A* és *B* bementi jelek (alsó ábra), *Dir* és *Imp* kimentek (felső ábra).



A bementi jelek négy kategóriába sorolhatók:

1. Az A jel késik a B jelhez képest, a kimeneti jeleken látható (érkező impulzusok és az irány).
2. Az B jel késik az 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 történik meg, amikor a tárcsa forgási iránya azelőtt változik meg mielőtt elérte volna a sötét mező a *B* foto tranzisztort.



Az elkészített „*Inkrementális Jelfeldolgozó modul1*” segítségével feldolgozhatom az érzékelő jeleit. A modul kimenetére majd újabb modulokat illeszthetünk, melyek segítségével mérjük a pozíciót vagy a szögsebességet.

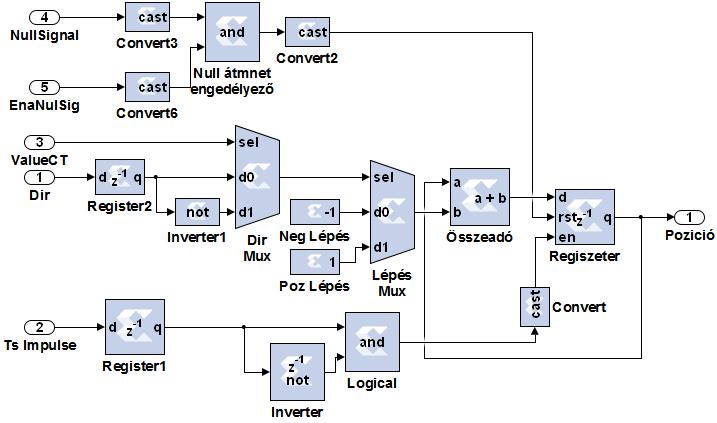
#### Pozíció mérése Inkrementális adó segítségével

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

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

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

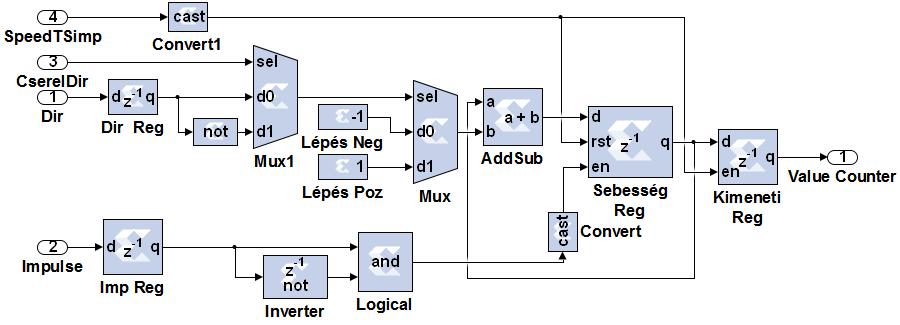
A számláló logika feladata, hogy növelje vagy csökkentse eggyel a regiszter tartalmát, annak függvényében, hogy a *Dir* jel milyen értékű. A „*ValueCT*” bemenet segítségével meg tudjuk fordítani a számolás irányát, így kényelmesebben tudjuk majd a robot vonatkoztatási rendszeréhez hangolni az érzékelőket.



#### Szögsebesség mérése Inkrementális adó segítségével

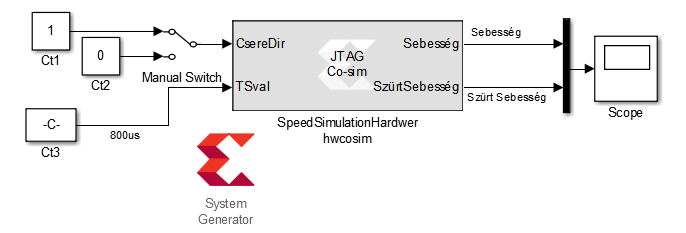
A sebesség mérésénél hasonlóképpen járunk el, mint a pozíció mérésénél. A sebességet, időegység alatt érkező impulzusok számát mérjük. Az impulzusok az inkrementális adó jeleinek a feldolgozó moduljától érkeznek. A modulban megtalálható a pozíció mérésénél kifejtet számláló logika, tároló logika és él detektáló logika.

A tároló regiszter minden mintavételi impulzus érkezésekor „reset” állapotba kerül, így az értéke 0 lesz. A „CsereDir” bemeneten keresztül megtudjuk változtatni a pozíció előjelét, erre a robothoz rögzített koordináta rendszerhez való illesztéskor lesz szükséges.



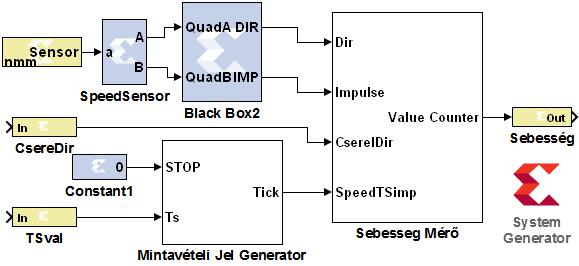
##### Hardveres mérések

A mérések során Kép. 3.31 látható egy DC motornak a tengelyén mért fordulatszám, miközben a motor maximális sebességen pörög. Ahhoz hogy eltudjuk végezni a szimulációkat, az inkrementális szenzort hardveresen kell illesztenünk a FPGA kivezetéseire, amelyet a „*Sensor*” modul old meg.



Kép. 3.25 DC motor sebességének mérése FPGA lapon

A Kép. 3.29 látható a Hardveres szimulációhoz szükséges logika. Sensor modul tartalmazza az FPGA azon kivezetéseit, amelyekre az érzékelő fizikailag kapcsolva van. Az érzékelő jelei a *SpeedSensor* (csak a jelek bekötésében segít) nevű modulon keresztül a feldolgozó modulba érkeznek be a jelek („*Black Box2*”). Ugyanakkor még megtalálható egy *SampleTime* *Generator* nevű modul is, amelynek a feladata ( ) periódusú impulzusok generálása, a periódust bemeneten adhatjuk meg. A kiszámolható ms-ban az alábbi összefüggéssel.



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

A sebességet adott időegység alatt beérkező impulzusok számával mérjük. A mérések során a rendszer tartalmazott egy 5 pontos átlagoló szűrőt.

Pirossal látható a szűrő kimente, de a rendszer a tárgyalt rendszert nem tartalmazza, mivel a mérések egy előző verzióban készültek.

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

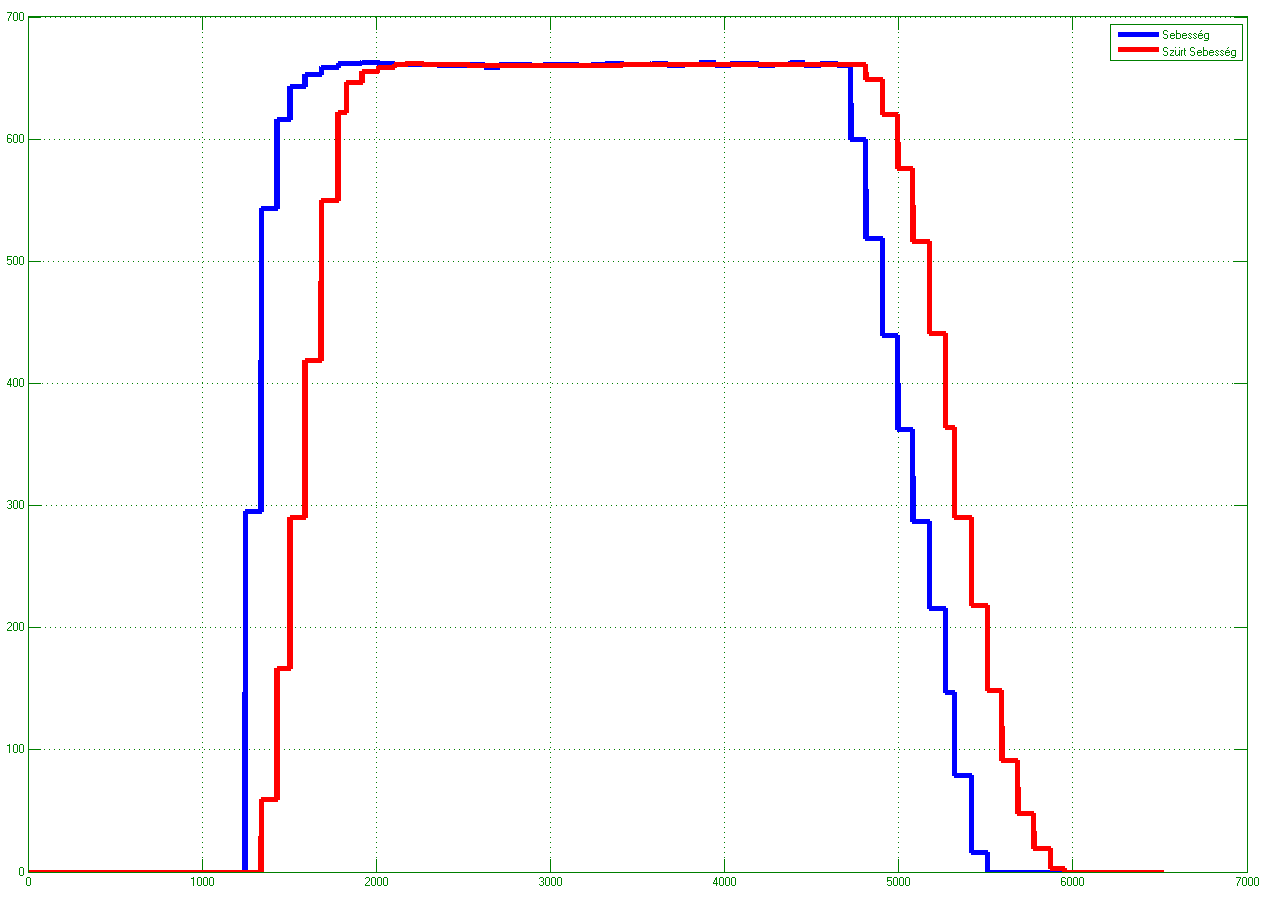
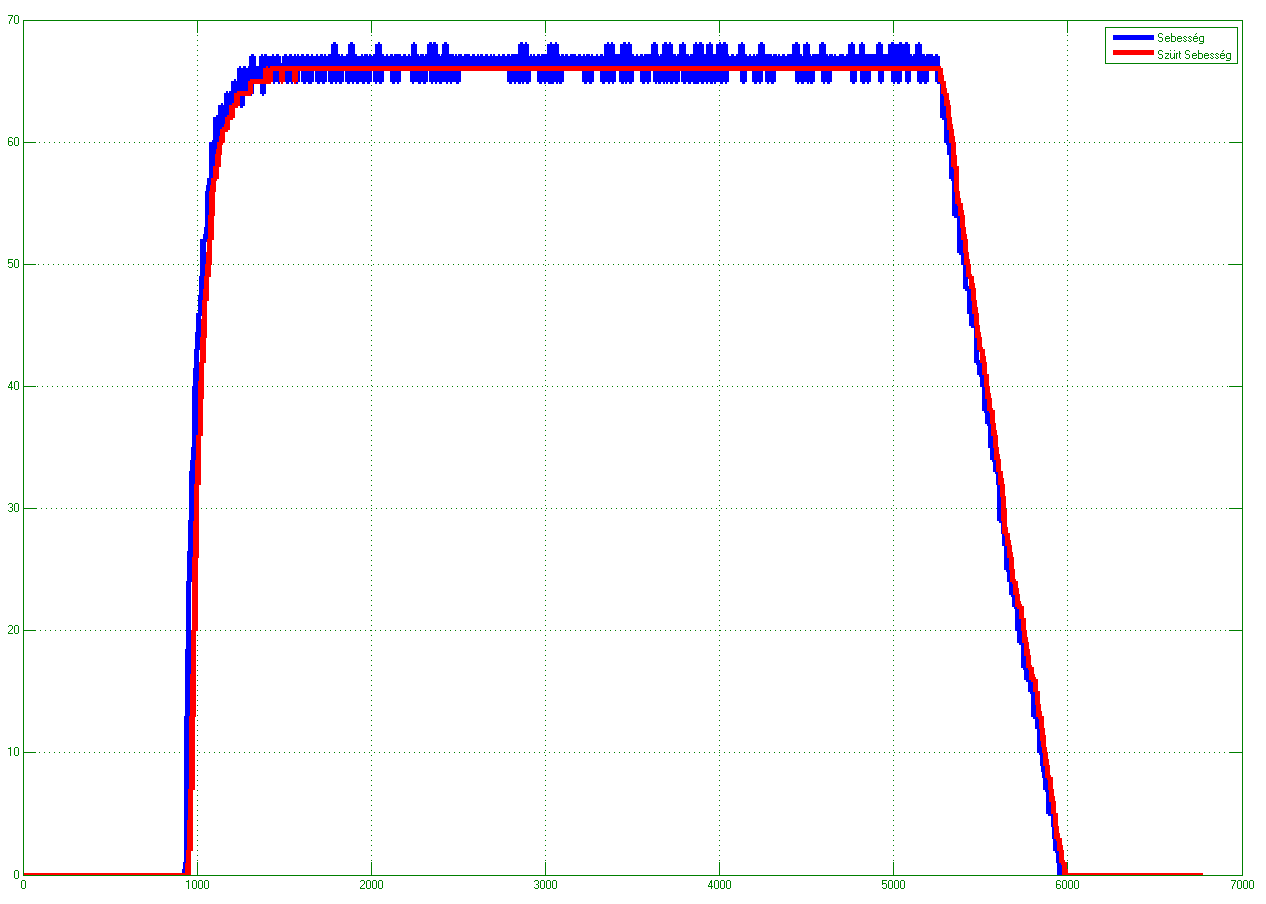
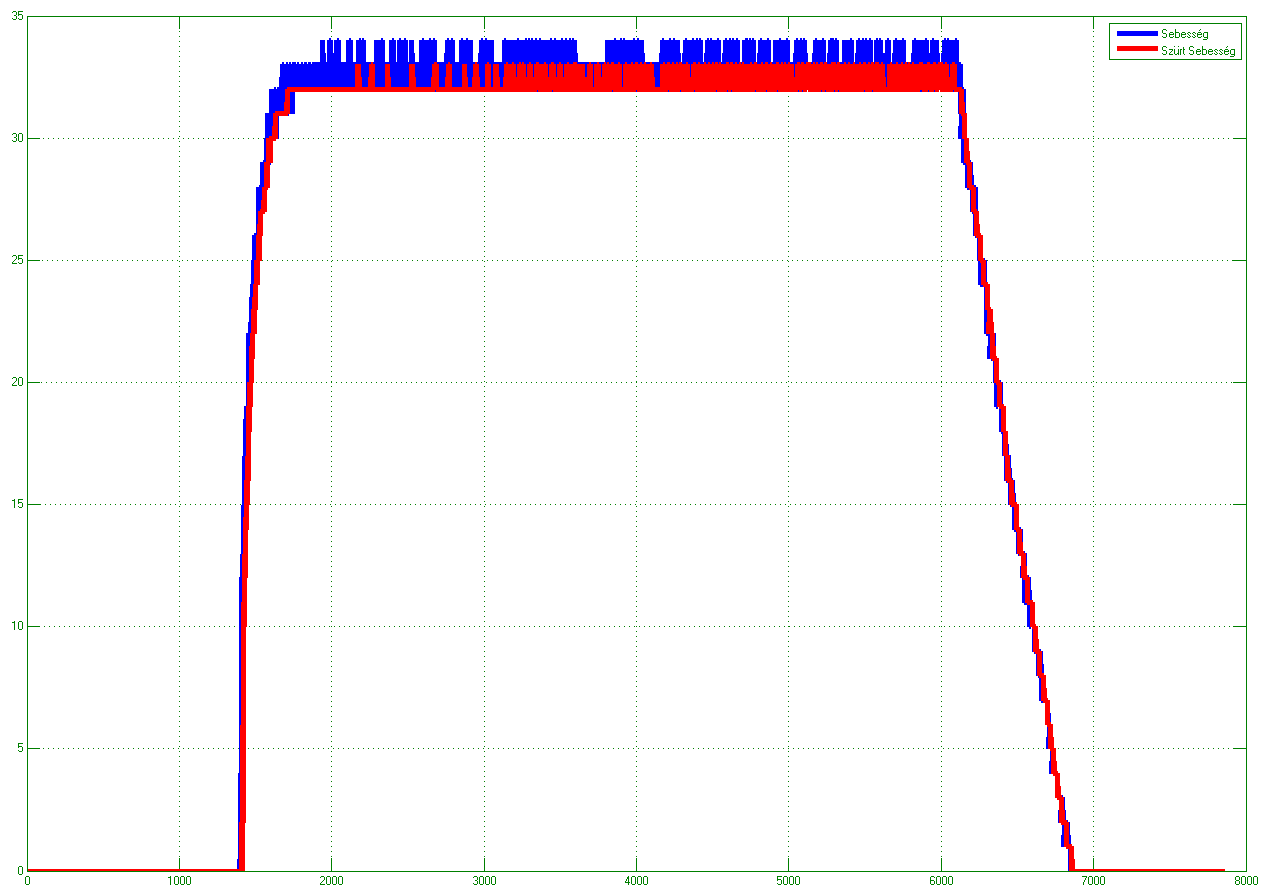
A fenti példában a az maximális fordulat, azért mert a motor a maximális megengedett feszültséggel volt táplálva, terhelés nélkül.

Ha ismerjük a motor maximális fordulatszámát, kiszámolható mérés felbontása.

A fenti példa esetében a felbontás 65 mivel a motor maximális fordulatszáma és 0 között 65 egész értéken van Kép. 3.32.

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

A Kép. 3.32.b képen a mintavételi időt 80 ms növelve, megnőtt a felbontás is megközelítőleg 650 re. A Kép. 3.32 c csökkentjük a mintavételi időt és ez megközelítőleg 33-ra csökkentette a rezolúciót.



## MPU-6050 giroszkóp és gyorsulásmérő

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

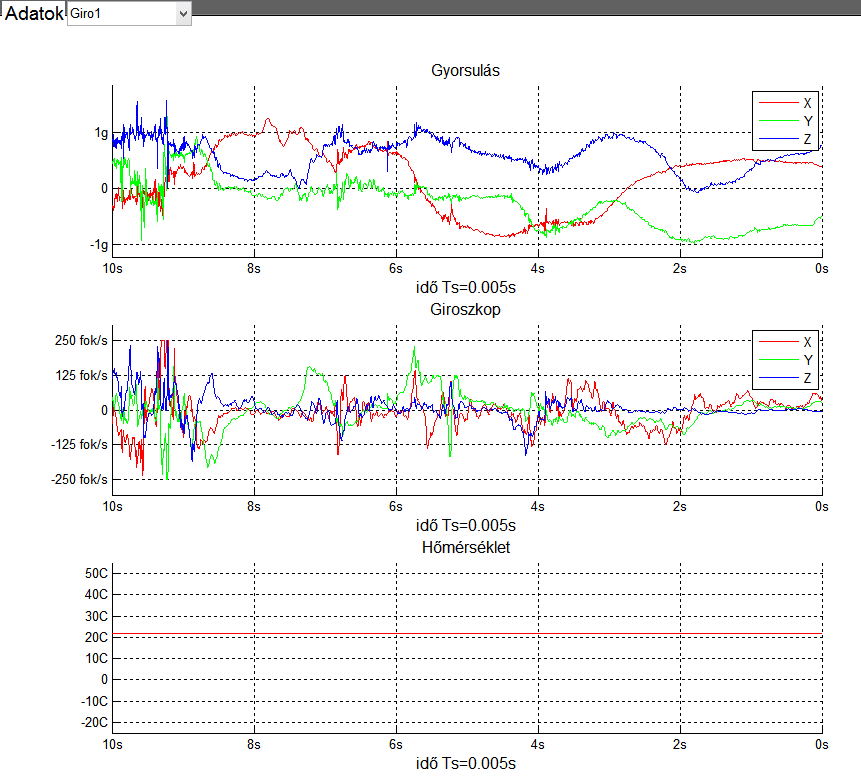
* FIFO memóriák kikapcsolása FIFO\_EN=0x00
* Gyorsulásmérők indítása ACCEL\_CONFIG=0xE7
* PWR\_MGMT\_1 =0x00 bealítjuk a szenzort ciklikus működésre és 8MHz órajelre.

A giroszkóp mérési határai változtathatóak: ±250, ±500, ±1000, ±2000 a giroszkóp, illetve ±2g, ±4g, ±8g, ±16g gyorsulási skálák közül választhatunk. A gyorsulást a FS\_SEL 3bites regiszter rész írásával tudjuk alítani, a AFS\_SEL 3 bites regiszter rész írásával állíthatjuk a gyorsulásmérő skáláját.

|  |  |
| --- | --- |
| FS\_SEL=3 ±2000 º/s | AFS\_SEL 0 ± 2g |
| FS\_SEL=1 ±500 º/s | AFS\_SEL 1 ± 4g |
| FS\_SEL=2 ±1000 º/s | AFS\_SEL 2 ± 8g |
| FS\_SEL=0 ±250 º/s | AFS\_SEL 3 ± 16g |

Az érzékelőbe beépített hőmérő ban mért értékét megkapjuk az alábbi összefüggés alapján:

Ahol: a egy 16 bites regiszter amely tartalmazza a mért hőmérsékletet.



A Kép. 3.30 láthatjuk a mért adatokat, amelyet a GUI felületen rajzoltam ki, a jeleket 0.001s mintavételezési periódussal mintavételeztem.

## Beavatkozó elemek:

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

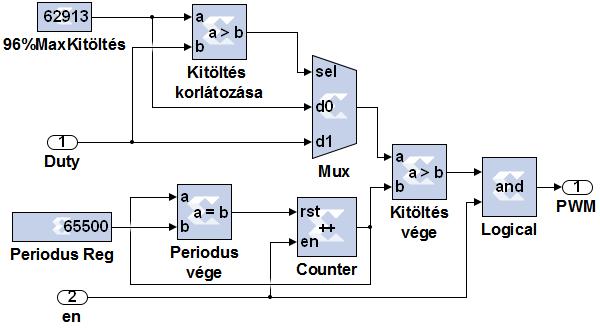
#### Megvalósítás

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

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

A PWM kitöltési tényezőjét egy „*Duty*” nevű bemeneten keresztül vihetjük be a rendszerbe, amelynek típusa megegyezik a számláló típusával. A kitöltési tényező értékének szüksége van egy skálázási eljárásra, amely segítségével illesztjük a frekvenciához.

A felépítésében be van iktatva egy korlátozás, amely segítéségével nem engedjük meg egy bizonyos százalék fölötti kitöltési tényezőt, „96%MaxKitöltés” nevű konstansba írhatjuk be 0 és 65535 közötti értékben. A kitöltési tényező maximálisan 0 és 16biten felírható maximális érték között lehet (65535). Az alábbi egyenlet segítségével kiszámíthatjuk a kitöltési tényező regiszterének az értékét.

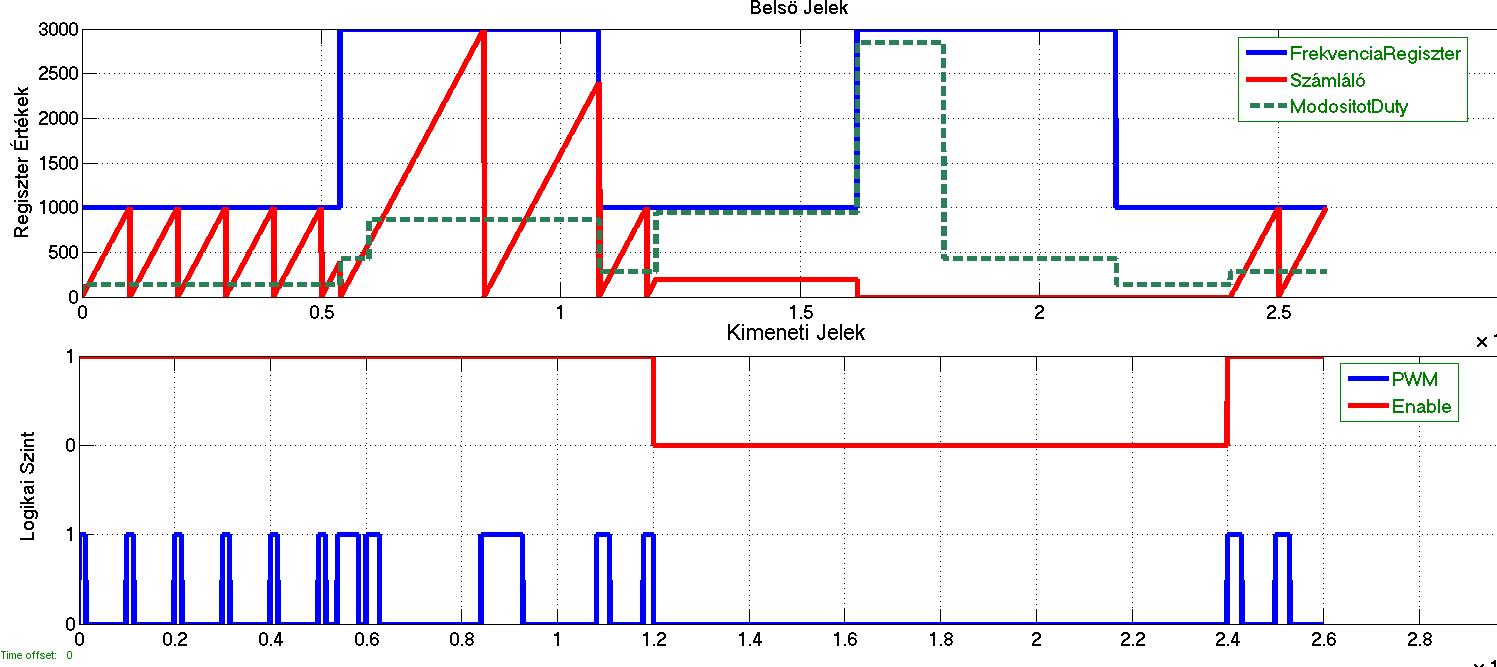


A PWM generátor kimenetét és a számláló (Counter) működését letilthatjuk az „*en*” bemenetre adott logikai 0 értékkel, máskülönben engedélyezve vannak.

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

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

A Kép. 3.33, a felső ábrán látható a kékkel jelölt frekvencia regiszter értéke, pirossal jelölt a számláló értéke, a zöld szaggatott a skálázott kiötlési tényező értéke. A kép alsó részén látható a kékkel jelölt PWM jel, illetve pirossal jelölt Enable jel.



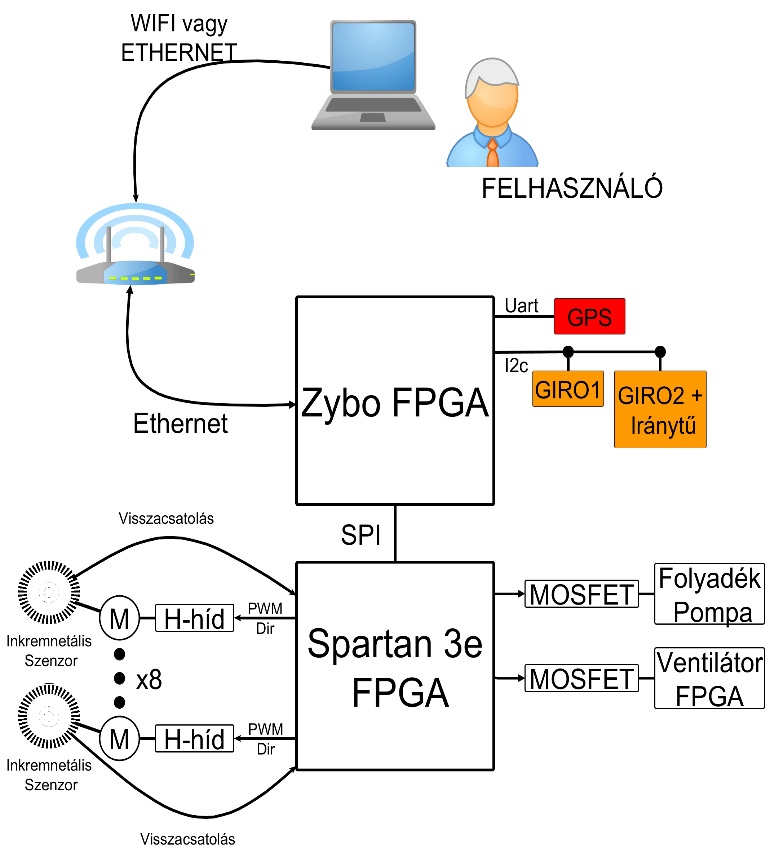
## Elektronika

### Digitális Elektronika

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

### FPGA Rendszer Felépítése

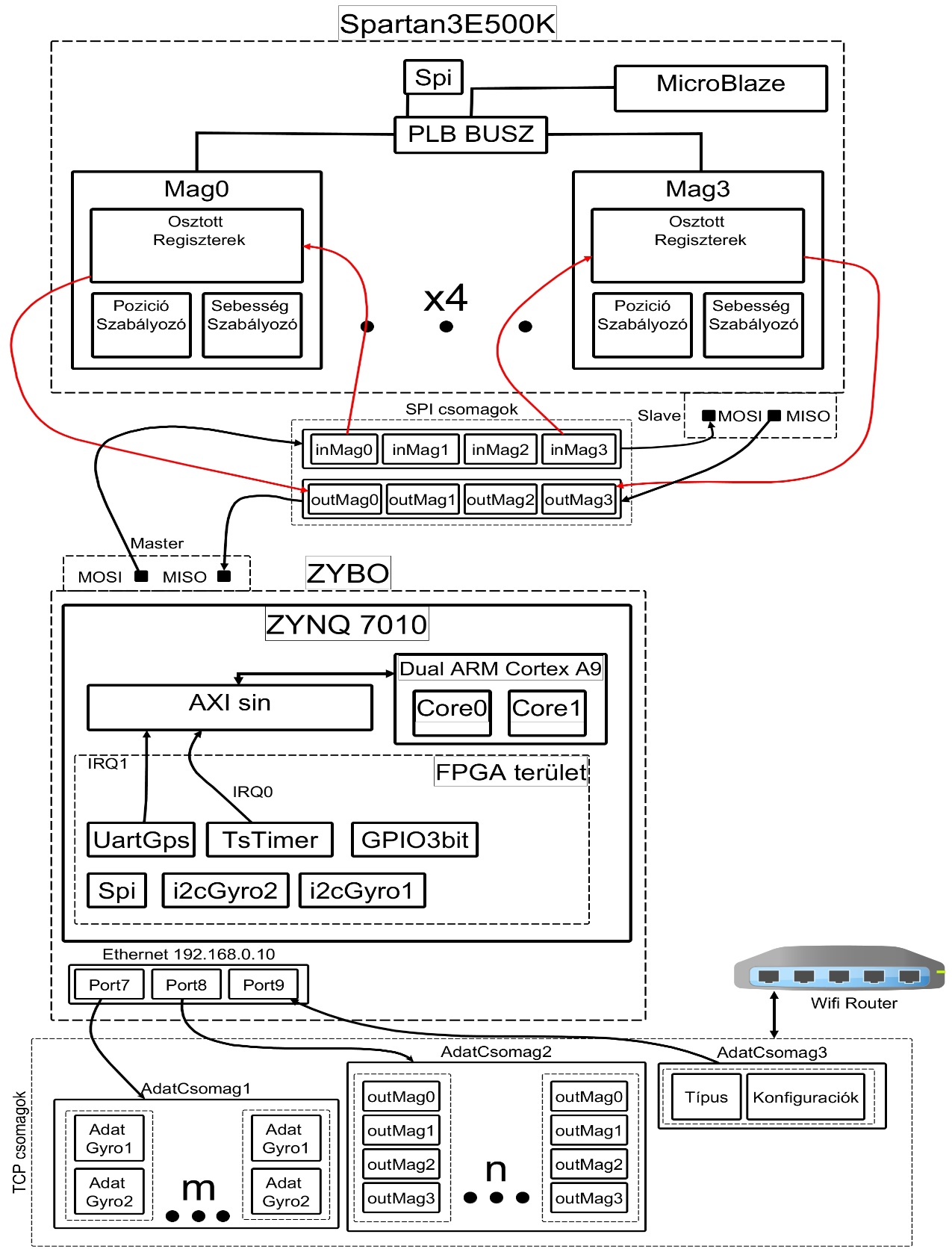
A rendszeren megtalálható két FPGA fejlesztő lap. Egy ZYBO, amely nagyobb erőforrással rendelkezik, de kevés a kivezetéséinek a száma. Valamint egy Spartan3e chippel rendelkező fejlesztőlap, amely kevés erőforrással bír, de 120 kivezetést tartalmaz.



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

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

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



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

#### Zybo FPGA fejlesztőlap

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

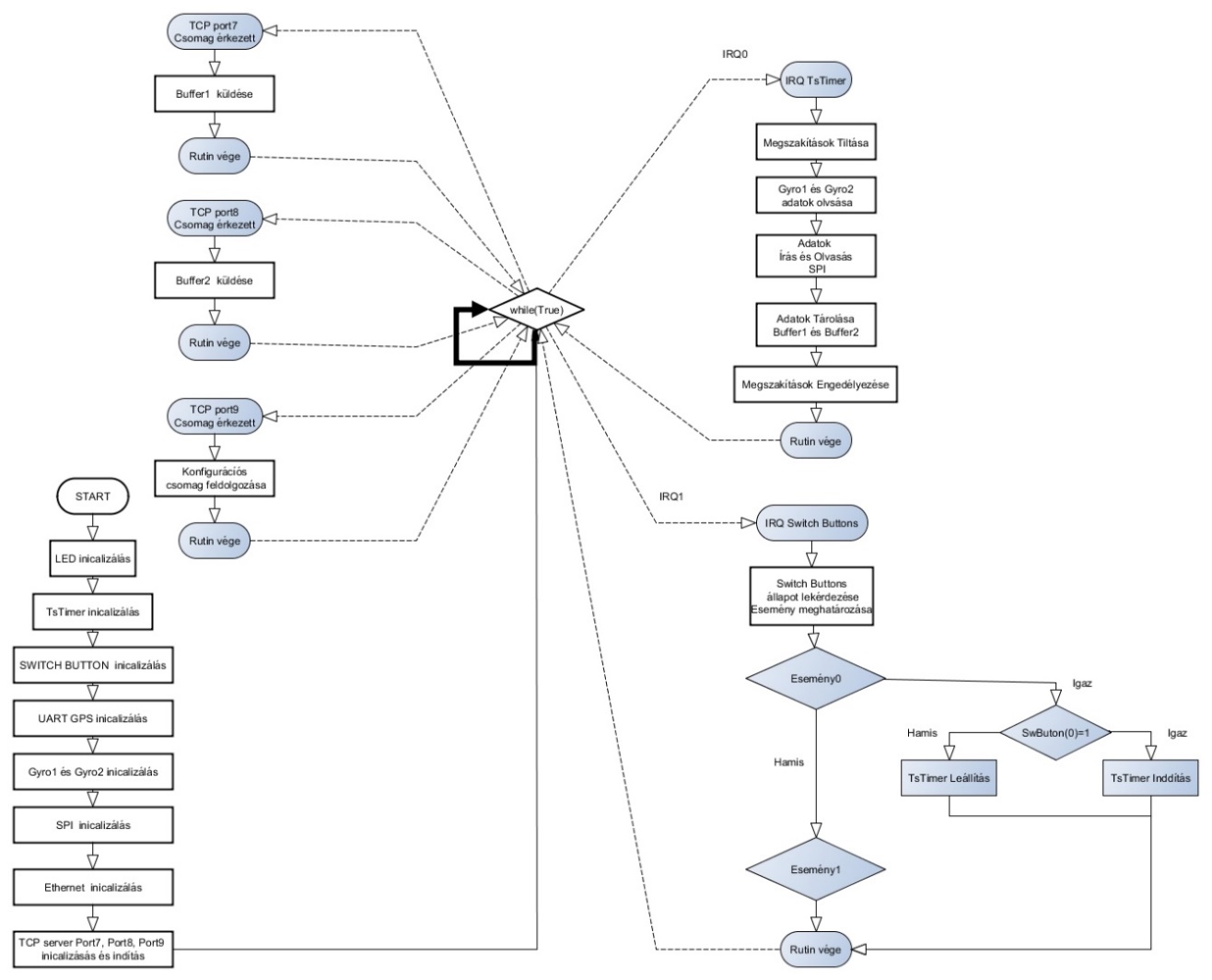
A Core0 processzor feladatai között szerepel a megszakítások lekezelése. A legfontosabb a mintavételi periódust generáló számlálótól érkező megszakítás, amelynek bekövetkeztekor a processzor begyűjti az adatokat a szenzoroktól (Giroszkóp 1 és 2). Lekezeli a megszakítást, amelyek az UART modultól érkeznek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

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

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

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

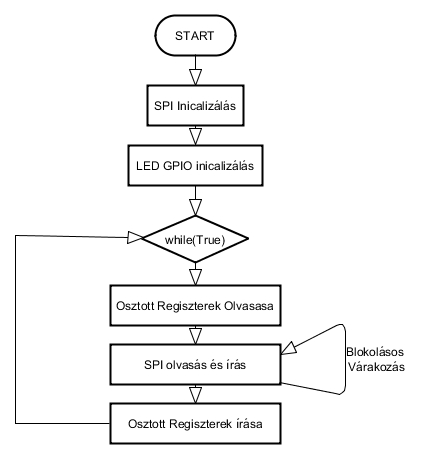
A program tartalmaz három TCP szervert is, amelyek a megadott port számon várják a kéréseket, minden kérés beérkezte után elküldik a pufferekben tárolt adatokat a kérést küldő kliensnek és kiürítik a puffereket.



Kép. 3.35 ZYBO Core0 program folyamat árbája

#### Spartan3e FPGA fejlesztőlap

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



Kép. 3.36 Spartan3e500, MicroBlaze szoftver Folyamat ábrája

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

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

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

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

#### Kommunikációs protokollok

##### SPI kommunikációs protokoll:

A Zybo mester míg a Spartan szolga egységként működik, a kommunikáció szinkron típusú, az órajel frekvenciája 1.56 MHz. Miközben a mester adatokat küld a szolga irányába a szolga egység is továbbit párhuzamosan adatokat a mester irányába. Az spi legkisebb csomagja minimum 32bit. Az spi kommunikáció a Kép. 3.35 látható SPI csomagszerkezeten keresztül történik.

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

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

s32addresReg; - cél regiszter szoftveres címe

s32dataReg; - cél regiszter adata

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

Az inMag0 tartalmazza azokat az adatokat, amelyek segítségével betudjuk konfigurálni a szabályzókat, és a paramétereket tudjuk frissíteni. Az Spi kommunikáción keresztül tudjuk beállítani a kirajzoláshoz szükséges adatregisztereknek az olvasási frekvenciáját. A Spartan rendszerben megtalálható MicroBlaze szoftvere, blokkolással várakozik az SPI csomagok érkezésére. A blokkolásból kikerülve kiolvassa az osztott regiszterek tartalmát és a bejött adatokat beírja más osztott regiszterbe, majd ismét blokkolásba kerül. Így tudjuk a Zybo rendszerből változtatni a mintavételezési frekvenciát anélkül, hogy bármit is kellene mordosítani a MikroBlaze szoftverében. A zybo rendszer az adatokat egy frissítési periódus késéssel kapja meg. A Zybo rendszerben minden 0.005s ként érkezik egy megszakítás, amelynek kiszolgálásakor a rendszer begyűjti az adatokat az érzékelőktől és küld egy SPI csomagot a Spartan rendszernek.

Az alább szemléltetett ábrán látható a SPI csomagban megtalálható 32bites változok. Az outMag0 struktúrát a Spartan rendszer küldi a Zybo fele, míg a inMag0-t a Zybo küldi párhuzamosan a Spartan rendszernek. Az inMag0 ban találhatók meg a szabályozók előirt értékei, és a konfigurációs beálltások.

**outMag0**

s32USebesseg0;

s32UPozicio0;

s32SebessegPozicio0;

s32AktPozicio0;

s32SzurtSebessegPozicio0;

s32AktSebesseg0;

s32SzurtSebesseg0;

s32eSebeseg0;

s32n20;

s32n30;

s32n40;

s32n50;

s32n60;

**inMag0**

s32Config0;

s32RefPos0;

s32RefSeb0;

s32Q0\_0;

s32Q1\_0;

s32Q2\_0;

s32Ts0;

s32TsL0;

s32Egyeb0;

s32PwmFrekREgH0;

s32PwmFrekRegL0;

s32sugarakAranya0;

s32qSav0;

##### Ethernet

A Zybo rendszeren megtalálható egy Ethernet csatlakozó 1GBit/s sebességre képes a programba beágyazva az lwip140 modult. A [9] adatlap alapján három TCP portot hozunk létre.

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

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

**A program:**

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

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

Az alhálózati maszk:

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

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

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

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

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

Következő lépésben aktualizáljuk a beállításokat a hardveren:

*init\_platform();*

*lwip\_init();*

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

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

*platform\_enable\_interrupts();*

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

**Szerver létrehozása:**

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

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

|  |  |  |
| --- | --- | --- |
|  | int start\_applicationSV1() | |
|  | { | |
|  | err\_t err; | |
|  | **unsigned** port = 9; | |
|  | pcb = tcp\_new(); | |
|  | **if** (!pcb) { | |
|  | xil\_printf("Error creating PCB. Out of Memory\n\r"); | |
|  | **return** -1; } | |
|  | err = tcp\_bind(pcb, IP\_ADDR\_ANY, port); | |
|  | **if** (err != ERR\_OK) { | |
|  | xil\_printf("Unable to bind to port %d: err = %d\n\r", port, err); | |
|  | **return** -2; } | |
|  | tcp\_arg(pcb, NULL); | |
|  | pcb = tcp\_listen(pcb); | |
|  | **if** (!pcb) { | |
|  | xil\_printf("Out of memory while tcp\_listen\n\r"); | |
|  | **return** -3; } | |
|  | tcp\_accept(pcb, accept\_callbackSV1); | |
|  | xil\_printf("TCP echo server started @ port %d\n\r", port); | |
|  | **return** 0; | |
|  | | } |

Program. 3.6.2 Server létrehozása C kód

### Feladatok Elosztása

#### Zybo fejlesztőlap

A Core0 processzor feladatai között szerepel a megszakítások lekezelése. A legfontosabb 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 érkeznek és a GPS adatait tartalmazza. Az adatok begyűjtése után elindítja a matematikai modell kiszámítását, amely a Core1 processzoron történik. Az Ethernet kommunikációhoz Szükséges Socketeket is kezeli.

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

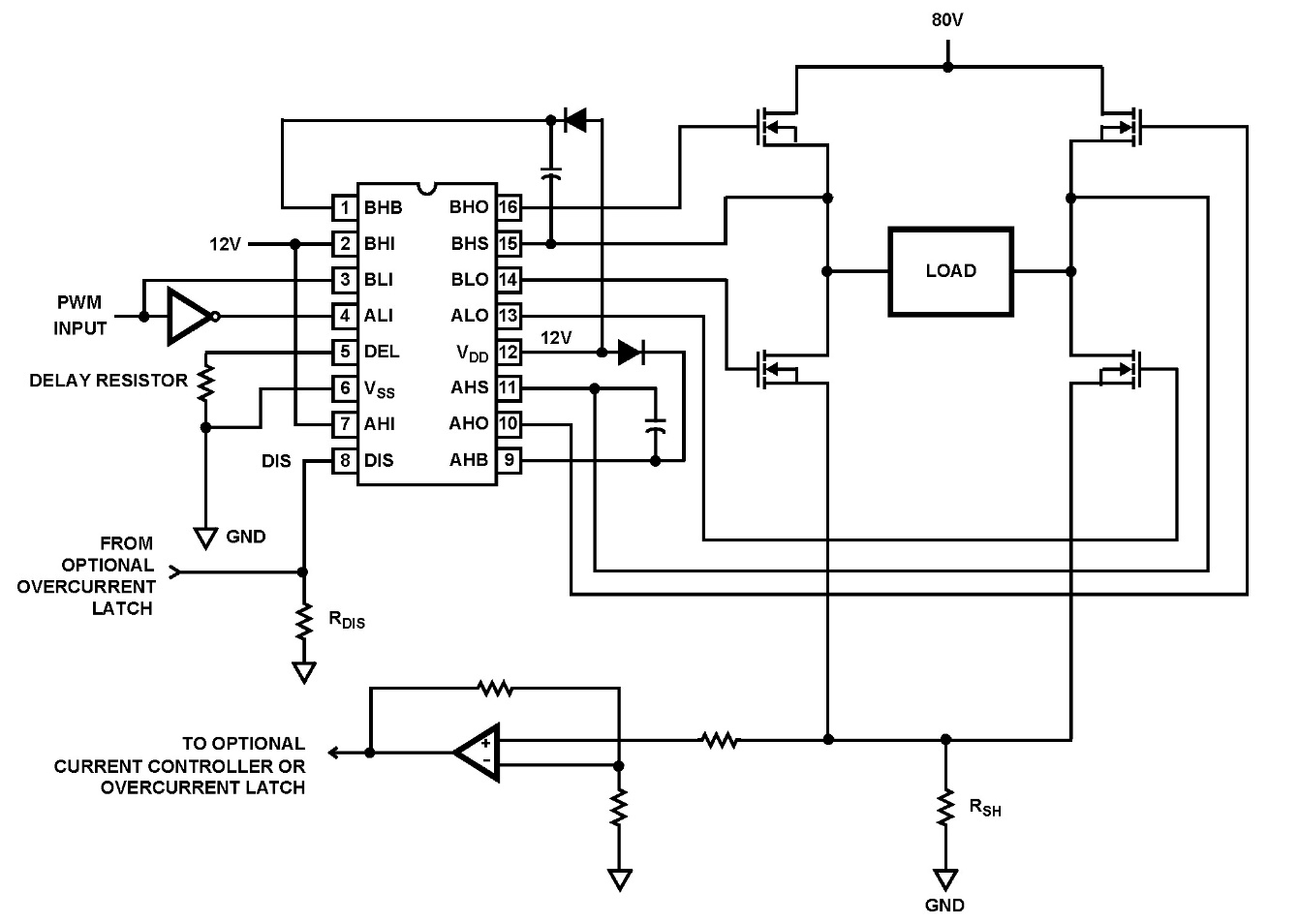
#### Spartan fejlesztőlap

Feladata kezeli a SPI kommunikáción beérkező adatokat és eljuttatja a megfelelő osztott regisztereken, valamint az visszaküldi a szabályozók adatait a Zybo rendszernek.

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

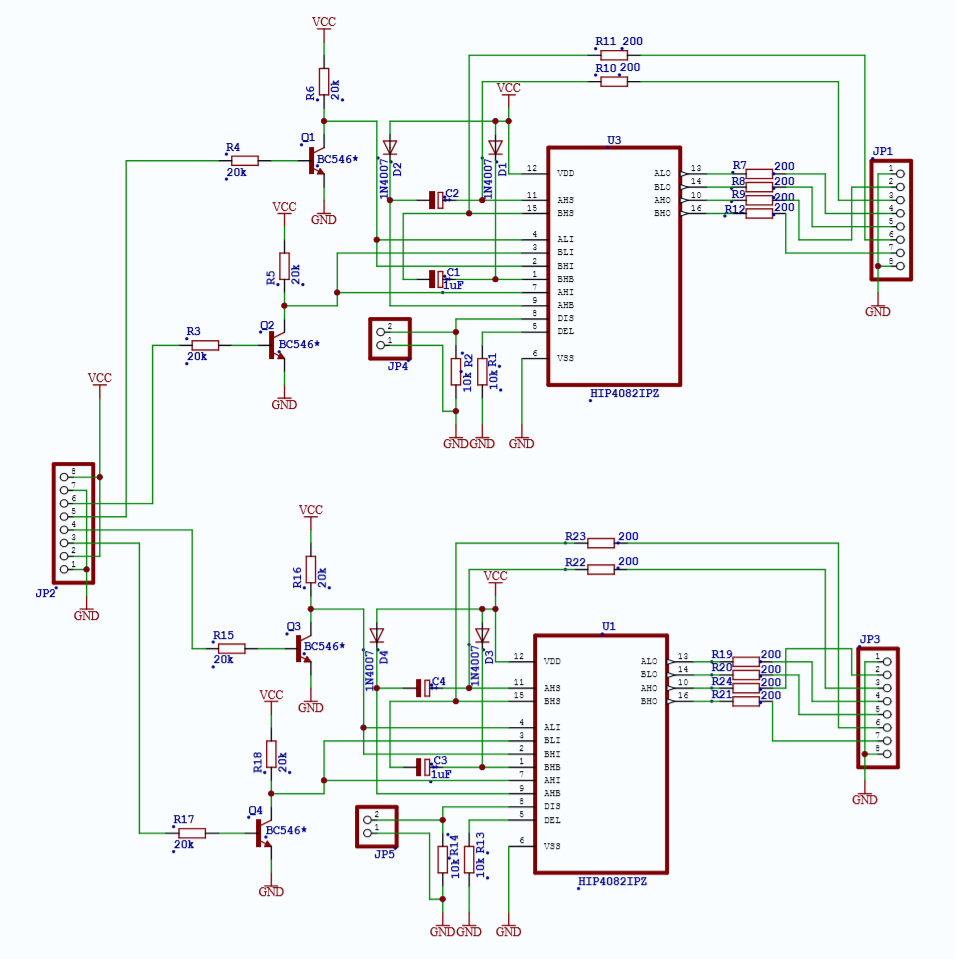
## Teljesítmény elektronika

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



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

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

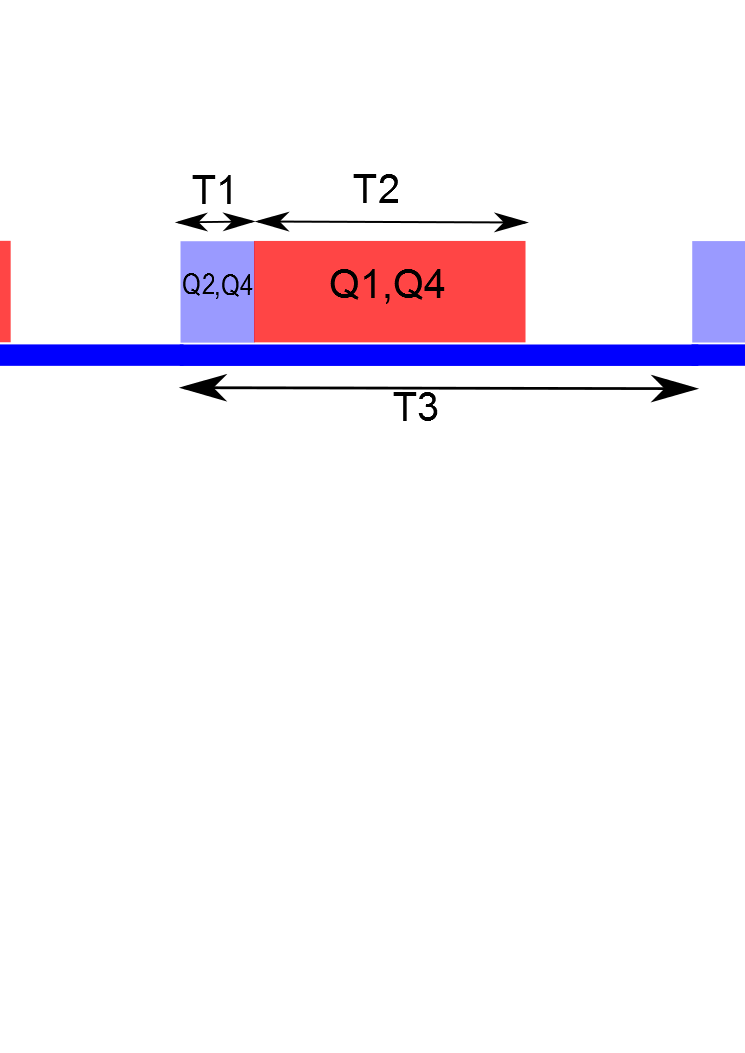


Kép. 3.38 Két hídvezérlő áramkör kapcsolási rajza HIP4082 integrált áramkörrel megvalósítva

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

A [10] adatlap alapján ALI, AHI, BLI, BHI bemenetek vannak, amelyek segítségével vezérelni tudjuk a tranzisztorokat. Az alsó két tranzisztor az ALO, BLO prioritást élveznek a felső AHO, BHO tranzisztorokkal szemben. Ami abból áll, ha bekapcsoljuk az alsó tranzisztorokat, akkor kikapcsolja a felső tranzisztorokat, ha azok bekapcsolt állapotban vannak. Ha kikapcsolt állapotban vannak és be szeretnénk kapcsolni. akkor nem engedi a bekapcsolásukat.

A Kép. 3.42 látható a pwm jel egy periódusának a felbontása. A pwm jel alapfrekvenciája 700Hz, így a T3=1,42ms. T1=0.5%T3=71us. A pwm jel kitöltése Duty=T1+T2.



Kép. 3.39 PWM és a tranzisztorok kapcsolása

A kívánt kapcsolási sorrendet úgy tudjuk elérni, hogy, a DIS bemeneten áramkorlát vagy más védelmeket iktathatunk be a rendszerbe. A [10] adatlapban ajánlott megoldás szerint méri az áramot az ellenálláson keresztül, egy műveleti erősítő segítségével, amelyet nem invertáló erősítő alapkapcsolásban használ, így dönti el, hogy áramkorlátban van vagy nincs. A DIS bemenet logikai 1 ben van, ha 2.5V fölött van, illetve logikai 0, ha 1V alatt van.

A Kép. 3.43 látható a nem invertáló erősítő kapcsolás, az Ube bemeneti feszültség, az árammérő ellenállástól érkező feszültség. A kapcsolás erősítése:



Kép. 3.40 Nem invertáló erősítő forrás [14]

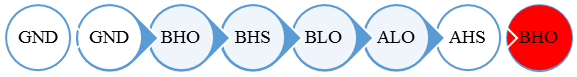
Ha , 10A szeretnénk az áramot korlátozni akkor a , ahol az a H hídon átfolyó áram.

Az erősítés: . A műveleti erősítős áramkorlát nem jelenik meg a megvalósításban, de a továbbfejlesztési lehetőség megvan rá. A Kép. 3.39 látható JP4 és JP5 jumper segítségével elengedhetjük a DIS bennetet földhöz képest. így kiiktatva az áramkorlátot, de illeszthető hozzá a jumperek helyére Kép. 3.43 képen látható áramérő és áramkorlátozó kialakítás.

A Kép. 3.41 képen látható kapcsolási rajz tartalmaz két HIP4082 hídvezérlő áramkört, az áramkörre a bemenet a JP2 bemeneten történik, amelyen keresztül betápláljuk 12V feszültséggel és 4 PWM jelet segítségével meg tudjuk hajtani a két hidat. A jeleket szalagkábel segítségével csatoljuk az áramkörhöz. A szalagkábel 8 vezetékből tevődik össze, rendeltetésük szerint:

Kép. 3.41 Dupla hídvezérlő áramkor vezérlő jelei JP2 csatlakozó a Kép. 3.39-n.

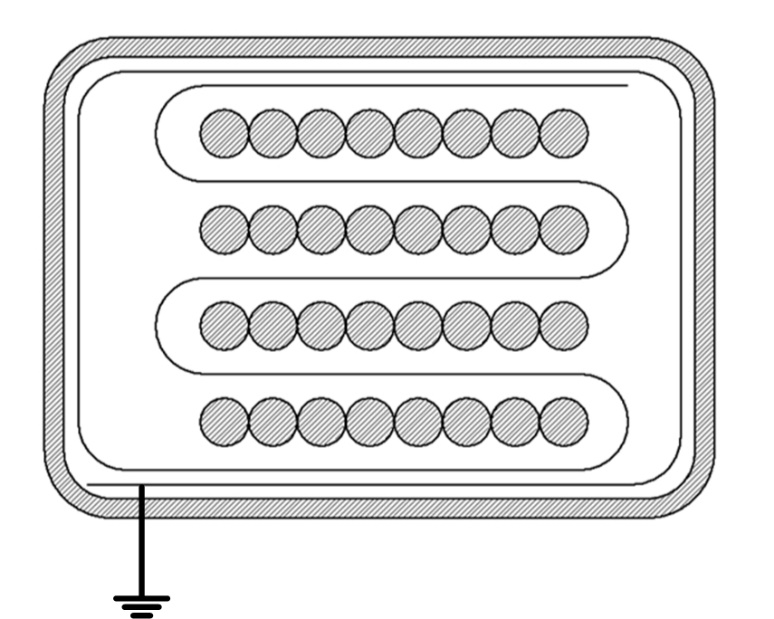
A pirossal megjelölt vezeték az 1 számú. A PWM4 egy 3,3V pwm jel, amely egy NPN (Q2) tranzisztoron keresztül kapcsolja az BLI, AHI bemeneteket, a tranzisztor a jelet megtagadja, ezért majd a FPGA pwm moduljába illesztünk egy tagadó, kaput, hogy semlegesítse egymást a két kapu. A PWM3 hasonlóak az előzőhöz,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



Kép. 3.42 H híd tranzisztorainak a Gate vezetékei

A Kép. 3.48 látható 4H-híd B és A dobozok tartalmaznak négy H hidat, a hidak kettesével rögzítve vannak egy rézlemezre, amelyeken keresztül tudunk vizet keringetni egy réz csővezeték segítségével így hűtve a tranzisztorokat. A tranzisztorok galvanikusan levannak választva a lemeztől egy elektromos szigetelő segítségedével, de ugyanakkor a szigetelő jó hővezető is. Egy hídban megtalálható tranzisztorok vezérléséhez szükséges vezetékek a Kép. 3.45 láthatok.

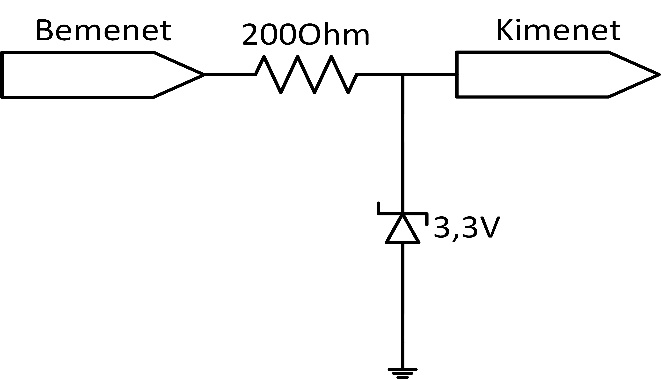
A négy híd vezérlésére négy (Kép. 3.45) szalagkábelre van szükségünk, amelyeket a Kép. 3.43 látható módon rendeztem el és árnyékoltam le a zavarforrásoktól.



Műanyag szigetelő

Alumínium fólia árnyékolás

Kép. 3.44 A négy Kép. 3.45 látható szalagvezeték jelenik meg a Buszvezetékben.



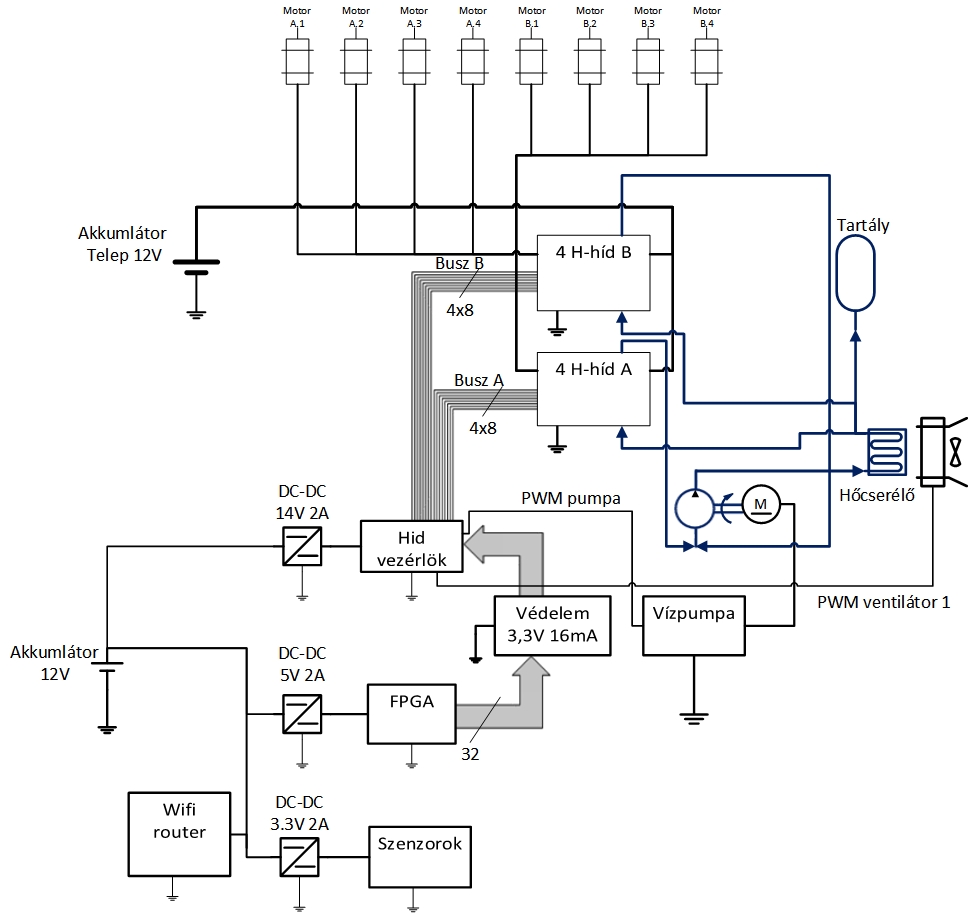
Kép. 3.43 FPGA kimentének a védelme

A szalagvezetékek között és körül alumínium fólia található, amelyek földpotenciálon vannak. A külső műanyag szigetelés véd a fizikai behatásoktól.

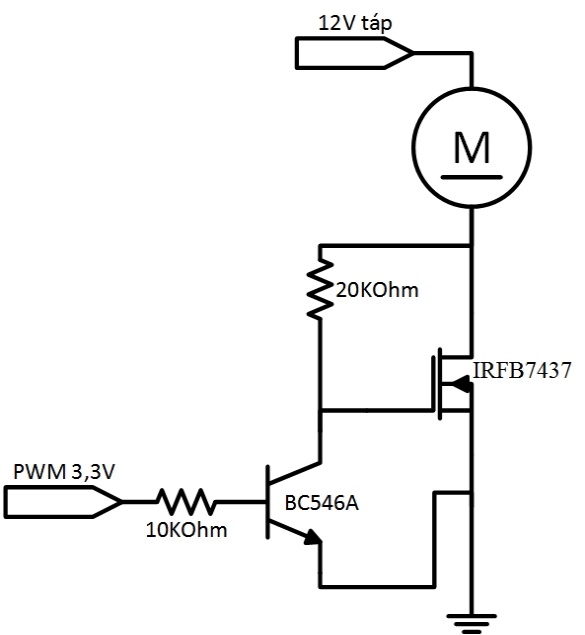
A Kép. 3.48 látható 32 bites buszvezeték, amely az FPGA rendszertől érkezik, és megtalálható benne a 8 motor hajtásához szükséges pwm beavatkozó jelek, amelyek az amplitúdója 3,3V. A busz szalagvezeték segítségével van kialakítva és megtalálható benne egy védelem is, amely megvédi az FPGA rendszert az esetleges visszahatásoktól. A védelem Kép. 3.44 látható, 3,3V ós zenre dióda segítségével történik, melynek feladata megakadályozza a 3,3V-ál nagyobb feszültségek az FPGA rendszerbe történő továbbhaladását. A diódával sorba van egy ellenállás is, amely az áramot korlátozza mivel az FPGA a bemenetén 16mA áramot visel el.

A Kép. 3.48 látható a robot energiaellátásának a terve. Az energiaellátás akkumulátorokkal fog történi, rendeltetésük szerint két csoportba oszthatók: egy 12V akkumulátor gondoskodik a rendszer digitális áramköreinek az ellátásáról. A digitális elemeket DC-DC konverteren keresztül táplálom be melyeknek a feszültsége állítható. A minimális feszültség ami szükséges a konvertereknek 3V, és a kimeneti feszültséget állíthatjuk 3-30V-ig. A konverterek maximálisan 2A tudnak leadni.

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



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

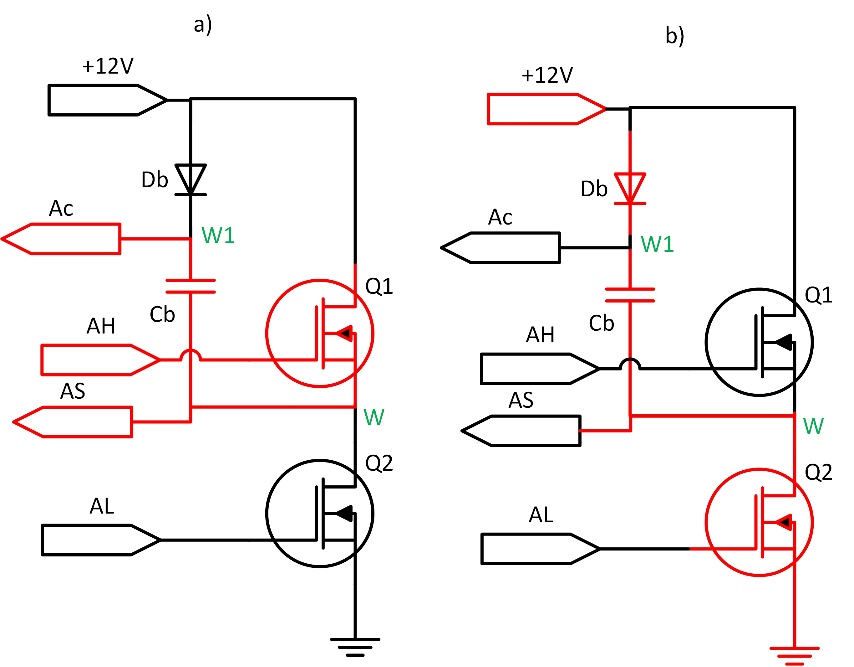


Kép. 3.46 Vízpumpa és a ventilátor motorjának vezérlő teljesítmény elektronikai kapcsolása

A vízpumpa és a ventilátorok motorja PWM jel segítségével van vezérelve egy N csatornás MOSFET tranzisztor segítségével, amelyet Kép. 3.48 ábra szemléltet.

### Bootstramp működése

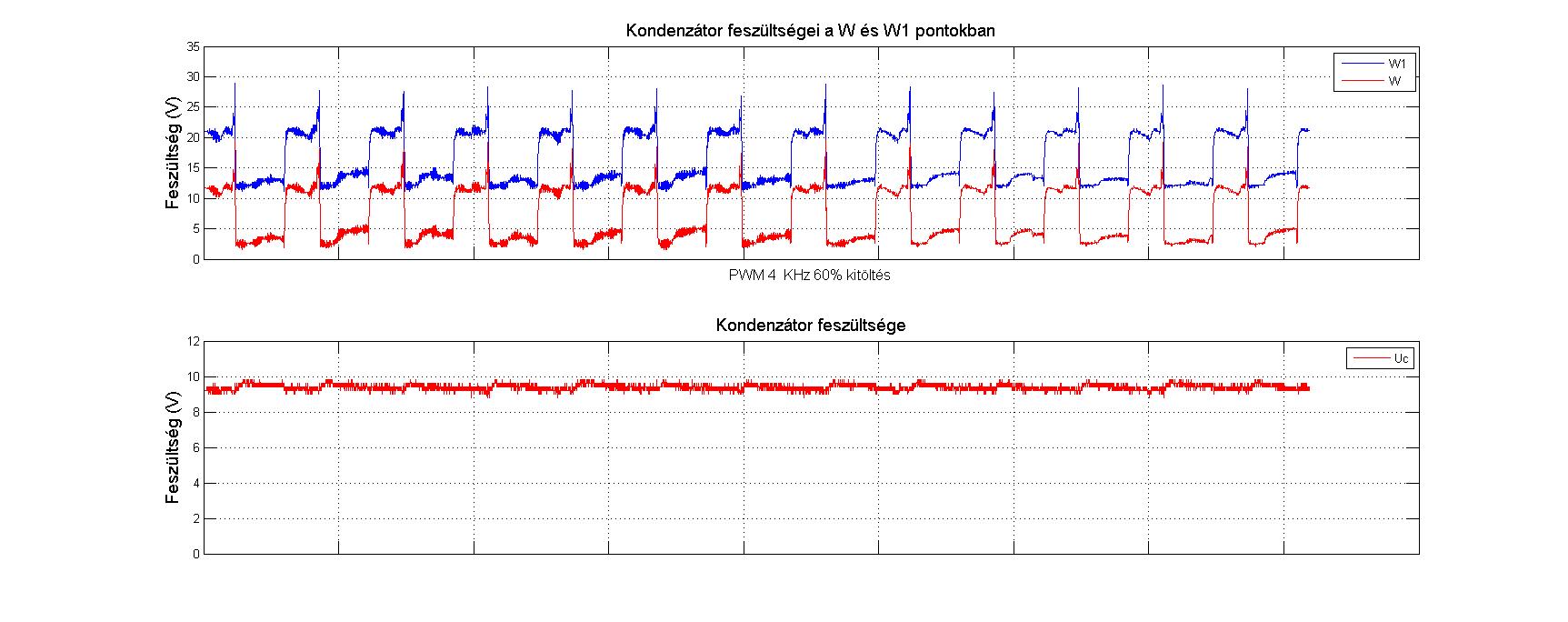
A [11] dokumentum alapján a bootstramp megoldás a Kép. 3.51 képen látható módon történik. A Cb kondenzátort töltjük fel a Db dióda segítségével azokban a pillanatokban, amikor a Q2 tranzisztor kinyitott állapotban van és a w potenciál elég alacsony, ahhoz hogy a Db dióda kinyisson és így feltöltve majdnem 12V feszültségre a kondenzátort.



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

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

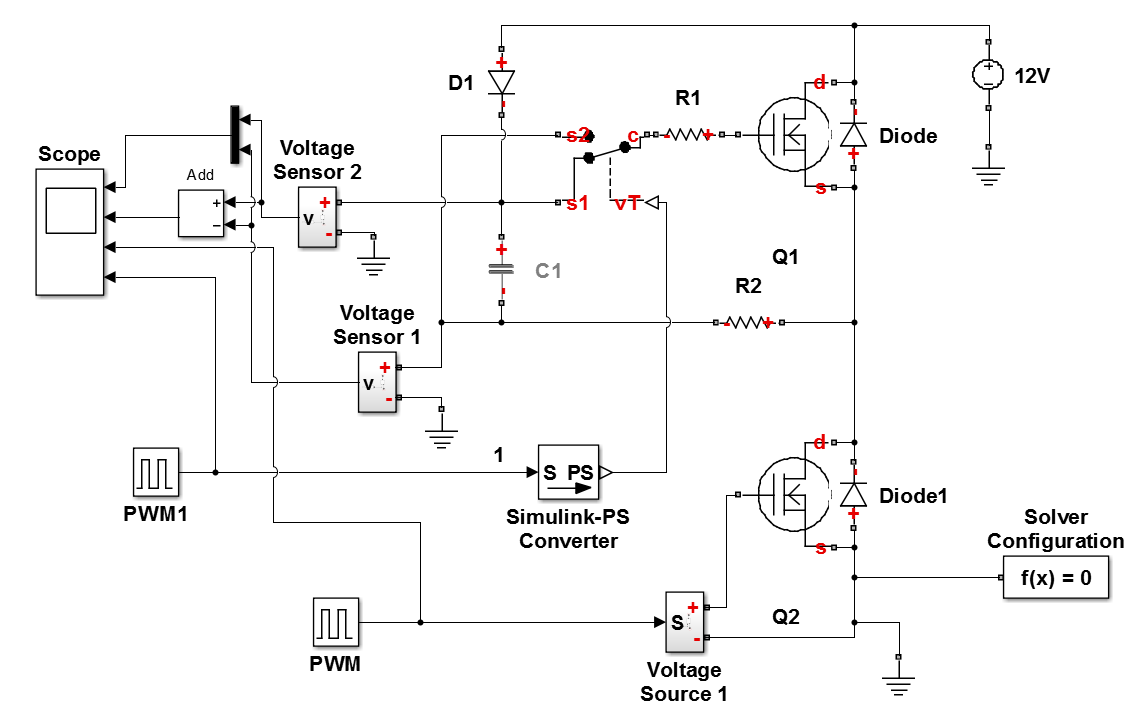
A Kép. 3.52 látható a mérése a Cb kondenzátor feszültségének, a méréseket oszcilloszkóp segítségével végeztem el és mentettem ki az adatokat, amelyeket majd ábrázoltam Matlab programmal.



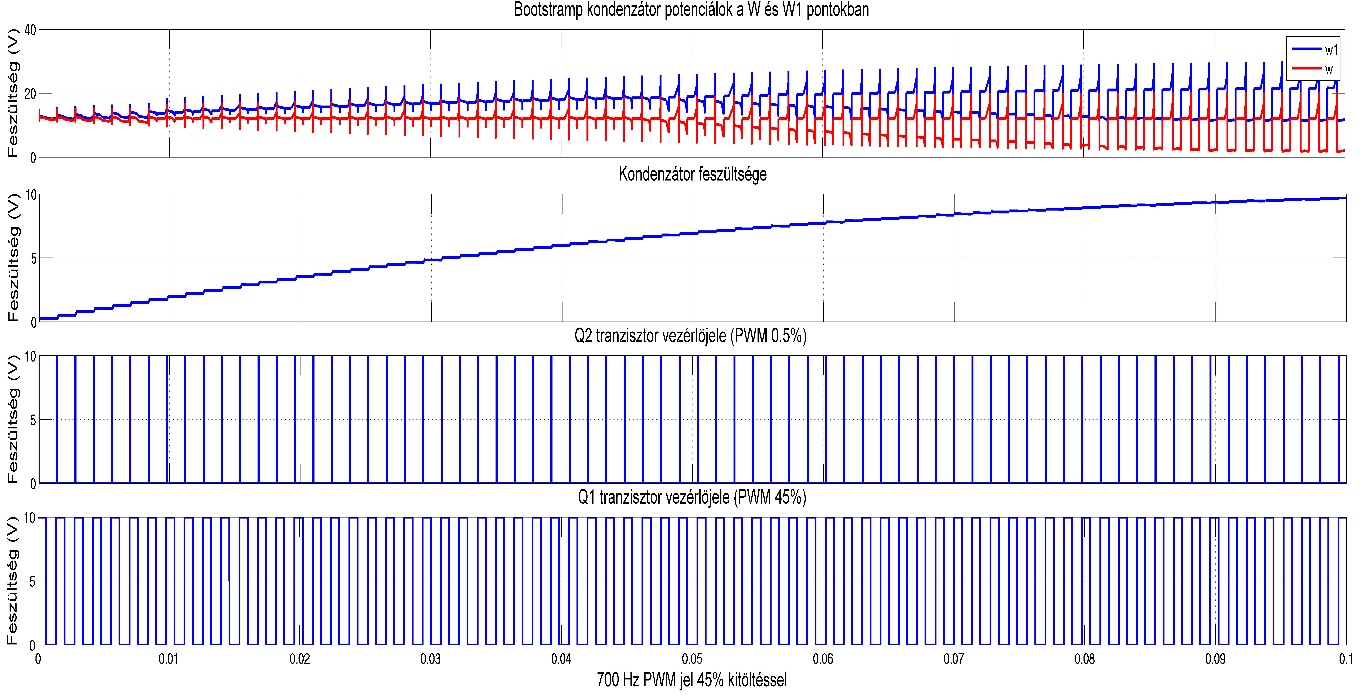
Kép. 3.48Bootstramp kondenzátor feszültsége a W és W11 pontokban

#### Szimuláció simulink környezetben

A szimuláció során előalítottam a Kép. 3.53 látható méréseknek megfelelő környezetet. Az eredmények szerint a Kép. 3.54 látható, ha a kondenzátor kezdeti feszültsége nulla, akkor a feszültség lassan kezd el felfutni rajta, emiatt a felső Q1 tranzisztor nem nyit ki teljesen és ezért veszteségek jelentkeznek rajta. A leg optimálisabb az lenne, ha egyszer feltöltenénk a kondenzátort és csak azután kezdenénk el a motor indítását. Amelyet úgy érhetünk el ha a Kép. 3.41 kapcsolási rajzot nézve a mindkét bemenetre 0V adunk. Ugyanis a tranzisztorokból kialakított tagadó kapu megtagadja és így a hídba mindkét alsó tranzisztor kinyitott állapotba kerül.



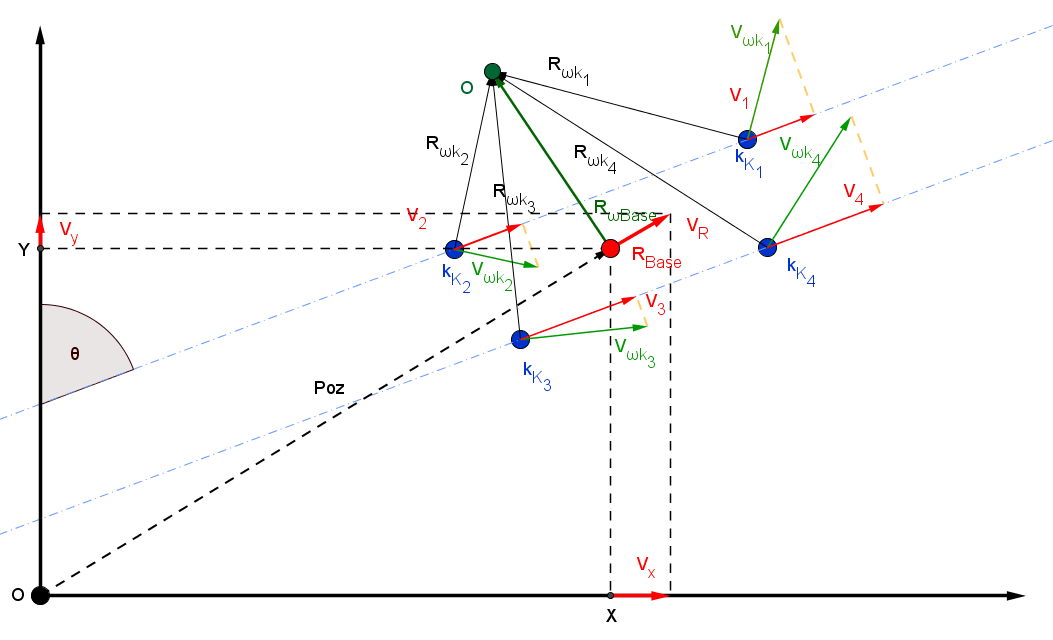
Kép. 3.49 Bootstramp működése, szimulációs modell MATLAB/SIMULINK környezetben



Kép. 3.50 Szimulációs eredmények Bootstramp

## Robot Modell

A [12] cikkben leírja egy négykerekű mobilis robot kinematikus és dinamikai modelljét. A rendszeren hasonlóképpen lehetne alkalmazni a tárgyalt modellt annyi eltéréssel, hogy ebben az esetben azok a pontok, amelyekben a robot érintkezik a talajjal nem szimmetrikusak.



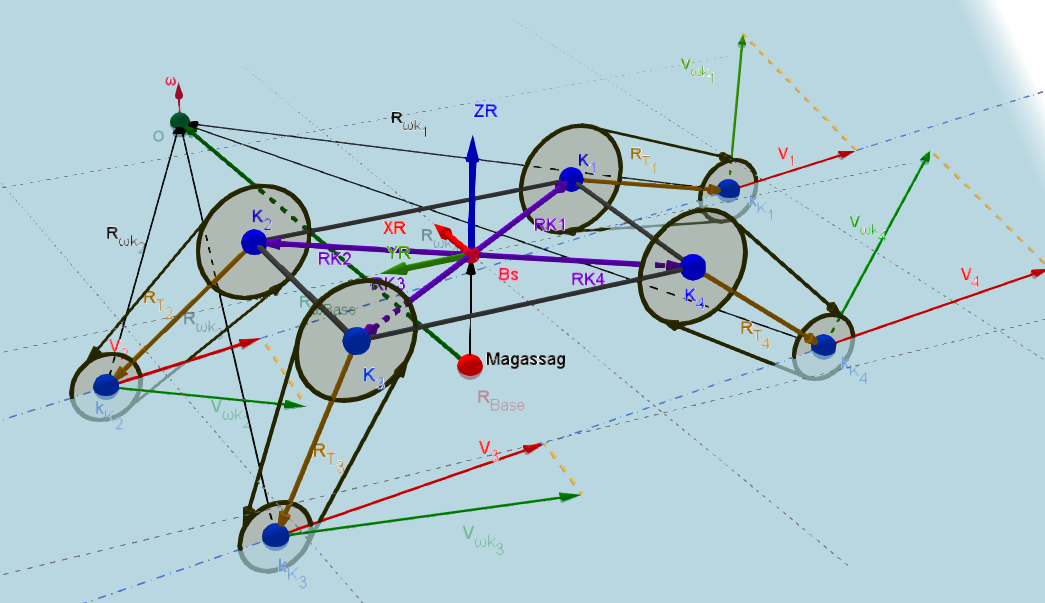
Kép. 3.51 Robot kerekek sebessége és a robot mozgásának viszonya

**Jelölések**:

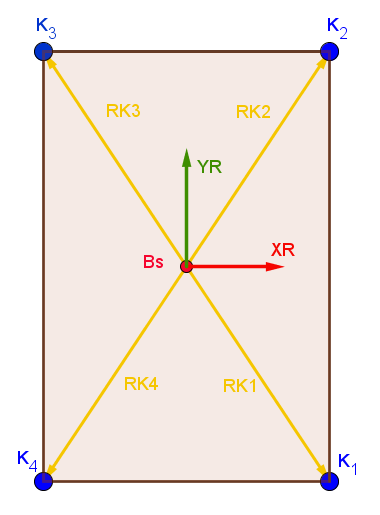
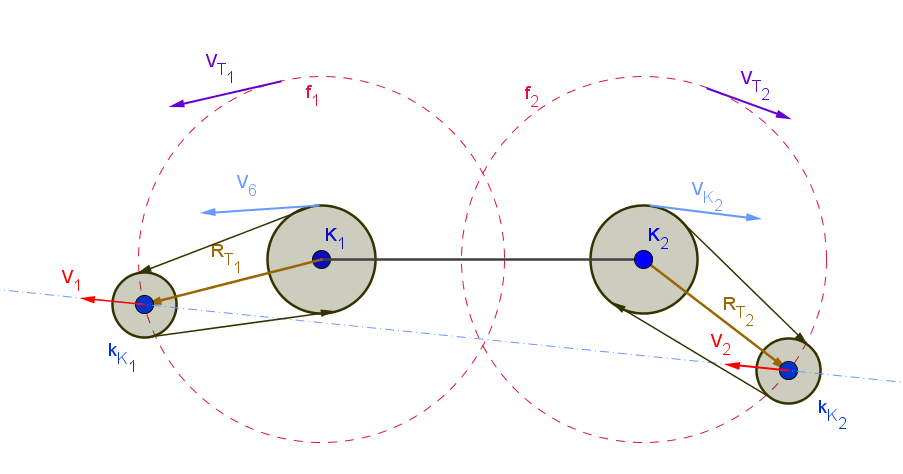
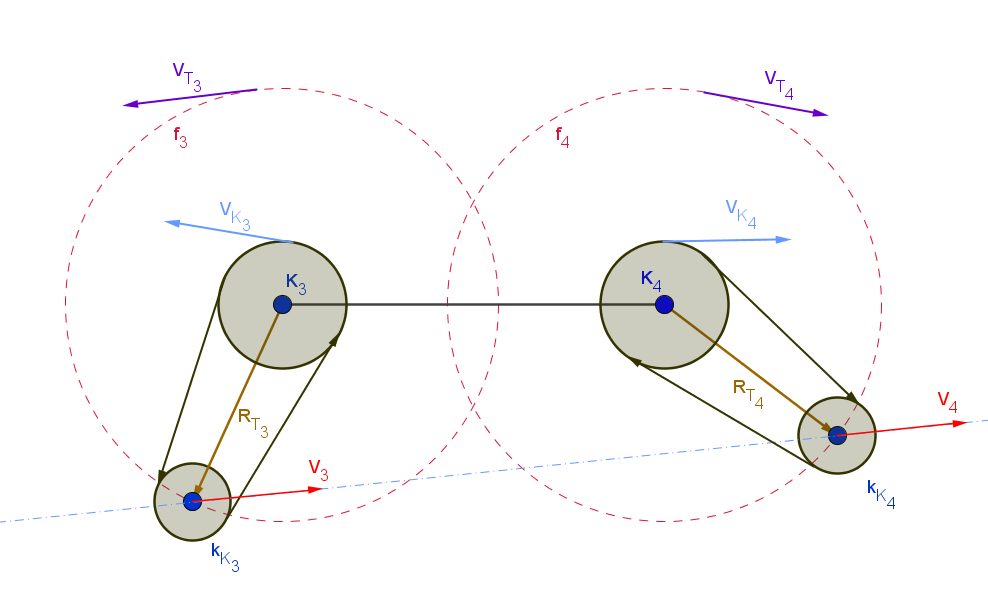
Szeretnénk, ha a robotunk egy adott körpályát írna le egy pont körül egy adott sebességgel. Jelen esetben O pont körül és, szögsebességgel.

Az [12] alapján felírhatók a következő összefüggések a vektorok között:

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



Kép. 3.52 Robot 3D vektorábrája



Kép. 3.53 Oldalnézetek és Felülnéztet, jelölések szemléltetése

# Robot Mechanikai Felépítése

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

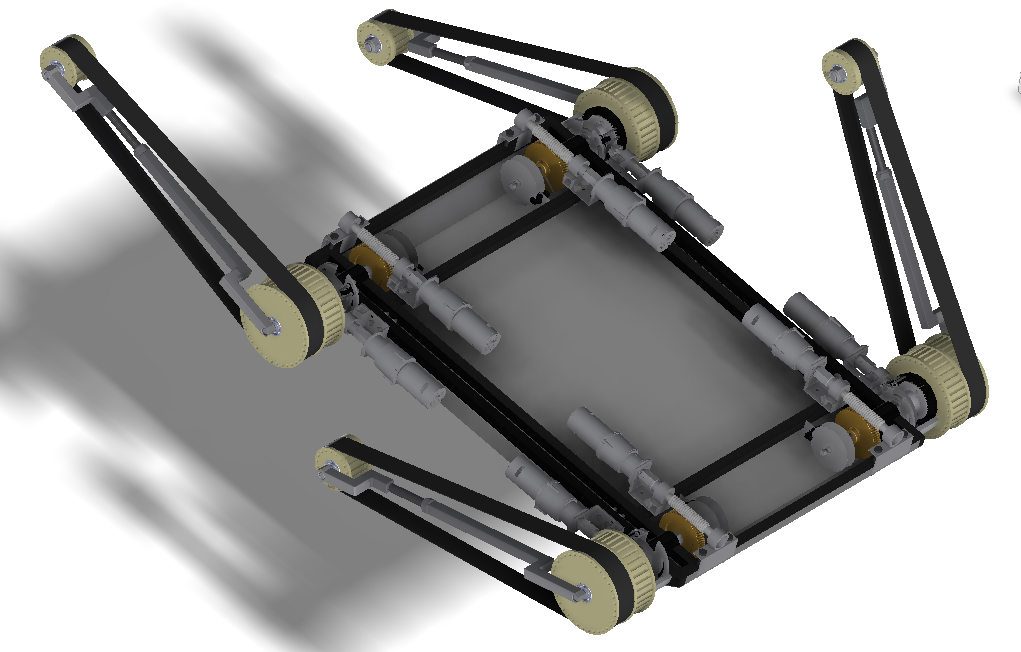
DC motorok betáplálási feszültsége: 12V, maximális terhelés alatt 10A áramot is felvehet.

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

* Nagy nyomaték kifejtésére képesek, jelen esetben 40:1 az áttétel arány. Egy motor által kifejtett nyomaték névlegesen 3-4Nm között van így a karok végén tengelyre kifejtett nyomaték elérheti a 60Nm-t.
* A terhelés nem képes visszafele hajtani, mert a mechanizmus lezárja, így akár a motort teljesen ki is kapcsolhatjuk, ha nem szeretnénk megváltoztatni a talp pozícióját.

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

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



Kép. 4.1 Robot vázának Inventoros 3D Képe

ForgóTalp1

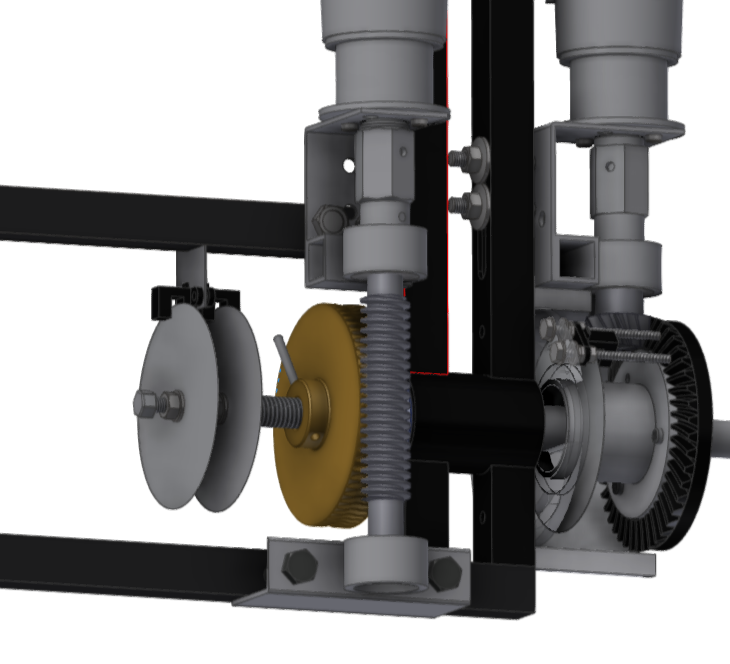
ForgóTalp4

ForgóTalp3

ForgóTalp2

KisKerék

NagyKerék



Null átmenti érzékelő tárcsája

Inkrementális szenzor tárcsa

# Elért eredmények, magvalósítások:

* Autodesk Invnetorban megterveztem a mechanikai rendszert
* A mechanikai rendszert 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 generátorban való szimulációja
* H-hídak megépítése, vízhűtés kivitelezése
* Grafikus vezérlőfelület elkészítése
* Nyáktervezés Altium tervező programban

# Következtetések:

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

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

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

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

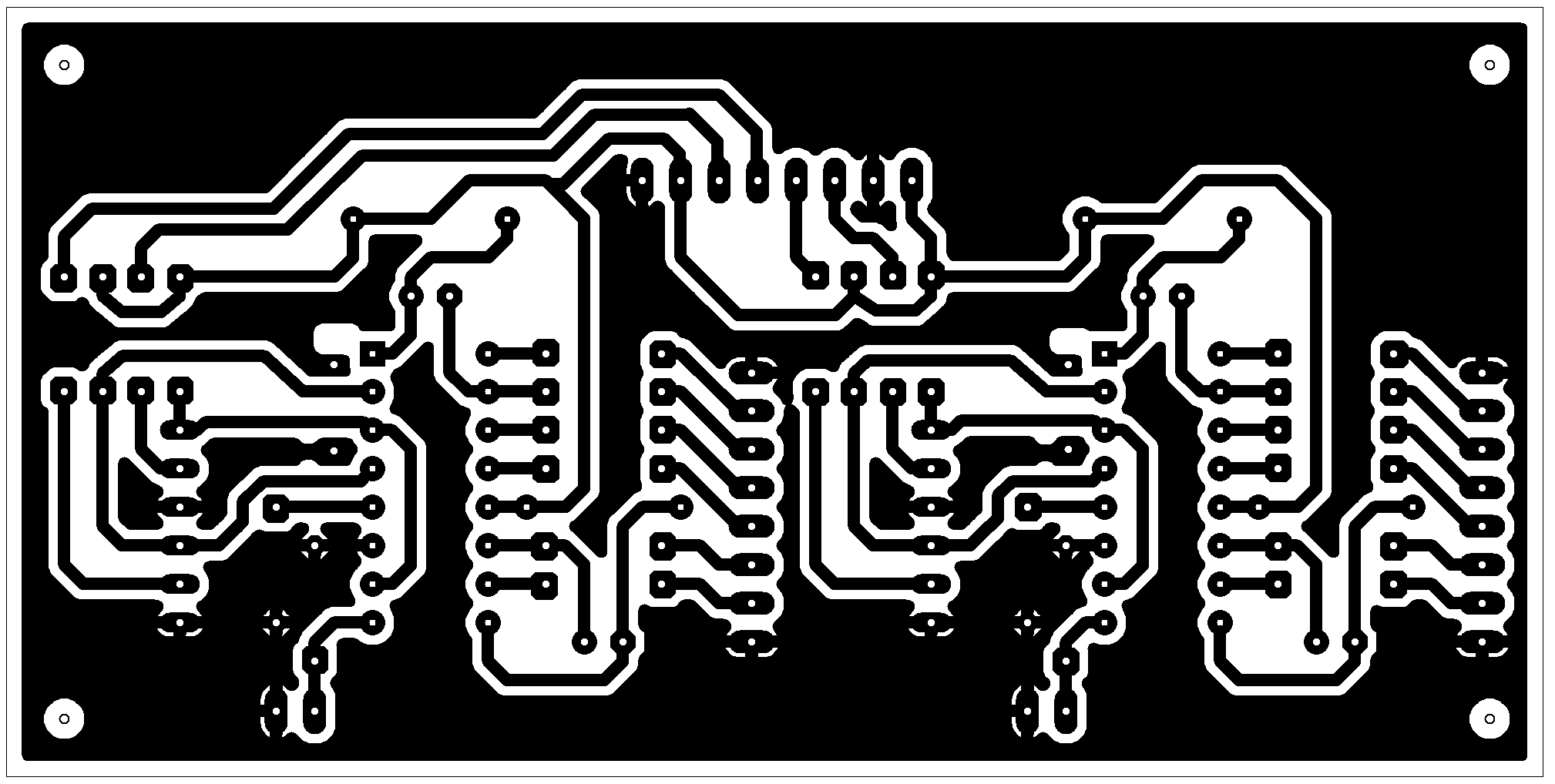
Az inkrementális szenzorok, amelyeket én készítettem, beváltak, nagyon olcsón tudtam előállítani a piaci árhoz viszonyítva. Fejlesztés szempontjából a tárcsák felbontása is növelhető lenne a tárcsák átmérőjének megnövelésével és a jobb minőségű lézeres nyomtató használatával.

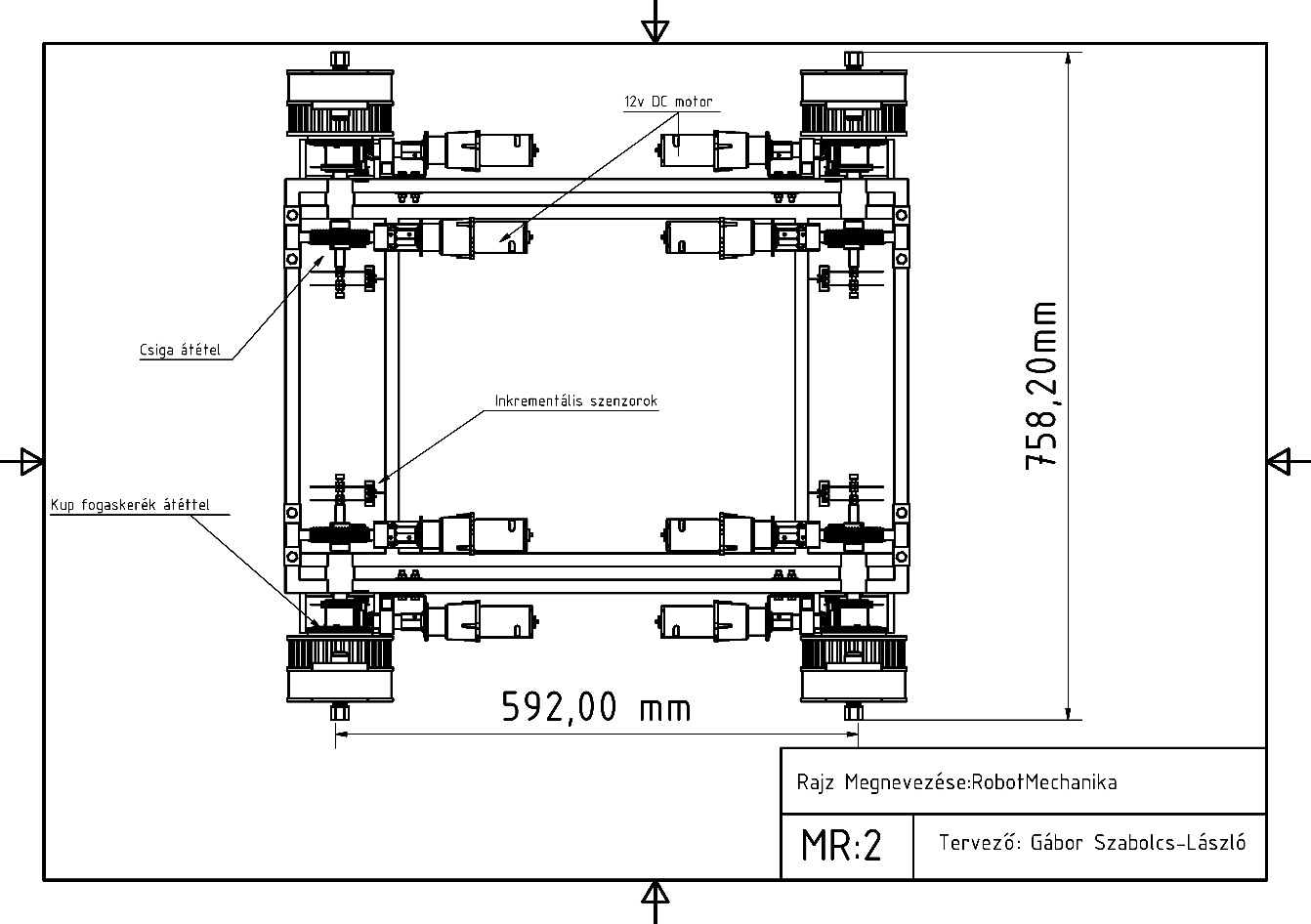
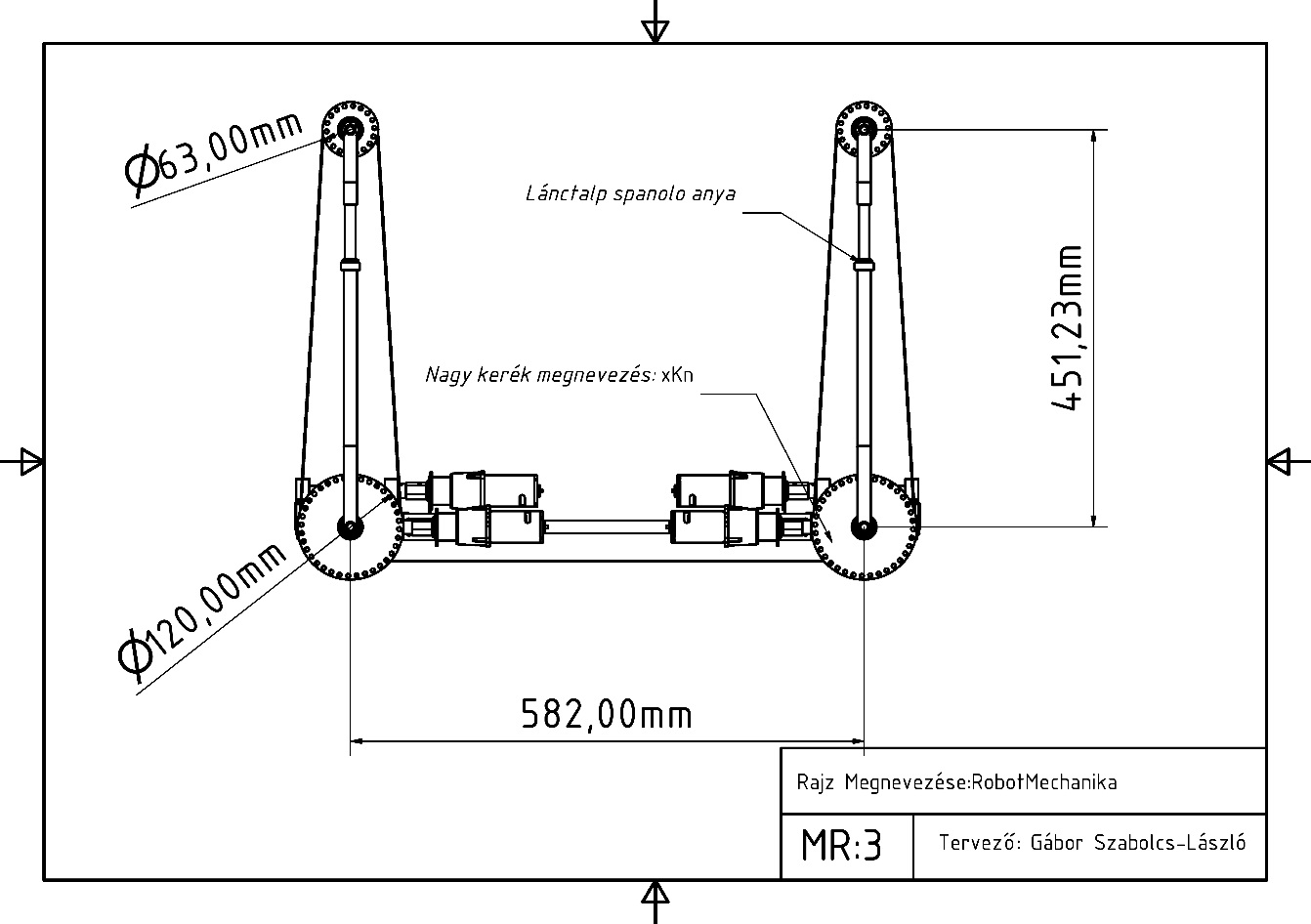
Összességében tekintve a rendszer nagyon jó szoftver és az elektronika fejlesztésére. A piaci ár alatt van jóval a rendszer előalításához szükséges pénz ahhoz képest, hogy ha egy kész rendszert kellett volna megvásárolni, de olcsó dolgoknak is ára van nem állná meg olyan jól a terepen a helyét.

# Bibliográfia

|  |  |
| --- | --- |
| [1] | D. K. K. D. R. P. S. Prof. Vikas Gupta, "Efficient FPGA Design and Implementation of Digital PID Controllers in Simulink," 2013. |
| [2] | M. Lőrinc, Irányítástechnika, Kolozsvár: Scientia, 2009. |
| [3] | xilinx, „http://www.xilinx.com/,” [Online]. Available: http://www.xilinx.com/support/documentation/sw\_manuals/xilinx14\_5/sysgen\_gs.pdf. [Hozzáférés dátuma: 26 01 2015]. |
| [4] | R. T. R. G. Rajesh Nema, „Design & Implementation of FPGA Based On PID Controller,” 2013. |
| [5] | „http://www.ms.sapientia.ro/elektronika,” [Online]. Available: http://www.ms.sapientia.ro/elektronika/fileok/jelerzekelok/szt\_lab08\_inkrementallis\_ado.pdf. |
| [6] | A. G. K. Krisztián LAMÁR, „IMPLEMENTATION OF SPEED MEASUREMENT FOR ELECTRICAL DRIVES EQUIPPED WITH QUADRATURE ENCODER IN LabVIEW FPGA,” 2013. |
| [7] | M. Lőrinc, „http://www.ms.sapientia.ro/~martonl/index.htm,” [Online]. Available: http://www.ms.sapientia.ro/~martonl/Docs/Labs/IRI\_L1.pdf. [Hozzáférés dátuma: 11 6 2015]. |
| [8] | M. Lőrincz, „http://www.ms.sapientia.ro/,” [Online]. Available: http://www.ms.sapientia.ro/~martonl/Docs/Lectures/Holtidos\_Folyamatok\_Iranyitasa.pdf. [Hozzáférés dátuma: 10 6 2015]. |
| [9] | I. Inc, „www.olimex.com,” [Online]. Available: https://www.olimex.com/Products/Modules/Sensors/MOD-MPU6050/resources/RM-MPU-60xxA\_rev\_4.pdf. [Hozzáférés dátuma: 11 6 2015]. |
| [10] | xilinx, „http://www.xilinx.com,” [Online]. Available: http://www.xilinx.com/support/documentation/application\_notes/xapp1026.pdf. [Hozzáférés dátuma: 10 6 2015]. |
| [11] | intersil, „http://www.intersil.com/,” [Online]. Available: http://www.intersil.com/content/dam/Intersil/documents/hip4/hip4082.pdf. [Hozzáférés dátuma: 11 6 2015]. |
| [12] | S. labs, „http://www.silabs.com/,” [Online]. Available: http://www.silabs.com/Support%20Documents/TechnicalDocs/AN486.pdf. [Hozzáférés dátuma: 11 6 2015]. |
| [13] | D. P. I. J. K. Kozłowski, „Modeling and control of a 4-wheel skid-steering”. |
| [14] | M. Trojnacki, „Dynamics Model of a Four-Wheeled Mobile Robot for Control Applications – A Three-Case Study,” in *Intelligent Systems'2014*, Springer, 2014, p. 111. |
| [15] | L. Lajos, „http://www.ms.sapientia.ro/,” [Online]. Available: https://moodle.sapidoc.ms.sapientia.ro/pluginfile.php/2771/mod\_resource/content/1/Losonczi\_Lajos\_-\_Analog\_Aramkorok\_3\_V1.pdf. [Hozzáférés dátuma: 11 6 2015]. |

# FÜGGELÉK





Kép. 8.1 A mechanikai rendszer műszaki rajza