

**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:

**Dr. Brassai Sándor Tihamér
egyetemi adjunktus**

Végzős hallgató:

Gábor Szabolcs-László

2015

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREŞ
SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

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

PROIECT DE DIPLOMĂ

Coordonator științific:
Dr.ing. Brassai Sándor Tihamér

Absolvent:
Gábor Szabolcs-László

2015

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREŞ
Specializarea: AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

Viza facultății

TEMĂ PROIECT DE DIPLOMĂ

Conducătorul temei:
Brassai Sándor Tihamér

Candidat Gábor László Szabolcs
Anul absolvirii: 2015

a) Tema proiectului de diplomă: *Proiectarea și implementarea unui robot mobil utilizând resurse hardware bazate pe circuite FPGA*

b) Problemele principale care vor fi tratate în proiect:

- Noțiuni privind implementarea regulatorului PID în circuit FPGA
- Noțiuni privind co-simulare hardware

c) Desene obligatorii:

- Schema bloc a aplicației
- Schema bloc a regulatorului PID implementat în System Generator
- Schema bloc a sistemului de reglare a poziției (vitezei) implementat în FPGA
- Schema bloc a circuitului de control în punte H a motoarelor de c.c.
- Schema logică a aplicației
- Schema electrică de alimentare a submodulelor sistemului robot

d) Softuri obligatorii

- Aplicație de comandă și monitorizare a parametrilor robotului mobil

Bibliografie recomandată:

[1] Roland SIEGWART, Illah R. NOURBAKHSH, Introduction to Autonomous Mobile Robots, MIT Press, 2011

[2] KRZYSZTOF KOZŁOWSKI, DARIUSZ PAZDERSKI, MODELING AND CONTROL OF A 4-WHEEL SKID-STEERING MOBILE ROBOT, Int. J. Appl. Math. Comput. Sci., 2004, Vol. 14, No. 4, 477–496

Termene obligatorii de consultații: - săptămânal

Locul practicii: Universitate

Primit la data de: 15. 05. 2014

Termen de predare: 26.06. 2015

Semnătura şefului de departament

Semnătura îndrumătorului științific

Semnătura candidatului

DECLARAȚIE,

Subsemnatul Gábor Szabolcs-László
absolvent al specializării Automatică și informatică aplicată

la **Facultatea de Științe Tehnice și Umaniste – Tîrgu Mureș , Universitatea „Sapientia” din Cluj-Napoca** certific prin prezenta că am luat la cunoștință de cele prezentate mai jos și că îmi asum, în acest context, originalitatea lucrării mele de licență/disertație cu:

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

coordonator Brassai Sándor Tihamér
prezentată în sesiunea 2015

La elaborarea lucrării de licență/disertație, se consideră plagiat una dintre următoarele acțiuni:

- ✓ 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 și referința 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 obținute sau a unor aplicații realizate de alții autori fără menționarea corectă a acestor surse;
- ✓ însusirea totală sau parțială 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 și indicarea referinței într-o listă corespunzătoare la sfârșitul lucrării;
- indicarea în text a reformulării unei idei, opinii sau teorii și corespunzător în lista de referințe 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 referințelor poate fi omisă dacă se folosesc informații sau teorii arhicunoscute, a căror paternitate este unanim acceptată.

Model tip a.

Declarație

Subsemnata/ul .. Gábor Szabolcs-László., absolvent(ă) al/a specializării Automatică și informatică aplicată , promoția...2015... cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Tîrgu Mureș,

Data:

Absolvent

Semnătura.....

Declarație

Subsemnata/Subsemnatul Brassai Sándor-Tihamér., funcția...șef lucrări., titlul științific...doctor... declar pe propria răspundere că Gábor Szabolcs-László absolventul specializării de Automatică și informatică aplicată . a întocmit prezenta lucrare cu îndrumarea mea.

Forma finală a lucrării a fost verificată de mine și aceasta corespunde cu cerințele de formă și conținut precizate de Consiliul Facultății de Științe Tehnice și Umaniste în baza reglementărilor Universității "Sapientia". Lucrarea/proiectul corespunde și cerințelor impuse de Legea Educației Naționale 1/2011 cu modificări ulterioare, Codului de etică și deontologie profesională a Universității Sapientia referitoare la furt intelectual.

Sunt de acord cu susținerea lucrării în fața comisiei de examen de licență/diplomă.

Tîrgu Mureș,

Data:

Semnătura îndrumătorului,

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

1 EXTRAS

Scopul lucrării este proiectarea unui robot mobil de teren, prezentarea elementelor necesare și implementarea sa. În lucrare accentul este pus pe proiectarea și implementarea componentelor electronice de control, implementarea sistemului senzorial, dar din punctul de vedere al înțelegерii funcționării întregului sistem este prezentată în ansamblu și partea mecanică a robotului mobil de teren proiectată integral de către autorul lucrării prin Autodesk Inventor.

Pe un șasiu masiv sunt atașate patru tălpi pivotante, care se pot rota cu 360 de grade față de șasiu. Pe tălpile pivotante sunt montate șenile care sunt puse în mișcare cu ajutorul motoarelor DC cu angrenaj de roți dințate conice.

Structura șasiului masiv este format din cadru metalic, cu profile din oțel, componentele sunt fixate între ele prin sudare. Structura cadrului, precum și sistemul sunt simetrice față de cele două axe.

În sistem sunt integrate 8 motoare DC cu ajutorul căroroare sunt puse în mișcare șenilele și se poate modifica unghiul tălpilor față de cadru. Motoarele sunt comandate prin punți H.

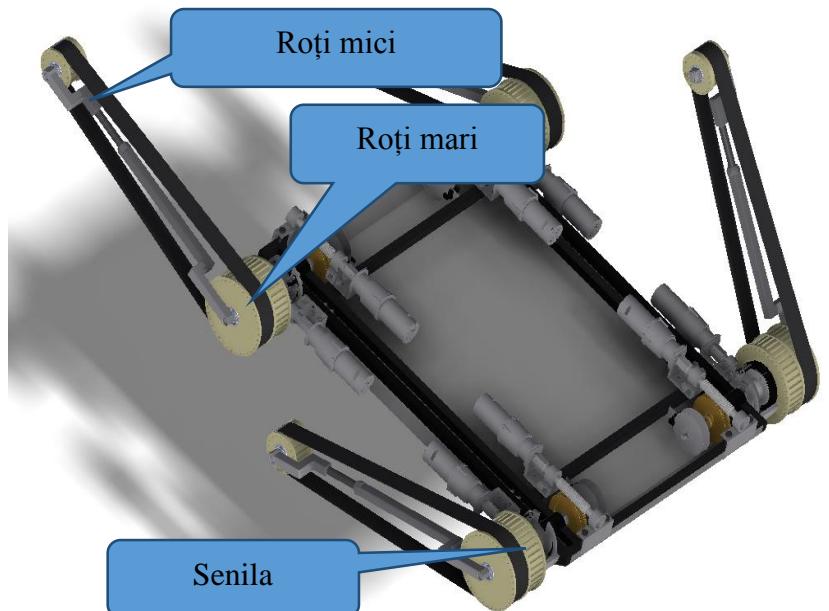


Fig. 1.1 Structura Robotului- Inventor 3D Foto

ALIMENTARE CU ENERGIE

Pe figura 1.2 se pot observa 8 punți H (modulele A și B conțin câte 4 punți H), fixate câte două pe o tablă din cupru. Circulația apei prin conductă cu scopul răcirii tranzistoarelor este realizată printr-o minipompă de apă. Tranzistoarele sunt separate galvanic de tablă, cu ajutorul unui izolator electric, cu toate acestea izolatorul are conductivitate termică bună. Căldura este extrasă prin intermediul unui sistem de răcire prin apă. Deoarece sistemul va funcționa pe teren, trebuie evitat ca praful să pătrundă în interiorul acestuia.

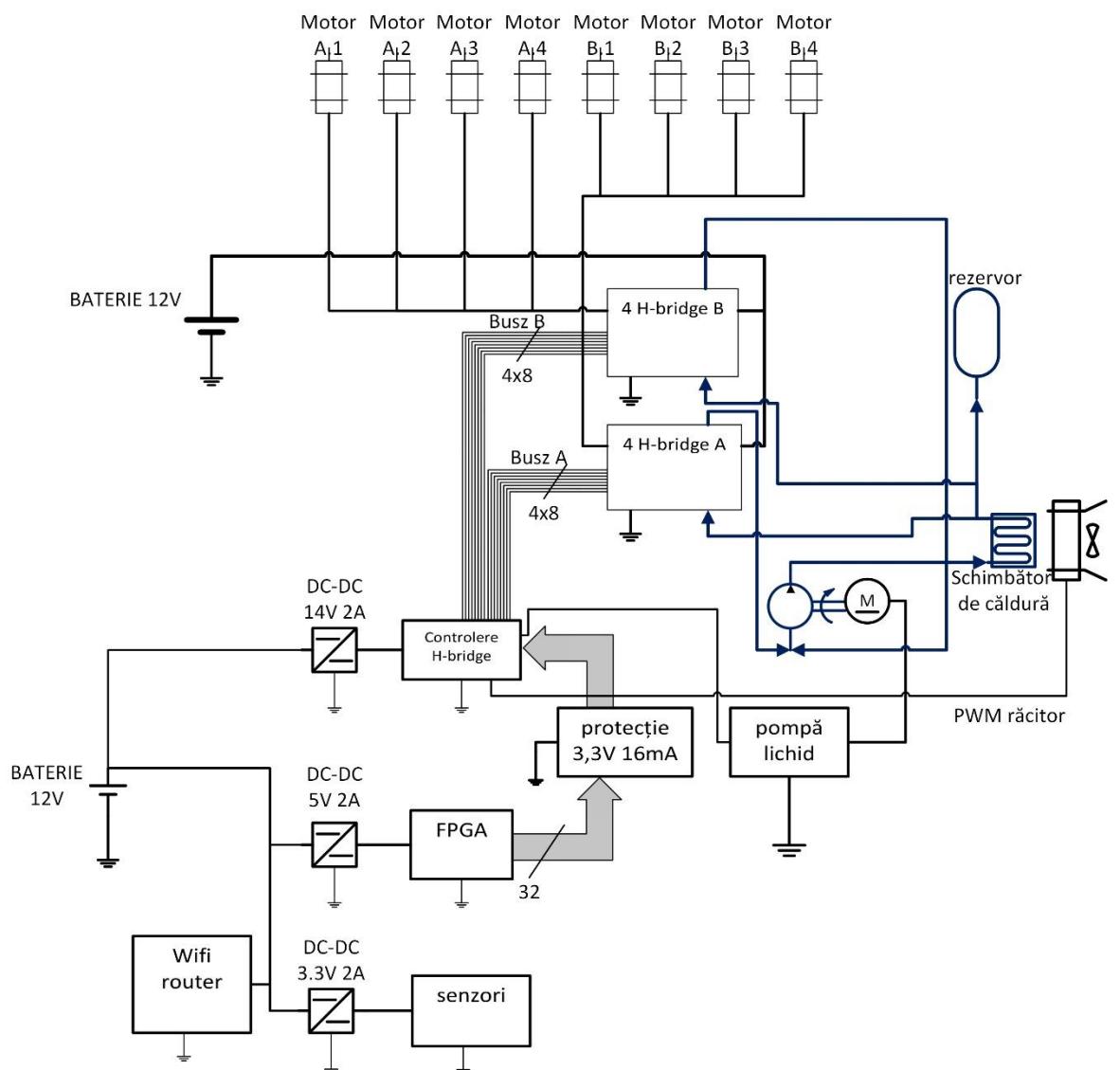


Fig. 1.2 Alimentarea cu energie electrică a robotului și structura sistemului de răcire

Pe figura 1.2 se poate vedea o magistrală de 32 biți (BUS), conectat la circuitul FPGA, și conține semnalele PWM (3,3 V amplitudine) pentru comanda celor 8 motoare de curent continuu. Pe magistrală este realizată și protecția circuitului FPGA. Protecția circuitului FPGA împotriva supratensiunilor dinspre punțile H este realizată prin diode zener de 3.3 V. Pe de altă parte rezistențele electrice conectate la ieșirile circuitului FPGA au rolul de limitare a curentului electric, deoarece curentul maxim suportat la ieșirile circuitului FPGA este de 16mA. Pe figura 3.48 este prezentat planul de alimentare cu energie a robotului. Alimentarea cu energie se va realiza cu acumulatoare, care în funcție de modulele pe care îi alimentează pot fi împărțite în două grupe: un acumulator de 12 V, care asigură alimentarea sistemului cu circuite digitale, respectiv a două surse de energie compusă din mai multe acumulatoare conectate paralel care alimenteză cu energie punțile. Elementele digitale sunt alimentate prin convertere DC-DC, cu tensiune reglabilă.

Pompa de apă și motorele ventilatoarelor sunt comandate cu ajutorul unui tranzistor MOSFET cu canal N.

STRUCTURA SISTEMULUI FPGA

Sistemul este structurat pe două plăci de dezvoltare FPGA. Una dintre ele este un sistem de dezvoltare ZYBO Zynq™-7000 Development Board cu putere de procesare mare, dar are un număr de ieșiri limitate. Placa cealaltă conține un circuit Spartan3e, cu mai puține resurse, dar are 120 de ieșiri. În circuitul Spartan este implementat sistemul de comandă a celor 8 motoare de curent continuu, care primește și prelucrează datele de la sistemul ZYBO. Sarcinile sistemului ZYBO sunt: colectarea datelor de la senzori (giroscop, modulul GPS) și comunicarea prin Ethernet.

Este necesară reglarea poziției a celor patru motoare ale robotului, regulatoare realizate în hardware reconfigurabil. De asemenea este necesară reglarea vitezei la patru motoare realizate cu regulatoare de tip PID, de asemenea implementate în hardware reconfigurabil. În bucla de control, feed-backul este realizat prin encoder incremental. Am folosit regulator de tip PID pentru reglarea vitezei și un regulator propriu implementat în hardware pentru reglarea poziției.

Regulatoarele au fost proiectate, realizate și simulate în System Generator. De asemenea testarea hardware s-a realizat prin efectuarea de măsurători cu System Generator prin co-simulare hardware.

Regulatorele de viteze și de poziție implementate în hardware au fost înglobate într-un nucleu IP. Parametrii regulațoarelor se pot defini prin registre. Conținutul registrelor este citită și scrisă de către procesorul MicroBlaze pe magistrala căreia este integrat nucleul IP. Acest procesor primește valorile de referință și parametri de acordare a regulațoarelor prin protocol de comunicare SPI de la sistemul ZYBO și le înscrie în registre. De asemenea, trimită în paralel datele buclelor de control (pozițiile și vitezele).

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 se pot prelua datele măsurate și se pot transmite comenzi pentru sistem, și CORE 1, care va avea ca sarcină calculul modelului matematic.

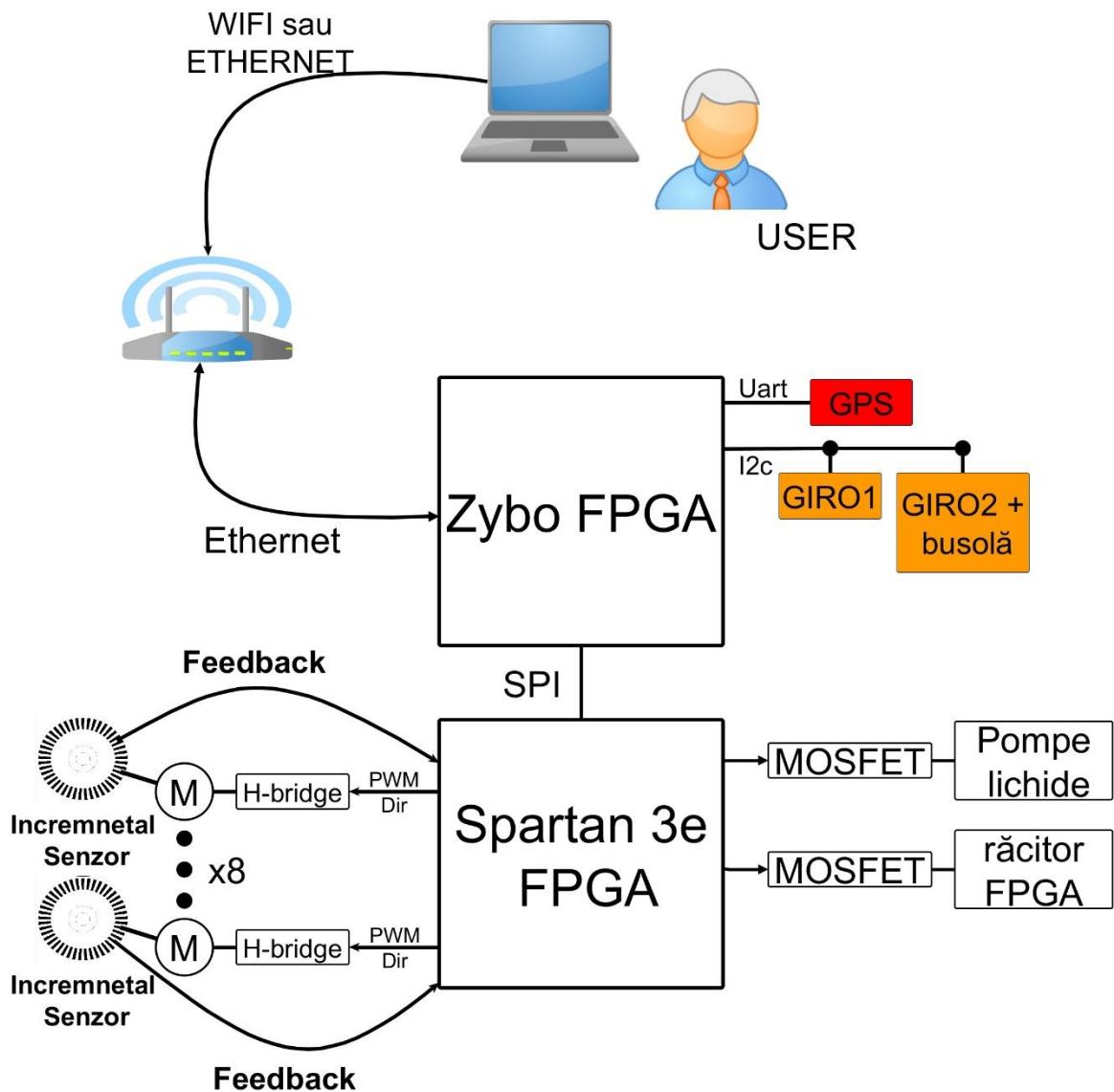


Fig. 1.3 Structura sistemului

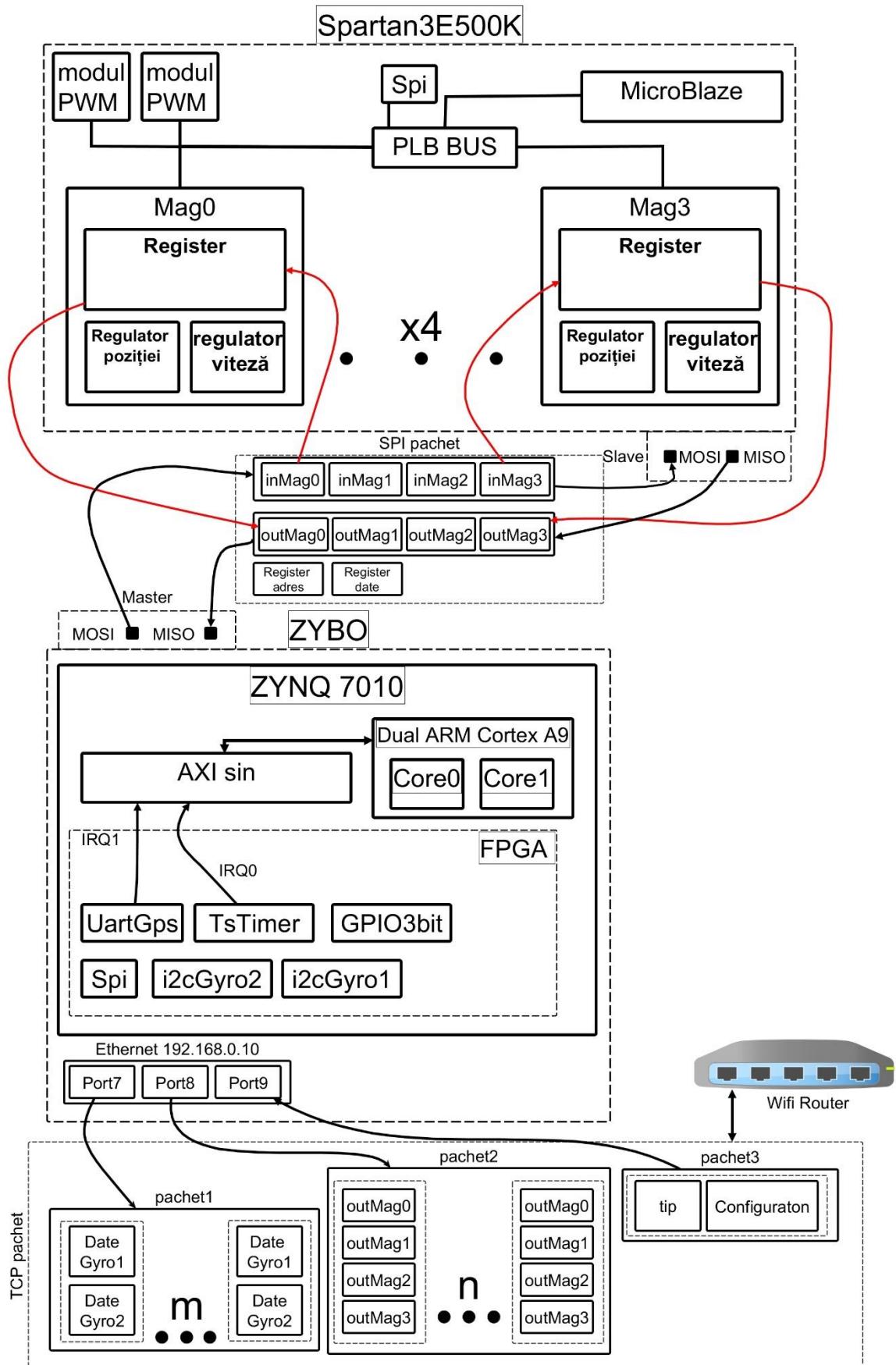


Fig. 1.4 Pachetele de comunicație și modulele proiectate în circuitele FPGA

Rolul modulelor:

Pe Figura 1.3 se poate vedea:

- Pompa de răcire-turația pompei este comandat prin semnal PWM
- Ventilator FPGA: asigură ventilația sistemului FPGA, viteza de rotație se poate regla prin semnal PWM

Modulele din Figura 1.4:

- Mag0..Mag3 nuclee IP, care conțin regulatoarele PID și poziție.
- Rolul modulelor PVM este generarea semnalului PWM, factorul de umplere se poate programa din procesorul Microblaze.
- SPI- modul comunicație, prin acest modul este realizată comunicația cu sistemul ZYBO..
- magistrala PLB – pe această magistrală sunt conectate elementele de bază din sistem.
- magistrala AXI – prin această magistrală este realizată conexiunea între procesor ARM și modulele legate în sistemul ZYBO.
- UartGPS recepționează datele de la modulul GPS prin protocol serial.
- TsTimer- 0.005 ms este utilizată pentru generarea perioadei de eşantionare.
- I2cGyro1,2 – prin acest modul este conectat modulul giroscop la sistem și sunt citite datele de la giroscop.
- Ethernet: se poate comunica cu sistem prin router wifi prin protocol TCP.
- TCP pachet : structura pachetelor TCP, care conțin datele măsurate de la senzori și de la nucleele IP .
- Wifi router- între sistemul ZYBO și router este o conexiune ethernet prin cablu.

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 frecările mecanice. Din această cauză este suficient dacă în momentul potrivit vom opri motorul. La schimbarea polarității motorului DC se schimbă și direcția de rotație a motorului, ajunge dacă intervenim în sistem cu semnal de reglare minim sau maxim.

Regulatorul realizat este definit cu ajutorul următoarelor ecuații:

$$\begin{cases} U = U_{MAX}, & \text{if } e_{sz} > 0, \\ U = U_{MIN}, & \text{if } e_{sz} < 0, \\ U = 0, & \text{if } e_{sz} = 0, \end{cases} \quad \begin{matrix} a1 \\ a_1 \\ a0 \end{matrix}$$

$$\begin{cases} \text{if } ref' = 0 \begin{cases} \text{then if } e > q \begin{cases} \text{then } e_{sz} = e, & a1 \\ \text{else } e_{sz} = 0, & a2 \end{cases} \\ \text{else } e_{sz} = e \end{cases} \end{cases}$$

Se poate spune că semnalul regulatorului de ieșire depinde de valoarea erorii e_{sz} . În sistemul mecanic există un anumit joc între piesele componente, de aici rezultă un zgomot, pe care dorim să-l filtrăm. Într-o astfel de situație, în care mecanismul se află într-o poziție adecvată, atunci într-un domeniu $+q, -q$ regulatorul nu va fi sensibil la semnalul de intrare, până când eroarea e nu ieșe din bandă sau nu se schimbă semnalul de referință.

Structura regulatorului este prezentată în figura următoare:

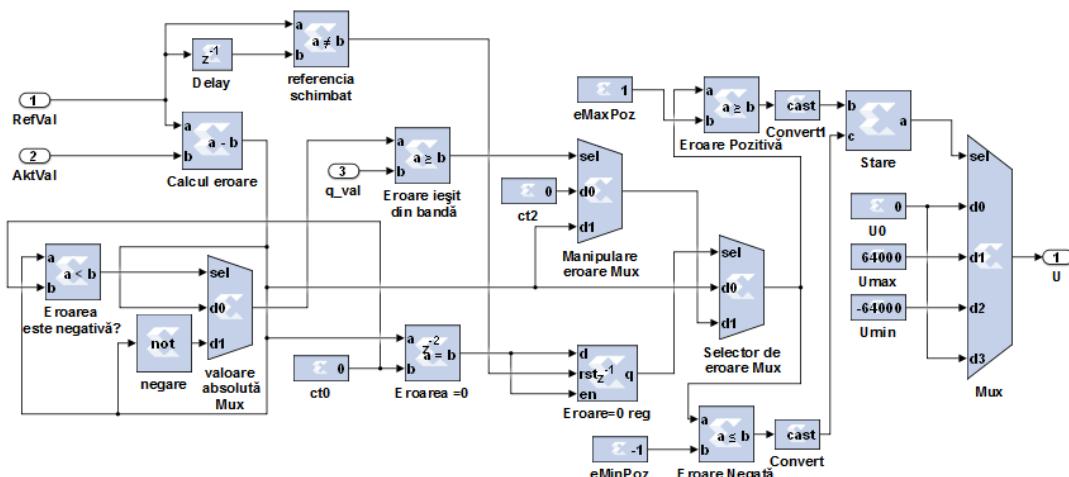


Fig. 1.5 Structura regulatorului de poziție proiectat și implementat în System Generator

Intrările: „RefVal”-poziție prescrisă măsurată în număr de impulsuri, reprezentată ca întreg cu semn pe 16 biți „AktVal”- poziția actuală măsurată în număr de impulsuri reprezentată ca întreg cu semn pe 16 biți; „U” - ieșire: întreg cu semn pe 17 biți;

Modulul „Calcul eroare” efectuează calculul erorilor din poziția actuală. Eroarea poate să fie chiar și negativă, de aceea cu ajutorul comparatorului „Eroarea este negativă?” selectăm eroarea calculată și decidem dacă este negativă sau pozitivă. După aceea cu ajutorul modulului „valoare absolută Mux” selectăm eroarea specificată, dacă este pozitivă, sau forma negată dacă este negativă.

Dacă valoarea erorii devine 0, atunci valoarea „Eroare=0 reg” este 1, până când poziția prescrisă nu se schimbă.

Modulul "Eroare ieșit din bandă" analizează valoarea erorii, și dacă această valoare este mai mică decât valoarea limită a bandei atunci cu ajutorul „Manipulare eroare Mux” valoarea eroare o să fie schimbată în 0.

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

Cu ajutorul registrelor „Umin”, „Umax”, „U0” reprezentate ca întreg cu semn de 17 biți se poate selecta valoarea maximă și minimă a semnalului de reglare. „Eroare Pozitivă” și „Eroare Negativă” decid în care domeniu este eroarea. Se pot distinge 3 domenii: negativ, pozitiv și eroare 0. Modulul multiplexor „Mux” selectează valoarea actuală a semnalului de control.

Reglarea poziției motorului DC pe robot:

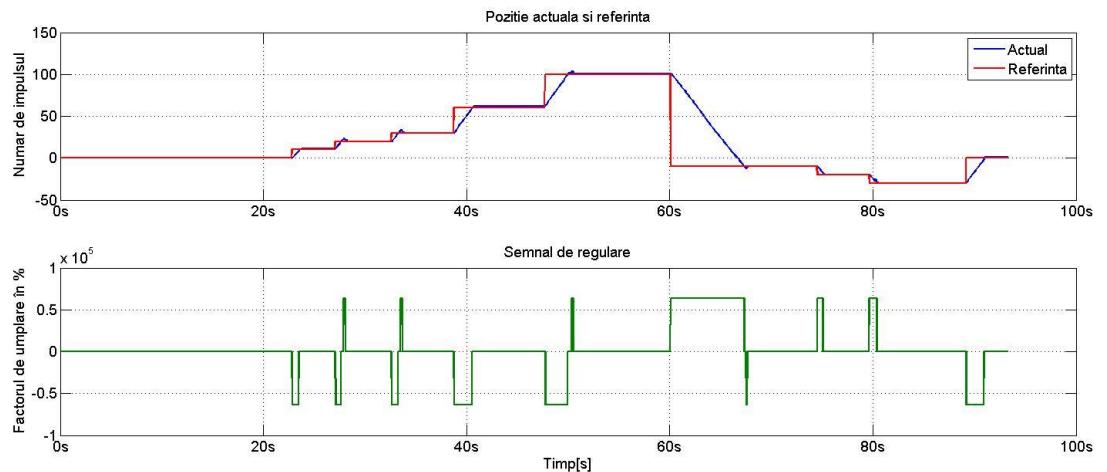


Fig. 1.6 Reglarea poziției rezultat de măsurare

Pe Figura 1.6, prima imagine este prezentată poziția actuală a sistemului și valoarea de referință reprezentată în impuls. (1 impuls = 2 grade). Pe a doua imagine se poate vedea valoarea semnalului de reglare, care reprezintă factorul de umplare pentru generatorul de semnal PWM.

REGULATOR PID-HARDWARE DISCRET

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

$$u_k = u_{k-1} + Q_0 e_k + Q_1 e_{k-1} + Q_2 e_{k-2} \quad (1)$$

$$Q_0 = K_P \left(1 + \frac{T_d}{T_s} + \frac{T_s}{T_i} \right), Q_1 = K_P \left(-1 - \frac{2T_d}{T_s} \right), Q_2 = K_P \left(\frac{T_d}{T_s} \right) \quad (2) [1]$$

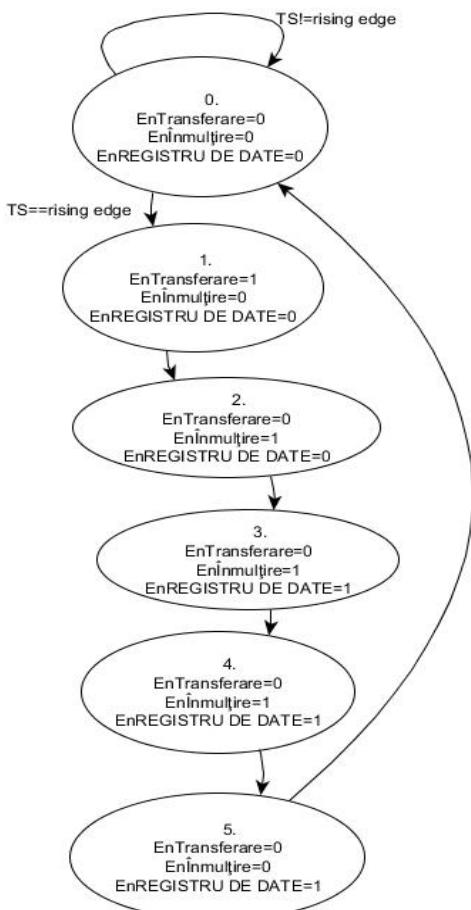


Fig. 1.7 Diagrama de stăre care descrie regulatorul PID discret

Regulatorul PID este implementat în hardware în circuit FPGA.

Pe baza datelor mai sus prezentate am proiectat un automat cu stări finite (ASF) cu cale de date realizat în System Generator.

Parametrii regulatorului PID se pot defini cu ajutorul parametrilor Q, care depind de parametrii cunoscuți: T_d - timp de derivare, T_i - timp de integrare, T_s perioada de eşantionare și K_p constanta de proporționalitate.

Automata cu stări finite implementată are 5 stări. La fiecare eşantion, ASF trece pas cu pas peste stări și revine la starea inițială. Operațiile în fiecare stare sunt efectuate pe frecvența semnalului de ceas a sistemului de dezvoltare FPGA. ASF-ul trece de la o stare la alta la fiecare semnal de ceas.

REALIZARE ÎN SYSTEM GENERATOR

Pentru selectarea căilor de date corespunzătoare este utilizat un numărător de 2. Căile de date sunt selectate cu două module de multiplexare MUXQ și MUXE.

Parametrii de intrare Q_0, Q_1, Q_2 sunt valori întregi cu semn de 16 biți, e semnal de eroare întreg cu semn de 16 bit respectiv T_s -boolean.

Ieșiri: semnalul U de 17 biți întreg cu semn. Pentru selectarea parametriilor Q este responsabil MUXQ, iar MUXE este responsabil pentru selectarea valorile de intrare e , ek-1 și ek-2 semnale întârziate în timp. Pe Figura 1.8 modulul „Înmulțire” înmulțește valoarea primită de la modulul de multiplexare după care adaugă la valoarea registrului „REGISTRU DE DATE”. Valoarea fiecărui modul dacă depășește valoarea minimă sau maximă este saturată. Astfel se poate evita creșterea continuă a valorii de ieșire al modulului din cauza componentei integrative, care ar răsturna funcționarea sistemului.

Pe figura 1.8 se poate observa registrele DELAY, care exploatează valorile erorilor e_k, e_{k-1}, e_{k-2} , din eşantioanele anterioare. Cele trei registre sunt legate între ele și sincronizate la perioada de eşantionare T_s . Valoarea erorii în fiecare eşantion este înscrisă în registrul DELAY1.

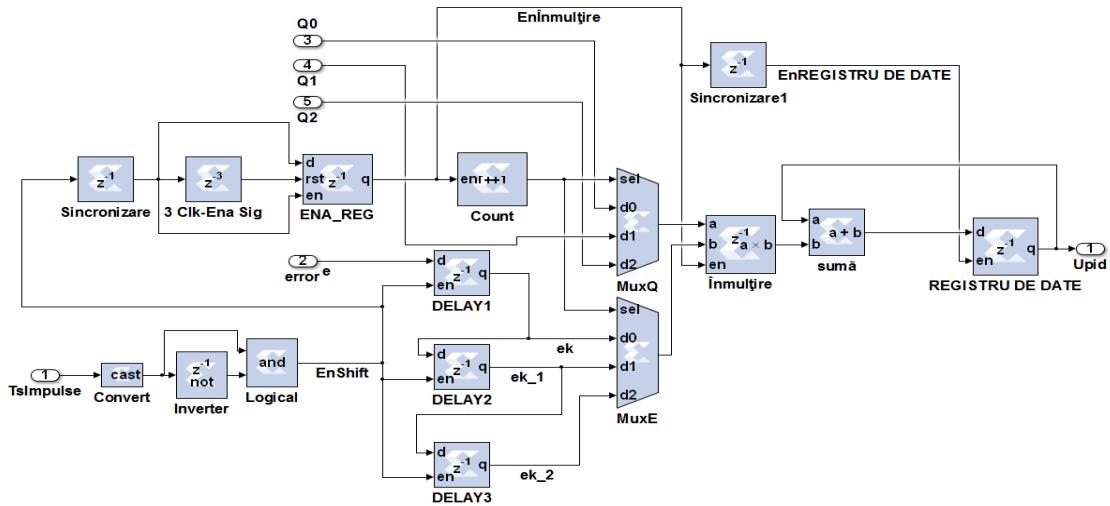


Fig. 1.8 Structura PID proiectat și implementat în System Generator

Numărătorul „Count” pornește numai după deplasarea datelor în registre, la semnalul generat de modulul „Sincronizare”. Semnalul de autorizare (en) pentru numărător (Count) activă pe durata a trei semnale de tact este generat prin modulele „ENA_REG” și „3 Clk-Ena Sig”.

Detectarea frontului de urcare este realizat prin „Inverter” și de o poartă logică „AND”. Prin compararea a celor două valori se poate detecta scimbarea semnalului. Cu ajutorul modulului „Sincronizare1” este autorizat înscriverea datelor în „REGISTRU DE DATE”.

Reglarea vitezei motorului DC pe robot

Pe Figura 1.10 se poate vedea rezultatul măsurătorilor de reglare a vitezei şenilelor de pe robot. Acordarea regulatorului PID este realizată prin metodă Oppelt.

Parametrii procesului:

timpul mort: 0.1s, constanta de timp: 0.9s, factor de amplificare : 0.71.

Parametrii PID:

KP=134, Ti=0.23, Td=0.002s perioadă de eşantionare: Ts=0.03s.

Pe prima imagine se poate vedea viteza actuală și de referință în grad/sec, iar pe a doua imagine ieșirea generatorului PID în %, care furnizează semnalul de intrare pentru generatorul PWM.

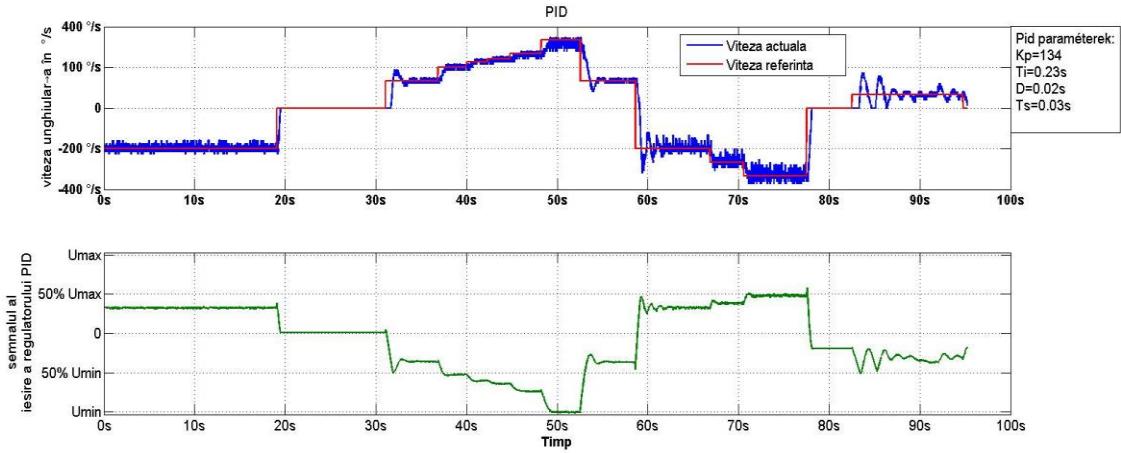


Fig. 1.9 Reglarea vitezei cu regulator PID hardware, rezultate de măsurare

Realizări proprii

- ✓ Proiectarea în Autodesk Inventor și construirea sistemului mecanic
- ✓ Configurarea și proiectarea componentelor hardware din circuitul FPGA respectiv proiectarea și implementarea softurilor aferente care rulează pe cele două sisteme:
 - Comunicație SPI între cele două FPGA
 - Realizarea elementelor hardware în System Generator
 - Sistemul de întrerupere software și hardware
 - Programe C pentru procesoare MicroBlaze și ARM
- ✓ Implementare în hardware a regulațoarelor și acordarea acestora utilizate la cele 8 motoare de curent continuu dc.
- ✓ Stand de măsurare construit pentru testarea regulațoarelor și sistemului senzorial.
- ✓ Achiziția datelor de la senzori: gyroskop1, gyroskop2, și cele 8 encoderi incrementali
- ✓ Comunicație Ethernet + client GUI (în Matlab)
- ✓ Construirea electronică de putere: controleri punte H, punte H, răcire cu apă
- ✓ Efectuarea de măsurători și punerea în funcțiune a sistemului

CONCLUZII

În urma implementării am întâlnit cu o mulțime de erori mici, dar care oferă oportunități de dezvoltare în viitor.

După realizarea lucrării consider că era o alegere bună utilizarea unui sistem de dezvoltare FPGA, deoarece oferă la bază bună pentru oportunități de dezvoltare privind atât partea hardware cât și partea software. Funcționalitatea regulatoarelor de viteză și poziție implementate în hardware este demonstrată din rezultatul măsurătorilor efectuate.

Modulul de măsurare viteză ar trebui extins cu un filtru pentru a reduce zgomotul ce rezultă la viteze de rotație redusă.

În ceea ce privește funcționarea sistemului, ar fi un avantaj măsurarea curentului electric la motoarele DC. Această măsurare ar fi de mare folos în reglare și protecție. La alimentarea sistemului cu energie electrică ar fi necesară de minim doi acumulatori independente, ca partea de electronica de performanță și cea digitală să primescă alimentarea de la diferite surse de energie, astfel circuitele digitale să aibă priorități în comparație cu alte elemente. Dacă dorim, ca sistemul să funcționeze în mod continuu pentru o perioadă mai îndelungată, ar trebui să integrăm încă o sursă de alimentare, de exemplu un colector de energie solară, care ar asigura aprovisionarea energiei pentru o anumită perioadă de timp.

Este validată și transmisia datelor prin protocolul TCP, cu scopul de monitorizare și setare a parametrilor din sistem. Router-ul ar ajuta în viitor la integrarea elementelor suplimentare, cum ar fi un braț de robot. Prin acest mod este posibil comunicarea cu acest braț prin router prin protocol TCP atât de la distanță dar și din sistemul FPGA.

Senzorii incrementali, pe care l-am realizat îndeplinește sarcinile conform așteptărilor, au fost realizate ieftin în comparație cu prețul pieței. Din punctul de vedere al dezvoltării este posibil și mărirea rezoluției discurilor, prin mărirea diametrului discului.

Bugetul necesar pentru realizarea robotului este sub prețul pieței, comparând cu achiziționarea unui sistem asemănător.

2 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 pozíóját koordinálják. A SPARTAN3e500 laphoz van illesztve 8 db. inkrementális érzékelő, amelyek a motorok pozíóját, 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, fünyíró, stb. Alkalmazhatósága elképzelhető akár a mezőgazdaságban is, mint gyomtalanító gép, vagy akár a biztonság technikában, mint beavatkozó eszköz.

Kulcsszavak: FPGA, beágyazott hardver szabályozó, PID szabályozó, hardver csimuláció, mobilis robot, inkrementális jeladó,

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, control 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, embedded hardware controller, PID controller, hardware co-simulation, mobile robot, incremental encoder,

1	EXTRAS.....	6
2	EXTRAS (MAGHIARĂ.....	18
REZUMAT FIGURI, TABELE		25
3	ÎNTRODUCERE	29
4	BIBLIOGRAFIE STUDIATĂ	31
4.1	PROIECTE SIMILARE A DEZVOLTĂRII REGULATOARELOR PID ÎN SISTEM FPGA	31
4.2	SENZOR INCREMENTAL.....	31
4.3	VITEZĂ UNGHİULARĂ MĂSURAT CU FPGA	32
4.4	MOTOARE DE CURENT CONTINUU.....	32
4.5	ACORDAREA REGULATORULUI PID DUPĂ METODĂ ZIEGLER-NICHOLS	33
4.6	ACORDAREA REGULATORULUI PID DUPĂ METODĂ OPPELT	34
4.7	Modulele SYSTEM GENERATOR	34
5	RENDSZER TERVEZÉSE.....	40
5.1	REGULATOARELE:	40
5.1.1	<i>Regulator PID Discret prin Hardware.....</i>	40
5.1.1.1	Implementare în System Generator	41
5.1.1.2	Rezultatele de simulare Simulink	43
5.1.1.3	Calculul parametrilor Q pe baza T_i, T_d, K_p, T_s	44
5.1.2	<i>Generarea semnalului a perioadei de eşantionare.....</i>	45
5.1.3	<i>Regulator poziție.....</i>	46
5.1.3.1	Structura regulatorului:	46
5.1.3.2	Simularea regulatorului	47
5.1.4	<i>Măsurări Hardware mérések.....</i>	48
5.1.4.1	Regularea vitezei motorului DC pe stand de măsurare.....	48
5.2	GENERAREA UNEP IP MASK, CARE CONȚINE REGULATOARE DE VITEZĂ ȘI DE POZIȚIE IN SYSTEM GENERATOR	49
5.2.1	<i>Măsurări pe sistem</i>	55
5.2.1.1	Regularea vitezei a şenilelor a	55
5.2.1.2	Regularea vitezei	57
5.3	SENZORI.....	59
5.3.1	<i>Senzor incremental</i>	59
5.3.1.1	Structura receptorului incremental optic	59
5.3.2	<i>Prelucrare a semnalului de senzor incremental cu ajutorul circuitului electric FPGA</i>	60
5.3.2.1	Simulație în System Generator	61
5.3.2.2	Măsurare poziție cu ajutorul postului incremental	62
5.3.2.3	Măsurare viteză unghiulară cu post incremental	63
5.4	MPU-6050 GIROSCOP ȘI ACCELEROMETRU.....	66
5.5	ELEMENTE DE INTERVENȚIE:	68
5.5.1	<i>Realizarea Generatorului Pwm prin circuit electric FPGA în mediu System Generator</i>	68

5.5.1.1	Realizare.....	68
5.6	ELECTRONICĂ.....	70
5.6.1	<i>Electronică digitală</i>	70
5.6.2	<i>Structura sistemului FPGA</i>	70
5.6.2.1	Zybo, placă de dezvoltare FPGA.....	72
5.6.2.2	Spartan3e, placă de dezvoltare FPGA	73
5.6.2.3	Protocoale de comunicatie.....	74
5.6.3	<i>Alocarea sarcinilor</i>	77
5.6.3.1	Zybo, placă de dezvoltare FPGA.....	77
5.6.3.2	Spartan3e, placă de dezvoltare FPGA	77
5.7	PERFORMANȚĂ ELECTRONICĂ.....	78
5.7.1	<i>Funcționare Bootstramp</i>	84
5.7.1.1	Simulație în mediu Simulink	85
5.8	MODEL ROBOT	87
6	STRUCTURA MECANICĂ A ROBOTULUI	90
7	REZULTATE OBȚINUTE, REALIZĂRI:	92
8	CONCLUZII:	92
9	BIBLIOGRAFIE	94
10	ANEXE	96

1	EXTRAS.....	6
2	KIVONAT	18
	ÁBRÁK, TÁBLÁZATOK JEGYZÉKE.....	25
3	BEVEVEZETŐ.....	29
4	BIBLIOGRÁFIAI TANULMÁNY.....	31
4.1	HASONLÓ FPGA FEJLESZTÖRENDSZEREN MEGVALÓSÍTOTT PID SZABÁLYZÓK.....	31
4.2	INKREMENTÁLIS ÉRZÉKELŐ	31
4.3	SZÖGSEBESSÉG MÉRÉSE FPGA SEGÍTSÉGÉVEL.....	32
4.4	EGYENÁRAMÚ MOTOROK	32
4.5	PID SZABÁLYOZÓ HANGOLÁSA ZIEGLER-NICHOLS MÓDSZERREL	33
4.6	PID SZABÁLYOZÓ HANGOLÁSA OPPelt MÓDSZERREL	34
4.7	SYSTEM GENERÁTOROS MODULOK.....	34
5	RENDSZER TERVEZÉSE.....	40
5.1	SZABÁLYOZÓK:.....	40
5.1.1	<i>Diszkrét Hardveres PID szabályozó</i>	<i>40</i>
5.1.1.1	Megvalósítás System Generátorban	41
5.1.1.2	Simulink szimulációs eredmények	43
5.1.1.3	Q paraméterek számolása Ti, Td, Kp, Ts alapján.	44
5.1.2	<i>Mintavételezési periódus jelének generálása.....</i>	<i>45</i>
5.1.3	<i>Pozíció Szabályzása.....</i>	<i>46</i>
5.1.3.1	A szabályozó felépítése:	46
5.1.3.2	Szabályozó szimulálása	47
5.1.4	<i>Hardveres mérések</i>	<i>48</i>
5.1.4.1	DC motor sebesség szabályzása mérőstandon.....	48
5.2	SEBESSÉG ÉS POZÍCIÓ SZABÁLYOZÓT TARTALMAZÓ IP MAG GENERÁLÁSA SYSTEM GENERATOR-BAN	
	49	
5.2.1	<i>Mérések a rendszeren</i>	<i>55</i>
5.2.1.1	A robot lánctalpának sebesség szabályozása	55
5.2.1.2	Pozíció szabályozása	57
5.3	SZENZOROK	59
5.3.1	<i>Inkrementális érzékelő</i>	<i>59</i>
5.3.1.1	Optikai inkrementális vevő felépítése	59
5.3.2	<i>Inkrementális érzékelő jeleinek a feldolgozása FPGA áramkör segítségével.....</i>	<i>60</i>
5.3.2.1	Szimuláció System Generatorban.....	61
5.3.2.2	Pozíció mérése inkrementális adó segítségével	62
5.3.2.3	Szögsebesség mérése inkrementális adó segítségével	63
5.4	MPU-6050 GIROSZKÓP ÉS GYORSULÁSMÉRŐ	66
5.5	BEAVATKOZÓ ELEMEK:	68
5.5.1	<i>PWM Generátor megvalósítása FPGA áramkörön System Generator környezetben.</i>	<i>68</i>

5.5.1.1	Megvalósítás	68
5.6	ELEKTRONIKA	70
5.6.1	<i>Digitális Elektronika</i>	70
5.6.2	<i>FPGA Rendszer Felépítése</i>	70
5.6.2.1	Zybo FPGA fejlesztőlap.....	72
5.6.2.2	Spartan3e FPGA fejlesztőlap	73
5.6.2.3	Kommunikációs protokollok.....	74
5.6.3	<i>Feladatok Elosztása</i>	77
5.6.3.1	Zybo fejlesztőlap	77
5.6.3.2	Spartan fejlesztőlap	77
5.7	TELJESÍTMÉNY ELEKTRONIKA.....	78
5.7.1	<i>Bootstramp működése</i>	84
5.7.1.1	Szimuláció Simulink környezetben.....	85
5.8	ROBOT MODELL	87
6	ROBOT MECHANIKAI FELÉPÍTÉSE	90
7	ELÉRT EREDMÉNYEK, MAGVALÓSÍTÁSOK:	92
8	KÖVETKEZTETÉSEK:	92
9	BIBLIOGRÁFIA	94
10	MELLÉKLET	96

Ábrák, táblázatok jegyzéke

Fig. 1.1 Structura Robotului- Inventor 3D Foto	6
Fig. 1.2 Alimentarea cu energie electrică a robotului și structura sistemului de răcire.....	7
Fig. 1.3 Structura sistemului.....	9
Fig. 1.4 Pachetele de comunicație și modulele proiectate în circuitele FPGA.....	10
Fig. 1.5 Structura regulatorului de poziție proiectat și implementat în System Generator .	12
Fig. 1.6 Reglarea poziției rezultat de măsurare	13
Fig. 1.7 Diagrama de stare care descrie regulatorul PID discret	14
Fig. 1.8 Structura PID proiectat și implementat în System Generator	15
Fig. 1.9 Reglarea vitezei cu regulator PID hardware, rezultate de măsurare	16
Kép. 4.1 Inkrementális érzékelő jelek	31
Kép. 4.2 A rendszer egységugrásra adott válasza és megközelítése egyenesekkel.....	34
Kép. 4.3 System Generator beállítása.....	35
Kép. 4.4 Új hardver profil létrehozása	36
Kép. 4.5 Összeadó modul és beállítása	37
Kép. 4.6 Szorzó modul	37
Kép. 4.7 Multiplexer modul és beállításai	38
Kép. 4.8 Logikai műveletek modul, és beállításai.....	39
Kép. 4.9 Simulink illesztő modulok	39
Kép. 5.1 a Pozíció és a sebesség szabályzási hurok elvi strukturális felépítése.....	40
Kép. 5.2 Állapot automata, amely leírja a Diszkrét PID szabályzót	41
Kép. 5.3 A PID felépítése System Generatorban	42
Kép. 5.4 PID Simulink szimulációs model.....	43
Kép. 5.5 Szimulációs eredmény amely tükrözi a konstans bementre a számolási lépésekét	43
Kép. 5.6 PID minimális periódusa	44
Kép. 5.7 GUI pid paraméterek.....	44
Kép. 5.8 Mintavételi taktust generáló modul	45
Kép. 5.9 Szimulációs eredmények mintavételi jelgenerátor.	46
Kép. 5.10 A Pozíció szabályozó System generátoros felépítése	47
Kép. 5.11 A pozíció szabályzás moduláris felépítése System Generator környezetben	48
Kép. 5.12 A pozíció szabályozó szimulálása	48

Kép. 5.13 Sebesség szabályozás PID szabályzóval.....	49
Az IP magok az FPGA-ban hardveresen vannak összealítva logikai kapuk és egyéb digitális elemek segítségével, ezért a benne található modulok minden az FPGA 50Mhz órajelére működnek. Az alábbi Kép. 5.15 a sebesség és pozíció szabályozást tartalmazó IP mag System generátoros felépítését mutatja be.....	51
Kép. 5.15 Sebesség és pozíció szabályozást tartalmazó IP mag System generátoros felépítése	52
Kép. 5.16 Pozíció szabályozó modul belső felépítése a Kép. 5.15	53
Kép. 5.17 Sebesség szabályozó modul felépítése a Kép. 5.15	54
Kép. 5.17 Szabályzó körök összekapcsolásának elvi kialakítása	54
Kép. 5.19 PID szabályozó a robot lánctalpának a sebességét szabályozva.....	55
Kép. 5.19 DC motor és a Kup fogaskerék áttétel szögsebessége maximális vezérlőjelre. .	56
Kép. 5.21 A rendszer egységugrásra adott válasza és megközelítése egyenesekkel.....	56
Kép. 5.21 Sebesség szabályzása PID-del Oppelt hangolási módszer után.....	57
Kép. 5.22Pozíció szabályozás csiga fogaskerék áttételeken keresztül.....	57
Kép. 5.23Forgatalp pozíciója szabályzás közben.....	58
Kép. 5.25 Optikai inkrementális vevő felépítése és elhelyezése	59
Kép. 5.26 Érzékelő tranzisztorok elhelyezése	59
Kép. 5.27 Rések és az éÉrzékelők közti kapcsolat.....	60
Kép. 5.28 Idődiagram a Tárcsa paraméterei függvényében	60
Kép. 5.29 Inkrementális jelfeldolgozó modul1 érzékelő modul belső felépítése.....	61
Kép. 5.30 Inkrementális érzékelőtől érkező jelek átalakító irány és impulzus jelekre	61
Kép. 5.31 Szimulációs eredmények a lehetséges bemenetekről az Black Box1 modulba..	62
Kép. 5.32 Inkrementális adóval mért pozíció, szimulációs modellje SytemGeneratorban.	63
Kép. 5.33 Sebesség mérő modul felépítése	64
Kép. 5.34DC motor sebességének mérése FPGA lapon	64
Kép. 5.35Dc motor Sebesség mérése FPGA rendszeren, System generatorban megvalósítva	65
Kép. 5.38.a Nmért = 65, Ts =8ms	66
Kép. 5.38.cNmért = 32, Ts =4ms	66
Kép. 5.38.bNmért = 650, Ts =80ms	66
Kép. 5.39 Giroszkóp mért adatainak az ábrázolása a GUI program segítségével	68
Kép. 5.40 A PWM generátor System Generátorban megvalósított szerkezete	68

Kép. 5.41 a PWM generátor bemenő, kimenő illetve néhány belső jele (Scope1)	69
Kép. 5.42 rendszer elvi felépítése.....	70
Kép. 5.43 Kommunikációs csomagok és az FPGA áramkörökbe programozott modulok elvi felépítése.....	71
Kép. 5.44 ZYBO Core0 program folyamat árbája	73
Kép. 5.45 Spartan3e500, microblaze szoftver Folyamat ábrája	73
Kép. 5.46 Spartan3e500, MicroBlaze szoftver folyamat ábrája.....	73
Kép. 5.48Hip4082 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.....	78
Kép. 5.48 Két hídvezérlő áramkör kapcsolási rajza HIP4082 integrált áramkörrel megvalósítva.....	79
Kép. 5.49 PWM és a tranzisztorok kapcsolása.....	80
Kép. 5.50 Nem invertáló erősítő forrás [15].....	81
Kép. 5.50 Dupla hídvezérlő áramkor vezérlő jelei JP2 csatlakozó a Kép. 5.47-n.	81
Kép. 5.52 H híd tranzisztorainak a Gate vezetékei.....	81
Kép. 5.53 A négy Kép. 5.53 látható szalagvezeték jelenik meg a Buszvezetékben.....	82
Kép. 5.54 FPGA kimentének a védelme	82
Kép. 5.55 A robot energia ellátása valamint a hűtő rendszer elvi felépítése.....	83
Kép. 5.56 Vízpumpa és a ventilátor motorjának vezérlő teljesítmény elektronikai kapcsolása	84
Kép. 5.57Bootstramp megoldás a felső tranzisztor Gate bemenetének a meghajtására.....	85
Kép. 5.58Bootstramp kondenzátor feszültsége a W és W11 pontokban.....	85
Kép. 5.59 Bootstramp működése, szimulációs modell MATLAB/SIMULINK környezetben	86
Kép. 5.60 Szimulációs eredmények Bootstramp.....	86
Kép. 5.61 Robot kerekek sebsége és a robot mozgásának viszonya	87
Kép. 5.62 Robot 3D vektorábrája.....	88
Kép. 5.63 Oldalnézetek és felülnéztet, jelölések szemléltetése.....	89
Kép. 6.1 Átételek	91
Kép. 6.2 Robot vázának Inventoros 3D Képe	91
Kép. 10.1 A mechanikai rendszer műszaki rajza.....	97
Táblázat. 4-1 Ziegler-Nichols módszerrel történő PID hangolás	34
Táblázat. 4-2 Oppelt módszer hangolás	34

Táblázat. 5-1 Manuálisan számolt értékek a szimuláció ellenőrzésére44

3 BEVEVEZETŐ

A dolgozat célja mobilis tereprobot tervezése és a megépítéséhez szükséges elemek tárgyalása. A mechanikai rendszer Autodesk Inventor-ban volt megtervezve, és az elkészített terv alapján kivitelezve. A következő részfeladat a vezérlő elektronika kialakításának a tervezése és a szenzoroknak a rendszerbe való integrálása volt. A rendszeren különböző szenzorok találhatók, amelyek közül talán a legfontosabb a sebesség és pozíció mérésére alkalmazott inkrementális jeladó. A rendszeren megtalálható két FPGA fejlesztő lap, egy nagyobb erőforrásokkal rendelkező Zybo (beépített ARM processzorral), amely a matematikai számítások elvégzésére hivatott, és egy kisebb kapacitású FPGA lap (SPARTAN3e500) amely tartalmaz 8 hardveresen megvalósított szabályozót és egy MicroBlaze 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 mérlik. A szenzorok adatait a Spartan lap egy gyors SPI kommunikáción keresztül küldi tovább a Zybo lapnak. A gyors hardveres PID szabályozó megvalósítása FPGA fejlesztőlapon Xilinx System Generator tervezőeszközzel készült, a szimulációkat hardver co-szimulációval végeztem el.

A tervezést a mechanikai rendszerrel kezdtem. Autodesk Inventor segítségével több változatot is megterveztem ameddig eljutottam a dolgozatban tárgyalt mechanikai struktúrához. A mechanikai rendszert, saját magam vitéleztem ki a tervezet 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 robot vázához rögzíteni lehet nagyobb tömegű kiegészítő tartozékokat pl.: robotkar, fünyíró, stb. Alkalmazhatósága elképzelhető a mezőgazdaságban, mint gyomtalanító gép, vagy akár a biztonság technikában, mint beavatkozó eszköz.

4 BIBLIOGRÁFIAI TANULMÁNY

4.1 HASONLÓ FPGA FEJLESZTŐRENDSZEREN MEGVALÓSÍTOTT PID SZABÁLYZÓK

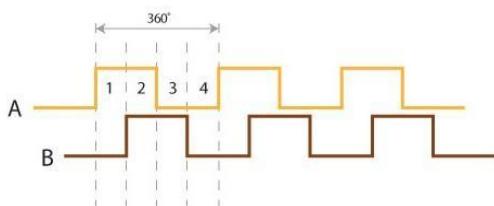
A [2]cikkben tárgyalt FPGA erőforráson kivitelezett PID szabályozó, amelyet a nagyobb működési sebesség kedvéért FPGA alapon valósított meg. A szabályozó paraméterei fordításkor vannak meghatározva, a hardverben kívülről nem lehet megadni, ami a hangolás szemszögéből nem előnyös. A [1] PID szabályozó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

$$u_k = u_{k-1} + Q_0 e_k + Q_1 e_{k-1} + Q_2 e_{k-2}$$
$$Q_0 = K_P \left(1 + \frac{T_d}{T_s} + \frac{T_s}{T_i} \right), Q_1 = K_P \left(-1 - \frac{2T_d}{T_s} \right), Q_2 = K_P \left(\frac{T_d}{T_s} \right) \quad [1]$$

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 is használt FPGA fejlesztőrendszere a [4] irodalomban találtam hasonló PID szabályzóra. 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 komplementens 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.

4.2 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



Kép. 4.1 Inkrementális érzékelő jelek

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.

4.3 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:

$$w[rpm] = \frac{60 * f_{clk}}{R * X * E}$$

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

$$\Delta w = \frac{E * R}{f_{clk}}$$

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

$$w[rpm] = \frac{60 * X}{T_m * R * E}$$

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. A sebességé mérésére egy időegység alatt beérkező impulzusokat számolom meg.

4.4 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 modellezzi. Ahol a i rotoron átfolyó áram. A Biot Savart és a Lenz törvények alapján: $e = c_1 \omega$, $\tau = c_2 i$. Ahol a c1 és c2 konstansok.

$$u = iR + L \frac{di}{dt} + e$$

Az elektromos egyenlet mellé még felírja a mechanikai egyenleteket is:

$$J_R \frac{d\omega}{dt} = c_2 i - F_f - \tau_{ext}$$

J_R – rotor inerciája, F_f – motorban fellépő súrlódási erők, τ_{ext} – külső nyomaték.

A motor dinamikus moteljéhez egyesíti a két egyenletet:

$$\begin{cases} L \frac{di}{dt} + iR + c_1 \omega = u \\ J_R \frac{d\omega}{dt} = c_2 i - F_f - \tau_{ext} \\ \frac{d\alpha}{dt} = \omega \end{cases}$$

Ahol az α a motor szög pizicíója.

A motor állapotteres motelje, a választott állapotok:

$$x = \begin{pmatrix} i \\ \omega \\ \alpha \end{pmatrix}$$

$$u = \begin{pmatrix} F_f \\ \tau_{ext} \\ u \end{pmatrix}$$

Ahol x - állapotok, és a u - bemenetek.

$$A = \begin{pmatrix} -\frac{R}{L} & -\frac{c_1}{L} & 0 \\ \frac{c_2}{J_R} & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, B = \begin{pmatrix} \frac{1}{L} & 0 & 0 \\ 0 & -\frac{1}{J_R} & -\frac{1}{J_R} \\ 0 & 0 & 0 \end{pmatrix}$$

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.

4.5 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 ($T_{lengés}$)

megmérve és ismerve a K_p erősítést, amelyen a lengések jelentkeznek, kiválasztatjuk a megfelelő PID paramétereket egy táblázat alapján. Mintavételezet megvalósítás esetén, a mintavételezési periódust $T \cong (0.1 \dots 0.3)T_{lengés}$ körüli értékre kell választani.

	KP	TI	TD
P	$0.45K_p$	-	-
PI	$0.45K_p$	$0.85T_{lengés}$	
PID	$0.6K_p$	$0.5T_{lengés}$	$0.12T_{lengés}$

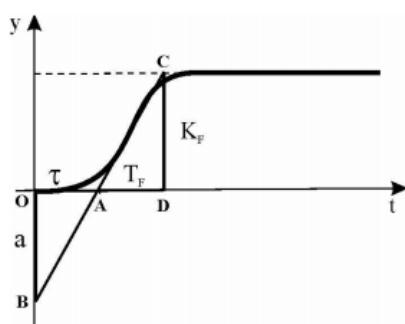
Táblázat. 4-1 Ziegler-Nichols módszerrel történő PID hangolás

4.6 PID SZABÁLYOZÓ HANGOLÁSA OPPELT MÓDSZERREL

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

$$H_F(s) = \frac{K_F}{T_F s} e^{-st}$$

A rendszert három paraméterrel lehet jellemzni: K_F – erősítés, T_F – időállandó, τ – holtidő.



	KP	TI	TD
P	$1/a$	-	-
PI	$0.8/a$	3τ	-
PID	$1.2/a$	2τ	0.24τ

Táblázat. 4-2 Oppelt módszer hangolás

Kép. 4.2 A rendszer egységugrásra adott válasza és megközelítése egyenesekkel.

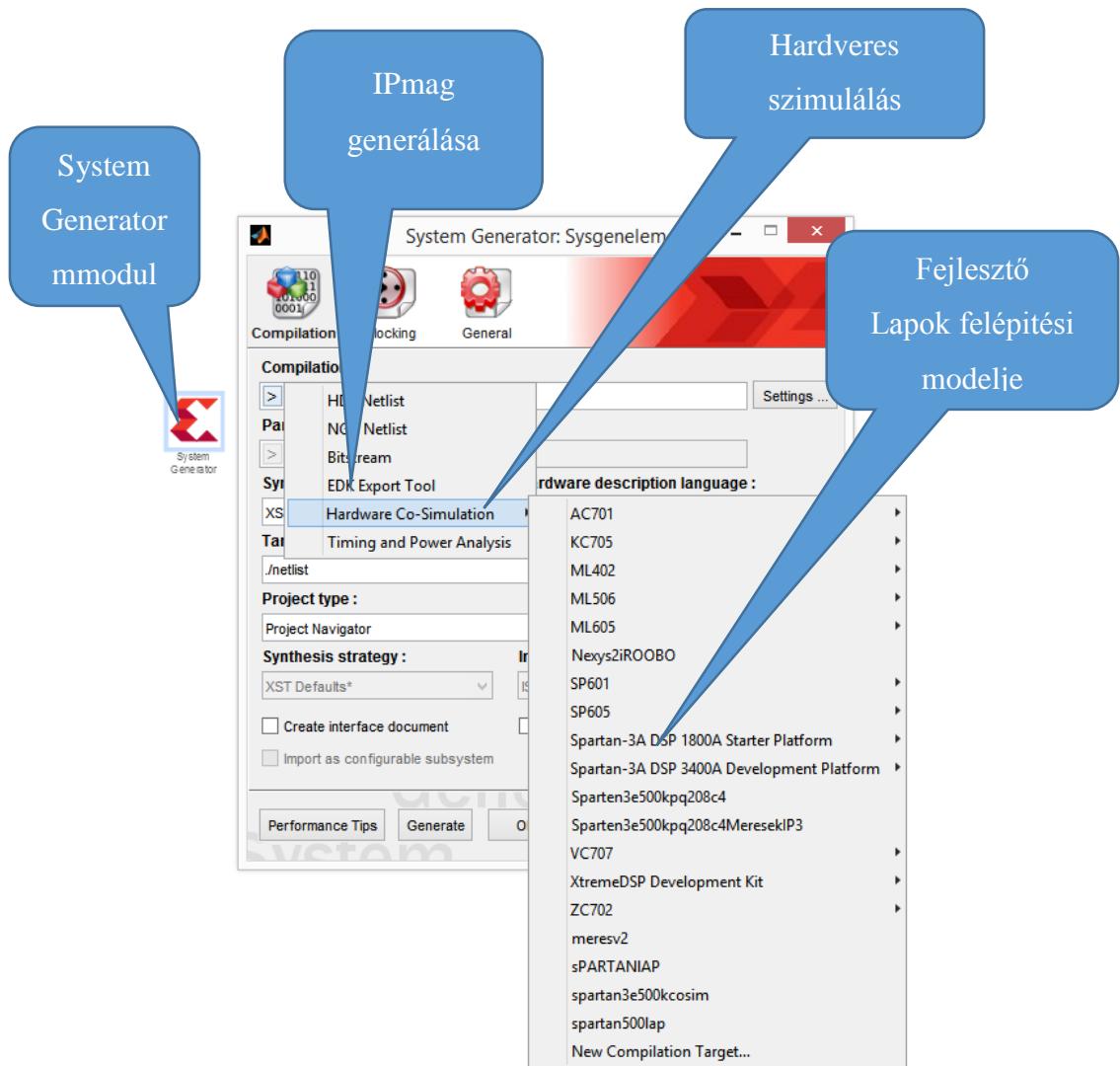
Mintavételes megvalósításnál a rendszer mintavételezési periódusát a $T \cong 0.3\tau$ értékre kell megválasztani. A módszer előnye, hogy nem kell a rendszert a stabilitás határára vinni, így biztonságosabb, valamint könnyű a bemenet előállítása.

4.7 SYSTEM GENERÁTOROS MODULOK

A System Generator egy Simulink eszköztár, amely segítségével Xilinx cég által gyártott FPGA fejlesztőlapokra tudunk rendszereket tervezni és leszimulálni szoftveresen és hardveresen. A tervet kigenerálva az eszközre is feltölthetjük és futathatjuk.

Sytem Generator modul

Simulink ablakhoz kell illeszteni ezt a modult, jelzi és tartalmazza a szükséges beállításokat, amelyek szükségesek a modell a működéshez. A modulban állíthatjuk be, hogy Hardveres-szoftveres szimulációkat (Co-szimuláció) végzünk, vagy IP magot generálunk. A Co-szimuláció esetében létrehozhatunk új modellt, ahol megnevezzük az FPGA órajelének a kivezetését, a fejlesztőlapon megtalálható FPGA csip típusát, és létrehozhatunk fizikai kivezetéseket, vagy bemeneteket.



Kép. 4.3 System Generator beállítása

Ha a „New Compilation Target” opciót válaszuk akkor létrehozunk egy új hardver modellt. A modellt szüksége van egy egyedi névre, meghatározzuk az órajel frekvenciáját és a bevezetést a rendszerbe. A „Target Device” mezőnél beállítjuk a fejlesztőlapon megtalálható FPGA cipet, és a tokozását. A „Non-Memory- Mapped Ports” mezőbe létrehozhatunk új portokat, minden port rendelkezik, egy egyedi névvel és iránnyal, amely meghatározza, hogy az FPGA fizikai kimenete vagy bemente. A portoknak van egy

szélessége, amely megadja, hogy hány bites. minden bithez illesztenünk kell egy fizikai kivezetést.

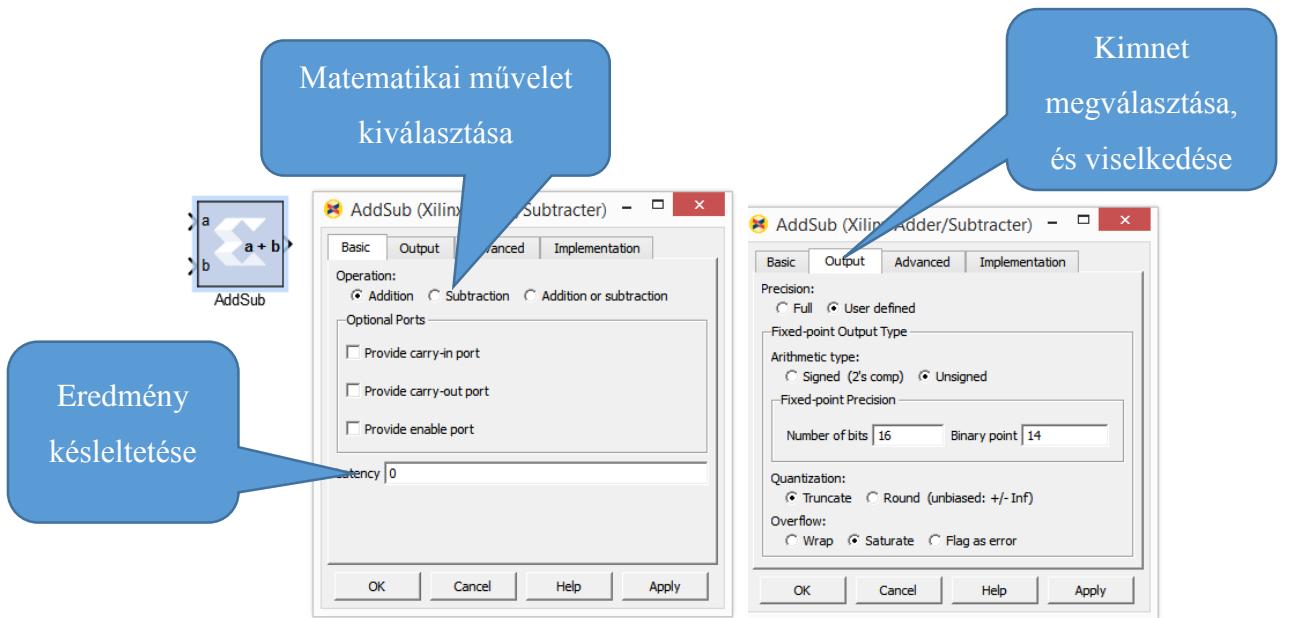


Kép. 4.4 Új hardver profil létrehozása

AddSub modul

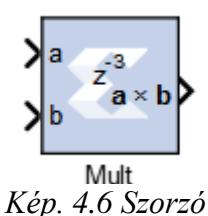
A modulnak két bemenete van: a,b amelyek egy több bites adatot képviselnek. A két bemenő adatot a beállításoktól függően összeadja vagy kivonja egymásból, és az eredményt a kimeneti portra kapcsolja. A kimenti port, nagyságát beállíthatjuk tetszőlegesen. Fix-pontos számábrázolásra is képes. A két bemenetén kívül más kontroljeleket is megadhatunk: enable jel segítségével engedélyezhetjük a modult, logikai 1-re aktív. Túlcordulás (Carry) bitet vezethetünk be vagy ki a modulból, amely jelzi, ha a művelet eredménye nem fér bele a kívánt biteken ábrázolt regiszterbe. A modul a számítást elvégzi 1 órajelen belül, így az eredményt azonnal megkapjuk ugyanabban az órajel periódusban, de ha szeretnénk, tudjuk késleltetni a „Latency” mezőben megadott órajel periódussal.

Abban az esetben, ha az eredmény túlcordulna, kiválasztjuk a „Saturate” tulajdonságot, amely meggátolja a túlcordulást negatív, és pozitív irányba. Az eredmény beszaturálódik a beállított bitszélességen felírható maximális pozitív vagy negatív értékre.



Kép. 4.5 Összeadó modul és beállítása

Mult (szorzás)



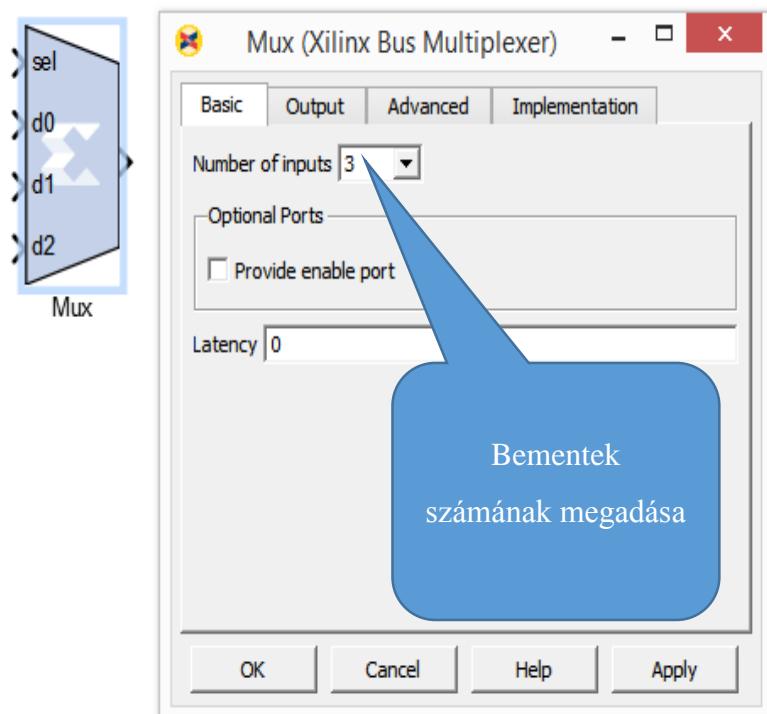
Kép. 4.6 Szorzó modul

A modulnak a beállításai hasonlítanak az AddSub moduléra. A modul a szorzás elvégzésére több órajel periódust vesz igénybe, amely ott tükrözödik, hogy az eredmény késleltetését nem tudjuk 3 órajel periódus alá vinni. Az „Implementation” ablakban kiválasztjuk a kivitelezés formáját: használhatunk előre

megépített szorzó áramkört, vagy létrehozhatunk az FPGA áramkörben új modult, amely sok erőforrást igényel.

Multiplexer modul

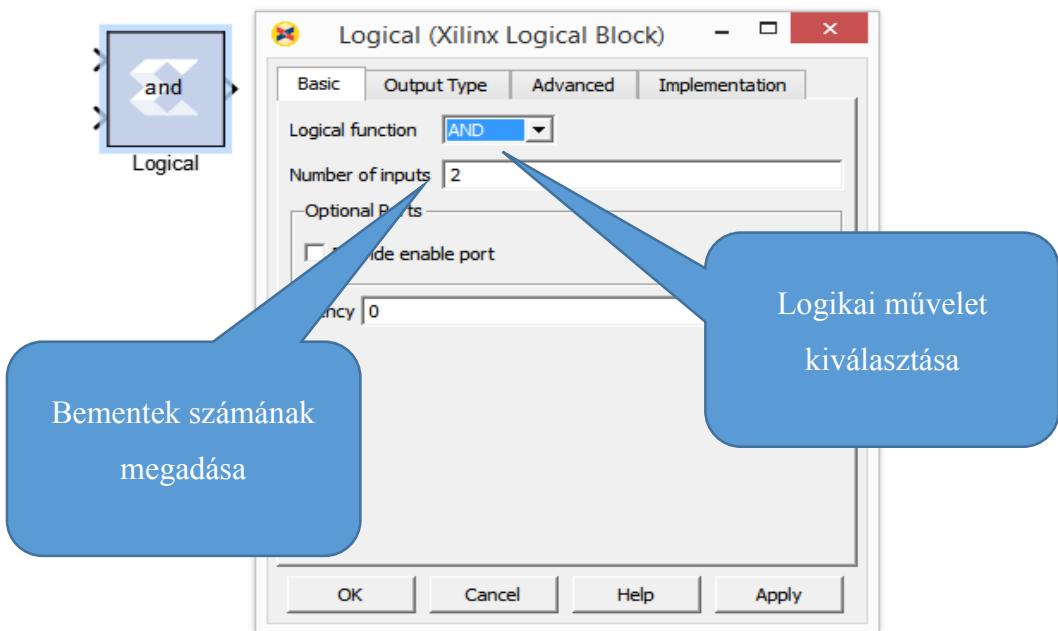
A modulnak meghatározhatjuk a bemeneteinek a számát: $d_0 \dots d_{n-1}$ ahol az n a bennetek száma. A bementeknek hasonlóképpen az AddSub modulnál megadhatjuk a bitszélességét. A multiplexer „sel” bemenetén, amelynek bitszélessége függ a bementek számától $n = 2^k$ ahol k a sel bemenet bitszélesége.



Kép. 4.7 Multiplexer modul és beállításai

Logical modul

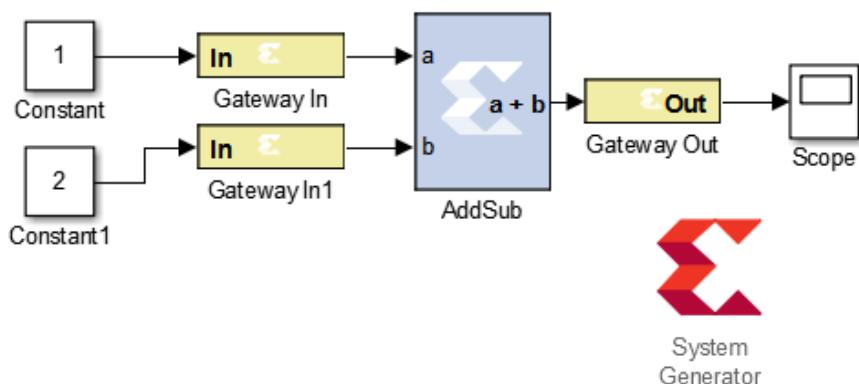
A modul segítségével beépíthetünk a rendszerbe logikai kapukat: AND, OR, XOR, NAND, NOR... amelyeknek megválaszthatjuk a bemeneteinek a számát. Egyszerre több bitre is alkalmazhatjuk, ha beálltjuk a bitszélességet, különben a bemenet bitszélességéhez alkalmazkodik.



Kép. 4.8 Logikai műveletek modul, és beállításai

Gateway In és Gateway Out modulok

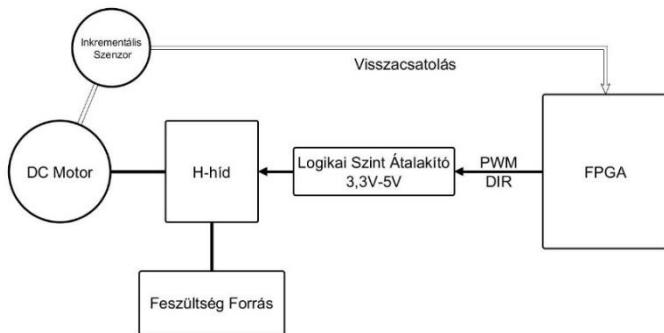
A két modul teremt kapcsolatot a Simulink elemek és a System Generator eszköztár között. Az IN modul segítségével beállított bitszélességű adatokat továbbíthatunk a hardvernek vagy a szimulációnak. Az OUT modul segítségével adatokat olvashatunk ki a hardverből, az adatok bitszélessége alkalmazkodik a hozzá csatolt System Generátoros modul bitszélességéhez.



Kép. 4.9 Simulink illesztő modulok

5 RENDSZER TERVEZÉSE

5.1 SZABÁLYOZÓK:



Kép. 5.1 a Pozíció és a sebesség szabályzásai hurok elvi strukturális felépítése

Visszacsatolást inkrementális érzékelő segítségével valósítottam meg.

5.1.1 Diszkrét Hardveres PID szabályozó

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

$$u_k = u_{k-1} + Q_0 e_k + Q_1 e_{k-1} + Q_2 e_{k-2} \quad (1)$$

$$Q_0 = K_P \left(1 + \frac{T_d}{T_s} + \frac{T_s}{T_i} \right), Q_1 = K_P \left(-1 - \frac{2T_d}{T_s} \right), Q_2 = K_P \left(\frac{T_d}{T_s} \right) \quad (2) [1]$$

Az általam elkészített PID szabályozó hardveresen van megvalósítva FPGA áramkörben, a minél kisebb mintavételezési periódus elérése céljából. A fent látható összefüggések (2) alapján egy adat utas automatát terveztem, amelyet majd System Generátorban építettem meg. A PID szabályozó paramétereit, a Q paraméterek segítségével adhatjuk meg, amelyek függenek az ismert paraméterektől: T_d - deriválási idő, T_i - integrálási idő, T_s mintavételezési periódus, valamint K_p proporcionális erősítés.

Az automata öt állapotot tartalmaz. minden mintavételre, az automata végigpörög az állapotokon és majd visszatér a kiinduló állapotba. Az állapotokban végzet műveletet az FPGA fejlesztő lap órajelének a 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.

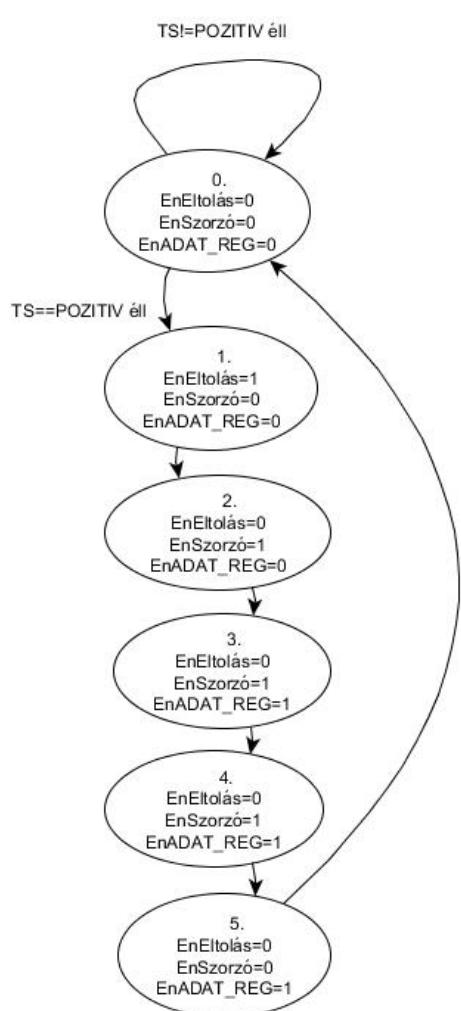
Az automata mindaddig 0 állapotban van, amíg a TS szignálon nem érkezik egy felfutó él. A 1. állapotban végrehajtja az e_k regiszterek eltolását, azáltal hogy EnEltolás

A DC motor valamint a hozzá csatolt mechanizmus pozíció és a sebesség szabályozására az egyhurkú kialakítást választottam Kép. 5.1. A feszültségen vezérelt DC motorok beavatkozó jelét PWM generátorral állítom elő. A

jelet logikai 1 re állítja, vagyis $e_{k-2} = e_{k-1}, e_{k-1} = e_k$ és e_k 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:



Kép. 5.2 Állapot automata, amely leírja a Diszkrét PID szabályzót

maximális értéke 2 lehessen így 0,1,2 értékeket veheti fel. Az adat utakat két 16 bites multiplexerrel MUXQ és MUXE válaszuk ki.

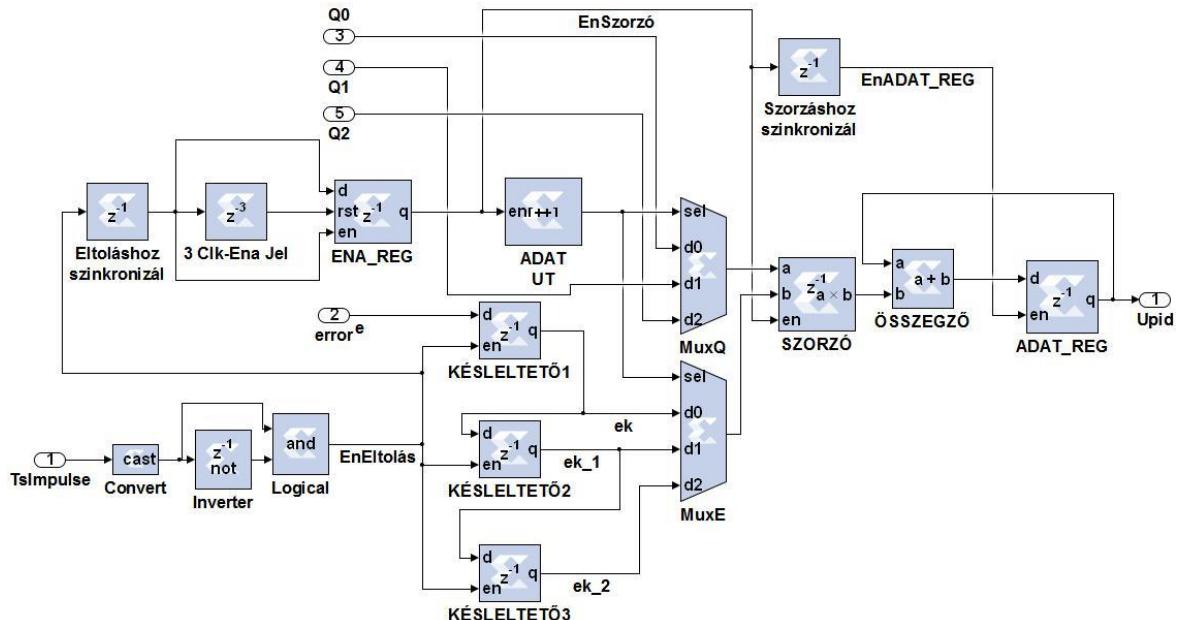
Bemeneti paraméterek a Q_0, Q_1, Q_2 16-bit előjeles egész értékek, e 16-bit előjeles egész érték, T_s -bool típusú.

Kimenetek: U 17bit előjeles egész érték. A MUXQ, a Q paraméterek kiválasztásáért valamint a MUXE az időben késleltetett e bemeneti értékek kiválasztásáért felelős. A Kép.

5.3 a „SZORZÓ” modul a két szelekciós multiplexertől kapott értéket összeszorozza, aztán hozzáadja az „ADAT_REG” regiszter értékéhez.

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

A Kép. 5.3 látható KÉSLELTETŐ regiszterek állítják elő e_k, e_{k-1}, e_{k-2} , múltbeli hiba értékeit, úgy hogy a három regiszter egymás után van láncolva és a T_s felfutó élére a következő regiszterbe csúszik át az érték. A KÉSLELTETŐ1 regiszterbe kerül minden az aktuális mintavételezett hiba értéke.

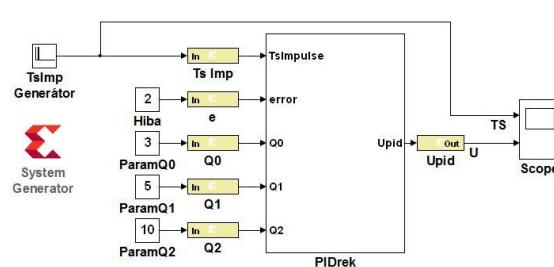


Kép. 5.3 A PID felépítése System Generatorban

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” együt valósítják meg a három órajelig tartó logikai engedélyező jelet, amely a számlálót indítja el.

Az él detektáló elemet egy tagadó kapu „Inverter” valamint egy „és” kapu (Logical) biztosítják, mégpedig úgy, hogy figyeljük egyazon jel előbbi periódus értékeit. 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 „EnSzorzo” engedélyező jeléhez képest. Erre azért van szükség, mert a szorzás eredménye 1 órajelet késik az elindítást követően és az eredményt szeretnénk eltárolni.

5.1.1.2 Simulink szimulációs eredmények

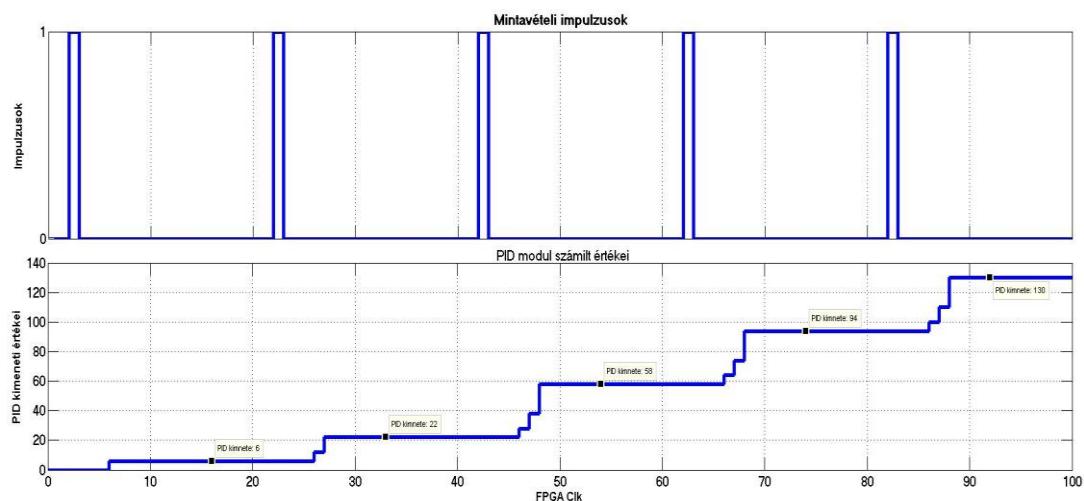


Kép. 5.4 PID Simulink szimulációs model

az adatokat Matlabban tudjuk kezelní.

A szimulációk során a számítások eredményét ellenőriztem le, amelyeket System Generatorban végeztem el Simulink segítségével, az eredményeket majd összehasonlítottam a manuálisan számolt értékekkel (Táblázat. 5-1).

Bemeneti paraméterek: $Q_0 = 3, Q_1 = 5, Q_2 = 10$, a bemenet konstans: $e = 2$



Kép. 5.5 Szimulációs eredmény amely tükrözi a konstans bementre a számolási lépéseket

A Kép. 5.6 látható a 6 szükséges órajel a számítások elvégzésére. Megjegyezném, hogy a szabályozó negatív bemeneti értékekre is működőképes.

ÓRAJEL Q_0 Q_1 Q_2 e_k e_{k-1} e_{k-2} U_k

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

A „PIDrek” modul tartalmazza a Kép.

5.3 képen látható modult, a bemenetekre és a kimentre illesztünk egy-egy konvertáló elemet, mely segítségével adatokat közölhetünk, vagy nyerhetünk a megtervezett Xilinx System Generátoros hardverrel, amely

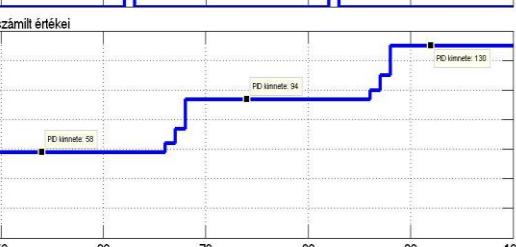
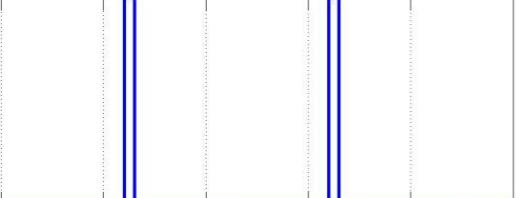
a szimulációk során a számítások eredményét ellenőriztem le, amelyeket System

Generatorban végeztem el Simulink segítségével, az eredményeket majd összehasonlítottam

a manuálisan számolt értékekkel (Táblázat. 5-1).

Bemeneti paraméterek: $Q_0 = 3, Q_1 = 5, Q_2 = 10$, a bemenet konstans: $e = 2$

Mintavételi impulzusok

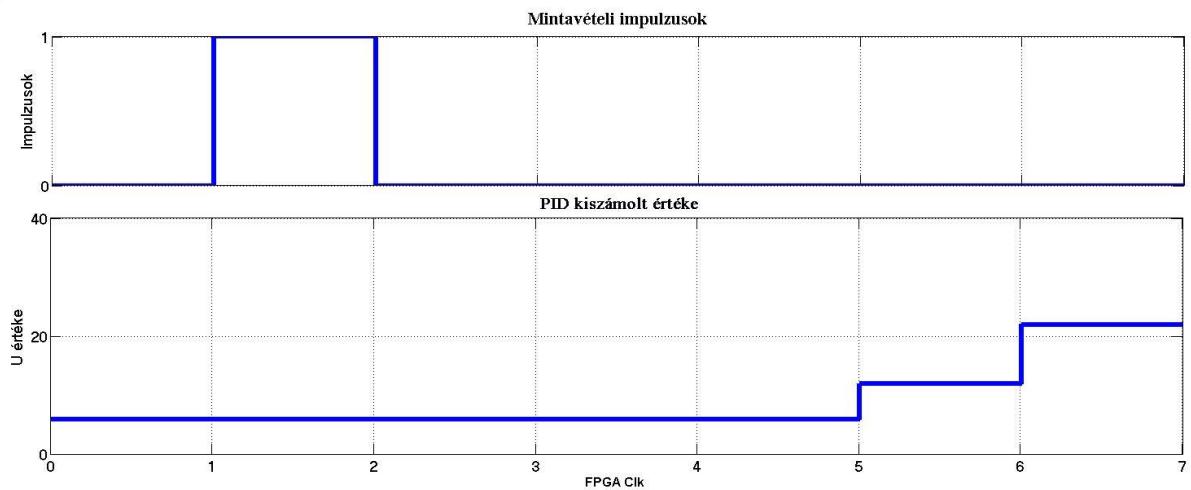


Kép. 5.5 Szimulációs eredmény amely tükrözi a konstans bementre a számolási lépéseket

A Kép. 5.6 látható a 6 szükséges órajel a számítások elvégzésére. Megjegyezném,

hogy a szabályozó negatív bemeneti értékekre is működőképes.

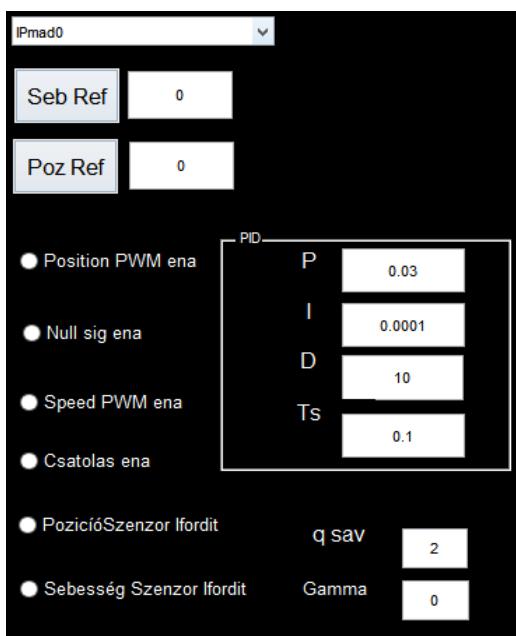
Táblázat. 5-1 Manuálisan számolt értékek a szimuláció ellenőrzésére



Kép. 5.6 PID minimális periódusa

A Szabályozó az elvártaknak megfelelő eredményeket töríti 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 $6 \times 20\text{ns} = 120\text{ns}$. Következetésképpen a PID szabályozó maximális mintavételezési periódusa 120ns.

5.1.1.3 Q paraméterek számolása Ti, Td, Kp, Ts alapján.



Kép. 5.7 GUI pid paraméterek

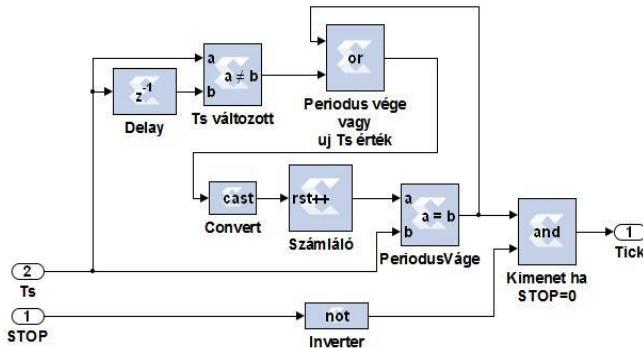
Adottak a (2) összefüggések, látható hogy Ts, Ti megjelennek a nevezőben, így fennáll annak a veszélye, hogy 0-val való osztás történik. Ezért soha ne válasszuk a Ti -t 0-nak, de lehet egy nullához közeli pozitív szám. A Ts mindenkor nagyobb, mint nulla.

A paraméterek értékének beállításával kitudjuk választani a szabályozó típusát is. PI szabályozó esetén a Td paramétert válasszuk 0-nak, mert nem okoz számítási problémát. PD szabályozó esetén célszerű a Ti -nek minél kisebb értéket beállítani, ami nem lehet egyenlő 0-val.

A Kép. 5.7 látható a grafikus felhasználói interfész PID nevű kertjében megadhatjuk a PID paramétereit: P proporcionális erősítés, I integrálási idő, D deriválási idő.

5.1.2 Mintavételezési periódus jelének generálása

Az Kép. 5.8 belső felépítése. A modulban a „Számláló” 32 bites számláló az FPGA órajelére számol, és az értékét összehasonlítjuk a „ T_s ” bemenet értékével. Ha az érték megegyezik, akkor generálódik egy impulzus, amely lenullázza a számlálót. A számláló



Kép. 5.8 Mintavételi taktust generáló modul

akkor is nullázódik, ha megváltozott a „ T_s ” értéke, amelyet a „ $Delay$ ” késleltető és a „ T_s 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 „ $Periodus vége$ ” összehasonlító modul abban a pillanatban, amikor a számláló elérte a „ T_s ” bemenet értékét „reset” állapotba hozza a számlálót.

Az **Error! Reference source not found.** látható a pirossal jelölt „ T_s ” 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.

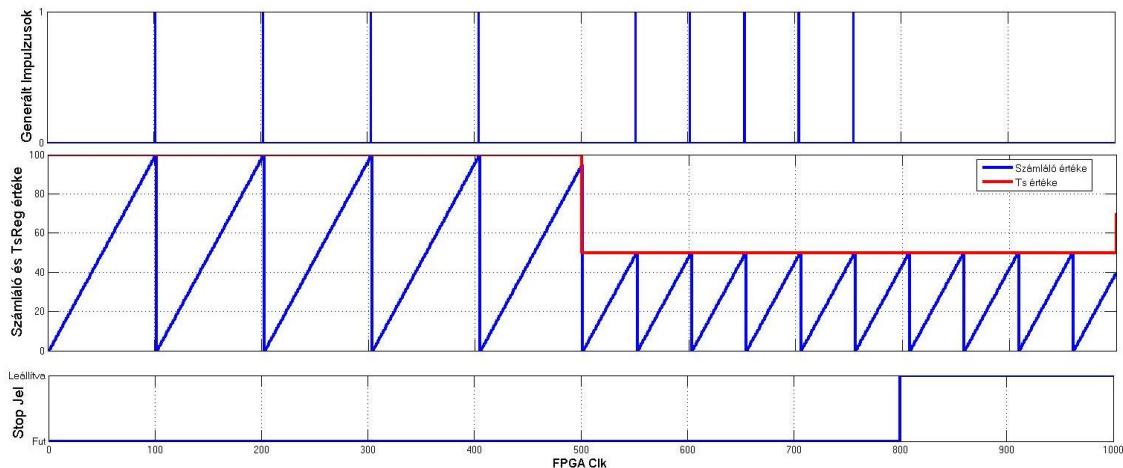
Impulzusok frekvenciája:

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

$$\omega = \frac{10^9}{20 * T_s}$$

A szimulációkat: $T_s = 100 \Rightarrow \omega = \frac{10^9}{20 * 100} = 500kHz$

$T_s = 50 \Rightarrow \omega = \frac{10^9}{20 * 50} = 1MHz$, végeztem el, és a következő eredményeket kaptam ω frekvenziának.



Kép. 5.9 Szimulációs eredmények mintavételi jelgenerátor.

5.1.3 Pozíció Szabályzása

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:

$$\begin{cases} U = U_{MAX}, & \text{ha } e_{sz} > 0, & a1 \\ U = U_{MIN}, & \text{ha } e_{sz} < 0, & a_1 \\ U = 0, & \text{ha } e_{sz} = 0, & a0 \end{cases}$$

$$\begin{cases} \text{ha } ref' = 0 \begin{cases} \text{akkor ha } e > q \begin{cases} \text{akkor } e_{sz} = e, & a1 \\ \text{maskep } e_{sz} = 0, & a2 \\ e_{sz} = e & \end{cases} \\ \text{maskep} \end{cases} \end{cases}$$

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

5.1.3.1 A szabályozó felépítése:

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

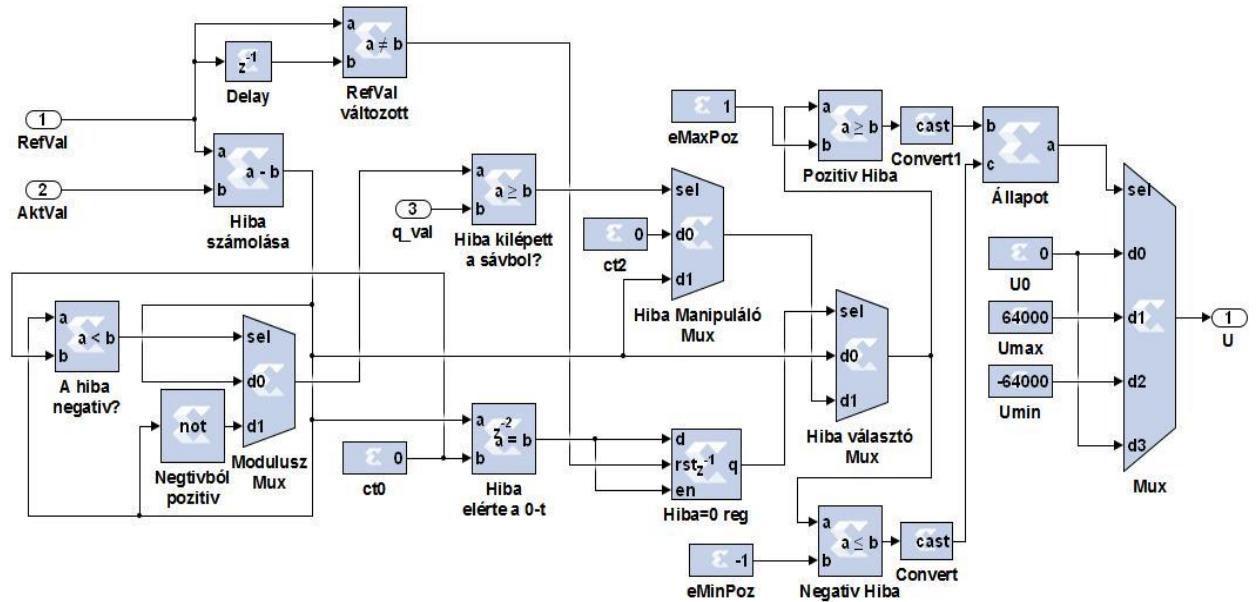
A „Hiba Számolása” modul végzi a hiba kiszámolását az aktuális és az előirt pozícióból. A hiba lehet negatív is ezért „A hiba negatív?” Komparátor segítségével eldöntjük, hogy negatív vagy pozitív a hiba. Azután a „ModuluszMux” segítségével kiválasztjuk a magát a számolt hibát, ha az pozitív, vagy a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

Ha a hiba elérte a 0-t akkor a „Hiba=0 reg” értéke 1 lesz mindenkor amíg az előirt 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.



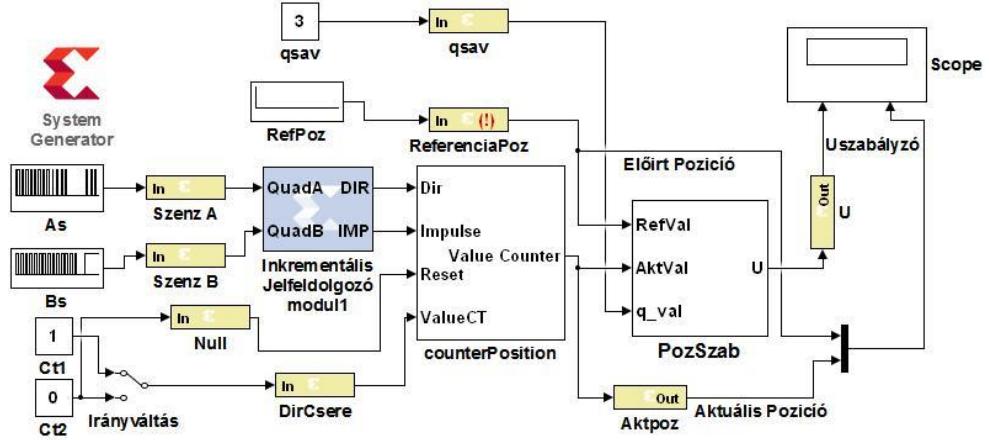
Kép. 5.10 A Pozíció szabályozó System generátoros felépítése

5.1.3.2 Szabályozó szimulálása

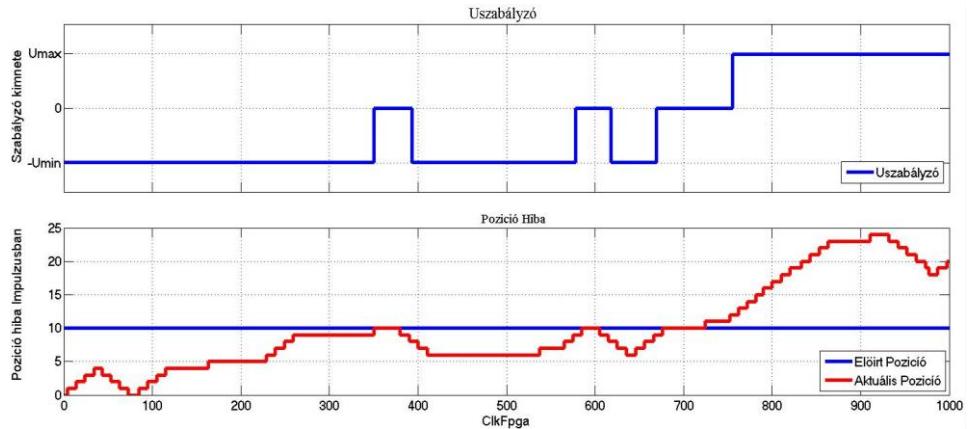
Az Kép. 5.11 látható a szimulációs logika, a „counter Position” és a „Inkrementális Jelfeldolgozó modul 1” megtalálható a pozíció mérése inkrementális adó segítségével. A szabályozót a „PozSzab” modul tartalmazza, belső felépítése a Kép. 5.10 alapján.

A Kép. 5.12 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 a nem idézheti elő az érzéketlenségi sávból való kilépést.

A szabályozó bemente impulzusban van megadva, amelyek az inkrementális tárcsától érkeznek, 1 impulzus megfelel 2 fokos elfordulásnak.



Kép. 5.11 A pozíció szabályzás moduláris felépítése System Generator környezetben



Kép. 5.12 A pozíció szabályozó szimulálása

5.1.4 Hardveres mérések

5.1.4.1 DC motor sebesség szabályzása mérőstandon

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 $\frac{\text{imp}}{\text{Ts}}$ -ben, vagyis impulzus per mintavételben mérjük, így a

referencia sebességet is ebben a mértékegységben kell megadnunk. Ezért átalakítást kell végezünk a következő összefüggés szerint:

N – tárcsa felbontás,

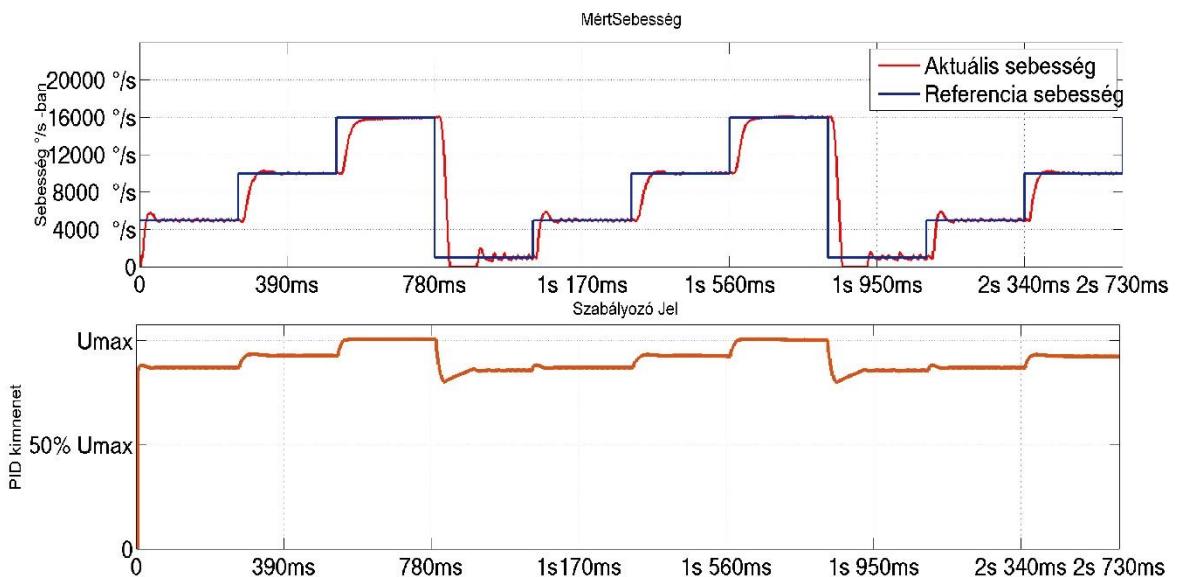
s_{Ts} – mintavételi periódus sec ban, a PID szabályzó mintavételi periódusa

$N_{mérő}$ – mintavételi periódus alatt érkezett impulzusok száma

ω – szögsebesség °/s ban

$$\omega = \frac{N_{mérő} * 360}{N * s_{Ts}}$$

A fenti képletben mindenkor csak egy ismeretlen van attól függően, ha referencia értéket kell számolnunk, akkor az $N_{mérő}$ paramétert fejezzük ki, vagy az ω paramétert.



Kép. 5.13 Sebesség szabályozás PID szabályzóval.

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

5.2 SEBESSÉG ÉS POZÍCIÓ SZABÁLYOZÓT TARTALMAZÓ IP MAG GENERÁLÁSA SYSTEM GENERATOR-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. 5.16 a „*Sebesség Szab*” modult tartalmazza, Kép. 5.25 látható modulokat és a sebesség szabályzására hivatott, míg a „*Pozíció Szab*” tartalmazza a Kép. 5.16 látható modulokat és a pozíció szabályzására 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. 5.29.

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 engedélyezése	PWM engedélyezése	Null sáv engedélyezése	Sebesség engedélyezése	Sebességek csatolása	Pozíció mérés irányváltás

Tábla. 1 Konfig 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 változon meg. A sebesség szabályozó referencia bemenetéhez hozzá kell adni a pozíció változását megszorozva egy arányossági tényezővel.

A Kép. 5.63 látható az 1 forgó talp V_{T_i} sebességet generál az f_i kör mentén, a robot lánctalpának az aktuális sebessége V_{K_i} .

$$V_i = V_{K_i} + V_{T_i}$$

A sebességek összeadását (Kép. 5.17 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 logikai 1-t 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:

$$\gamma = \frac{\text{Forgótalpak hossza}}{\text{nagykerék kerülete}} = \frac{40\text{cm}}{37,68\text{cm}} = 1,06$$

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

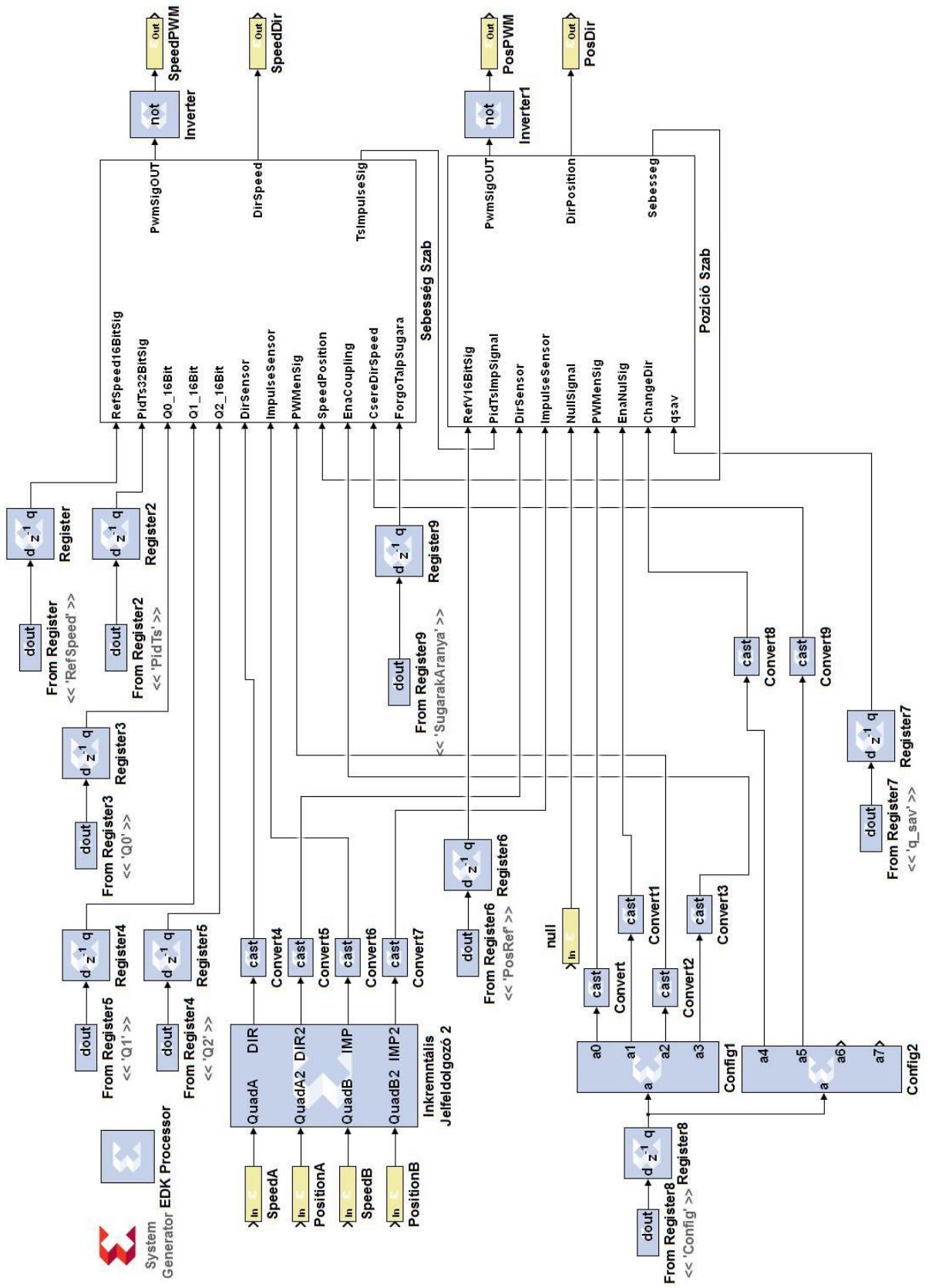
Az osztott regiszterek olyan hardveres elemek, amelyeket az FPGA-ban hozunk létre, szoftveresen a memóriába illesztett címekkel rendelkeznek, amelyeket írhatunk vagy olvashatunk. A System generátorban a *ToRegister* illetve *FromRegister* elemekkel tudjuk kivitelezni. A *ToRegister* típusú regiszterek 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 ír a megadott változóból a megadott memória címre.

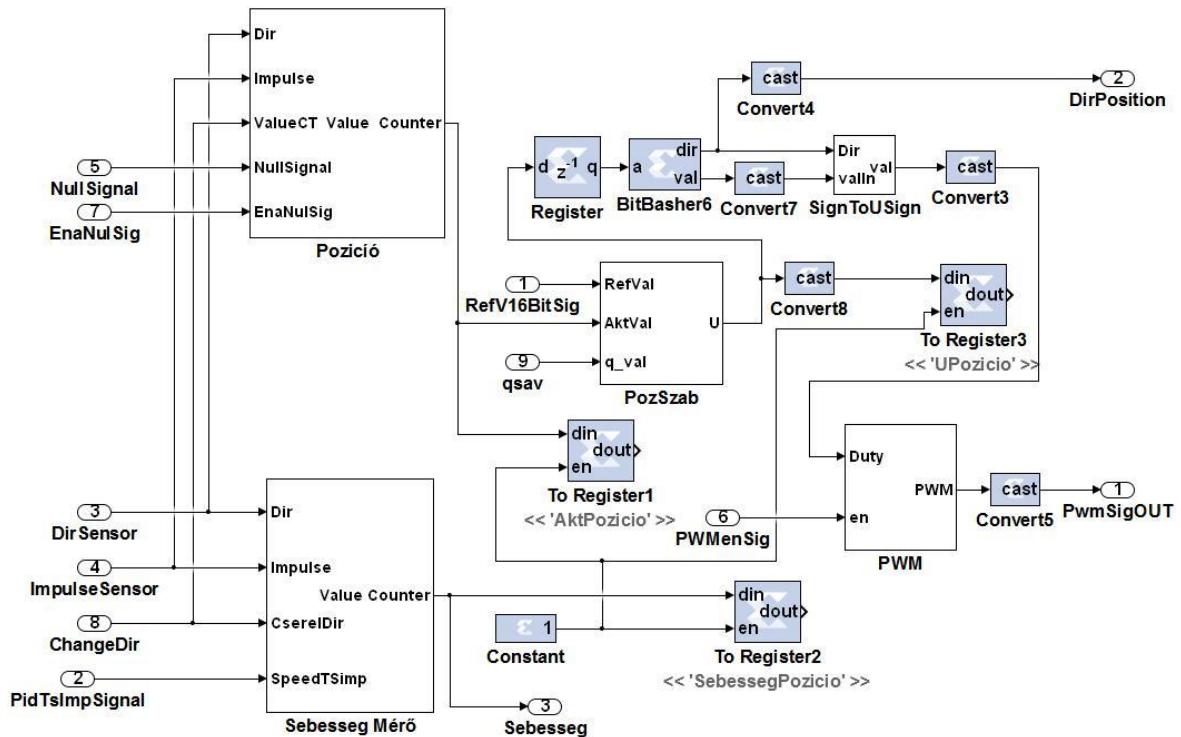
Változó neve =Xil_In32(*regiszter címe*), utasítással egy 32 biten értelmezett értéket olvasunk ki a megadott címről a megadott változóba. 16 bites értékek esetén hasonlóan járunk el annyi különbséggel, hogy a **Xil_Out16**, **Xil_In16** utasításokat használjuk.

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

Az IP magok az FPGA-ban hardveresen vannak összealítvány logikai kapuk és egyéb digitális elemek segítségével, ezért a benne található modulok minden az FPGA 50Mhz órajelére működnek. Az alábbi Kép. 5.14 a sebesség és pozíció szabályozást tartalmazó IP mag System generátoros felépítését mutatja be.



Kép. 5.15 Sebesség és pozíció szabályzását tartalmazó IP mag System generátoros felépítése



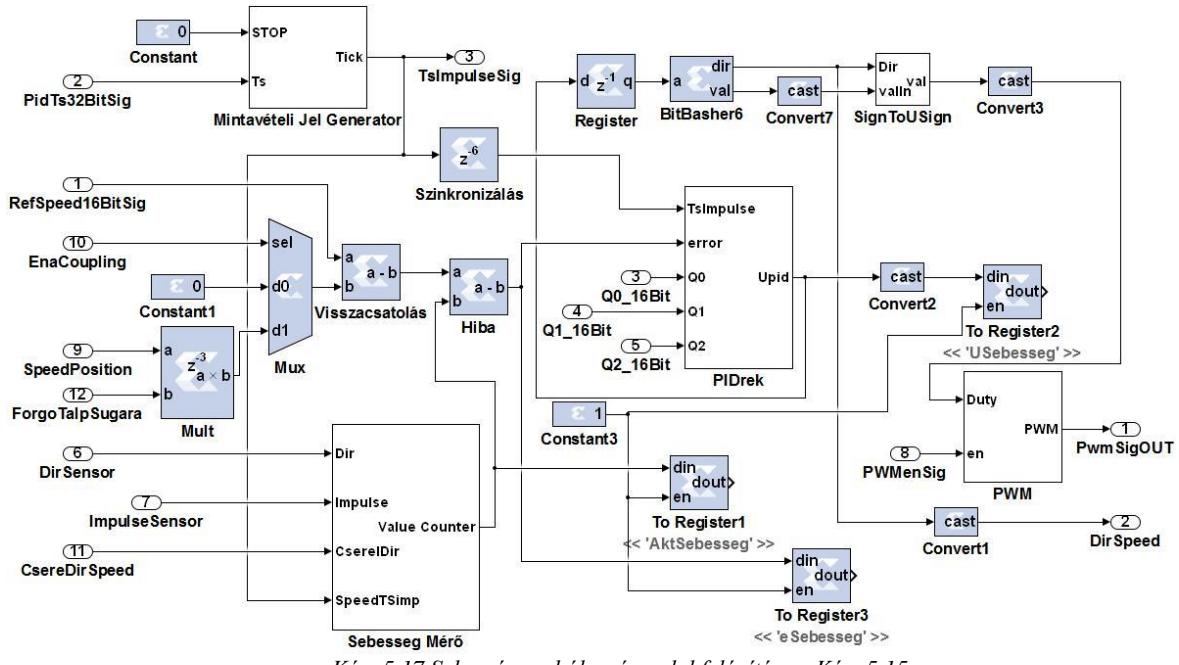
Kép. 5.16 Pozíció szabályozó modul belső felépítése a Kép. 5.15

Pozíció szabályozó modulok:

- A „Pozíció” nevű modul a Kép. 5.32-en látható, feladata a relatív pozíció mérése impulzusban, kimenete egy 16 bites előjeles szám.
- A „Sebesség Mérő” modul (Kép. 5.33 moduljai) feladata a sebesség mérése impulzus per mintavételben, a kimenete egy 16 bites előjeles szám.
- A „Pozíció Szab” nevű modul (Kép. 5.10 moduljai) feladata a pozíció szabályozása.
- A „PWM” nevű modul (Kép. 5.40 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 jelé, 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 „Pozíció” modul segítségével megkapjuk a szöget, amelyben a Forgóalp pillanatnyilag áll. A Sebesség Mérő visszatéríti a pozíció deriváltját, a szögsebességet impulzus/mintavételben. A szögsebesség mérése szinkronizálva van „PidTsImpSignal”

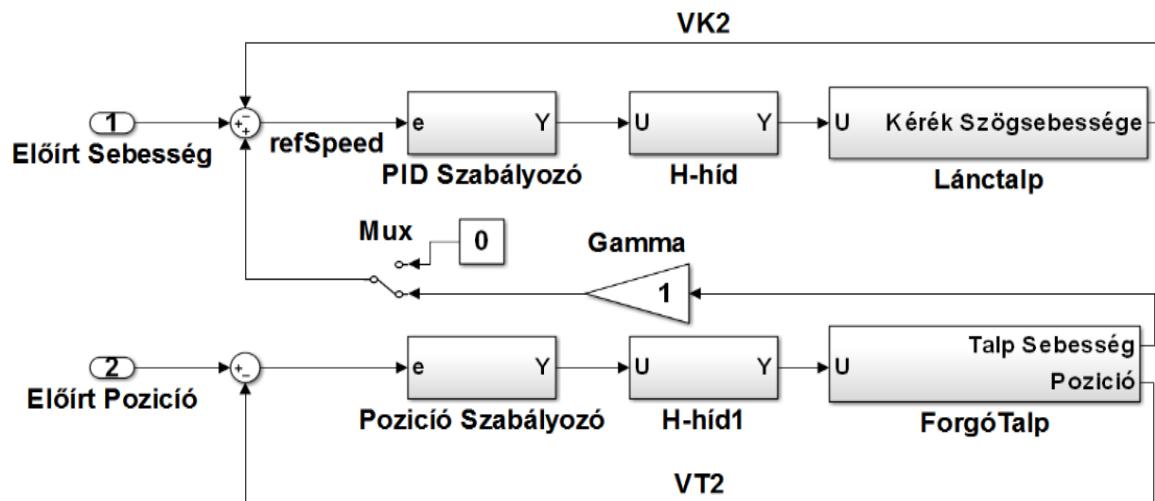
bemeneten keresztül a PID szabályozó mintavételi periódusával, amelyet a Kép. 5.17 látható „Mintavételi jel Generátor” állít elő.



Kép. 5.17 Sebesség szabályozó modul felépítése a Kép. 5.15

A „Sebesség Szab” modul felépítése hasonló a pozíció szabályozó felépítésére, annyi eltérés van, hogy itt nem jelenik meg pozíció mérés csak sebesség. Viszont megjelenik a „Mintavételi Jel Generátor” (lásd Kép. 5.8), melynek feladata, hogy biztosítja a mintavételi periódust a sebesség mérő modulok PID szabályzók számára.

A „PIDrek” nevű modul tartalmazza a pid szabályozót a Kép. 5.3-an látható kialakításban.



Kép. 5.18 Szabályzó körök összekapcsolásának elvi kialakítása

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

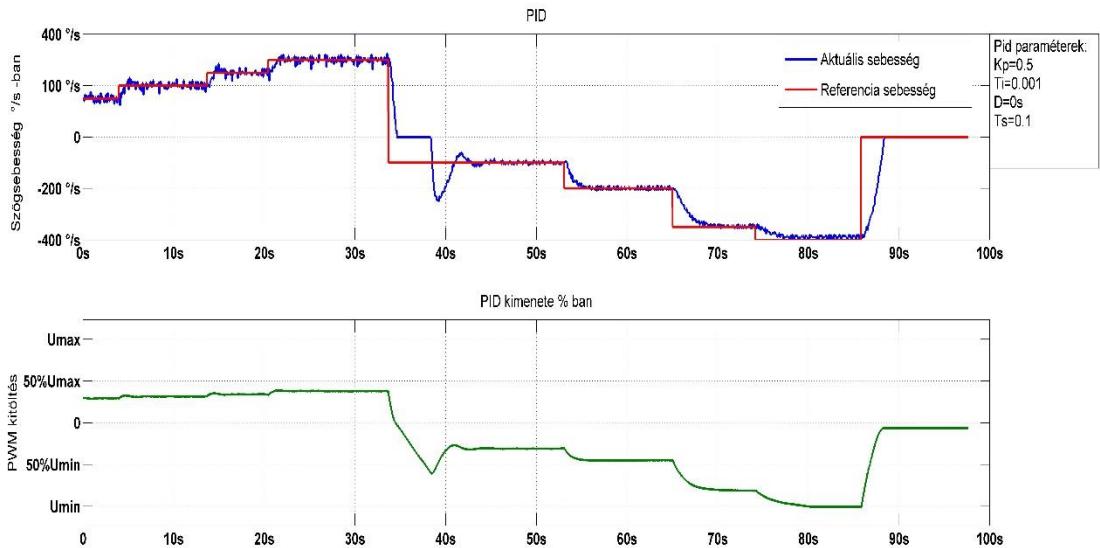
A Kép. 5.63 látható alsó ábrán a pozíció változása V_{T_2} sebességet generál a kis keréknek, ha mi a V_{K_2} sebességet szeretnénk tartani, akkor a következő a teendő: előírjuk a sebesség szabályzónak hogy ne változon meg a sebesség: $\text{refSpeed} = V_{T_2} * \gamma + V_{K_2}$, ahol az γ egy arányos ági tényező.

5.2.1 Mérések a rendszeren

5.2.1.1 A robot lánctalpának sebesség szabályozása

Első lépésként megpróbáltam beállítani a szabályozó paramétereit kézzel. Az Kép. 5.19 látható mérési eredményeket kaptam szabályzás közben.

A PID paraméterei: Kp:0.5, Ti:0.001s, D:0s, Ts:0.1s



Kép. 5.19 PID szabályozó a robot lánctalpának a sebességét szabályozva.

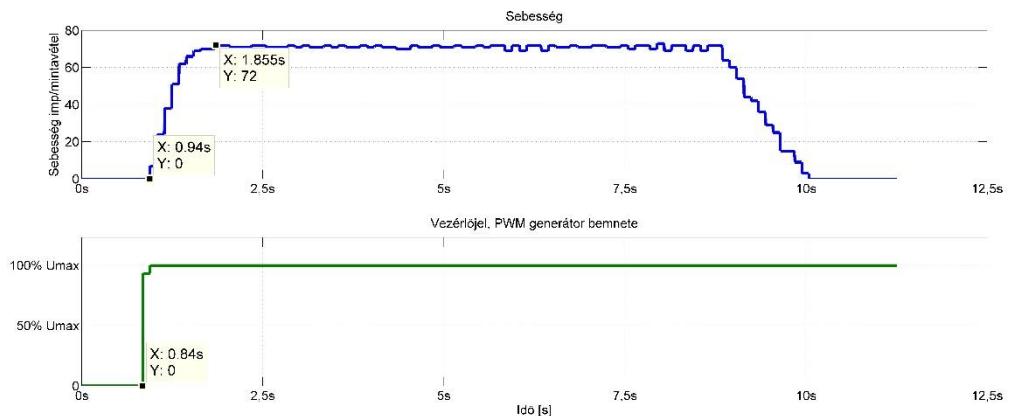
A fenti képen látható a lánctalpak sebességének a szabályozása kúpkerék áttételen keresztül.

A PWM generátorok, minden a sebesség minden 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

Hangolás Oppelt módszerrel

A motorra maximális vezérlőjelet kapcsoltam kiiktatva a szabályozót, és mértem a rendszernek a szögsebességét (impulzus/mintavételi idő). A motorra a vezérlőjelet a megépített PWM generátor segítségével kapcsoltam rá H-hídon keresztül. A PWM generátor bemenetét is elmentettem, látható a Kép. 5.20.

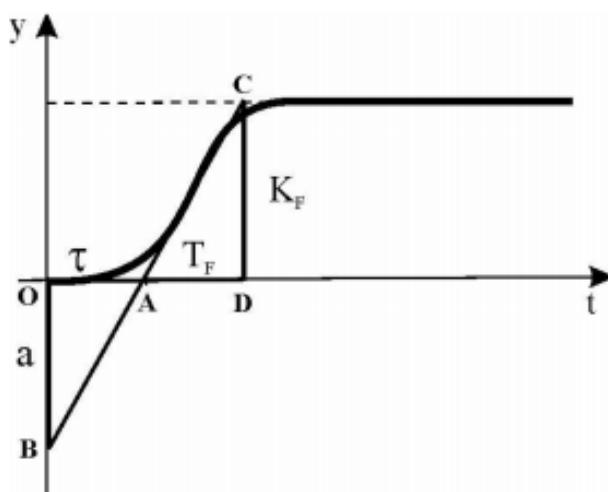


Kép. 5.20 DC motor és a Kup fogaskerék áttétel szögsebessége maximális vezérlőjre. A mérés során a rendszer: holtideje: $\tau = 0.1\text{s}$, időállandója $T_F = 0.915\text{s}$

$$\text{erősítése: } K_F = \frac{72}{100} = 0.72 \quad \frac{K_F}{T_E} = \frac{a}{\tau} \Rightarrow a = \frac{K_F * \tau}{T_E} = \frac{0.72 * 0.1}{0.915} = 0.078$$

A Táblázat. 4-2 alapján a PID paramétereinek az értékei:

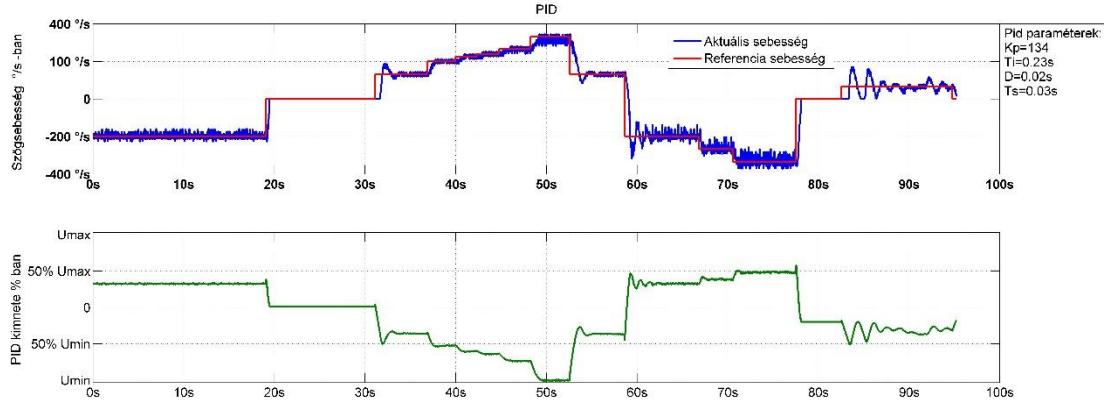
$$Kp = 15,3 \quad Ti = 0,2s \quad Td = 0,024$$



Kép. 5.21 A rendszer egységugrásra adott válasza és megközelítése egyenesekkel.

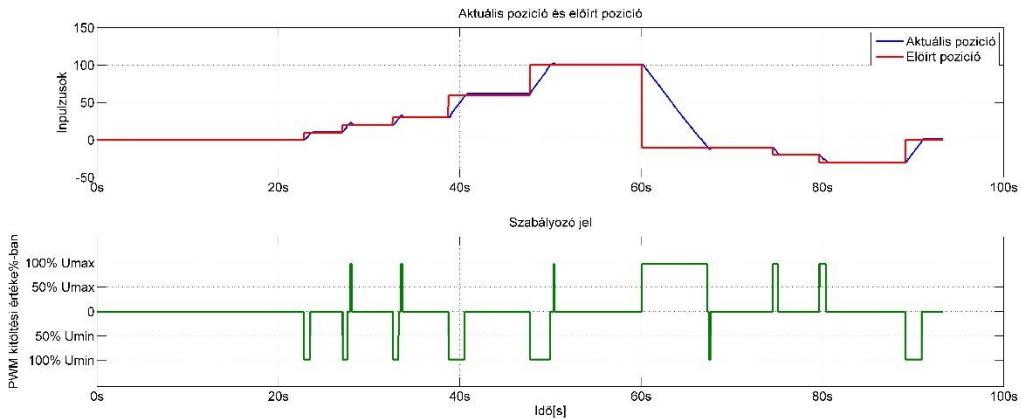
Mintavételeles megvalósításnál a rendszer mintavételezési periódusát a $T_s \cong 0.3\tau = 0,03$ s értékre választottam.

Mért eredmények a hangolás után:



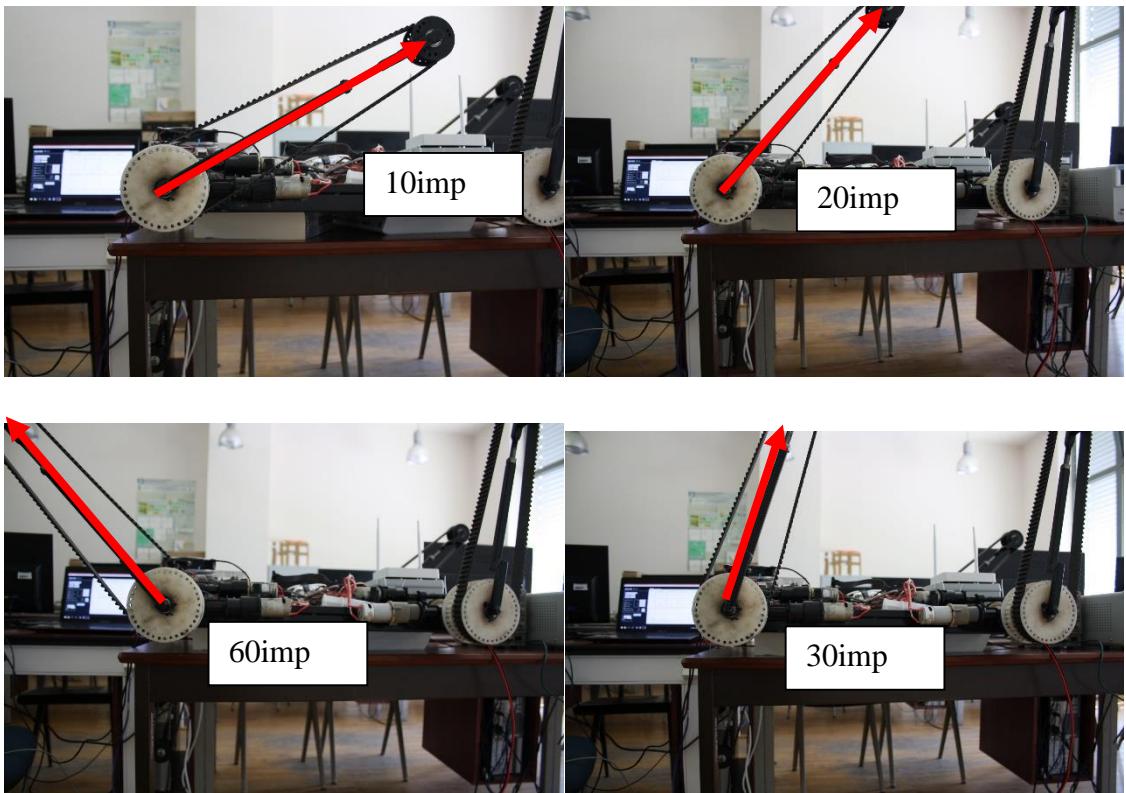
Kép. 5.22 Sebesség szabályzása PID-del Oppelt hangolási módszer után

5.2.1.2 Pozíció szabályozása



Kép. 5.23 Pozíció szabályozás csiga fogaskerék áttételen keresztül

A Kép. 5.23 Pozíció szabályozás csiga fogaskerék áttételen keresztül Kép. 5.23 látható a felső ábrán mért és az előírt pozíció impulzusokban megadva, az alsó ábrán a beavatkozó PWM jel %-ban megadva. A szabályozónak különböző pozíciókat írtam elő: 10imp, 20imp, 30imp, 60imp. Az alsó képeken látható a rendszer forgóalpának a pozíciója különböző értékekre.



Kép. 5.24 Forgóalp pozíciója szabályzás közben

5.3 SZENZOROK

5.3.1 Inkrementális érzékelő

5.3.1.1 Optikai inkrementális vevő felépítése

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

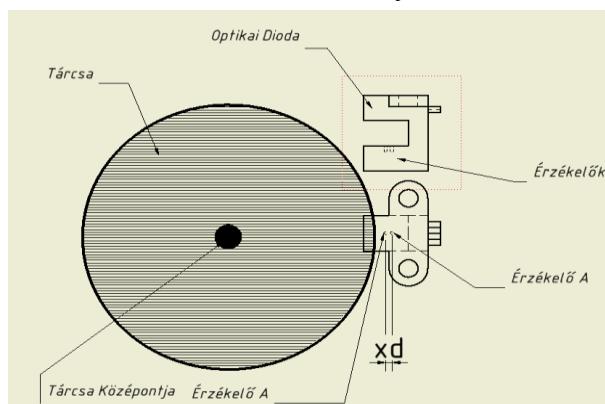
Két vezeték segítségével táplálhatjuk be a piros (3,3V-5V), fekete (GND), a sárga és a kék vezetékek kimeneti jelek az érzékelőtől.

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

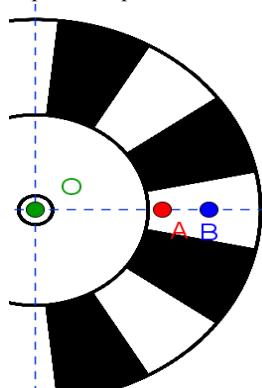
Az érzékelő számára a tárcsát Kép. 5.25 látható módon kell illeszteni.

Könnyen belátható, hogy a tárcsan a rések mérete és dőlés szöge befolyásolja az A, B jelek időbeni eltolását. A könnyebb kivitelezés kedvéért, a tárcsákat lézeres nyomtató

segítségével átlátszó fóliára szeretnénk nyomtatni.



Kép. 5.25 Optikai inkrementális vevő felépítése és elhelyezése



Kép. 5.26 Érzékelő tranzisztorok elhelyezése

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

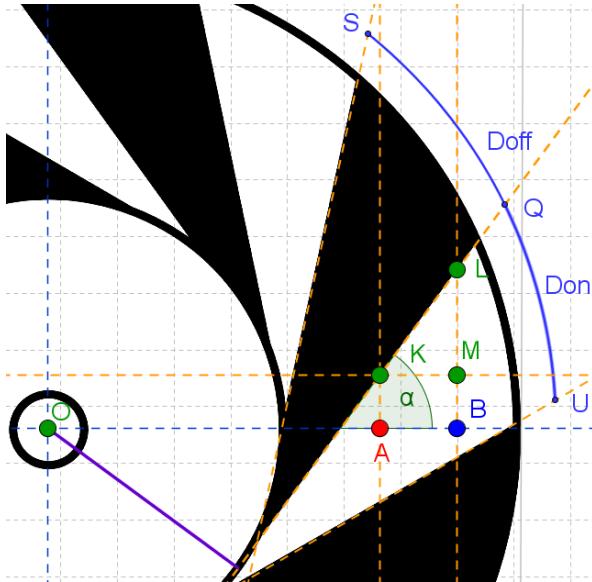
Az O pont az inkrementális tárcsa középpontja, amely körül Omega szögsebességgel forog.

Az ábrán (Kép. 5.26) a fehér mezők az inkrementális tárcsa réseinél képviselik. A rések száma megadja, a tárcsa felbontását N.

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

Ha az A, B, O pontok egy egyenesen találhatók (könnyebb az érzékelő felfogatása), akkor meg kel dölteni a réseket az A, B pontok által meghatározott egyeneshez képest α szöggel (Kép. 5.27).

Ismert adatok: R – tárcsa sugara,



Kép. 5.27 Rések és az érzékelők közti kapcsolat

N – tárcsa felbontása, Don – (QU) a résekhez tartozó körív hossza,

$Doff$ – (SQ) a sötétmezőkhöz tartozó körív

x_d – (A és B pontok közti távolság) érzékelők közti távolság,

h – LM szakasz hossza.

A logikai magas és alacsony állapot közti arány egyenesen arányos a Don és a $Doff$ szakaszok arányával, amint látható a bal oldali ábrán.

Az érzékelő Kép. 5.27 módon van illesztve a tárcsához, akkor felírható az összefüggés, amely meghatározza a két jel közti késést.

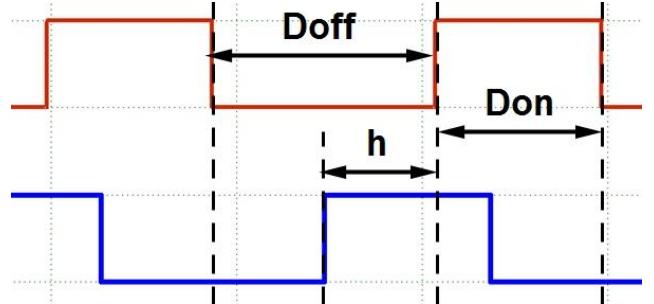
$$h = \tan(\alpha) x_d, \alpha = [0^\circ, 90^\circ], SU = \frac{2\pi R}{N}, SU = Don + Doff.$$

A h -nak minden esetben nagyobbnak kell lennie, mint a x_d távolságnak.

5.3.2 Inkrementális érzékelő jeleinek a feldolgozása FPGA áramkör segítségével

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

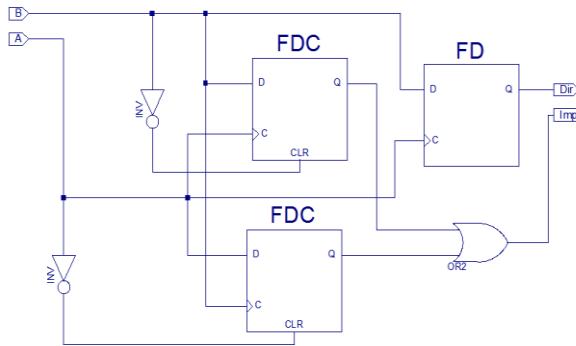
A modul VHDL programnyelven készült a Kép. 5.29 látható kialakítás szerint. BlackBox modul segítségével integráljuk a System Generátor környezetben (BLACK BOX1). A ki és bemeneti portókat illesztjük a System Generátor környezetben található elemekhez, majd létrehozzuk a szimulációs bemeneti jelet, amelyeket az A, B sárgával jelölt



Kép. 5.28 Idődiagram a Tárcsa paraméterei függvényében

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).

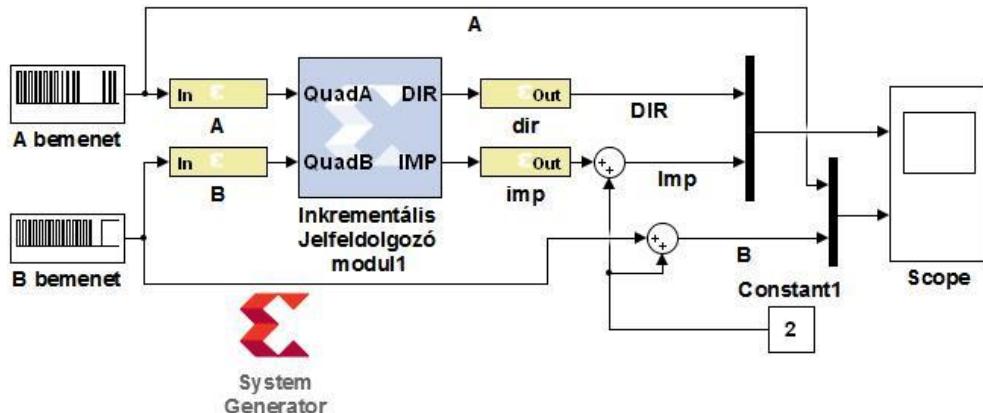
A beérkező A és B jeleket egy tagadó kapu segítségével bekötjük a FDC tárolók CLR bemenetére, a Kép. 5.29 látható módon. Egy harmadik tároló segítségével meghatározhatjuk a forgás irányát. Az „*imp*” impulzus kimenete akkor lesz logikai 1, ha valamely FDC tároló Q kimenete is logikai 1 lesz.



Kép. 5.29 Inkrementális jelfeldolgozó modul1 érzékelő modul
belsı felépítése

5.3.2.1 Szimuláció System Generatorban

Az ábrán (Kép. 5.31) látható a szimulációs eredmények az A és B bemeneti jelek (alsó ábra), *Dir* és *Imp* kimentek (felső ábra).

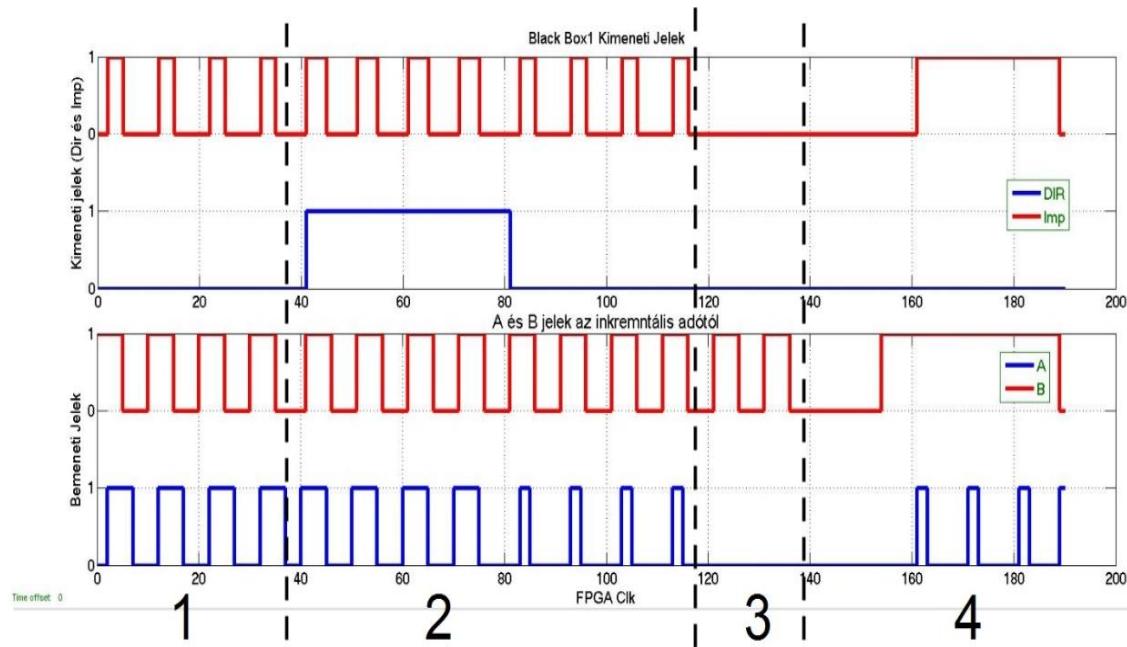


Kép. 5.30 Inkrementális érzékelőtől érkezı jelek átalakító irány és impulzus jelekre

A bemeneti jelek négy kategóriába sorolhatók:

1. Az A jel késik a B jelhez képest, a kimeneti jeleken látható az 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 fotótranzisztor.



Kép. 5.31 Szimulációs eredmények a lehetséges bemenetekről az Black Box1 modulba

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

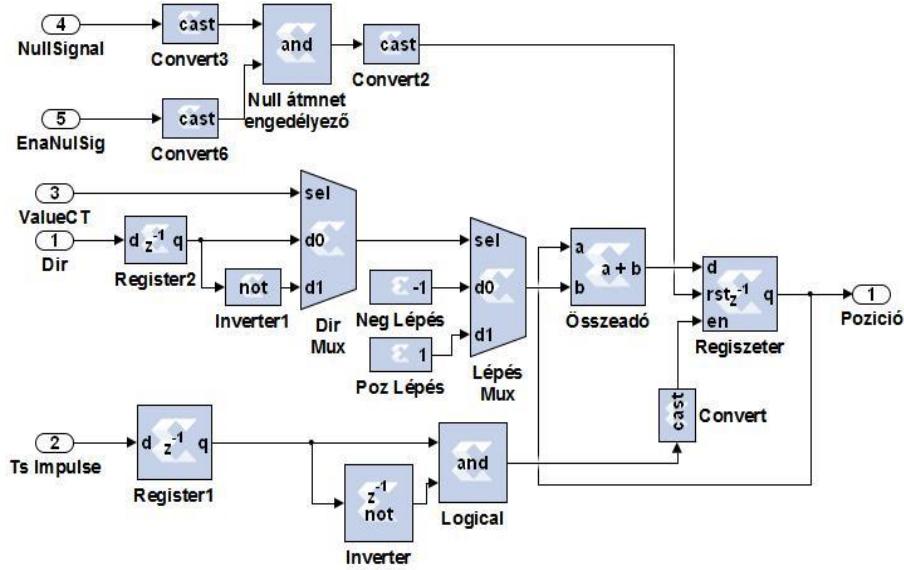
5.3.2.2 Pozíció mérése inkrementális adó segítségével

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 mindenig 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.

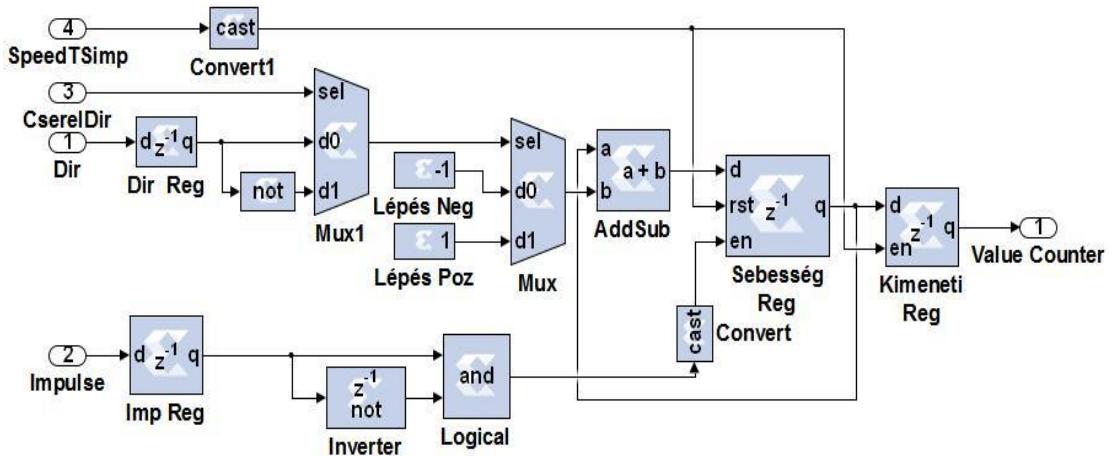


Kép. 5.32 Inkrementális adóval mért pozíció, szimulációs modellje SystemGeneratorban

5.3.2.3 Szögsebesség mérése inkrementális adó segítségével

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

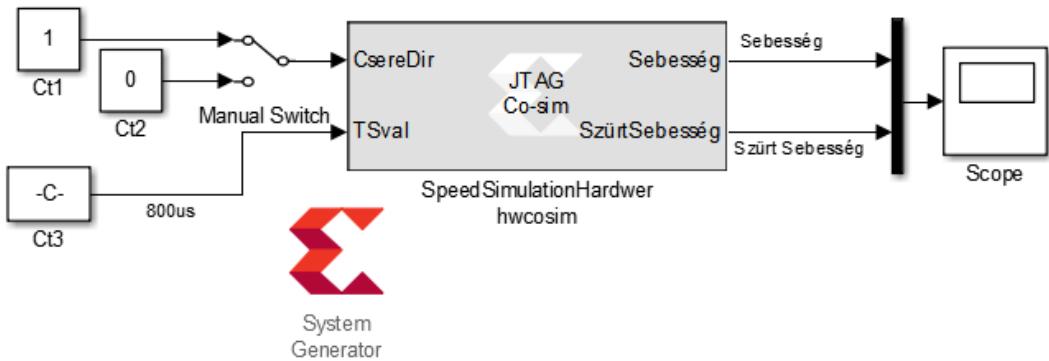
A tároló regiszter minden mintavételi impulzus érkezésekor „reset” állapotba kerül, így az értéke 0 lesz. A „*CsereDir*” bemeneten keresztül megtudjuk változtatni a pozíció előjelét, erre a robothoz rögzített koordináta rendszerhez való illesztéskor lesz szükséges.



Kép. 5.33 Sebesség mérő modul felépítése

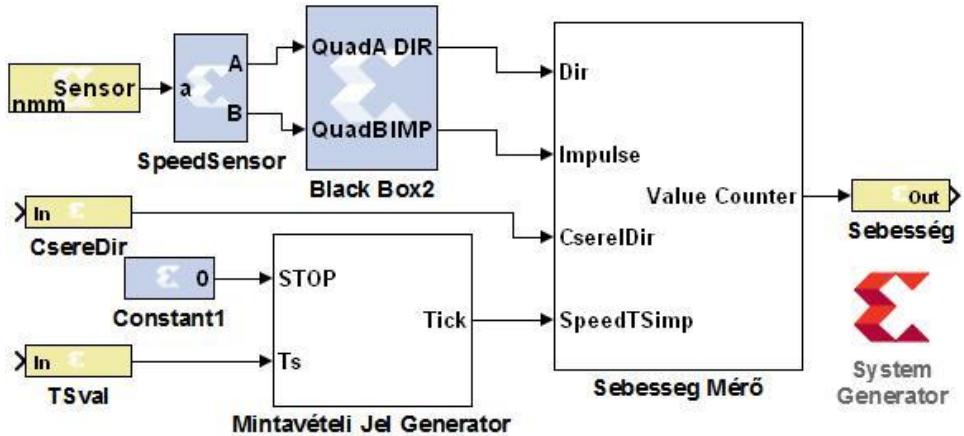
5.3.2.3.1 Hardveres mérések

A mérések során az ábrán (Kép. 5.38) 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. 5.34 DC motor sebességének mérése FPGA lapon

Az ábrán (Kép. 5.35) 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 *Tper* periódusú impulzusok generálása, a periódust *TSval* bemeneten adhatjuk meg. A *Tper* kiszámolható ms-ban az alábbi összefüggéssel.



Kép. 5.35 Dc motor Sebesség mérése FPGA rendszeren, System generatorban megvalósítva

$$T_{per} = T_{Sval} * CLK_{per}$$

A terv kigenerálása után kapunk egy újabb modult SpeedSimulationHardware 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 az aktuális változat nem tartalmazza, mivel a mérések egy előző verzióban készültek.

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

$$\text{Jelölések: } \omega_{mérő} = 65, \left\{ \frac{\text{Imp}}{T_s} \right\}. \text{ A tárcsa réseinek száma: } N = 180, T_s = 8ms$$

$$\omega_{fordulat} = \frac{\omega_{mérő} * 60 * 10^3}{N * T_s} = 2700, \{RPM\}$$

A fenti példában a $\omega_{mérő}$ az maximális fordulat, azért mert a motor a maximális megengedett feszültséggel volt táplálva, terhelés nélkül.

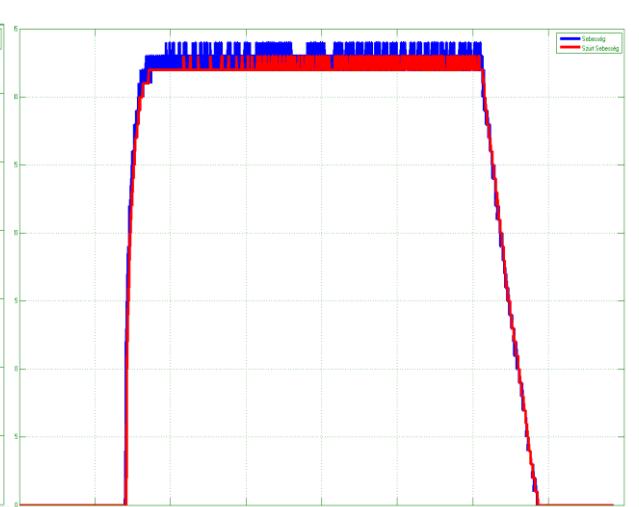
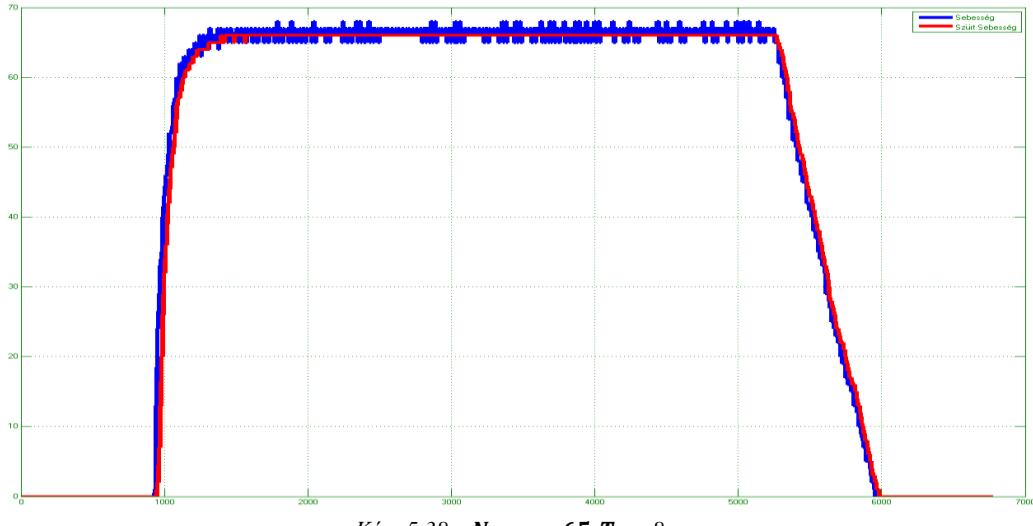
Ha ismerjük a motor maximális fordulatszámát, kiszámolható mérés felbontása.

A fenti példa esetében a felbontás 65 mivel a motor maximális fordulatszáma és 0 között 65 egész értéken van Kép. 5.38.

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

$$T_s = \frac{Res * 60 * 10^3}{RPM_{max} * N}, \{ms\}$$

A Kép. 5.38.b képen a mintavételi időt 80 ms-ra növelte, megnőtt a felbontás is megközelítőleg 650 re. A Kép. 5.38 c csökkentjük a mintavételi időt és ez megközelítőleg 33-ra csökkentette a rezolúciót.



5.4 MPU-6050 GIROSZKÓP ÉS GYORSULÁSMÉRŐ

Az érzékelő modulban 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 [9] adatlap alapján a következő beállításokat végeztem el:

- FIFO memóriák kikapcsolása FIFO_EN=0x00
- Gyorsulásmérők indítása ACCEL_CONFIG=0xE7

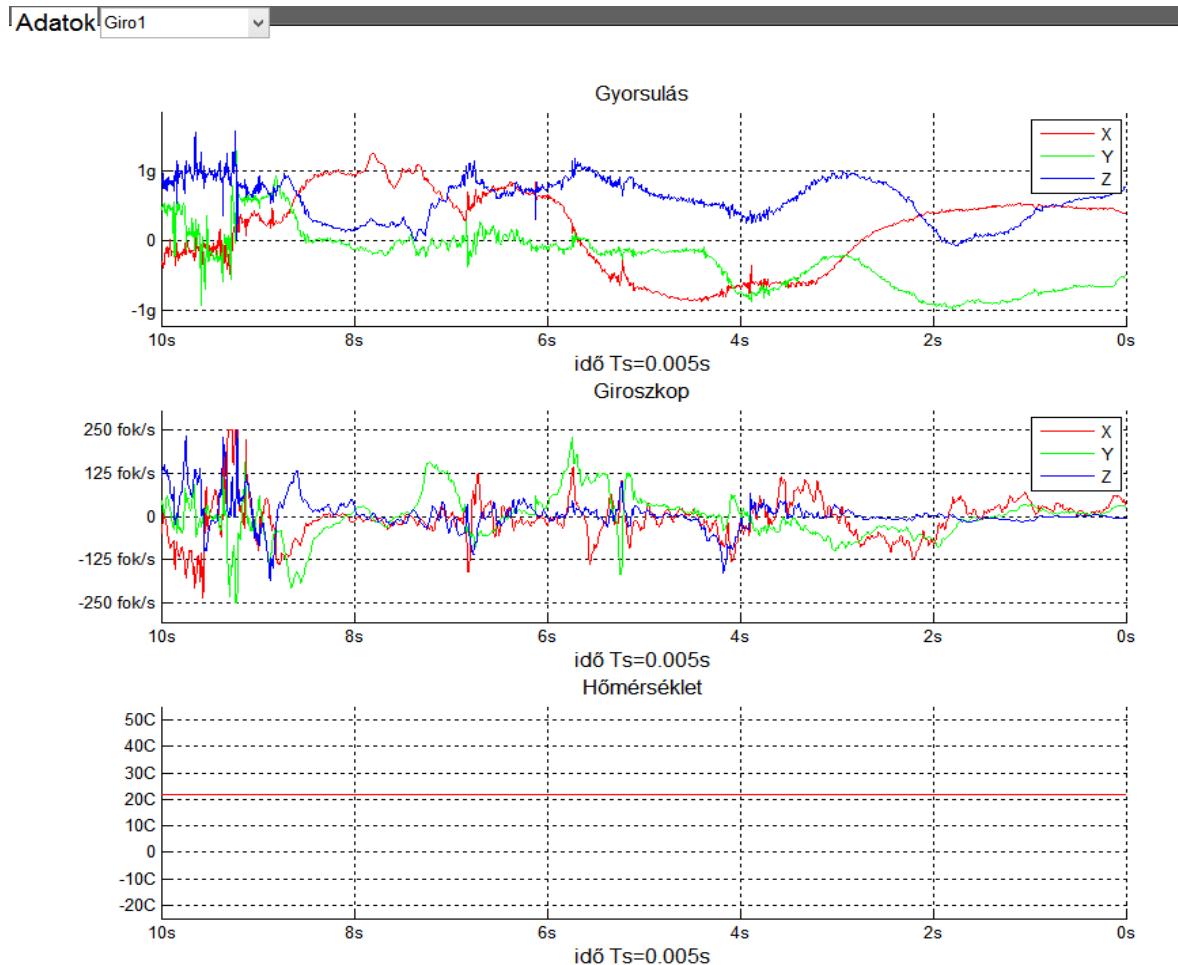
- PWR_MGMT_1 =0x00 beállítjuk a szenzort ciklikus működésre és 8MHz órajelre. A giroszkóp mérési határai változtathatóak: ± 250 , ± 500 , ± 1000 , $\pm 2000 \text{ }^{\circ}/\text{s}$ a giroszkóp, illetve $\pm 2\text{g}$, $\pm 4\text{g}$, $\pm 8\text{g}$, $\pm 16\text{g}$ gyorsulási skálák közül választhatunk. A gyorsulást a FS_SEL 2 bites regiszter rész írásával tudjuk állítani, a AFS_SEL 2 bites regiszter rész írásával állíthatjuk a gyorsulásmérő skáláját.

FS_SEL=3 $\pm 2000 \text{ }^{\circ}/\text{s}$	AFS_SEL 0 $\pm 2\text{g}$
FS_SEL=1 $\pm 500 \text{ }^{\circ}/\text{s}$	AFS_SEL 1 $\pm 4\text{g}$
FS_SEL=2 $\pm 1000 \text{ }^{\circ}/\text{s}$	AFS_SEL 2 $\pm 8\text{g}$
FS_SEL=0 $\pm 250 \text{ }^{\circ}/\text{s}$	AFS_SEL 3 $\pm 16\text{g}$

Az érzékelőbe beépített hőmérő $^{\circ}\text{C}$ ban mért értékét megkapjuk az alábbi összefüggés alapján:

$$T = \left(\frac{\text{Tempreg}[15:8] \ll 8 + \text{Tempreg}[7:0]}{340} + 36.53 \right)$$

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



Kép. 5.39 Giroszkóp mért adatainak az ábrázolása a GUI program segítségével

A Kép. 5.39 láthatjuk a mért adatokat, amelyet a GUI felületen rajzoltam ki, a jeleket 0.005s mintavételezési periódussal mintavételeztem.

5.5 BEAVATKOZÓ ELEMEK:

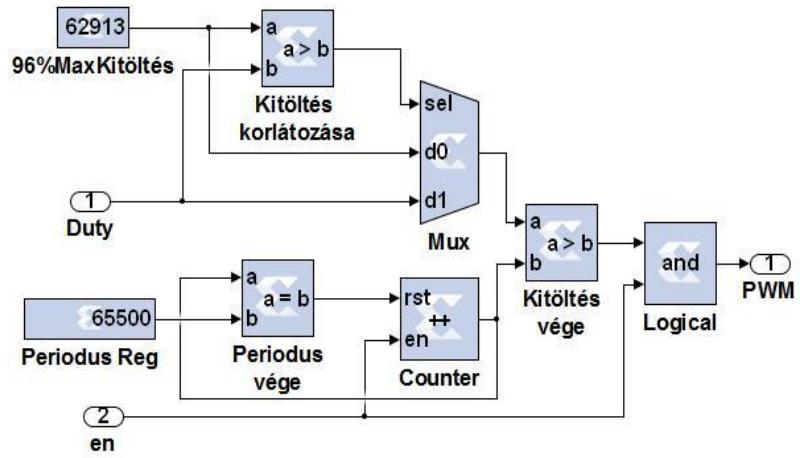
5.5.1 PWM Generátor megvalósítása FPGA áramkörön System Generator környezetben.

5.5.1.1 Megvalósítás

Egy hardveres számláló segítségével, amely az FPGA órajelére számol, egy komparátor segítségével összehasonlítjuk a számláló értékét az általunk megadott kitöltési tényező értékével. 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 16-bites 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.



Kép. 5.40 A PWM generátor System Generátorban megvalósított szerkezete

A felépítésében be van iktatva egy korlátozás, amely segítségével 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.

$$DutyReg = \frac{65535 * Duty\%}{100}$$

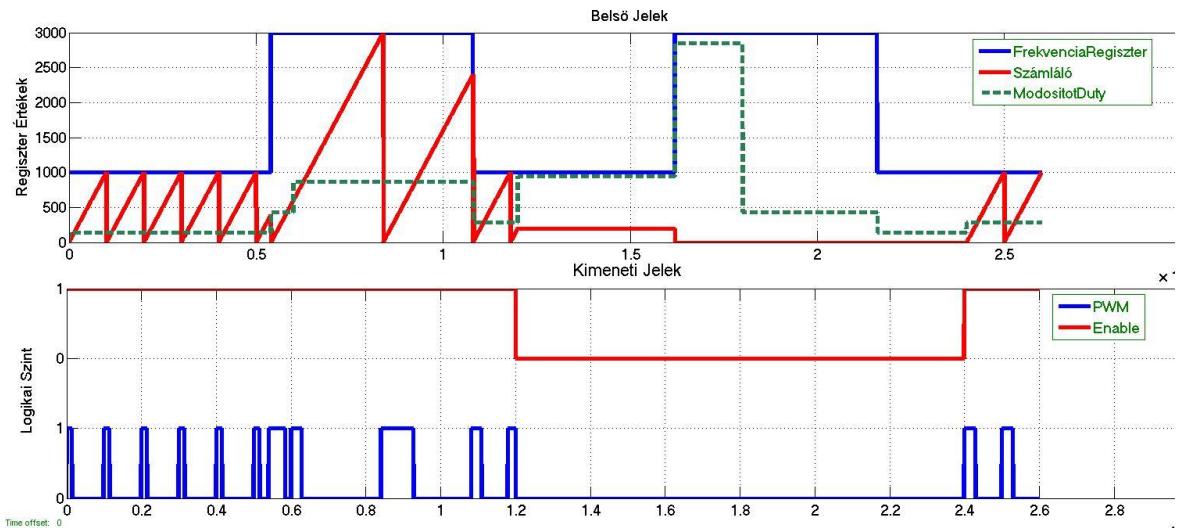
A PWM generátor kimenetét és a számláló (Counter) működését letilthatjuk az „en” bemenetre adott logikai 0 értékkel, máskülönben engedélyezve vannak.

A frekvencia megadásához ismernünk kell az FPGA órajelének a periódusát, amely jelen esetben $T_{Clk} = 20ns = 20 * 10^{-9}, s - ban$ $f = \frac{1}{T_f}, Hz - ben$.

$$f = \frac{1}{FrekReg * T_{Clk}} = \frac{1}{FrekReg * 20 * 10^{-9}}, Hz - ben.$$

A *Periodus Reg* 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 *Periodus Reg* között, így ha növeljük a frekvenciát, csökkeni fog a rezolúció.

A Kép. 5.41, 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.



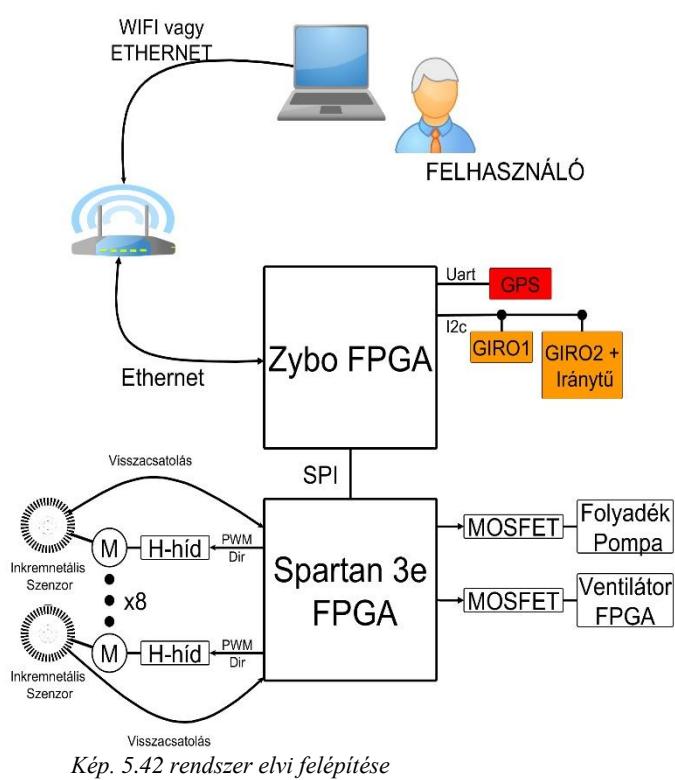
Kép. 5.41 a PWM generátor bemenő, kimenő illetve néhány belső jele (Scope1)

5.6 ELEKTRONIKA

5.6.1 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.

5.6.2 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.

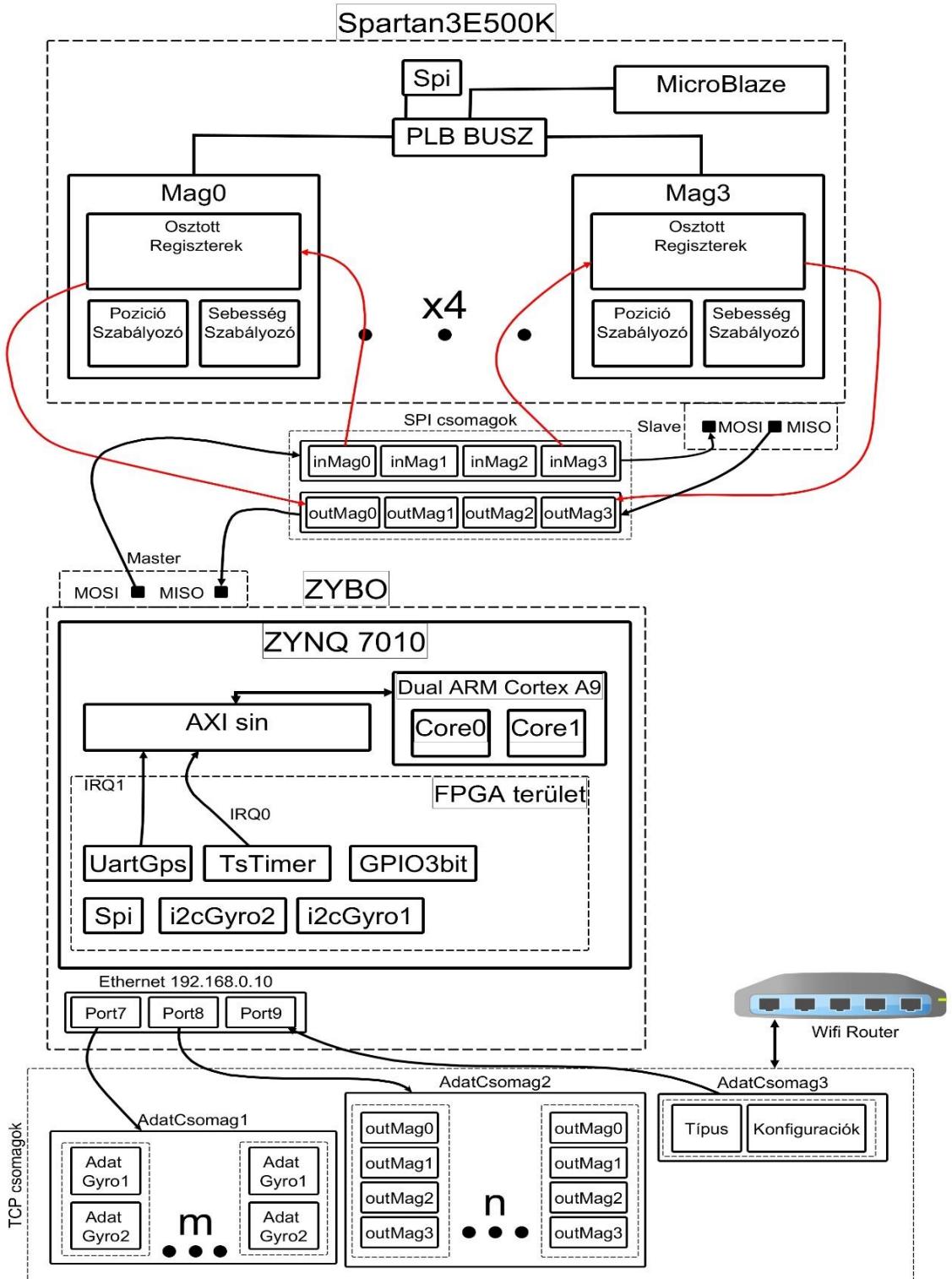
A ZYBO fejlesztőlapon levő ZYNQ 7010 chip tartalmaz két beépített ARM Cortex A9 processzort. A processzor mellett található újrakonfigurálható mag és

egy előre elkészített periferikus elemeket tartalmazó rész. A processzorok a körülöttük levő elemekkel az AXI busznak nevezet sínrendszeren keresztül tudnak kommunikálni.

A Spartan FPGA-ban kialakítunk egy 32 bites *MicroBlaze* processzort és a hozzá szükséges PLB sínrendsزert, a sínrendszer 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 IP mag paramétereit osztott regisztereken keresztül állíthatjuk be vagy olvashatunk ki értékeket, a regiszterek a PLB sínre vannak illesztve. A

Zybo lapon található Ethernet modulon keresztül kapcsolódunk egy Wifi routerhez, amely Access pontként működik. A routerhez még csatlakoztathatunk három más vezetékes eszközt, amelyek lokális hálózatba lesznek kötve a ZYBO-val.



Kép. 5.43 Kommunikációs csomagok és az FPGA áramkörökbe programozott modulok elvi felépítése

5.6.2.1 Zybo FPGA fejlesztőlap

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

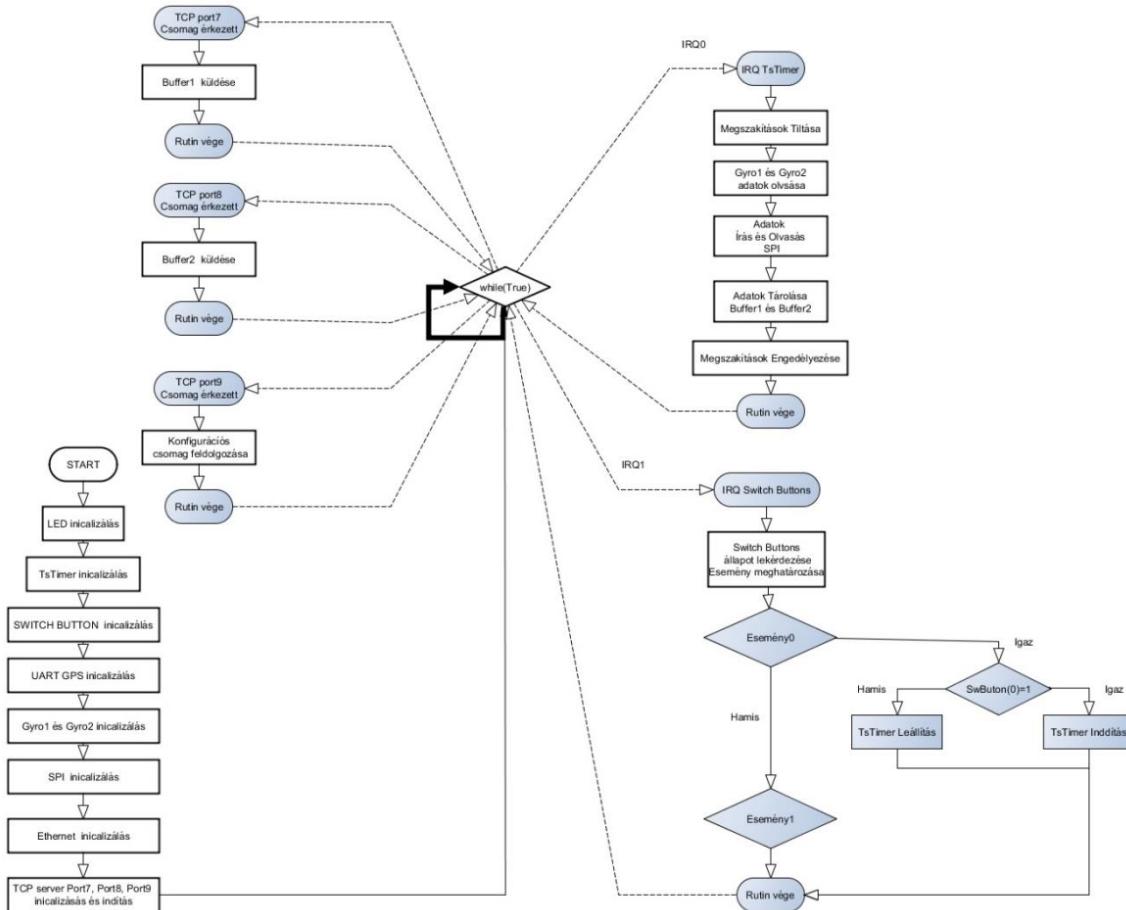
A Core0 processzor feladatai között szerepel a megszakítások lekezelése. A legfontosabb a mintavételi periódust generáló számlálótól érkező megszakítás, amelynek bekövetkeztekor a processzor begyűjti az adatokat a szenzoroktól (Giroszkóp 1 és 2). Lekezeli a megszakítást, amelyek az UART modultól érkeznek 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 futtatja.

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 vérehajtásakor.

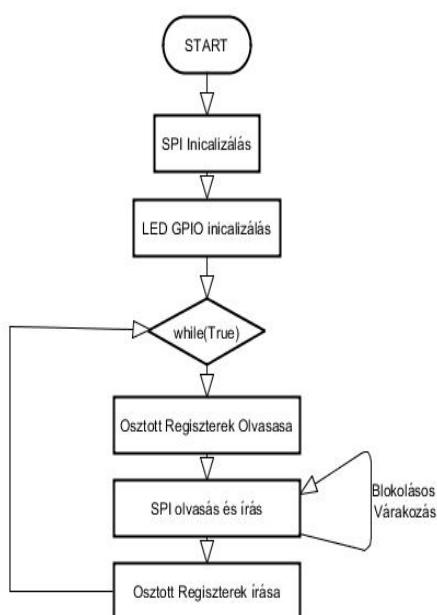
SwitchButtonok megszakítása nem nagy prioritású, célja hogy a manuálisan beállítható paraméterek futás közben változtathatóak. A megszakítás érkezésekor lekérjük a *SwitchButton* kapcsolok állapotait, majd generálunk egy eseményt annak függvényében, hogy mely kapcsoló állapota változott meg.

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. 5.44 ZYBO Core0 program folyamat árbája

5.6.2.2 Spartan3e FPGA fejlesztőlap



Kép. 5.46 Spartan3e500, MicroBlaze szoftver folyamat árája

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

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

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.

5.6.2.3 Kommunikációs protokollok

5.6.2.3.1 SPI kommunikációs protokoll:

A ZYBO mester még a Spartan szolga egységeként működik, a kommunikáció szinkron típusú, az órajel frekvenciája 1.56 MHz. Miközben a mester adatokat küld a szolga irányába a szolga egység is továbbít párhuzamosan adatokat a mester irányába. Az SPI legkisebb csomagja minimum 32bit. Az SPI kommunikáció az ábrán (Kép. 5.43) látható SPI csomagszerkezeten keresztül történik.

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

Az IP mag csomagok mellett megtalálható két regiszter, amely segítségével a nem kritikus modulokat tudjuk megcímzni és adatokat továbbítani nekik pl.: ventillátor PWM modul, pumpa PWM modul.

```
s32addresReg; - cél regiszter szoftveres címe  
s32dataReg; - cél regiszter adata
```

Az outMag0 csomag tartalmazza egy beépített IP mag 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 módosítani a MicroBlaze 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 32 bites változok. Az outMag0 struktúrát a Spartan rendszer küldi a Zybo fele, míg az inMag0-t a Zybo küldi párhuzamosan a Spartan rendszernek. Az inMag0 ban találhatók meg a szabályozók előirt értékei, és a konfigurációs beálltások.

<u>outMag0</u>	<u>inMag0</u>
s32 <u>USebesseg0</u> ;	s32 <u>Config0</u> ;
s32 <u>UPozicio0</u> ;	s32 <u>RefPos0</u> ;
s32 <u>SebessegPozicio0</u> ;	s32 <u>RefSeb0</u> ;
s32 <u>AktPozicio0</u> ;	s32 <u>Q0_0</u> ;
s32 <u>SzurtSebessegPozicio0</u> ;	s32 <u>Q1_0</u> ;
s32 <u>AktSebesseg0</u> ;	s32 <u>Q2_0</u> ;
s32 <u>SzurtSebesseg0</u> ;	s32 <u>Ts0</u> ;
s32 <u>eSebeseg0</u> ;	s32 <u>TsL0</u> ;
s32 <u>n20</u> ;	s32 <u>Egyeb0</u> ;
s32 <u>n30</u> ;	s32 <u>PWMFrekREgH0</u> ;
s32 <u>n40</u> ;	s32 <u>PWMFrekRegL0</u> ;

5.6.2.3.2 Ethernet

A Zybo rendszeren megtalálható egy Ethernet csatlakozó 1Gbit/s sebességre képes a programba beágyazva az lwip140 modult. A [10] 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 maguktól beérkezett adatokat. A harmadik szerver segítségével konfigurációs parancsokat küldhetünk a rendszernek, amelyeket értelmez és végrehajtja.

A csomagok szerkezete a Kép. 5.43 látható a „TCP csomagok” feliratnál.

A program:

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

IP4_ADDR(&ipaddr, 192, 168, 0, 10);

Az alhálózati maszk:

```
IP4_ADDR(&netmask, 255, 255, 255, 0);
```

A router 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, amellyel fog rendelkezni az eszköz:

```
unsigned char mac_etheren_address[] = { 0x00, 0x0a, 0x35, 0x00, 0x01, 0x02 };
```

Következő lépésekben aktualizáljuk a beállításokat a hardveren:

```
init_platform();
lwip_init();
xemac_add (echo_netif, &ipaddr, &netmask, &gw, mac_etheren_address,
PLATFORM_EMAC_BASEADDR)
netif_set_default(echo_netif);
platform_enable_interrupts();
netif_set_up(echo_netif);
```

Szerver létrehozás:

A Program. 5.6.1 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. lépésekben társítjuk az IP címet és a portszámot a TCP protokollal, 18. sorban társítjuk a kapcsolat kérésekor végrehajtandó eljárást.

Az **accept_callbackSV1** eljárás társítja a csomagok érkezésekor meghívódó rutint, amelyben majd történik az adatok visszaküldése a feladónak. A **recv_callbackSV1** történik az adatok kiolvasása.

```

1. int start_applicationsv1()
2. {
3.     err_t err;
4.     unsigned port = 9;
5.     pcb = tcp_new();
6.     if (!pcb) {
7.         xil_printf("Error creating PCB. Out of Memory\n\r");
8.         return -1;
9.     }
10.    err = tcp_bind(pcb, IP_ADDR_ANY, port);
11.    if (err != ERR_OK) {
12.        xil_printf("Unable to bind to port %d: err = %d\n\r", port, err);
13.        return -2;
14.    }
15.    tcp_arg(pcb, NULL);
16.    pcb = tcp_listen(pcb);
17.    if (!pcb) {
18.        xil_printf("Out of memory while tcp_listen\n\r");
19.        return -3;
20.    }
}

```

5.6.3 Feladatok Elosztása

5.6.3.1 Zybo fejlesztőlap

A Core0 processzor feladatai között szerepel a megszakítások lekezelése. A legfontosabb a mintavételei 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.

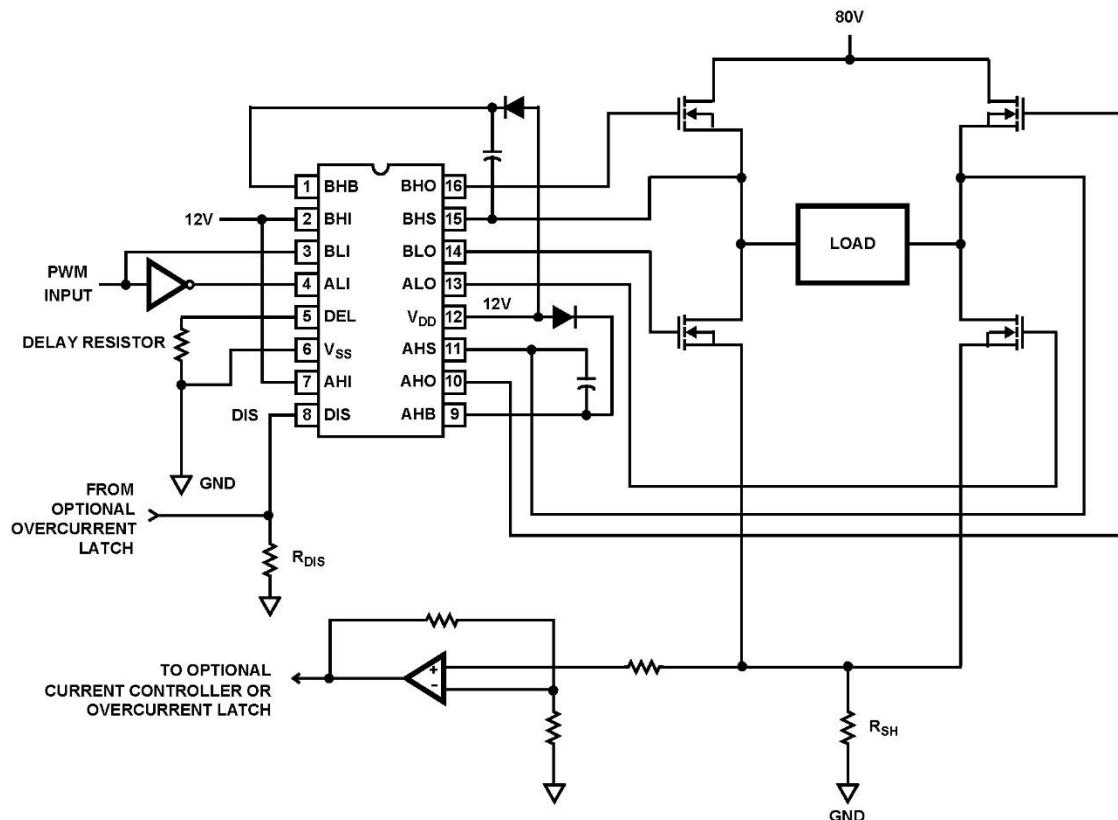
5.6.3.2 Spartan fejlesztőlap

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

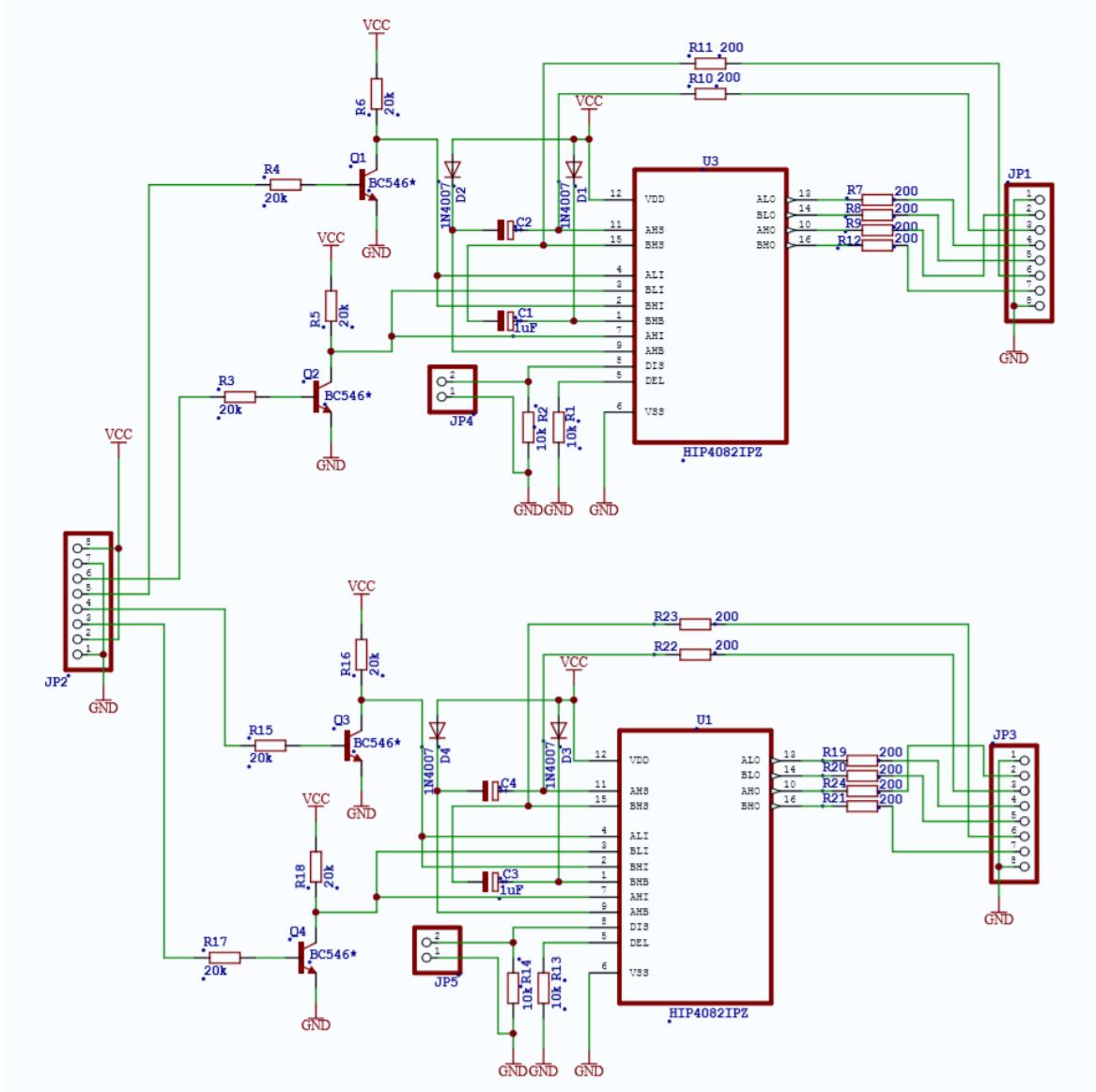
Abban az esetben, ha megszakad az SPI kommunikáció egy adott időn belül letiltja a PWM modulok kimenetét így állítva meg a rendszert.

5.7 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 tranzisztoroknak 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. 5.47 Hip4082 alkalmazása H hid kapcsolásban. Forrás: <http://www.intersil.com/en/products/space-and-harsh-environment/harsh-environment/half--full-bridge-and-three-phase-drivers/HIP4082.html>



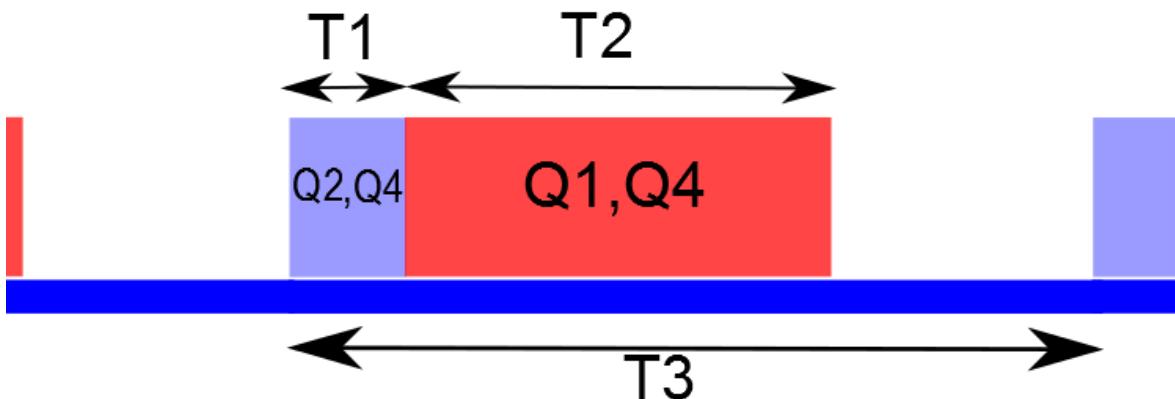
Kép. 5.48 Két hídvezérlő áramkör kapcsolási rajza HIP4082 integrált áramkörrel megvalósítva

Az alkalmazott N csatornás MOSFET tranzisztorok típusa IRFB7437, adatlapí adatok szerint az 195A áramot tud vezetni maximálisan, 40V feszültséget bír el, valamit a teljesen kinyitott állapotban az ellenállása kisebb, mint $2\text{m}\Omega$.

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, hogy a kondenzátor töltődjön fel és majd a HIP4082 integrált áramkörön keresztül tudjuk rákapcsolni a felső tranzisztorok Gate bemenetére. A [11] adatlap alapján AHB, BHB az integrált áramkör azon bemenetei, amelyeknek feszültsége el van tolva az AHS, BHS közös pontokhoz képest, a H hídban 10V-ra feltöltött kondenzátor

feszültségével. Az AHO, BHO azok a kimenetek, amelyek a felső tranzisztorokat vezérik, az ALO, BLO az alsó tranzisztorok vezérlő kivezetései. A HIP4082 áramkörben van beépítve egy késleltető, amely garantálja, hogy ne alakuljon ki rövidzár a hídban, abban az esetben, amikor a felső és az alsó tranzisztorokat kapcsoljuk át ugyan azon a fél híd oldalon.

A [11] 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.



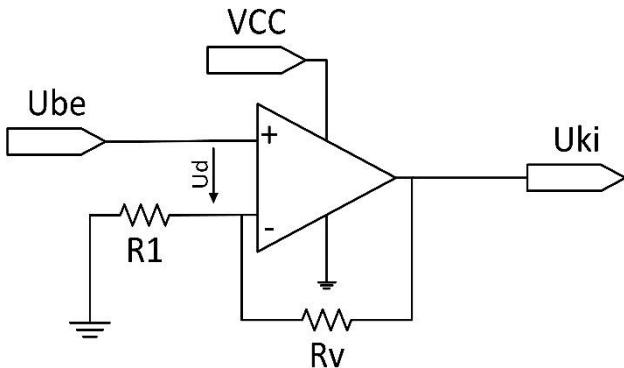
Kép. 5.49 PWM és a tranzisztorok kapcsolása

A Kép. 5.49 látható a PWM jel egy periódusának a felbontása. A PWM jel alapfrekvenciája 700Hz, így a $T_3=1,42\text{ms}$. $T_1=0.5\%T_3=71\text{us}$. A PWM jel kitöltése Duty= T_1+T_2 .

A kívánt kapcsolási sorrendet úgy tudjuk elérni, hogy, a DIS bemeneten áramkorlát vagy más védelemeket iktathatunk be a rendszerbe. A [11] adatlapban ajánlott megoldás szerint méri az áramot az R_{SH} ellenálláson keresztül, egy műveleti erősítő segítségével, amelyet nem invertáló erősítő alapkapsolásban 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. 5.50 látható a nem invertáló erősítő kapcsolás, az Ube bemeneti feszültség, az árammérő ellenállástól érkező feszültség. A kapcsolás erősítése: $A = 1 + \frac{R_v}{R_1}$

Ha $R_{SH} = 0.1 \text{ Ohm}$, 10A szeretnénk az áramot korlátozni akkor a $U_{be} = R_{HS}I = 1V$, ahol az I a H hídon átfolyó áram.



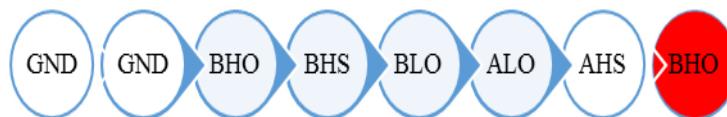
Kép. 5.50 Nem invertáló erősítő forrás [15]
áramkorlátot, de illeszthető hozzá a jumperek helyére az ábrán (Kép. 5.50) látható áramérő és áramkorlátozó kialakítás.

Az ábrán (Kép. 5.48) látható kapcsolási rajz tartalmaz két HIP4082 hídvezérlő áramkört, az áramkörre a bemenet a JP2 bemeneten történik, amelyen keresztül betápláljuk 12V feszültséggel és 4 PWM jel segítségével meg tudjuk hajtani a két hidat. A jeleket szalagkábel segítségével csatoljuk az áramkörhöz. A szalagkábel 8 vezetékből tevődik össze, rendeltetésük szerint:



Kép. 5.51 Dupla hídvezérlő áramkor vezérlőjei JP2 csatlakozó a Kép. 5.48-n.

A pirossal megjelölt vezeték az 1 számú. A PWM4 egy 3,3V PWM jel, amely egy NPN (Q2) tranzisztoron keresztül kapcsolja a 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 hasonlóképpen az előzőhez csak a ALI, és BHI bemeneteket vezérli a (Q1) tranzisztorokon keresztül.

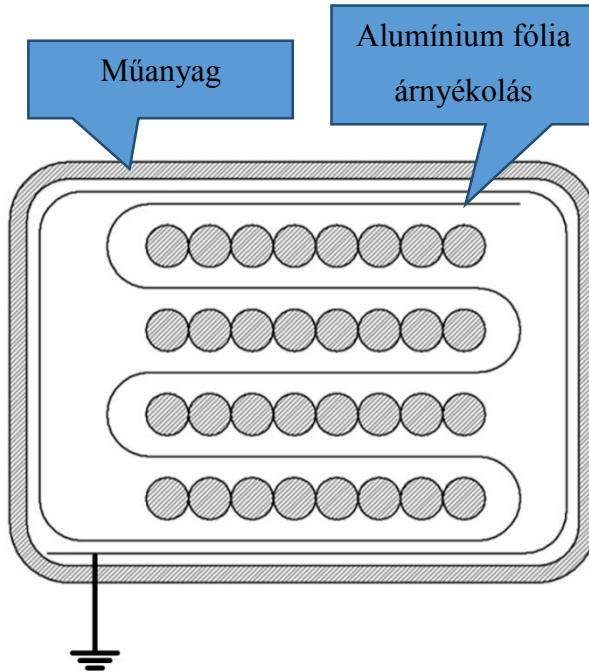


Kép. 5.52 H hid tranzisztorainak a Gate vezetékei

Az ábrán (Kép. 5.55) 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. 5.52 vannak szemléltetve.

A négy híd vezérlésére négy (Kép. 5.52) szalagkábelre van szükségünk, amelyeket

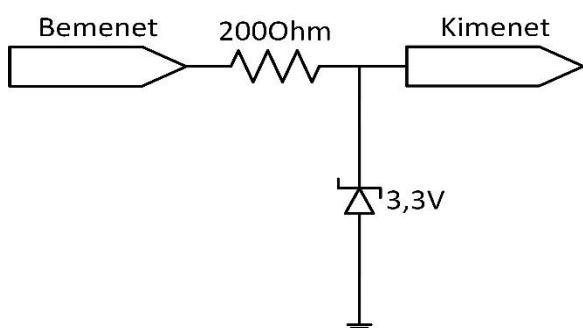


Kép. 5.53 A négy Kép. 5.52 látható szalagvezeték jelenik meg a Buszvezetékben.

az ábrán (Kép. 5.53) látható módon rendeztem el és árnyékoltam le.

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

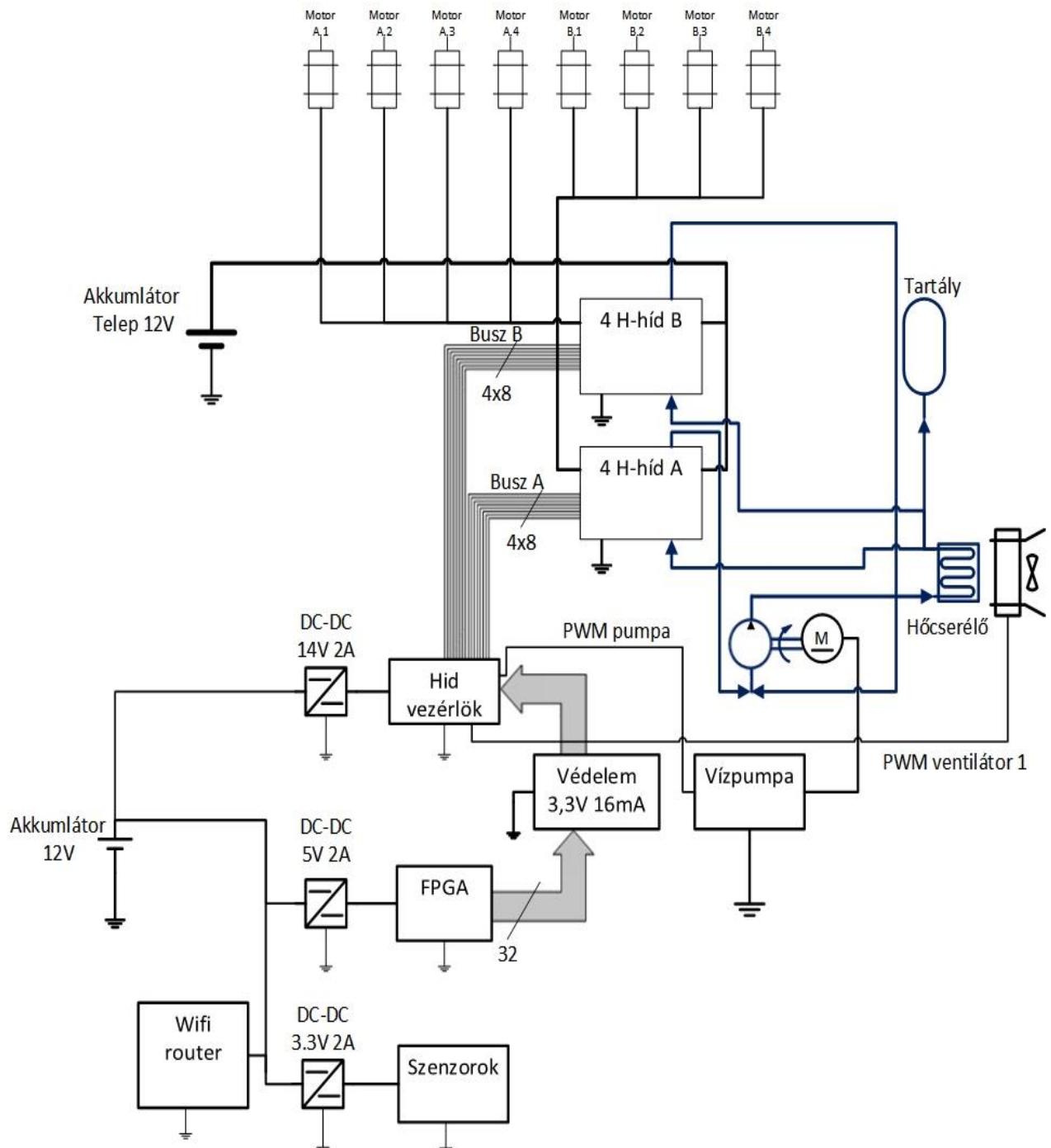
A Kép. 5.55 látható 32 bites buszvezeték, amely az FPGA rendszertől érkezik, és megtalálható benne a 8 motor hajtásához szükséges PWM beavatkozó jelek, amelyek az amplitúdója 3,3V. A busz szalagvezeték segítségével van kialakítva és megtalálható benne egy védelem is, amely megvédi az FPGA rendszert az esetleges visszahatásuktól. A védelem



Kép. 5.54 FPGA kimentének a védelme

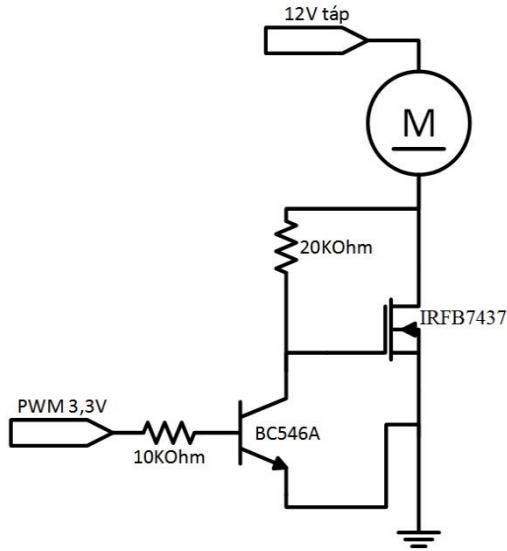
Kép. 5.54-en látható, 3,3V-os zener dióda segítségével történik, melynek feladata megakadályozza a 3,3V-ál nagyobb feszültségek FPGA rendszerbe történő továbbhaladását. A kimenetre van kötve egy ellenállás is, amely az áramot korlátozza mivel az FPGA a bemenetén a legtöbb 16mA áramot visel el.

A Kép. 5.55 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 konverteknek 3V, és a kimeneti feszültséget állíthatjuk 3-30V-ig. A konvertek maximálisan 2A tudnak leadni.



Kép. 5.55 A robot energia ellátása valamint a hűtő rendszer elvi felépítése

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. 5.56 Vízpumpa és a ventilátor motorjának vezérlő teljesítmény elektronikai kapcsolása

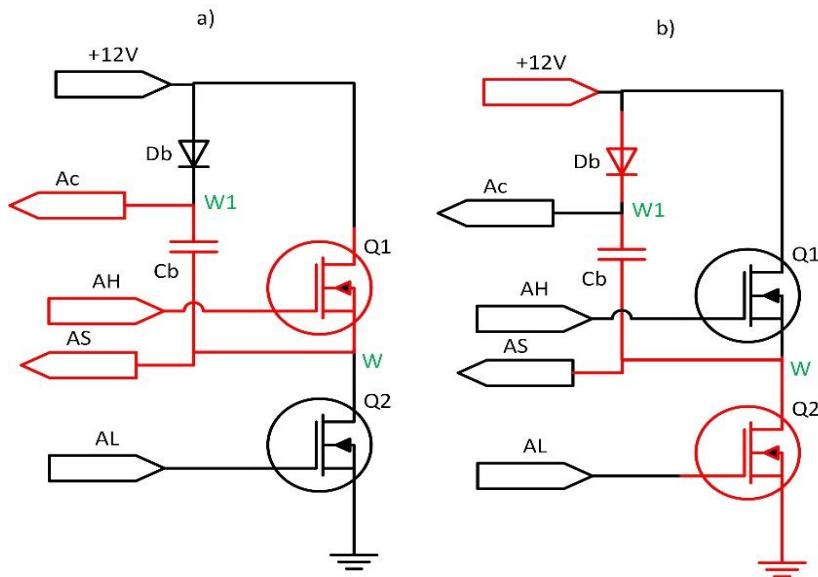
A vízpumpa és a ventilátorok motorja PWM jel segítségével van vezérelve egy N csatornás MOSFET tranzisztor segítségével, amelyet Kép. 5.56 ábra szemléltet.

5.7.1 Bootstramp működése

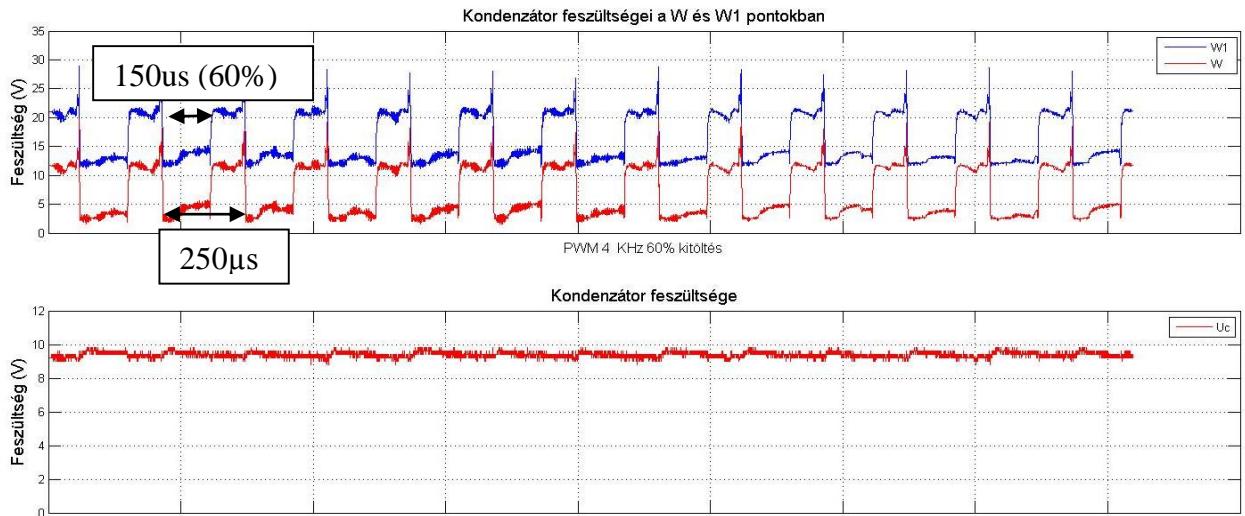
A [12] dokumentum alapján a bootstramp megoldás a Kép. 5.57 képen látható módon történik. A C_b kondenzátort töltjük fel a D_b dióda segítségével azokban a pillanatokban, amikor a Q₂ tranzisztor kinyitott állapotban van és a w potenciál elég alacsony, ahhoz hogy a D_b dióda kinyisson és így feltöltve majdnem 12V feszültségre a kondenzátort.

A Kép. 5.57 látható b) ábrán látható amint a Q₂ tranzisztor tölti a C_b 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.

Az ábrán (Kép. 5.58) látható a mérése a C_b kondenzátor feszültségének, a méréseket oszcilloszkóp segítségével végeztem el és mentettem ki az adatokat, amelyeket majd ábrázoltam Matlab programmal.



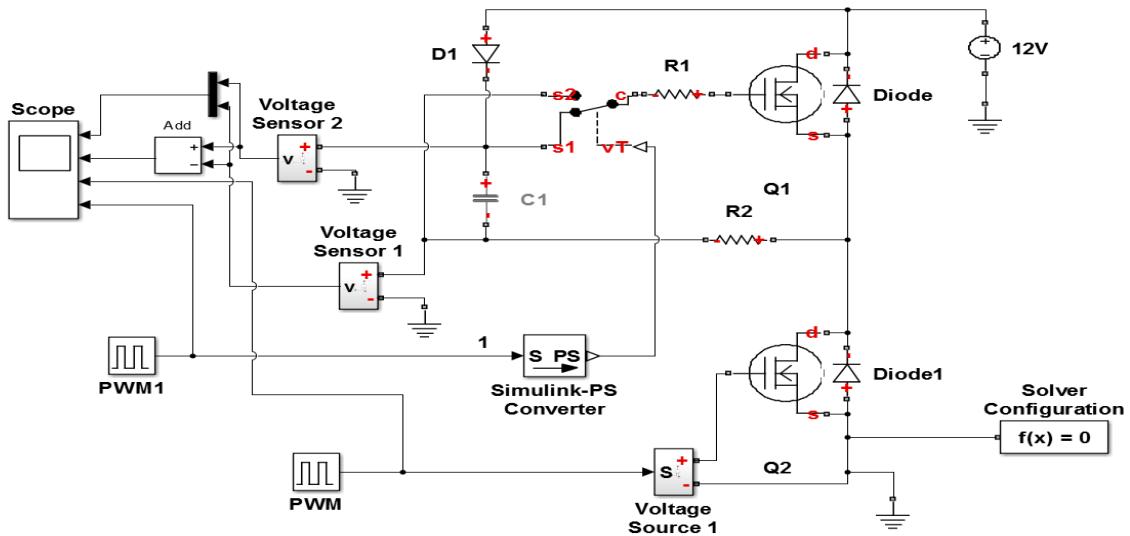
Kép. 5.57 Bootstramp megoldás a felső tranzisztor Gate bemenetének a meghajtására



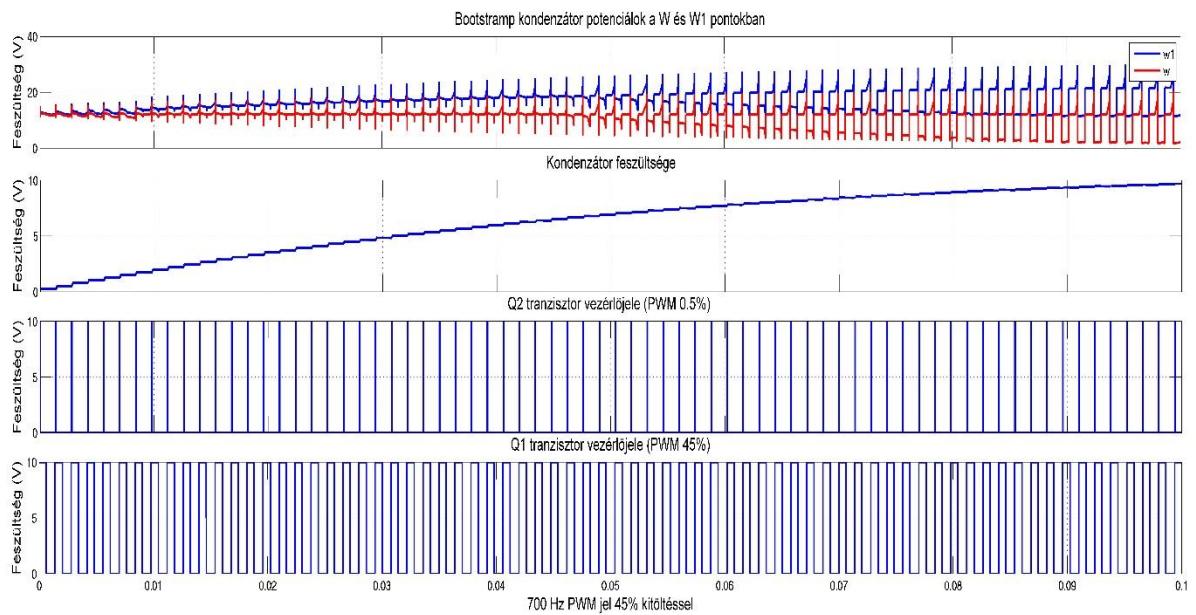
Kép. 5.58 Bootstramp kondenzátor feszültsége a W és W11 pontokban

5.7.1.1 Szimuláció Simulink környezetben

A szimuláció során előalítottam az ábrán (Kép. 5.59) látható méréseknek megfelelő környezetet. Az eredmények szerint (Kép. 5.60) 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 kapcsolási rajzot nézve (Kép. 5.48) ha a mindenre 0V kapcsolunk. Ugyanis a tranzisztorokból kialakított tagadó kapu megtagadja és így a hídba minden két alsó tranzisztor kinyitott állapotba kerül.



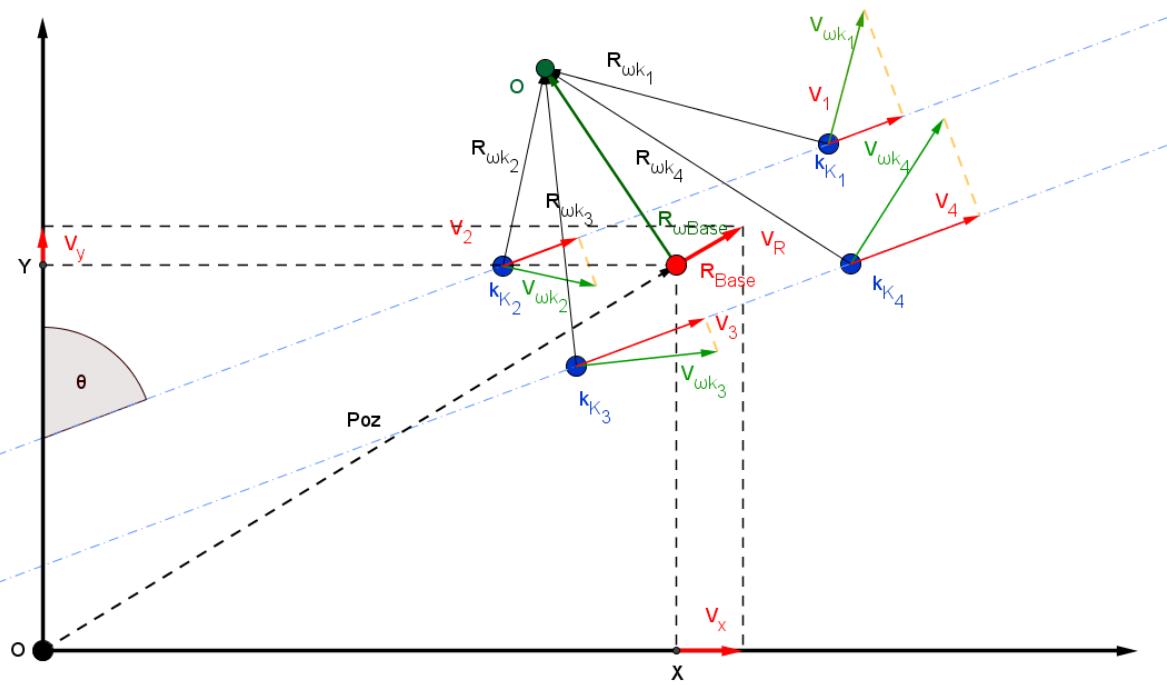
Kép. 5.59 Bootstrapping működése, szimulációs modell MATLAB/SIMULINK környezetben



Kép. 5.60 Szimulációs eredmények Bootstrapping

5.8 ROBOT MODELL

A [13] cikkben leírja egy négykerekű mobilis robot kinematikus és dinamikai modelljét. A rendszeren hasonlóképpen lehetne alkalmazni a tárgyalt modellt annyi



Kép. 5.61 Robot kerekek sebsége és a robot mozgásának viszonya
eltéréssel, hogy ebben az esetben azok a pontok, amelyekben a robot érintkezik a talajjal nem szimmetrikusak.

$$\dot{x} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_z \\ \omega \end{bmatrix}$$

Jelölések:

$$i \in (1,2,3,4) - index jelölése$$

$$\vec{P} = (x, y, z) - egy vektorfelépítése$$

R_{T_i} – a robot i talpa kis kerekének a pozició vektora, K_i – nagykeréktől nézzve,

\overrightarrow{RK}_i – a roboti nagy kerekének a pozició vektora

Bs – a robot vázához rögzített kordináta rendsze origója

R_{BASE}

– a robot Bs pontjának a vetülete a talaj síkjára a robot kordináta rendszerében

$\overrightarrow{V_{T_i}}$ – a roboti lánctalpának a pozició sebessége

$$\overrightarrow{V_{K_i}} - a roboti lánctalpának a sebessége$$

f_i – a roboti talpán levő kis kerekének a középpontjának a pályája

XR, YR, ZR a robothoz rögzítet koordináta rendszer tengelyei

$$\overrightarrow{\text{Magasság}} = \overrightarrow{R_{Base}} \text{ és } aBs \text{ közötti vektor}$$

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

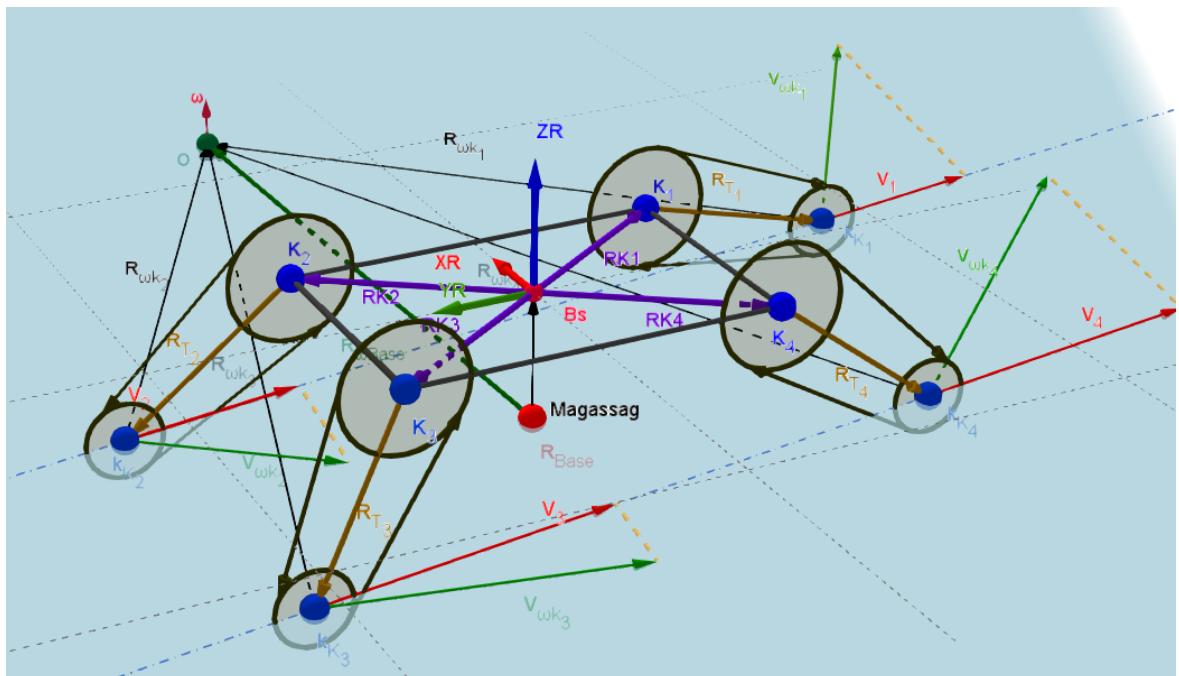
Az [13] alapján felírhatók a következő összefüggések a vektorok között:

$$\overrightarrow{R_{\omega K_l}} = \overrightarrow{R_{\omega Base}} - \overrightarrow{R_{K_l}}$$

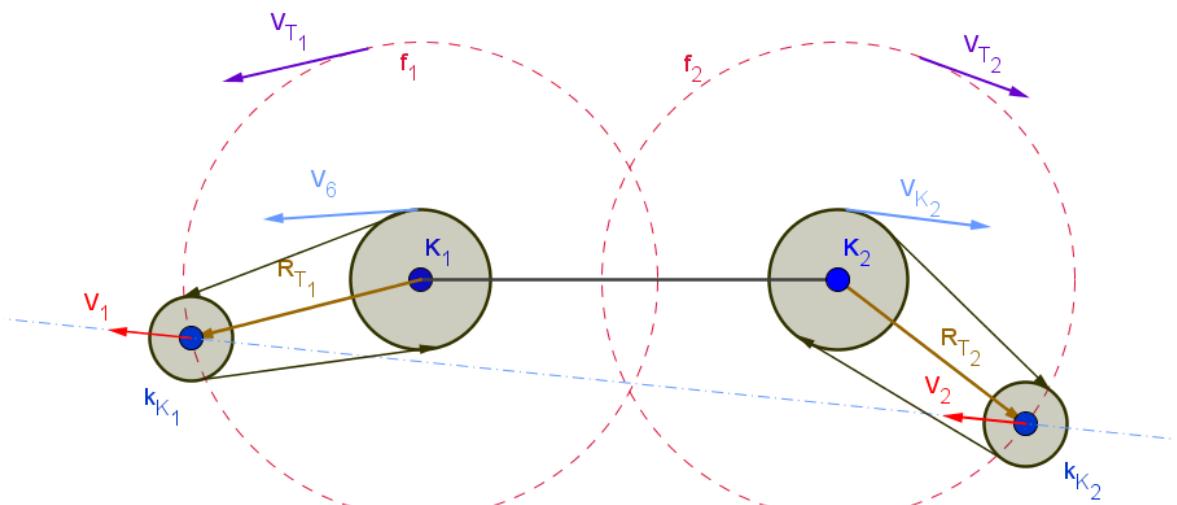
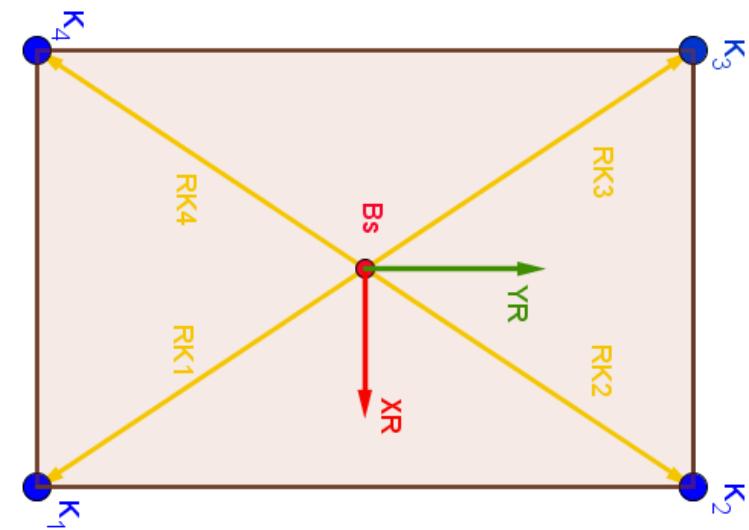
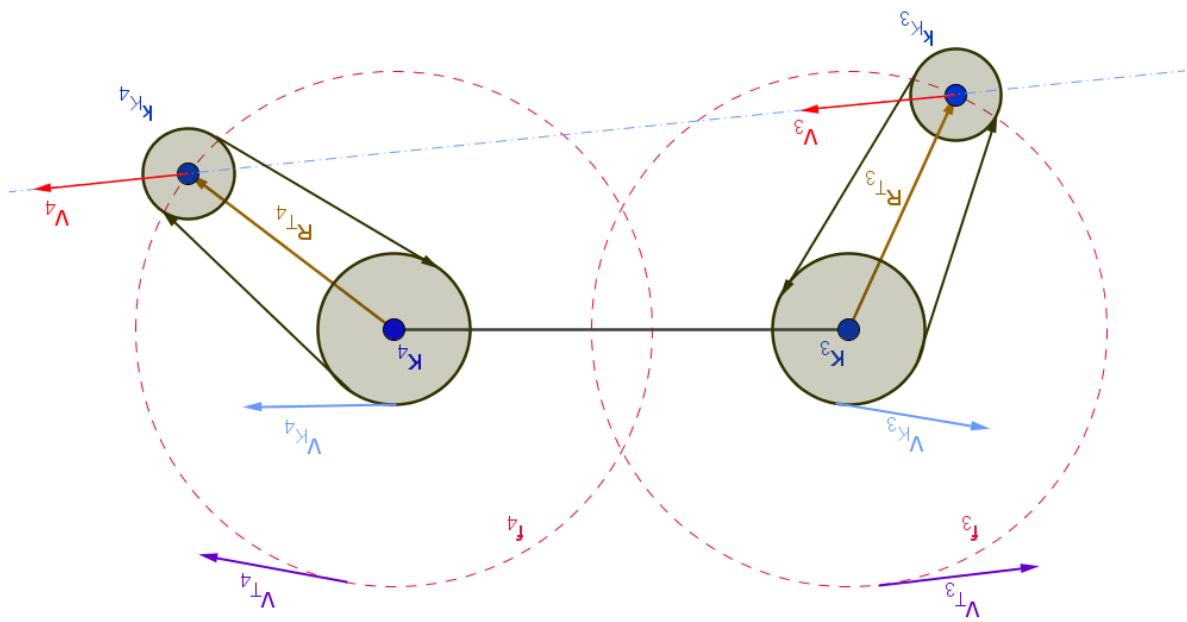
$$\overrightarrow{V_{\omega K_l}} = \overrightarrow{R_{\omega K_l}} \times \overrightarrow{\omega}$$

Ha ismerjük a $\overrightarrow{R_{\omega Base}}$, és $\overrightarrow{\omega}$ kitudjuk számolni a $\overrightarrow{V_{K_l}}$ sebességeket. Tudva hogy a rendszer csak az YR tengelye mentén tud sebességet generálni így:

$$V_{K_i} = \overrightarrow{V_{\omega K_l}} - \text{nak az } YRszerinti \text{ komponensével.}$$



Kép. 5.62 Robot 3D vektorábrája



Kép. 5.63 Oldalnézetek és felülnézetek, jelölések szemléltetése

6 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.

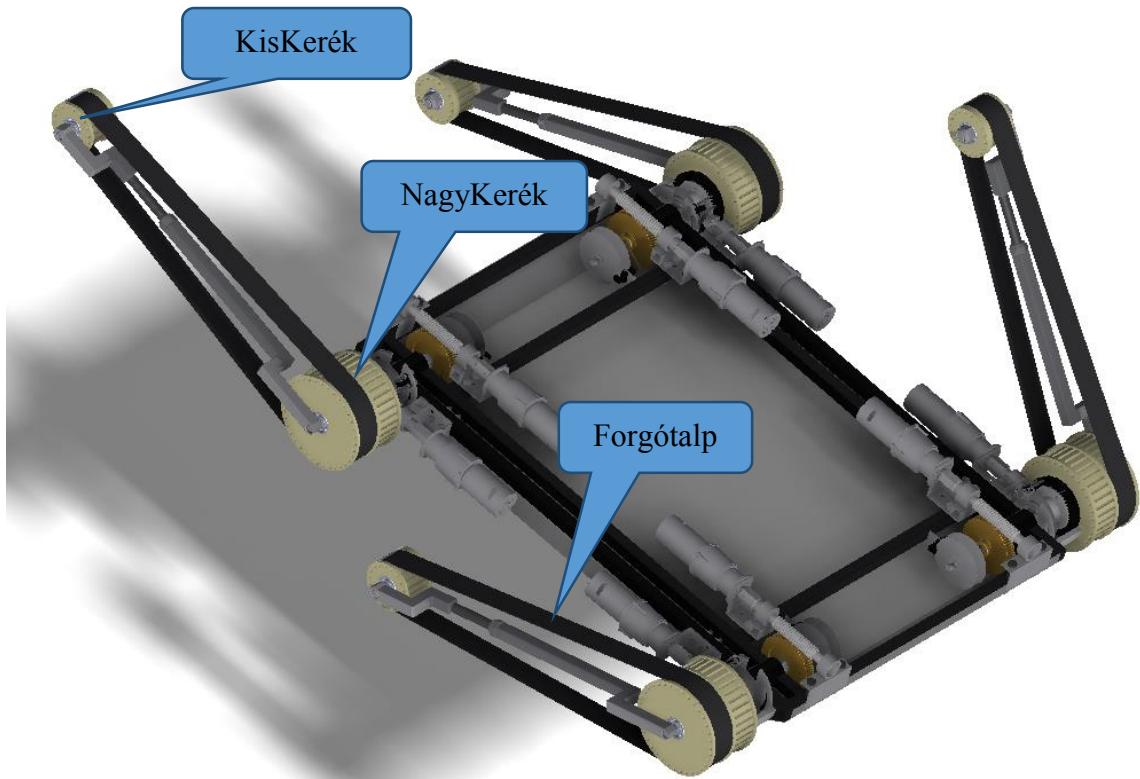
DC motorok betáplálási feszültsége: 12V, maximális terhelés alatt 10A áramot is felvehet.

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

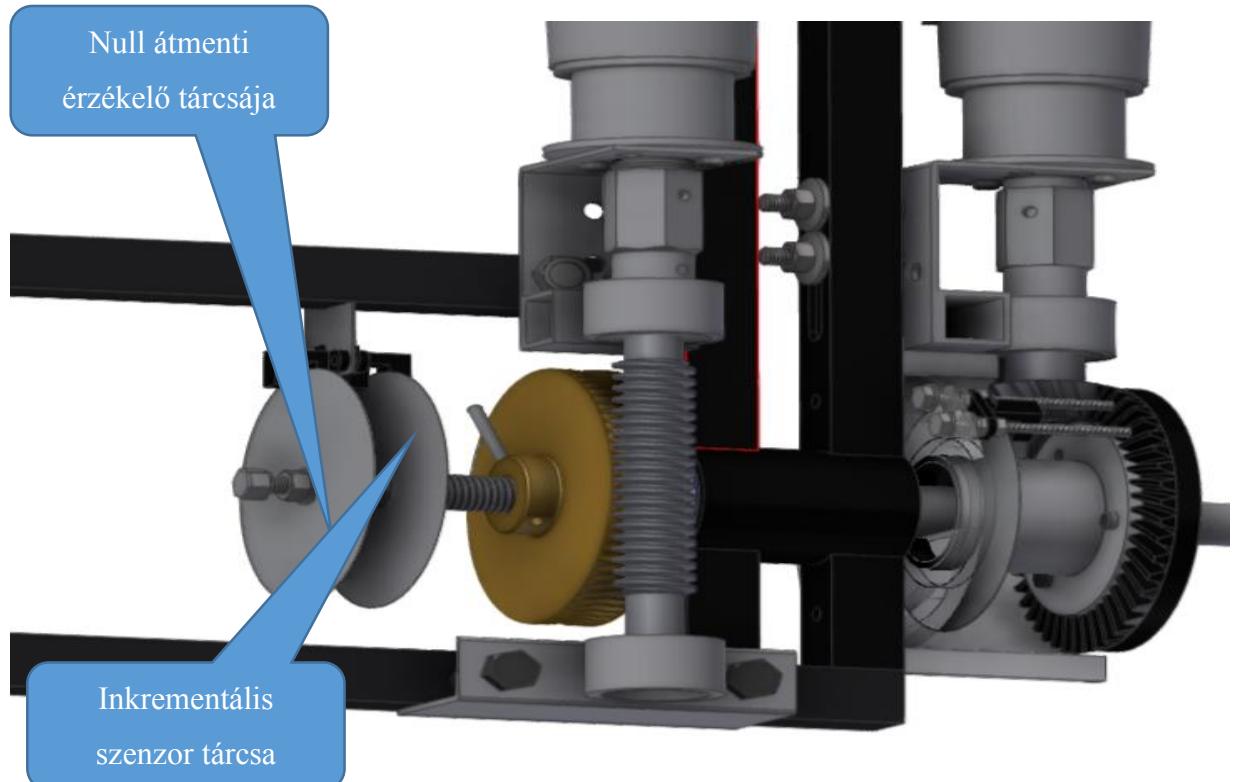
- Nagy nyomaték kifejtésére képesek, jelen esetben 40:1 az áttétel arány. Egy motor által kifejtett nyomaték névlegesen 3-4Nm között van így a karok végén tengelyre kifejtett nyomaték elérheti a 60Nm-t.
- A terhelés nem képes visszafele hajtani, mert a mechanizmus lezárja, így akár a motort teljesen 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 $Motor_i S$ hajt meg egy másik fogaskeréken keresztül. A $Motor_i P$ a csiga áttételen keresztül változtatja a lánctalpak szögét. A mechanikai rendszer terve az alábbi ábrákon van szemléltetve.



Kép. 6.2 Robot vázának Inventoros 3D Képe



Kép. 6.1 Átételek

7 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 alapkönfigurációs kialakítása Xilinx Platform Studio-val a két fejlesztőrendszeren
- A beágyazott processzorokon futó programok megvalósítása Xilinx Software Development Kit eszközzel
- IP mag tervezése (sebesség+pozíció szabályozó)
- Az egyes alegységek Simulink System generátorban való szimulációja
- H-hídak megépítése, vízhűtés kivitelezése
- Nyáktervezés Altium tervező programban
- Grafikus vezérlőfelület elkészítése

8 KÖVETKEZTETÉSEK:

A kivitelezés során sok olyan apró hibára bukkantam, amelyek jó továbbfejlesztési lehetőséget nyújtanak a jövőben.

A dolgozat befejeztével, úgy gondolom, hogy jó választás volt az FPGA fejlesztő rendszer, mert nagyon jó alapot nyújt mind a szoftveres mind a hardveres továbbfejlesztési lehetőségekre. A sebesség szabályozók, valamint a hardveresen kifejlesztett pozíció szabályozók működőképességét alátámasztják a mérési eredmények.

A sebesség mérő modult még ki kellene egészíteni egy zajszűrővel, ami az alacsony sebességből adódik. A mechanikai rendszeren is lehetne átalakításokat végezni. A rendszeren kívül levő motorokat be kellene vinni a vázon belülre. A lánctalpákat is át kellene alakítani, mert nem fognak megfelelni a kültéri követelményeknek.

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 és 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.

Mindezek kivitelezésében az anyagi háttér nagy részben akadályozta munkámat.

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

Az inkrementális szenzorok, amelyeket készítettem, az elvárásoknak megfelelően teljesítik feladataikat, 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 ez a rendszer nagyon jó lehetőségekre mutat rá szoftver és elektronika továbbfejlesztés szempontjából. A rendszer előállítási költsége jóval a piaci ár alatt van, viszonyítva egy hasonló rendszer közbeszerzési értékéhez..

Sok minden megtanulhattam a munkafázisok során, nemcsak elektronika, hanem mechanika terén is. Az így szerzett tapasztalatokat szeretném majd továbbvinni a jövőben.

9 BIBLIOGRÁFIA

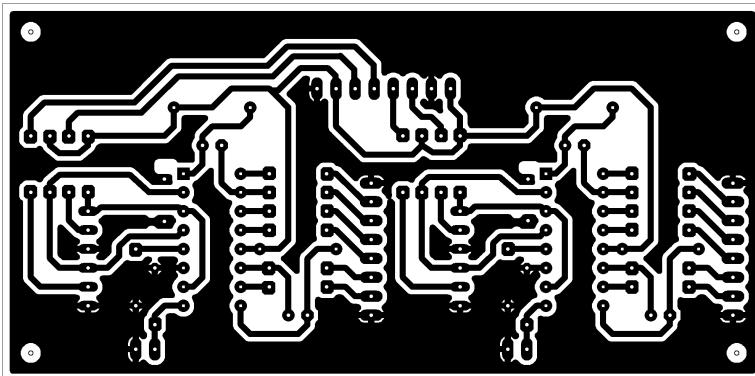
- [1] Márton Lőrinc, *Irányítástechnika*. Kolozsvár: Scientia, 2009.
- [2] Dr Kavita Khare, Dr R. P. Singh Prof. Vikas Gupta, "Efficient FPGA Design and Implementation of Digital PID Controllers in Simulink," 2013.
- [3] xilinx. http://www.xilinx.com/. [Online].
http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_5/sysgen_gs.pdf
- [4] Nema Rajesh, Thakur Rajeev, and Gupta Ruchi, "Design & Implementation of FPGA Based On PID Controller," 2013.
- [5] http://www.ms.sapientia.ro/elektronika. [Online].
http://www.ms.sapientia.ro/elektronika/fileok/jelerzekelok/szt_lab08_inkrementallis_ado.pdf
- [6] András Gergő KOCSIS Krisztián LAMÁR, "IMPLEMENTATION OF SPEED MEASUREMENT FOR ELECTRICAL DRIVES EQUIPPED WITH QUADRATURE ENCODER IN LabVIEW FPGA ,," 2013.
- [7] Márton Lőrinc. http://www.ms.sapientia.ro/~martonl/index.htm. [Online].
http://www.ms.sapientia.ro/~martonl/Docs/Labs/IRI_L1.pdf
- [8] Márton Lőrincz. http://www.ms.sapientia.ro/. [Online].
http://www.ms.sapientia.ro/~martonl/Docs/Lectures/Holtidos_Folyamatok_Iranyitasa.pdf
- [9] InvenSense Inc. www.olimex.com. [Online].
https://www.olimex.com/Products/Modules/Sensors/MOD-MPU6050/resources/RM-MPU-60xxA_rev_4.pdf
- [10] xilinx. http://www.xilinx.com. [Online].
] http://www.xilinx.com/support/documentation/application_notes/xapp1026.pdf
- [11] intersil. http://www.intersil.com/. [Online].
] <http://www.intersil.com/content/dam/Intersil/documents/hip4/hip4082.pdf>
- [12] Silicon labs. http://www.silabs.com/. [Online].
] <http://www.silabs.com/Support%20Documents/TechnicalDocs/AN486.pdf>
- [13] D. Pazderski, I.Rudas, J.Tar K. Kozłowski, "Modeling and control of a 4-wheel skid-] steering".

[14 Maciej Trojnacki, "Dynamics Model of a Four-Wheeled Mobile Robot for Control Applications – A Three-Case Study," in *Intelligent Systems'2014*.: Springer, 2014, p. 111.

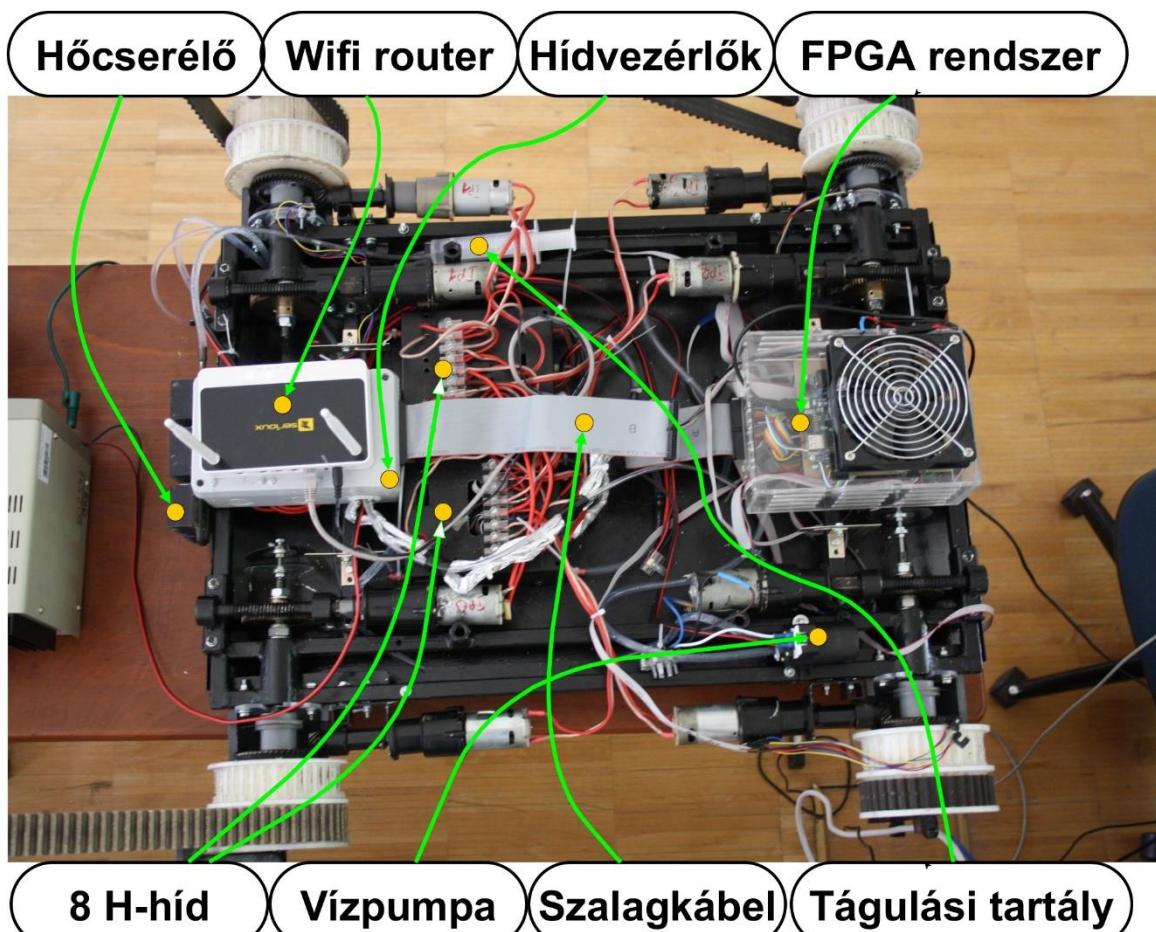
[15 Losonczi Lajos. <http://www.ms.sapientia.ro/>. [Online].
] https://moodle.sapidoc.ms.sapientia.ro/pluginfile.php/2771/mod_resource/content/1/Losonczi_Lajos_-_Analog_Aramkorok_3_V1.pdf

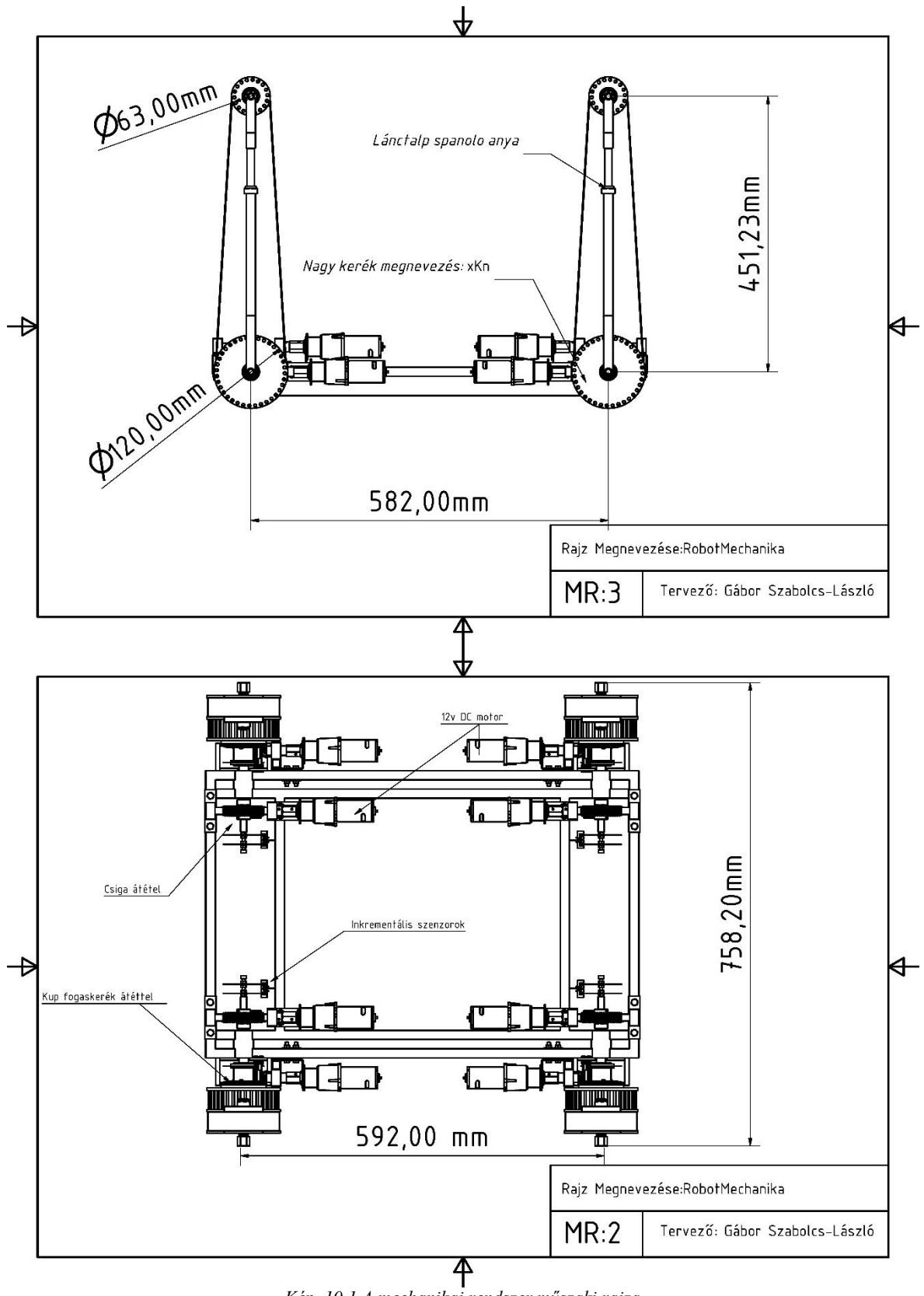
10 MELLÉKLET

Nyákterv UV-s nyomtatási módszerhez



A megépített rendszer és a modulok elhelyezése





Kép. 10.1 A mechanikai rendszer műszaki rajza

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREŞ
SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

Vizat decan

S. 1. .Dr. ing Kelemen András
József

Vizat şef catedră

§.1. Dr. ing. Domokos