SZAKDOLGOZAT

Jándi Tibor

Debrecen
2010

Debreceni Egyetem
Informatika Kar

Kétdimenziós pozicionálás NI eszközökkel

Témavezető:
Dr. Cserháti Csaba
egyetemi docens
DE TTK
Szilárdtest Fizika Tanszék

Készítette:
Jándi Tibor
mérnök informatikus
hallgató

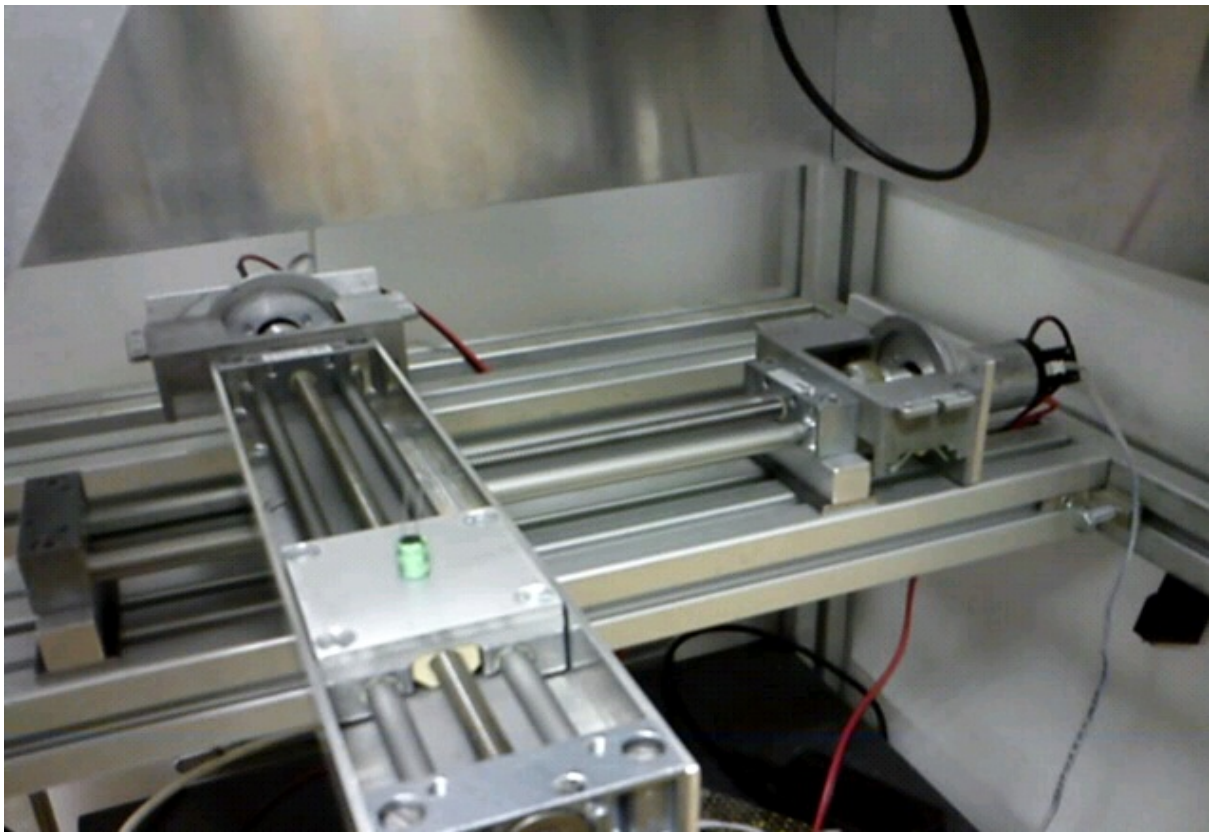
Debrecen
2010

Tartalomjegyzék

Tartalomjegyzék....3
Bevezetés.... 4
Mi is az NI Labview? 7
Virtuális műszerezés....10
Röviden az FPGA-ról....11
A Labview FPGA modul....14
A projekt megnyitásához szükséges szoftveres és hardveres követelmények....17
Csatlakozás az eszközhöz / futtatás / MAX....19
A feladat megoldása / HOST VI....23
Az FPGA VI....30
Összefoglalás....38
Irodalomjegyzék....39
Köszönetnyilvánítás....40

Bevezetés

A szakdolgozatom témája a Kétdimenziós pozicionálás NI eszközökkel. Szakdolgozatom készítése során a DE TTK Szilárdtest Fizika tanszékén működő optikai vizsgálóállomás két dimenzióban mozgatható mintatartóasztalának pontos (tized-milliméteres) mozgatását kellett megvalósítanom. Ezt NI eszközökkel végeztem el, pontosabban NI c-RIO eszközzel és a motormozgatáshoz kifejlesztett NI 9505-ös modullal. A kódolást Labview fejlesztői környezetben végeztem.

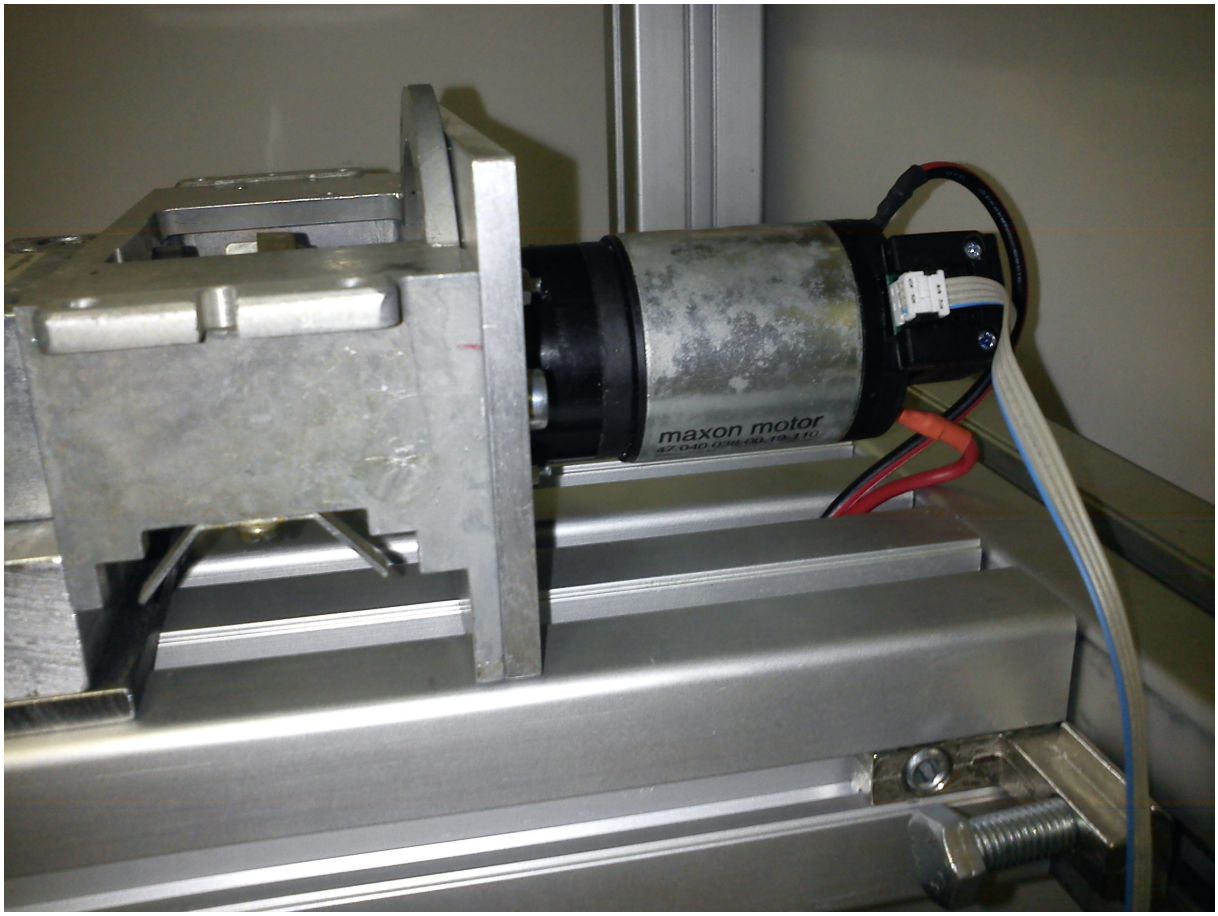


Kétdimenziós pozicionálás

A projektemhez két darab maxon DC motort használtam, amihez maxon tacho enc 22-es encoder van csatlakoztatva.

A maxon motor kiváló minőségű, állandó mágnessel szerelt egyenáramú motor. A motor lelke a System maxon nevű vasmag nélküli tekercselés. Kicsi az induktivitása és a tehetetlensége.

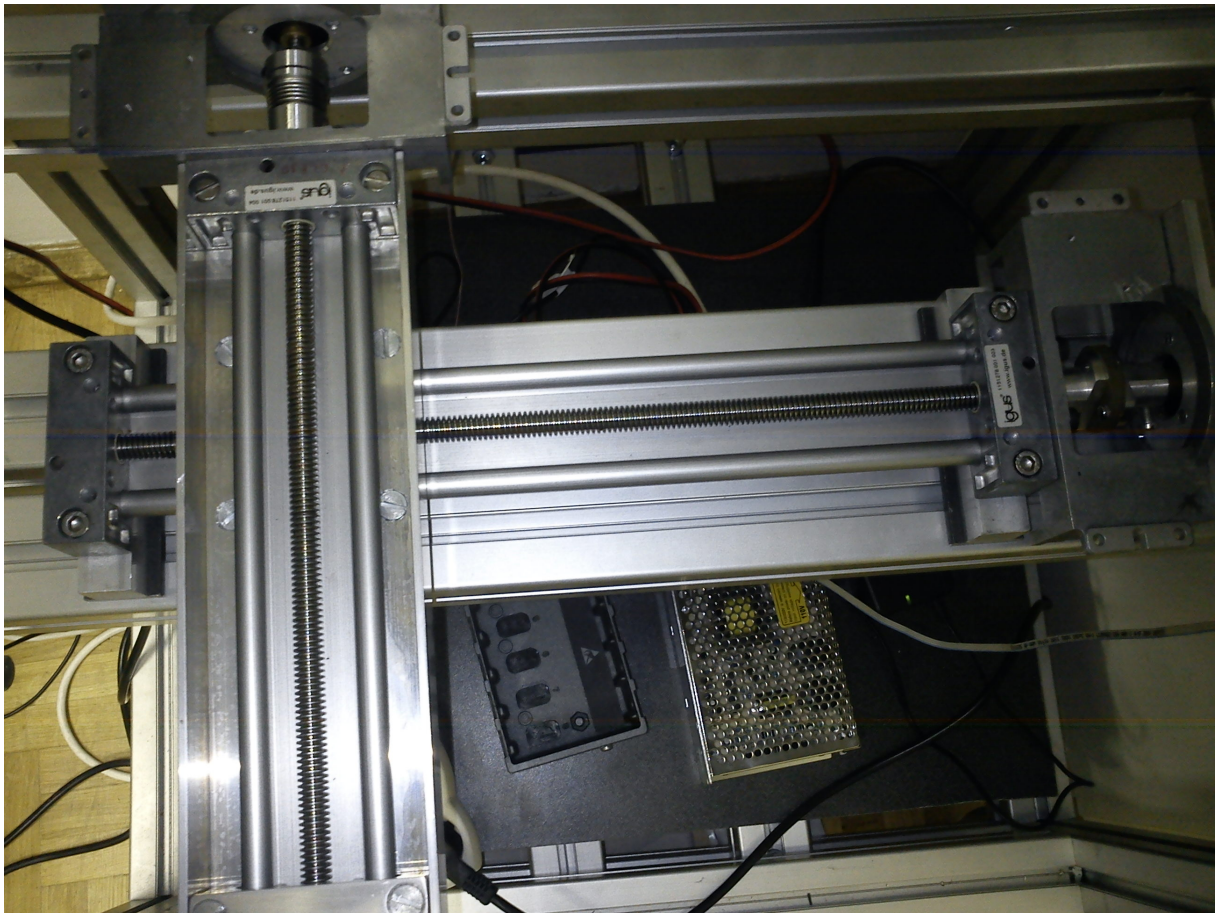
Az alábbi képen látható a motor, ahol a piros és fekete vezetékek, az áramot szolgáltatja a motornak, a kék-fehér vezeték pedig az encoderből továbbít adatokat a CompactRIO-nak.



Maxon DC motor, maxon tacho enc 22-es encoderrel

A motorok egy-egy csigaműves tengelyhez csatlakoznak egy-egy kuplung segítségével. Az X irányú mozgatóegység a berendezés állványához van rögzítve, míg az Y irányú csigaműves meghajtóegység az előbbi asztalához van rögzítve. Így valósul meg a két egymásra merőleges irányú mozgás és pozicionálás. Feltöltöttem egy videót a Youtube-ra, ahol működés közben is meg lehet nézni:

http://www.youtube.com/watch?v=sfORdkYozlY&list=UL63-YzQyri4c&playnext=1



Mi is az NI Labview?

A *virtuális műszerezés* kifejlesztésében vállalt úttörő szerepet a National Instruments cég, amely 1976-ban alakult. A cég legfőbb célja a méréstechnika kutatása és fejlesztése, és hogy az ügyfeleiket mérőeszközökkel, és a hozzájuk kapcsolódó szoftverekkel lássák el.

A hardvereiket csak Austinban, és Debrecenben állítják elő, fejlesztői központok viszont több helyen is vannak a világon. A National Instruments több szoftvert is forgalmaz, például NI TestStand, NI VeriStand, DIAdem, MATRIXx, de ezek közül a legjelentősebb a Labview, amely lényegében egy saját programnyelvre épülő fejlesztői környezet.[1]

A Labview egyik hatalmas előnye, hogy nagyságrendekkel eltér a szokásos szöveges programnyelveket használó fejlesztői környezetektől, mert a programozási feladatokat grafikusan oldja meg, ezáltal sokkal áttekinthetőbb, személetesebb kódot eredményez, és a program megírása is közelebb áll az emberi gondolkodásmódhoz.

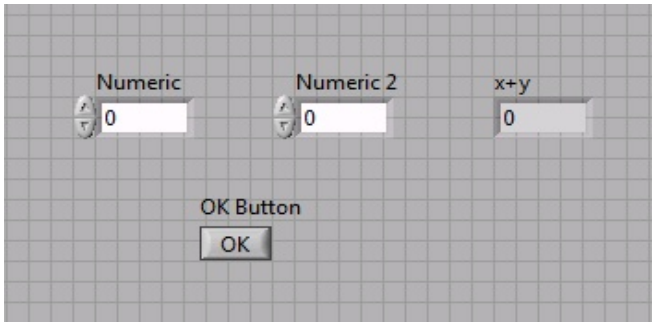
A másik nagy előnye, hogy a National Instruments összes termékén kívül nagyon sok másik cég termékeivel is képes együtt működni.

A különböző mérő és beavatkozó készülékek felismerése, kalibrálása automatizált, rendkívül felhasználóbarát.[2]

A LabView fejlesztői környezet részei:

Front panel

Ez a felhasználói felületünk. Itt találhatóak meg az input (contoll), és az output (indicator) eszközök. Attól függően, hogy milyen típusú adatot szeretnénk manipulálni, választjuk meg a bemenet, és az kimenet típusát.

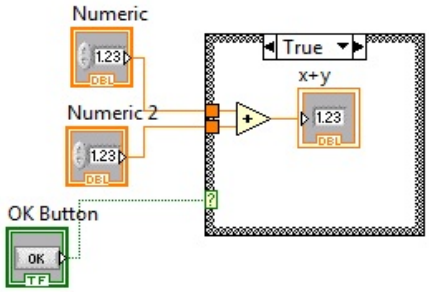


Front Panel

Blokk-diagram

A programozás itt történik. Két csomópont közt huzalozással teremtjük meg a kapcsolatot. A csomópontok funkciójuk szerint lehetnek függvények, struktúrák és SubVI-ok. A fejlesztő saját maga is definiálhat SubVI-okat.

A SubVI-ok tetszőleges mértékig egymásba ágyazhatóak, és a rekurzió és egyszerűen megvalósítható. A programban a huzalok külön szálaknak tekintendők, és akár párhuzamosan is futhatnak egymással.

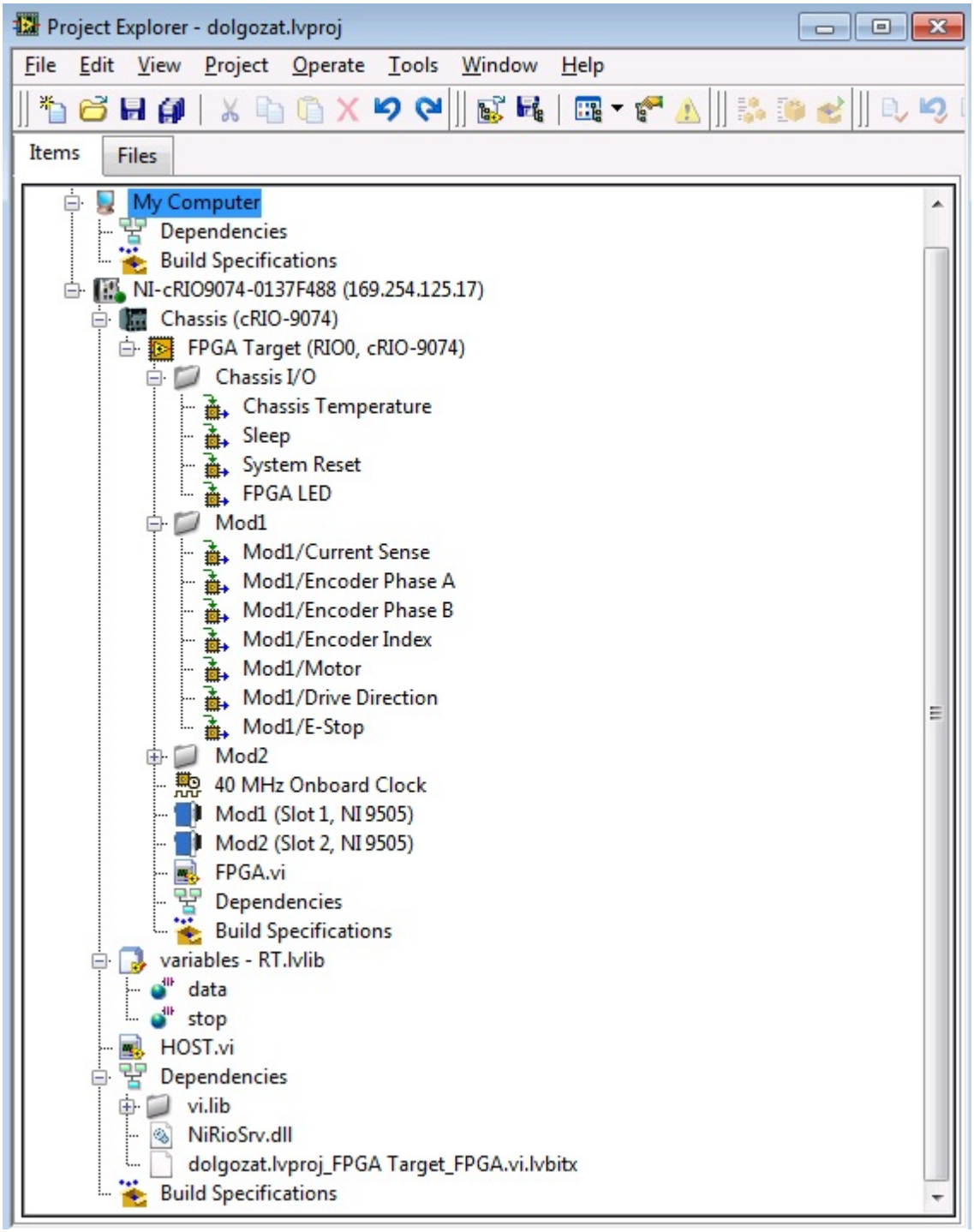


Blokk-diagram

Források: [1] https://inf-moodle.duf.hu/mod/resource/view.php?id=2927

[2] https://inf-moodle.duf.hu/mod/resource/view.php?id=2928

A Labview-nak egy eszköze a Labview Project, amely egy projektnek szervezi össze a fájljait, az NI hardvereket, adatokat és különböző beépített specifikációkat. A projekt információ egy .lvproj fájlban van tárolva.



A szakdolgozatom Projektje

Virtuális műszerezés[3]

A Labview-ban a programokat virtuális műszereknek (virtual instruments) VI-nak nevezik.

A méréstechnikában általában az a feladat, hogy valamilyen fizikai folyamat egy adott paraméterét megmérjük, kiértékeljük, az értékelés után döntést hozzunk, és a döntésnek megfelelően beavatkozzunk a folyamatba. Ennek megfelelően többnyire valamilyen érzékelővel van mérve az adott paraméter, a bejövő jeleket egy erre a célra kialakított berendezéssel feldolgozzuk, majd az eredményektől függően valamilyen beavatkozó szervet, aktuátort, vezérlünk. Ebben a folyamatban a hardverrel van a probléma, mert vagy huzalozott elektronikájúak, ami eleve nagyon nehezen módosítható, vagy módosításuk nagyon nagy szakértelmet, rengeteg idő és energia ráfordítást igényel. Emiatt az előállításuk, módosításuk és az új feladatra való átállás nehézkes, és emiatt nagyon drága.

Ezt a problémát oldja meg a virtuális műszerezés. Itt ugyanúgy megvannak az érzékelők és aktuátorok, a jeleknek a kiértékelését és a feldolgozását nem hardveresen, hanem szoftveresen, programozással oldja meg.

Ennek a módszernek az a legnagyobb előnye, hogy egy átlagos programozói ismerettel bármilyen probléma megoldható egyetlen egy fejlesztői környezet ismeretével. Az elkészült programot bármikor lehet módosítani és folyamatosan fejleszthető. Ezek miatt sokkal olcsóbbá teszi a terméket.[3]

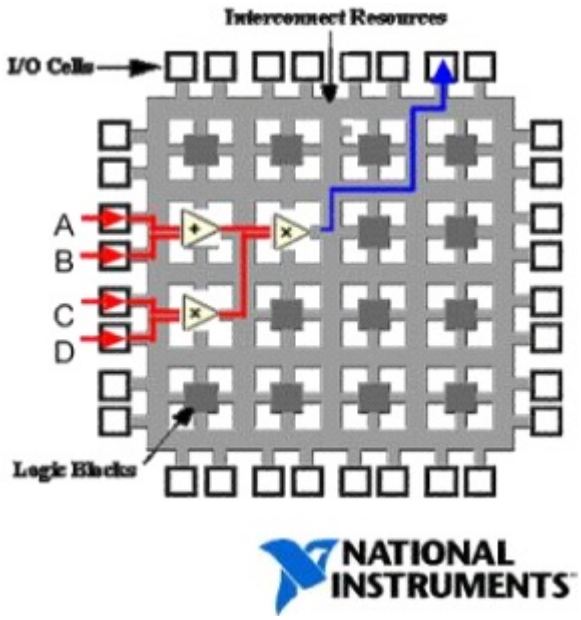
Forrás:

[3] https://inf-moodle.duf.hu/mod/resource/view.php?id=2926

Röviden az FPGA-ról[4]

Az FPGA (Field-Programmable Gate Array = helyszínen programozható, logikai kapukat tartalmazó tömb) olyan félvezető eszköz, amely logikai blokk-oknak nevezett programozható logikai komponenseket és programozható összeköttetéseket tartalmaz.

Például az F=(A+B)*C*D műveletnek az értékét, ahol az F a kimenet, az A, B, C és D pedig a bemenetek, a következőképpen implementálja a VI:[5]



Jellemzői:

Az FPGA logikai blokkok úgy programozhatók, hogy azok olyan logikai kapuk funkcionalitásával rendelkezzenek, mint az ÉS (AND) és kizáró VAGY (XOR) kapuk, vagy akár olyan bonyolultabb funkcionalitással, mint egy dekóder vagy egy matematikai függvény megvalósítása.

A programozható összeköttetések hierarchiája lehetővé teszi, hogy logikai blokkok a rendszertervező által igényelt módon kapcsoljanak össze. Az FPGA legyártása után a logikai blokkokat és összeköttetéseiket a felhasználó vagy a tervező programozhatja.

Az FPGA-k általában lassúbbak, mint alkalmazás-specifikus integrált áramkörök, nem is tudnak olyan bonyolult tervezést kezelni mint az ASIC, és bármilyen félvezető-gyártási technológia esetén nagyobb teljesítményt vesznek fel. Nagy előnyük a rövidebb piacra kerülési idő, a helyszíni újraprogramozhatóság és az alacsonyabb költség.

Xilinx FPGA-chip[6]

Alkalmazások

Az FPGA alkalmazási területe egyre növekszik, szélesedik:

digitális jelfeldolgozás, DSP, szoftveres rádió, űrkutatási és katonasági rendszerek, ASIC prototípuskészítés, orvosi képalkotás, beszédfelismerés, kriptográfia, bioinformatika, számítógép hardver emuláció.

Az FPGA-kat különösen gyakran használják olyan algoritmikus területeken, amelyeken jól használhatók az architektúrájuk által kínált masszív párhuzamosság. Az egyik ilyen terület kriptográfiai algoritmusokban a kódfeltörés, különösen a nyers erő módszerű támadások esetén.

Az FPGA-kat növekvő mértékben használják a hagyományos nagyteljesítményű számítási alkalmazásokban, ahol a számítás, nagy részét az FPGA végzi a mikroprocesszor helyett.

Az FPGA logikai erőforrásaiból eredő párhuzamosság a számítási teljesítményt jelentősen megnöveli. Az FPGA flexibilitása még nagyobb teljesítményt is lehetővé tesz, ha a párhuzamos aritmetikai egységek számának megnövelése kedvéért engedünk a számábrázolási formátum pontosságából és számábrázolási tartományából.

Az FPGA-k felhasználását nagyteljesítményű számításokban jelenleg az korlátozza, hogy az FPGA-k tervezése sokkal bonyolultabb, mint a hagyományos szoftveré és extrém hosszúságú fejlesztési idők alatt valósítható meg.

FPGA programozáshoz hardver leíró nyelvet használunk (HDL = Hardware Description Language), mint például VHDL vagy Verilog.

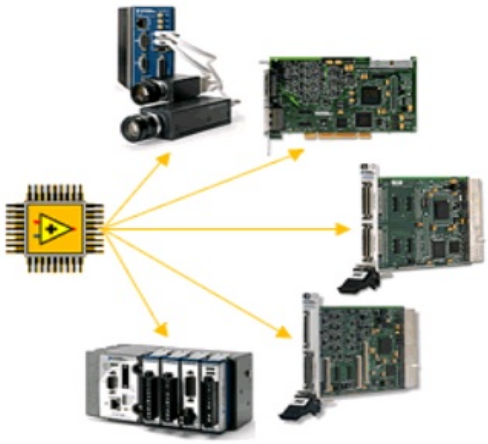
Források:

[4] http://hu.wikipedia.org/wiki/FPGA_%28Field-programmable_gate_array%29
[5] ftp://ftp.ni.com/pub/devzone/tut/fpga_training_slides.zip
[6] http://www.nowires.com.br/blogger/uploaded_images/FPGA-708477.jpg

A Labview FPGA modul

A Labview FPGA modul egy kiegészítő modulja a Labviewnak. Ezeket használva, mindenféle elsődleges HDL tudás nélkül lehet fejleszteni.
A Labview FPGA modullal VI-okat lehet készíteni, amit rá lehet tölteni az FPGA vagy a RIO hardverre. A számítógépre is lehet készíteni VI-t, amely segítségével irányítani lehet az FPGA-t, adatokat lehet rá továbbítani.

A következő NI hardver platformok léteznek:
CompactRIO, PXI Rseries Multifunction RIO, PCI R Series Multifunction RIO,
Compact Vision System, FlexRIO,RIO Instruments.



NI hardver platformok[7]

Forrás: [7] http://www.ni.com/fpga/rio.htm

Hogyan is működik?

A Labview FPGA modul lefordítja az általunk elkészített Labview alkalmazást az FPGA hardverre egy fordító program segítségével. A grafikus kód lefordul egy szöveg alakú VHDL kódra, ezután a beépített Xilinx ISE fordító eszközök optimalizálják és előállítják a Labview kód hardver áramkörét. Végül egy bitfolyam, ami tartalmazza a programutasításokat, letöltődik az FPGA-ra.

A Labview FPGA két legfontosabb előnye, hogy VHDL nyelv ismerete nélkül lehet programozni az FPGA-t és a Labview blokk diagramon keresztül közvetlen hozzáférésünk van a hardver terminálhoz.

Mi a CompactRIO?

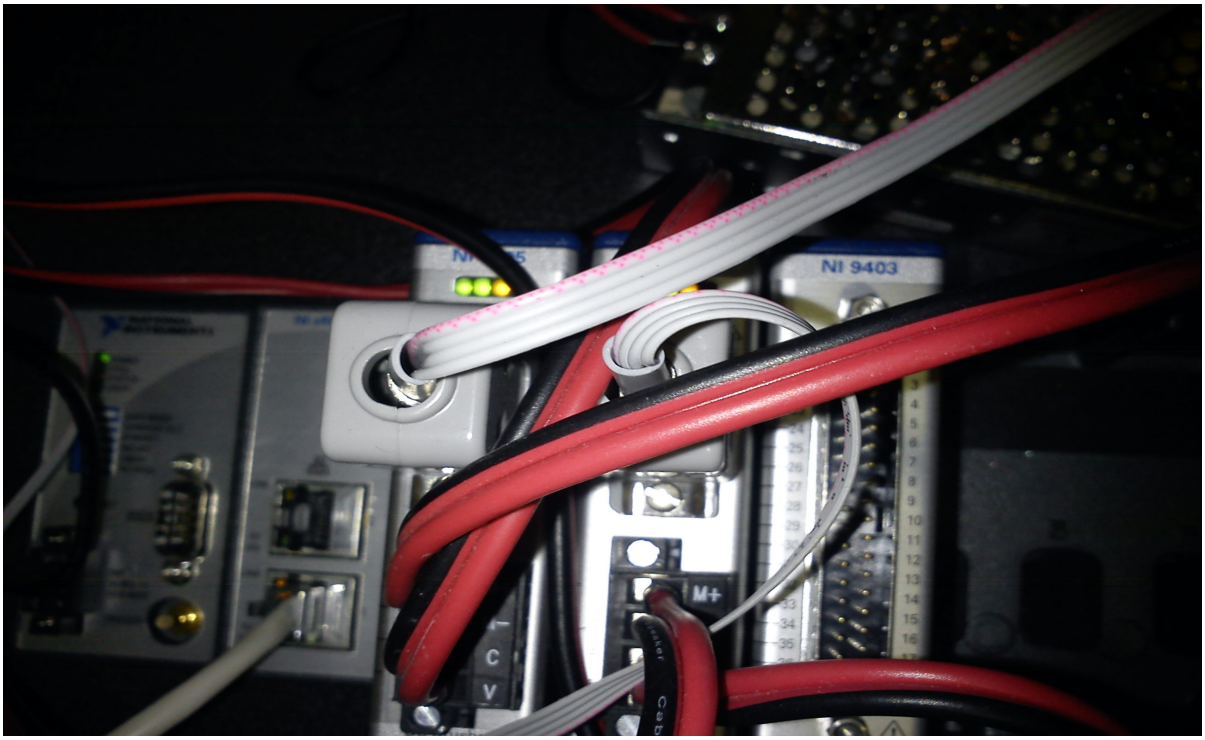
A CompactRIO (Compact Reconfigurable I/O) egy beágyazott rendszer, amit gépek irányítására, adatgyűjtésre és hordozható NVH alkalmazásokhoz használják.
Beágyazott rendszereknek nevezzük, azokat a processzor alapú eszközöket, illetve az ezekből alkotott rendszereket, amelyek képesek a befogadó környezetüket, érzékelők segítségével autonóm módon megfigyelni és amennyiben szükséges, beavatkozók segítségével befolyásolni.
CompactRIO rendszerek integrált hardver architektúrával rendelkeznek, amelyek a beágyazott valósidejű processzort és az újrakonfigurálható FPGA-t egyesítik egyetlen cRIO keretben. Ennek a felépítésnek köszönhetően csökken az OEM alkalmazásokra szánt CompactRIO eszközök ára.
A controller modul egy beágyazott valósidejű processzor, ami soros porton és Etherneten keresztül kommunikál. Etherneten keresztül való kommunikációhoz egy helyi hálózaton kell lenniük, vagy lehet közvetlenül is csatlakoztatni, lehetséges hogy a tűzfalat ki kell kapcsolni.

A cRIO-9074 CompactRIO modul beágyazott hardver architektúrával rendelkeznek, amely az erőteljes lebegőpontos processzor, valamint az újrakonfigurálható FPGA és I/O modulok kombinációját jelenti. A megbízhatóság és a minőség növelése mellett ez a standard architektúra csökkenti az egyedi berendezések piacra jutási idejét is. A standard struktúrának, valamint a LabVIEW grafikus fejlesztői környezet eszközeinek köszönhetően a fejlesztők sokkal gyorsabban végre tudják hajtani a tervezés és prototípus létrehozásának lépéseit, majd

ezt rövid idő alatt adaptálni lehet a költséghatékony cRIO-907x CompactRIO rendszerekre.

A prototípus készítés és az üzembe helyezés során a fejlesztők újra használhatják a korábban megírt LabVIEW kódokat, mely jelentősen csökkenti a piacra jutási időt és egyúttal növeli a gép megbízhatóságát is.

A CompactRIO együttműködik a NI LabVIEW FPGA és a LabVIEW Real-time technológiákkal, ami lehetővé teszi a felhasználóknak a személyre szabott beágyazott rendszerek tervezését, fejlesztését és terjesztését számtalan iparágban mint például autógyártás, hadi/légi ipar, folyamat- és gépvezérlések.[8]



CompactRIO és a hozzá csatlakoztatott modulok

Forrás:

[8] http://www.cegnyilvantarto.hu/cikkek/62/uj-compactrio-rendszer/

A projekt megnyitásához szükséges szoftveres és hardveres követelmények:

Szoftveres

Labview (NI Academic Site License) 2009 Core Software

Labview Device Driver 2009 August

Ni-RIO 3.2.1

Labview Realtime

Hardveres:

NI cRIO-907x[9]

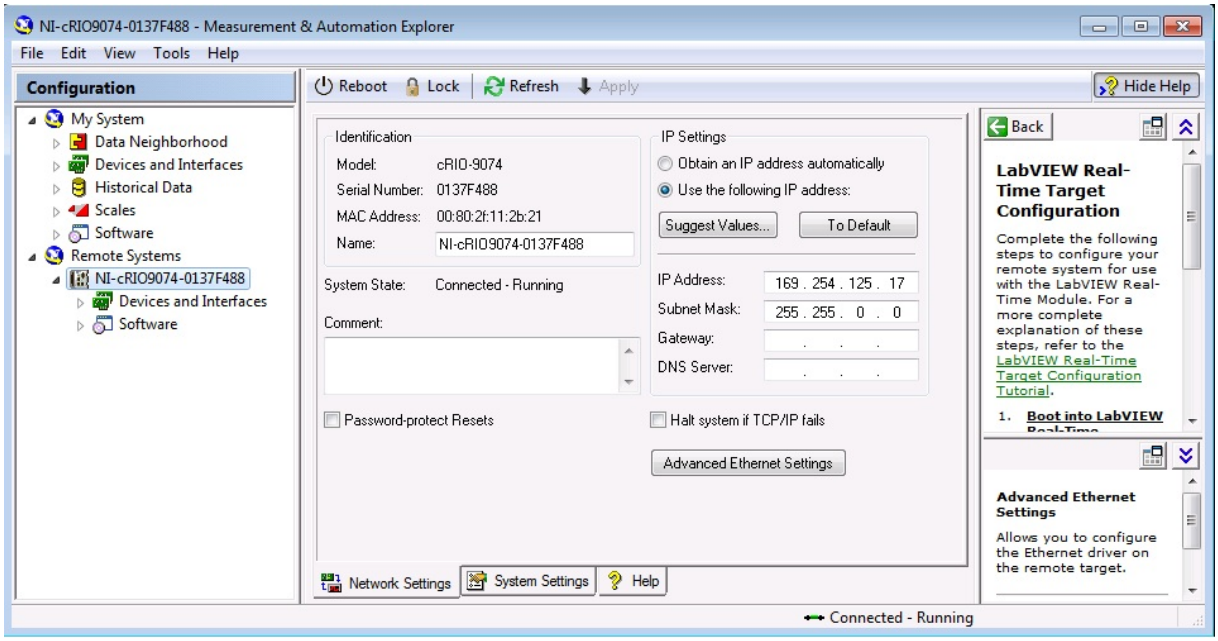
NI 9505 Full H-Bridge Brushed DC Servo Drive Module[10]

Források:

[9] http://sine.ni.com/np/app/main/p/ap/global/lang/en/pg/1/sn/n24:cRIO/fmid/102

[10] http://sine.ni.com/nips/cds/view/p/lang/en/nid/202711

Csatlakozás az eszközhöz/ futtatás/ MAX

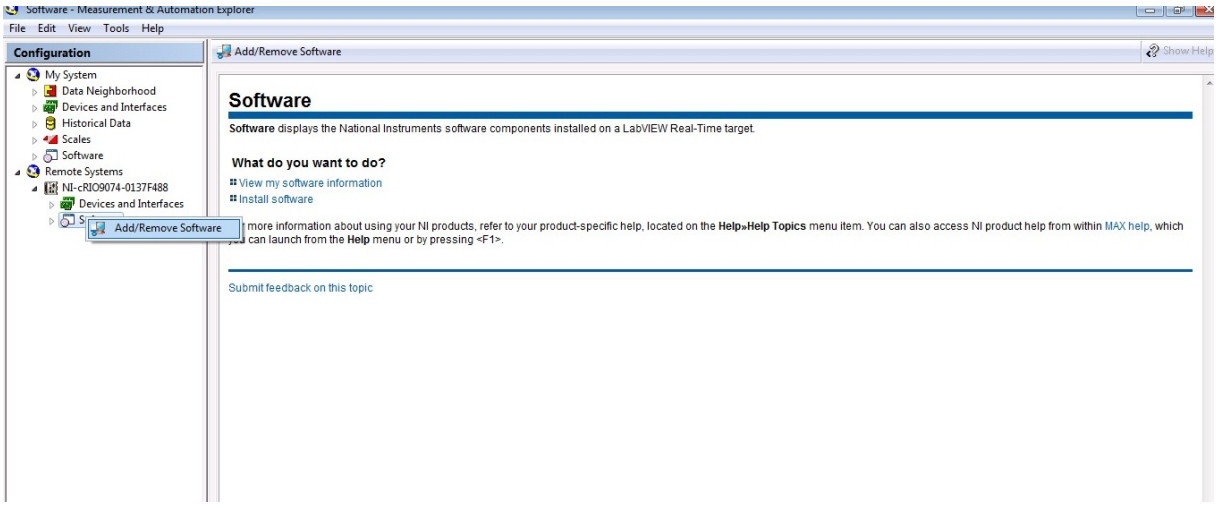


A Measurement & Automation Explorer-t elindítva, lehet megnézni, hogy milyen eszköz van a számítógépünkhöz csatlakoztatva, különböző információk állnak a rendelkezésünkre: a modell neve, sorozatszám, MAC cím, neve és a rendszer státusza. Itt lehet konfigurálni az eszköz hálózati beállításait: IP cím, ami az eszközünk IP címe, jelen esetben 169.254.125.17, Subnet maszk, alapértelmezett átjáró, DNS szerver. Ha az alapértelmezett átjáró és a DNS szerver nem elérhető, vagy nincsen, akkor üresen kell hagyni.

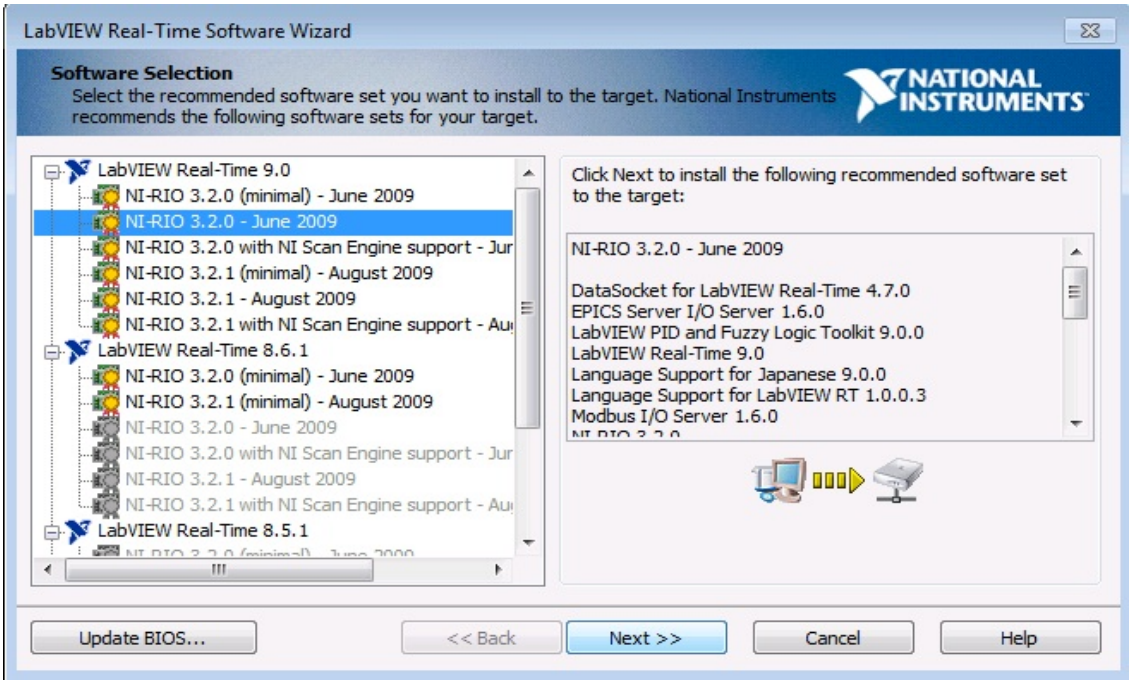
Az információk felett található a Reboot gomb, amivel újra lehet indítani a CompactRIO rendszert, a Lock gombbal le lehet zárni a beállításokat, a Refresh-sel frissíteni lehet a rendszer státuszát, az Apply gomb pedig a változtatások alkalmazására szolgál.

Mielőtt nekikezdenénk a munkának be kell álltani a CompactRIO eszközünknek, hogy milyen verziójú Labview-val programozunk, ha ezt nem tesszük le akkor nem fognak lefutni a projektjeink.

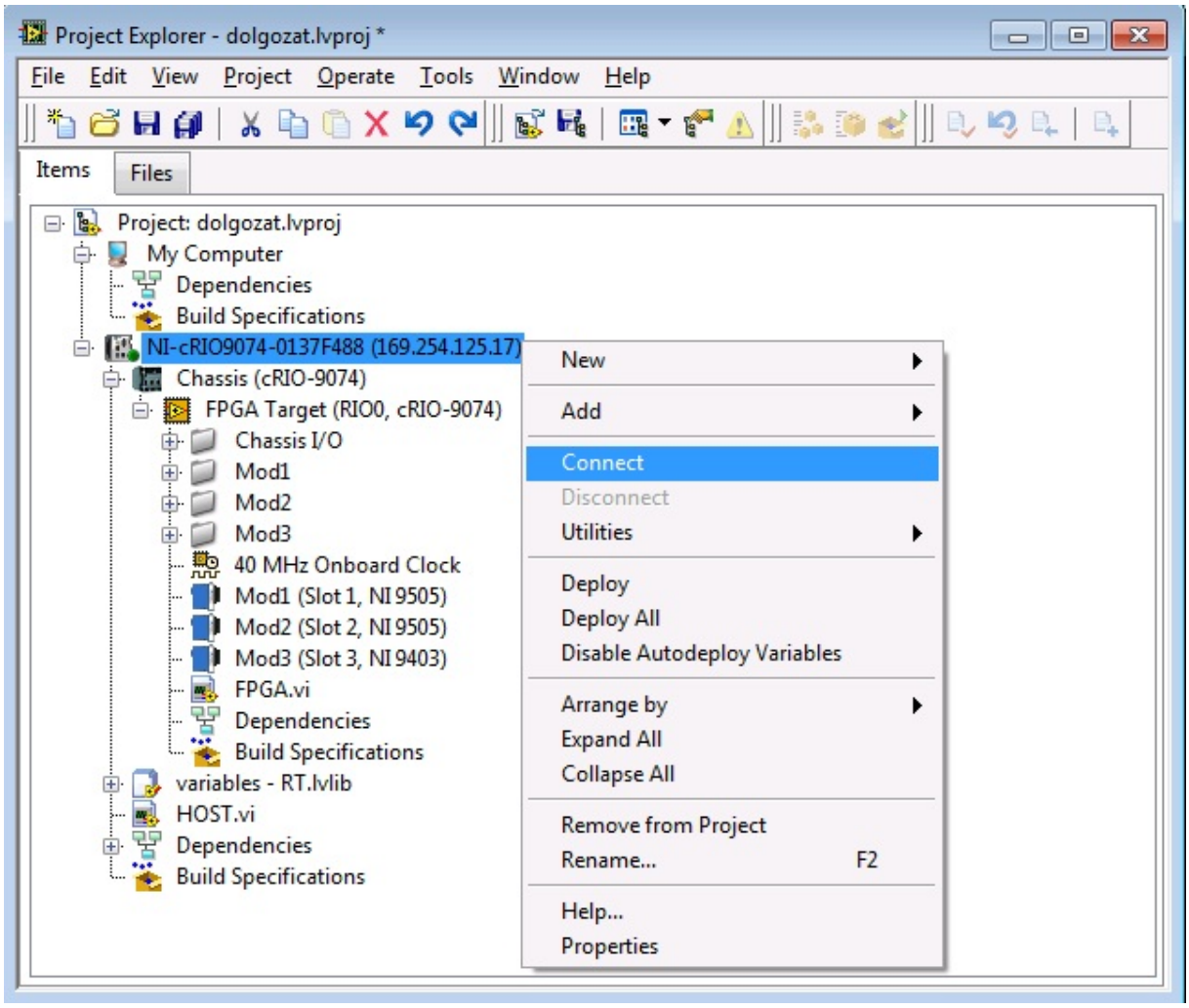
A MAX-ben kiválasztjuk az eszközünket, majd rákattintunk a Software-re és ott kiválasztjuk az Add/Remove Software menüpontot.



Ezután előugrik a Labview Real-Time Software varázsló, ahol kiválasztjuk a gépünkre telepített szoftverünket és a Next gombra kattintva telepítjük a CompactRIO-ra.



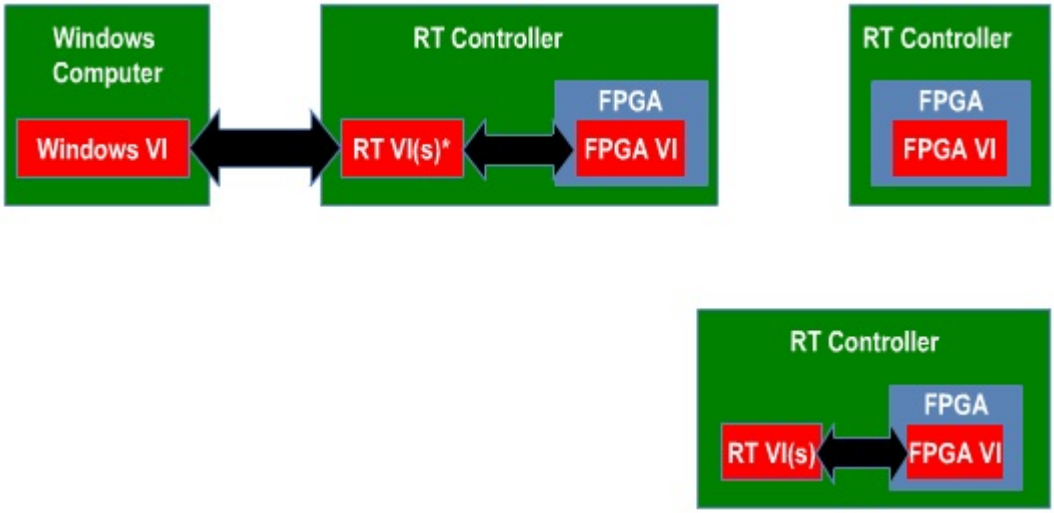
A Project Explorerben jobb kattintással az eszközre, majd a Connect gomb kiválasztásával lehet csatlakozni a CompactRIO-hoz.



Csatlakozás a CompactRIO rendszerhez

A c-RIO-t két, elvileg különböző üzemmódban lehet használni:

1. A Real-Time kontrolleren fut egy VI, ami irányítja a folyamatot és átadja illetve átveszi a paramétereket az FPGA-ról, ahol az FPGA-ra megírt a program fut. Ez az az üzemmód, melyben leginkább kihasználhatjuk a a rendszer lehetőségeit. Ekkor a mérési és beavatkozási folyamatokat szabadon időzíthetjük, triggerelhetjük. Késleltethetjük az adatfeldolgozást és maximálhatjuk az adatgyűjtési sebességet. Amennyiben szükség van felhasználói beavatkozásra, a felhasználói felület a hálózaton a c-RIO eszközhöz kötött számítógépen jelenik meg.
2. A számítógépen semmilyen folyamat nem fut, csak a Real Time kontroller továbbítja a mért adatokat a felhasználónak. Teljes mértékben a CompactRIO Real Time részében fut a program. Ilyenkor az FPGA néhány 100Hz-en periodikusan szkenneli az I/O modulokat és adja át az adatokat a Real Time modulnak. Kis sebességek esetén ez a módszer elegendő.



FPGA irányítási módok[11]

A projektemben az első irányítási mód fut, mivel a Real-Time operációs rendszere sokkal stabilabb mint a Windows-é és a Real-Time eszköznek 1MHzes órajele is lehet, míg a Windowsnak csak 1kHz, valamint ekkor használhatjuk ki az FPGA lehető legtöbb előnyét.

Forrás: [11] ftp://ftp.ni.com/pub/devzone/tut/fpga_training_slides.zip

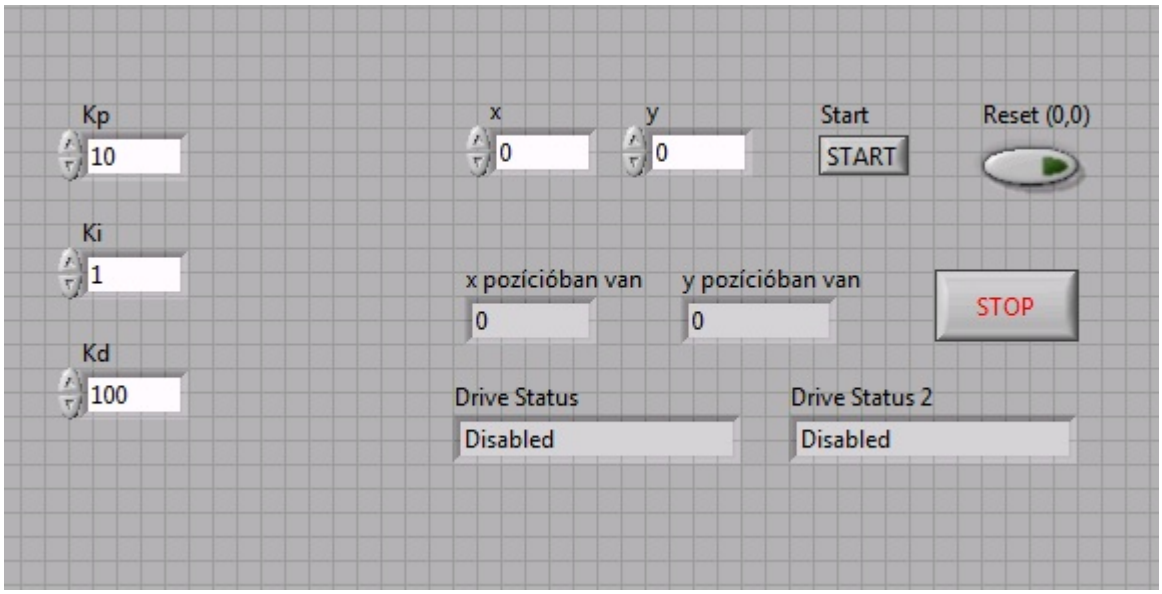
A feladat megoldása

A szakdolgozatom során készült Labview projekt két legfontosabb része az FPGA.vi és a HOST.vi

Először a HOST.vi működéséről írok részletesebben.

A Labview fejlesztői környezet két részből áll: Front Panelből és Blokk Diagramból.

A felhasználók a következő képen látható front panelt látják:



Front Panel

A Run gombra kattintva futtathatjuk a VI-t. A nyíl ikonja ekkor megváltozik, egy feketével kitöltött szaggatott nyíllá.

A következő részek találhatók a Front Panelen:

A szürke „dobozban” lévő információk az indikátorok, a program kimenete, míg a fehér, a program bemenete, a kontroll, ezeket bármikor, akár futtatás közben is lehet változtatni.

x,y: itt lehet beállítani melyik pozícióba kerüljön, centiméterben van megadva
x,y pozícióban van: éppen hol vannak a motorjaink, centiméterben megadva
Start: rákattintva elindulnak a megadott koordinátákra
Drive Status, Drive Status 2: ez a két helyzetjelző mutatja a felhasználónak, hogy engedélyezve van e a két motor, két értéket vehet fel: Enabled és Disabled
Reset (0,0): visszahelyezi a 0,0 pozícióra a motorokat
STOP: szabályosan leállítja a programot

A PID szabályzó beállítása:

Az értékeket a Ziegler-Nichols módszerrel állítottam be.

Kp: P tag

Ki: I tag

Kd: D tag

A PID szabályozó egy lineáris rendszerek szabályozásánál gyakran alkalmazott, soros kompenzáción alapuló szabályozótípus. A PID rövidítés a szabályozó elvére utal, a szabályozó által kiadott végrehajtójel

- a hibajellel (**P**: *proportional*),
- a hibajel integráljával (**I**: *integral*), valamint
- a hibajel változási sebességével, deriváltjával (**D**: *derivative*)

arányos tagokból adódik össze. Ezen tagok közül nem mindig valósítják meg mindet, ilyenkor beszélhetünk P, PI, PD szabályozókról.[12]

A Ziegler-Nichols módszer[13]

A Ziegler-Nichols módszert csak olyan folyamatoknál lehet alkalmazni, amelyeknél a technológia megengedi, hogy a szabályozási kört a stabilitás határán működtessük.
A hangolási módszernél nem szükséges ismerni a rendszer válaszát.
A hangoláshoz a szabályozási körből kiiktatjuk az integráló és deriváló csatornát a. Így a szabályozó egy erősítőre redukálódik. A szabályozó Kp erősítését nulláról kell növelni addig, amíg a zárt rendszer eléri a stabilitás határát. A stabilitás határán állandósult állapotban a folyamat kimenete szinuszosan leng az alapjel körül. Jelölje K_{Pkrtit} a kritikus erősítést, vagyis a szabályozó erősítését a konstans amplitúdójú lengések bekövetkeztekor. Jelölje T_{krit} a kritikus periódust, a konstans amplitúdójú lengések periódusát. A szabályozó hangolása ezen értékek alapján történik.

P	$K_P = 0.45\,K_{Pkrit}$	-	-
PI	$K_P = 0.45\,K_{Pkrit}$	$T_i = 0.85\,T_{krit}$	-
PID	$K_P = 0.6\,K_{Pkrit}$	$T_i = 0.5\,T_{krit}$	$T_D = 0.12\,T_{krit}$

Ziegler-Nichols módszer - hangolás

A szabályozási kör csillapítása a táblázat alapján is ζ=0.25, ami 40% körüli túllövést eredményez.
Mintavételes megvalósításnál a mintavételi periódust $T \approx (0.1 \ldots 0.3)T_{krit}$ körüli értékre kell választani.
A hangolás előnye, hogy nincs szükség az egységugrásra adott válaszra, az egyetlen paraméter, amit a folyamatról ismerni kell a T_{krit}. A hátránya egyrészt hogy a szabályozási rendszert el kell vinni a stabilitás határára, másrészt, hogy ha az irányított folyamat lassú, a hangolás időigényes. Másik előnye, hogy a kidolgozott táblázat használható az önhangoló szabályozások megvalósításánál.

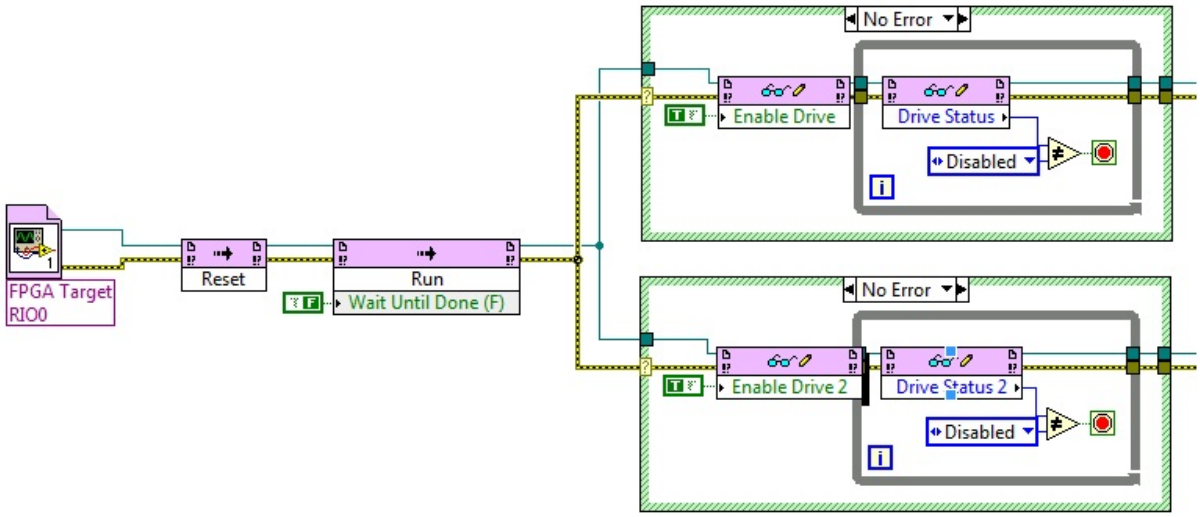
Forrás:

[12] http://hu.wikipedia.org/wiki/PID_szab%C3%A1lyoz%C3%B3

[13] http://www.ms.sapientia.ro/~martonl/Docs/Labs/IRI_L7.pdf

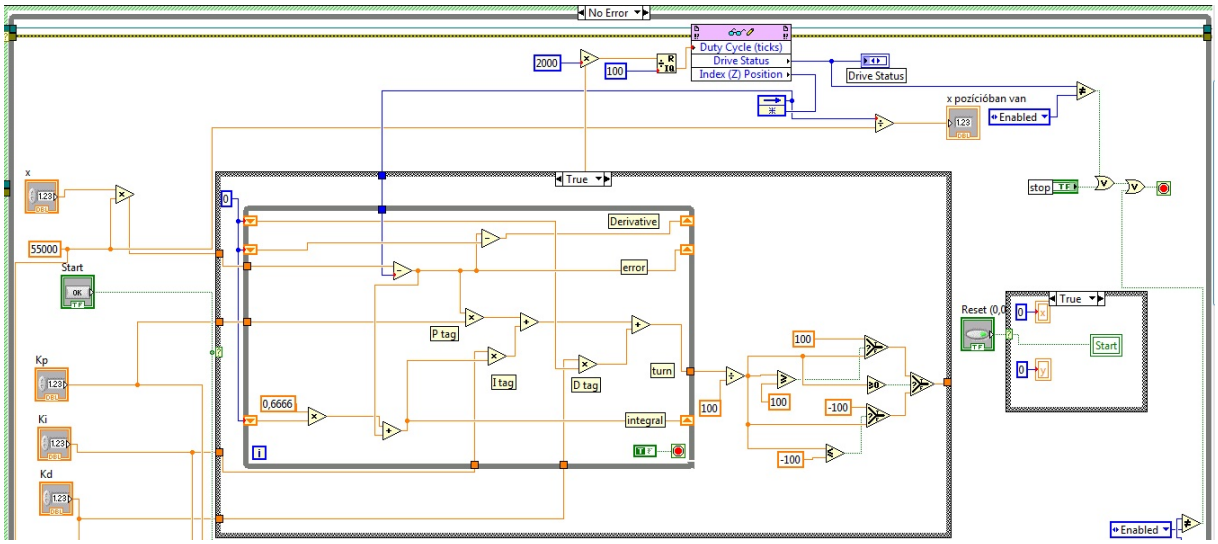
Blokk Diagram:

A Ctrl+E megnyomásával előugrik a blokk diagram, ami gyakorlatilag a program. Mivel -ahogyan arról korábban mér volt szó- a Labview egy un. adatfolyam programozási nyelv, a blokk diagramot is balról jobbra olvassuk:



Az első függvény az Open FPGA VI Reference Out megnyitja az FPGA VI hivatkozását, majd az Invoke Method, segítségül hív egy FPGA Interface metódust, ami elindítja az FPGA-t, majd elkezdi futtatni az eszközre írt és korábban lefordított programot. A Case struktúránál, ha nincs hiba engedélyezi a motorokat, ha a Drive Status egyenlő a Disabled -del, akkor leállítja a VI-t. Ha van hiba, akkor nem engedi futni a programot.

A következő képen, a X motor irányítása található, az Y motoré is ugyanígy néz ki, csak a változók nevei eltérőek. A while ciklust körül veszi egy Case struktúra, ami ha hibát észlel nem engedi futni a programot. A while ciklus akkor áll le, ha megnyomjuk a STOP gombot, vagy valamelyik Drive Status értéke „Disabled”.



A képernyő jobb oldalán találhatóak, az értékadások, x, y, Kp, Ki, Kd, valamint egy 55.000 értékű konstans, amivel megszorozzuk az x,y-t, így egy egység egy centiméternek felel meg, majd elosztjuk ezzel a számmal az encoder pozíciótól kapott értet, ezzel megkapjuk pontosan hány centiméternél van jelenleg az x,y motor. A Start gomb, amivel elindítjuk a Case struktúrát. Ha, hamis értéket kap, tehát nincsen lenyomva a gomb, akkor 0 sebességet ad át az FPGA-nak.

Ha igaz értéket kap, akkor elindul a belső while ciklus, ahova betöltődnek a kezdő értékadások. Itt található öt belső változó:

error: ez az aktuális hibaérték, ami úgy számoljuk ki, hogy a megadott x értékből kivonjuk az encoder értékét

last error: egy shift regiszter segítségével átadjuk a ciklusnak az utolsó hiba értéket

derivative: a deriváltat úgy számoljuk ki, hogy a jelenlegi hibából, kivonjuk az utolsó hibát

integral: az integrált eredményét úgy számoljuk ki, hogy az előző integral érték háromnegyedéhez, ami kezdőértéknek nulla, hozzáadjuk az aktuális hiba értéket.

turn: ez a motor sebességének megadása százalékban, -100 és +100 közötti értékeket ad át az FPGA-nak, ahol a pozitív előjeles számok óramutató járásával megegyezően, a negatív előjeles számok az óramutató járásával ellentétesen forognak. A -4 és +4 közötti számokat nullára konvergálja. A turn értékének kiszámításához PID szabályozást alkalmaztam.

PID szabályzás

A stabilis folyamatokra pontos elméleti módszerek állnak rendelkezésre, amelyek alapján a legkülönbözőbb esetekre meghatározható az optimális szabályozó struktúrája és a paramétereinek optimális értéke. Ezek az elméleti módszerek kidolgozása előtt kialakult a szabályozó berendezések egy olyan osztálya, a PID szabályzó, amely mind a mai napig meghatározó jelentőséggel bír az ipari folyamatok és technológiák irányításában.

A PID szabályzást elve a következő:

A szabályozási hiba aktuális értéke (P tag) arányosan, a múltjára a hiba integráljával (I tag) arányosan reagáljon, a jövőt pedig a hiba differenciálhányadosával (D tag)vegye figyelembe

P csatorna: Kp * (megadott érték – encoder érték)

I csatorna: Ki * ¾ * (integrál + hiba)

D csatorna: Kd * (hiba – utolsó hiba)

A turn értéket a következő módon számoljuk ki: Kp * error + Ki * integral + Kd * derivative.

A turn nagyságrendekkel nagyobb, illetve kisebb értéket vehet fel, ezért a könnyebb számolás érdekével elosztjuk 100-al, majd behatároljuk -100 és +100 közé:

Ha a kapott turn,

nagyobb vagy egyenlő nullával, és

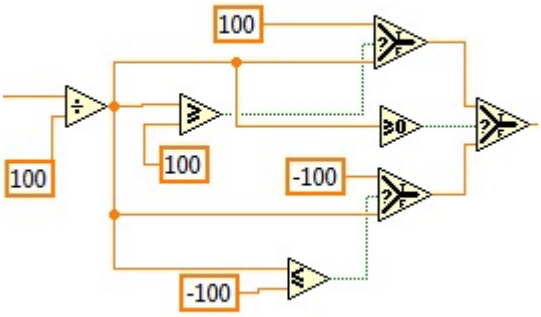
1: ez az érték nagyobb mint 100, akkor 100as értéket ad tovább.

2: ez az érték nem nagyobb mint 100, akkor az értéket adja tovább.

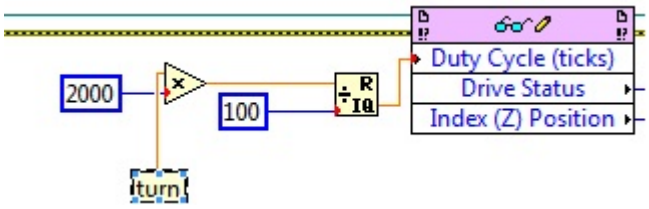
NEM nagyobb vagy egyenlő nullával és

1: ez az érték kisebb mint -100, akkor -100as értéket ad tovább.

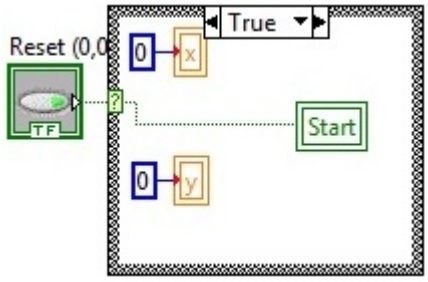
2: ez az érték nem kisebb mint -100, akkor az értéket adja tovább.



A turn -t megszorozzuk 2000-rel és a Quotient & Remainder művelettel kiszámolja az egész hányadosát, majd átadja az értéket a Read/Write Control segítségével az FPGA VI-nak.A Control segítségével nyerünk információt a motorok státuszáról és az encoder pozícióról.



A Reset gomb megnyomásával 0,0 koordinátára beállítódik az x,y és igaz értéket ad a Start gombnak, így a megadott sebességgel a 0,0 pozícióba állnak be a motorok.



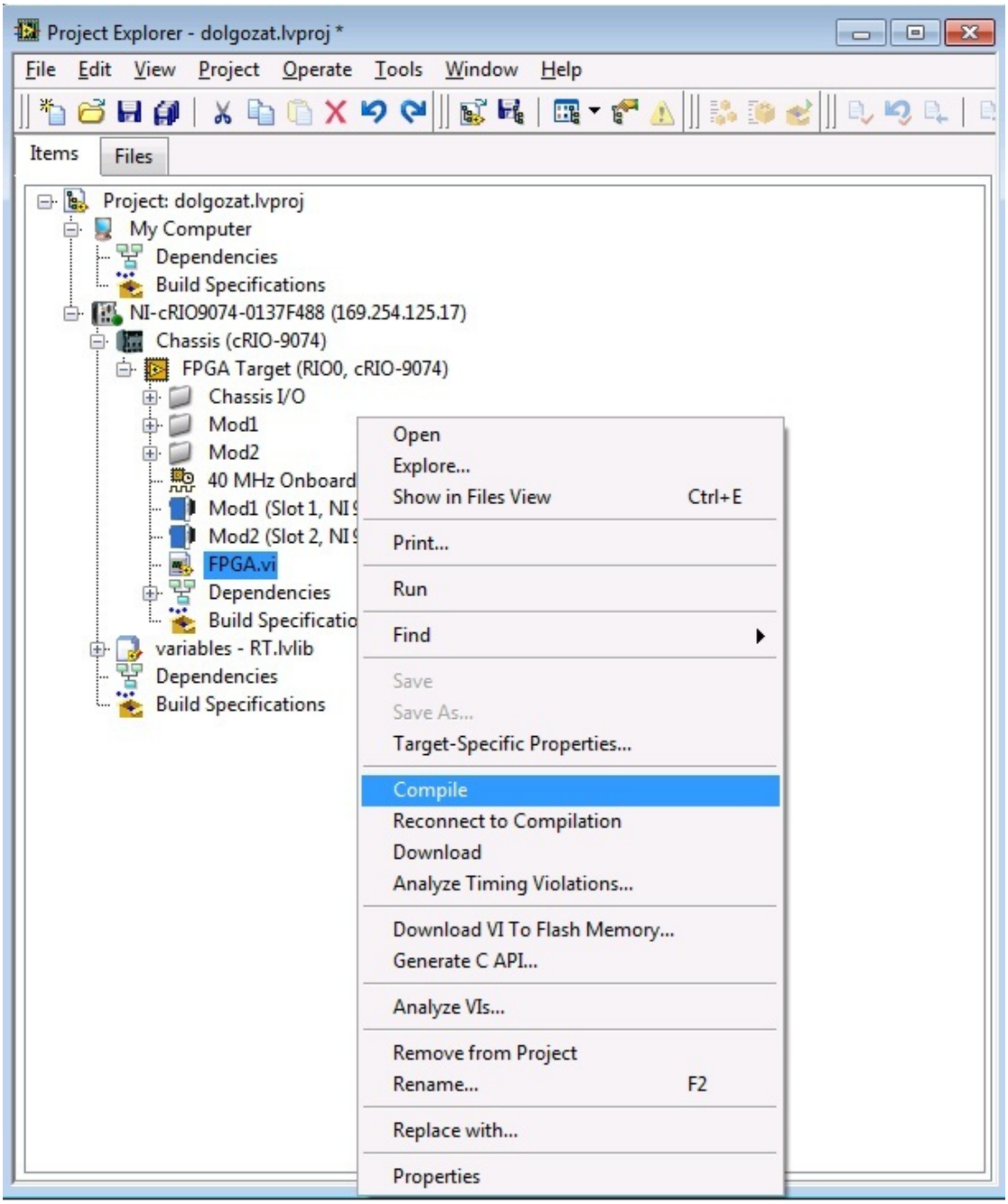
Ha valamelyik státusz nem Enabled, vagy le nyomják a Stop gombot, akkor leáll a while ciklus.

A Host.vi-t a Close FPGA VI Reference zárja le, amely leállítja az FPGA VI futását az FPGA-n és bezárja a hivatkozását a VI -nak.

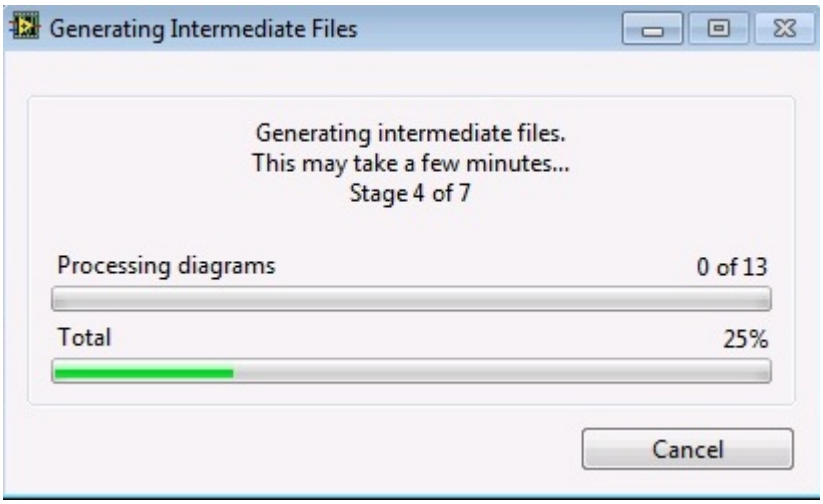
Az FPGA.vi

Ez a VI kerül rá a CompactRIO FPGA-jára, előtte le kell fordítani, ami egyéb fejlesztői környezetekhez viszonyítva, relatíve sok időt vesz igénybe, ez függ persze a számítógéptől amin írjuk a programot. Egy Intel Core I3-370M-es 2,4GHz-es, 4GB DDRIII-as PC-vel körülbelül 10 percig tart.

A projekt megnyitása után az FPGA.vi -ra jobb gombbal kell kattintani és a compile-ra kattintva lehet lefordítani a VI-t.



Ezután generálja a közvetítő fájlokat, és átalakítja a grafikus kódot VHDL kóddá.



Majd megnyílik a Labview FPGA Compile Server ablak, ahol a fordítás menetéről tekinthetünk meg különböző információkat:

Compile Status: az aktuális fordításról ad információkat.

Server Service ID: megadja a fordítandó VI-ok számát.

Client Name: megadja a számítógép nevét, amin az FPGA VI található.

Client Service ID: megadja az azonosítóját a VI-nak

Status: megadja a fordítás aktuális helyzetét

Start Time: megadja a fordítás elkezdésének az idejét

Last Update Time: megadja mikor frissítette utoljára a Compile Status-t

Details: információkat ad meg a VI-ról bitstream-be való átalakításáról, ami le lett töltve az FPGA-ra

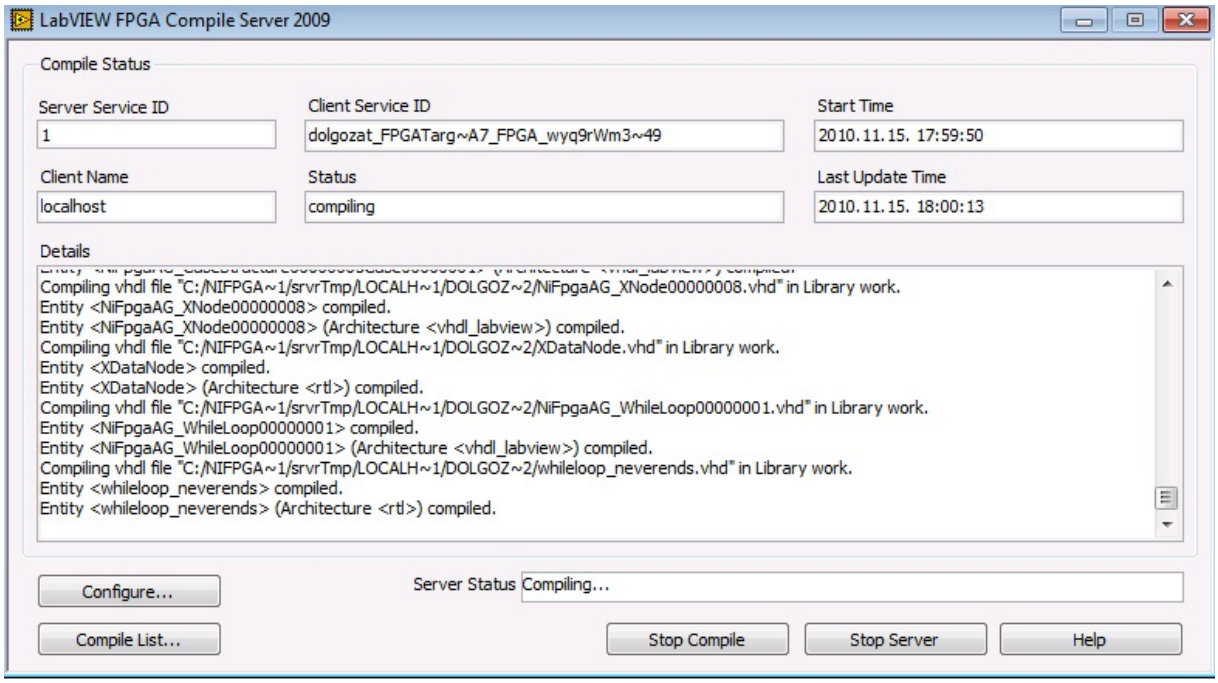
Configure: ezzel a gombbal lehet előhozni a szerver konfigurációs ablakot, ahol be lehet állítani a Labview FPGA Compile Server-t

Compile List: a felugró ablakban, meg lehet keresni az összes régebbi fordításokat és az aktuális fordítást a fordítási sorban.

Server Status: a Labview FPGA Compile Server helyzetét mutatja

Stop Compile: törli az aktuális fordítási műveletet

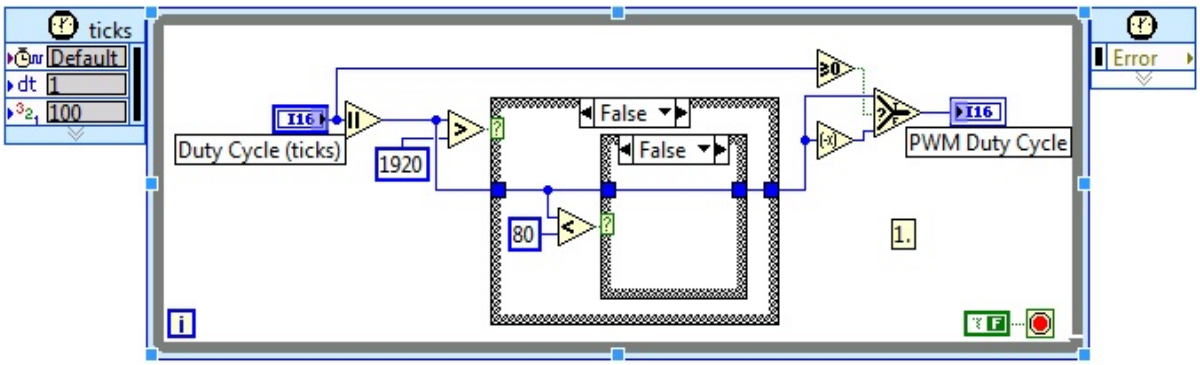
Stop Server: leállítja a szervert, törli az aktiális fordításokat



Labview FPGA Compile Server

Az FPGA.vi három időzített (Timed Loop) és egy while (While Loop) ciklusból áll.

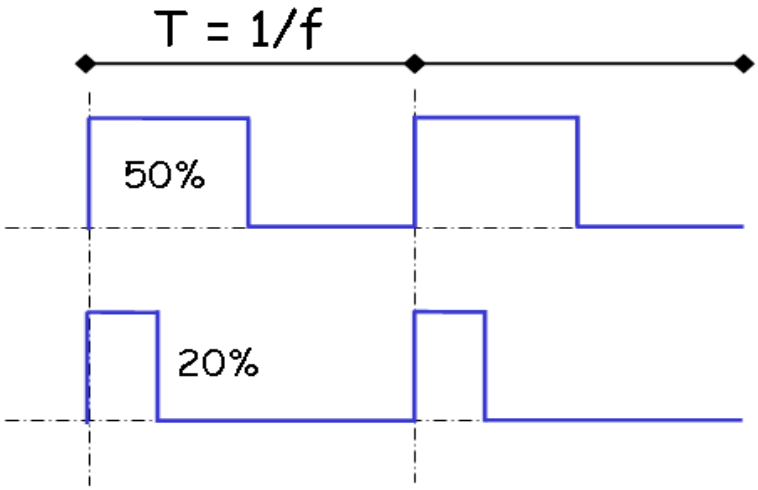
PWM generálás:



Ezt a vezérlőt arra használjuk, hogy generálja a PWM-et (Pulse Width Modulation: Impulzus-szélesség modulált) a motornak. A beállítható sebesség tartománya 4%-96%. Habár, lehet generálni 0%-ot és 100%-ot is. Ha 96% felett van akkor kikényszeríti a 100%-ot, ha 4% alatt akkor kikényszeríti a 0%-ot. A képen látható PWM szabályozást a Labview beépített példái között találtam.

Röviden a PWM-ről:[14]

A vezérlési és szabályozási folyamatok analóg feszültségjelei helyettesíthetők digitális impulzussorozat-jelekkel, amelyek hosszabb időtartamra vonatkoztatott átlagfeszültsége egyenértékű az analóg feszültségjellel.



50 és 20 százalékos kitöltési tényezőjű PWM jel.

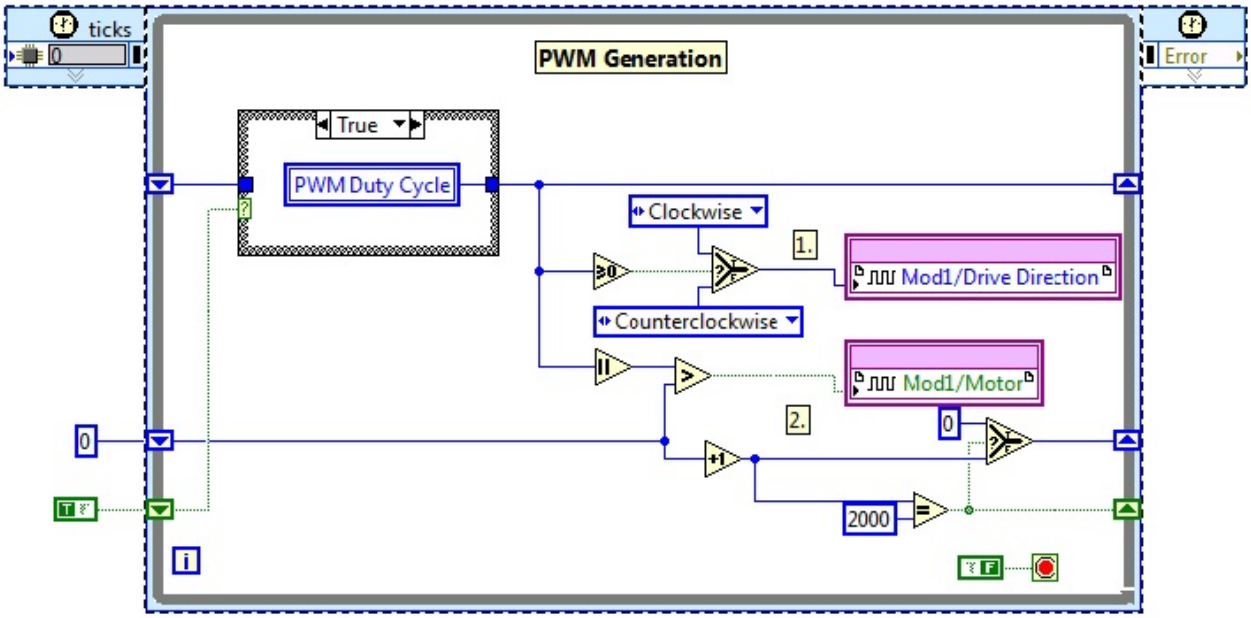
A digitális impulzussorozat frekvenciáját úgy kell megválasztani, hogy az, a vezérelt vagy szabályozott eszköz megfelelő működését biztosítsa, ebben az esetben a motorok ne lökésszerűen változó szögsebességgel forogjanak.

Az ilyen digitális jelek egyik jól használható változatai a PWM jelek, amelyek olyan állandó periódusidejű és frekvenciájú jelek, ahol az átlagfeszültség beállítása a jel kitöltési tényezőjének változtatásával történik.

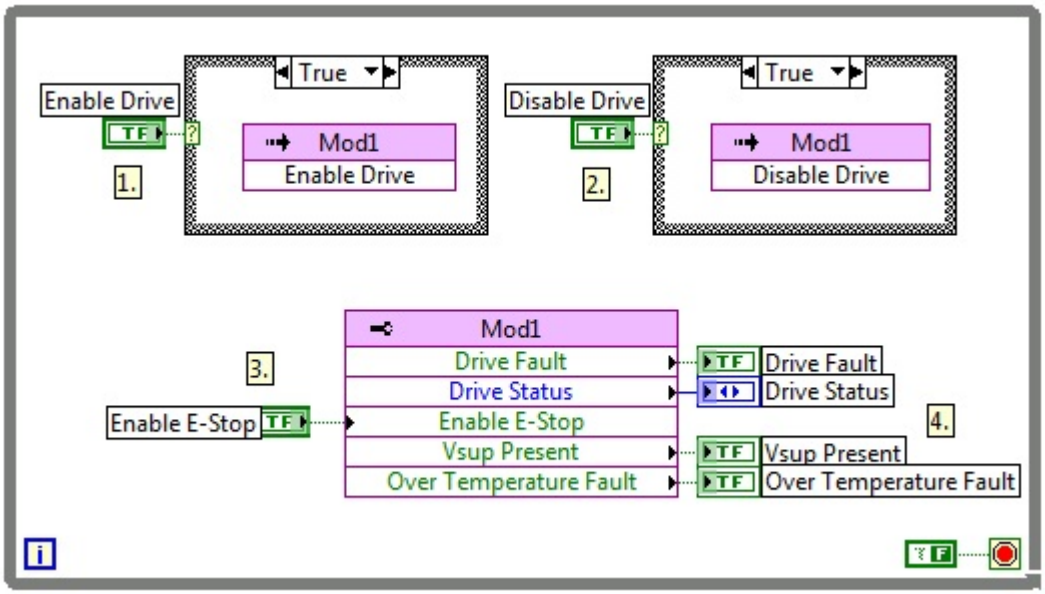
A PWM Duty Cycle előjelbitje adja meg a forgás irányát. Egy pozitív szám az óramutató járásával megegyező, egy negatív szám, az óramutató járásával ellentétes irányba mozgatja a motort.

Egy számláló egy egyszerű-ütemű időzített ciklussal együtt, feltételezve egy 40MHz-es FPGA óraidőt, generál egy 20kHz-es PWM frekvenciát (2000 ütem).

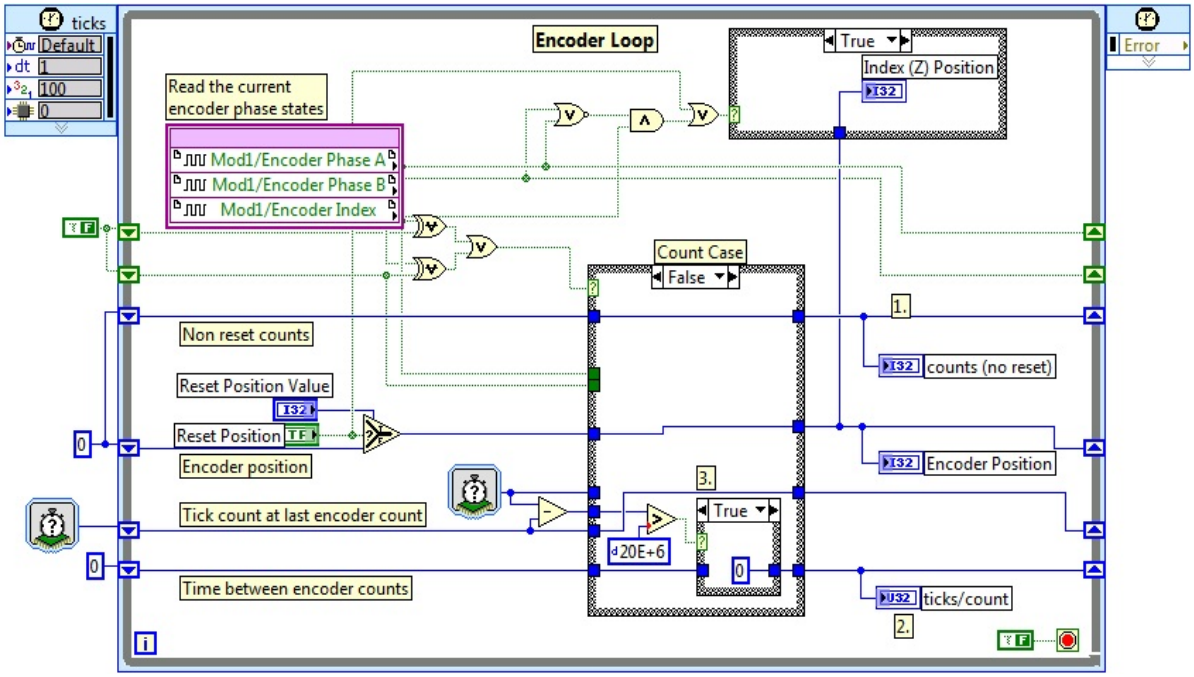
Forrás: [14] http://www.freeweb.hu/t-t/elokep/pic/felhkk/kk/sz1604.htm



PWM generálás



1. A motor engedélyezése egy metódus, amelyet meg kell hívni mielőtt használnánk.
2. A motor lezárása egy metódus, amely meg van hívva, hogy megszüntesse a motor irányítását
3. Az E-Stop engedélyezésével, ha bármilyen hiba lép fel letiltja a motor kezelését, ezzel leállítva a programot, alapértelmezetten nincs engedélyezve.
4. Ezek a jelek alapvető állapotsorok.



1 a „counts (no reset)” -t arra használjuk, hogy megkönnyítsük a sebesség számítását az FPGA-nak.

2 a „ticks/count” arra használjuk, hogy kiszámoljuk a sebességet felhasználva az FPGA rendszeridejének pontosságát.

Mi az encoder?[15]

Egy olyan jeladó, ami egy tengely elfordulását érzékelve az elfordulás szögével arányosan valamilyen elektromos jelet szolgáltat.

Sok egyéb névvel is illetik. Pl.: szög adó, forgás jeladó angular encoder, rotary encoder, stb. Nem keverendő össze azonban a tachométerrel, ami nem az elfordulás szögét, hanem a ez elfordulás sebességét méri.

Mire jó az encoder?

Elmozdulás mérésére, pozíció érzékelésre, pozicionálásra. A zárt hurkú vezérléssel működtetett pozicionáló hajtások visszacsatoló jeleit is szolgáltathatják encoder-ek. CNC technikában nagyon gyakoriak, de PLC-s vezérléseknél is előfordul.

Pozíció érzékelésre, útmérésre használva végálláskapcsolókat, érzékelőket válthat ki. Alkalmazása lehetővé teszi, hogy a megállási vagy műveleti pontok paraméterezhetően módosíthatóak legyenek. Az encoderekkel nagy pontossággal mérhető az elmozdulás.

Az encoderben egy tengely forog, amihez belül egy tárcsa van rögzítve. A tárcsa általában átlátszó üveg, amire a pereme közelében átlátszatlan rovátkák vannak felgőzölve. Esetleg perforált fém tárcsa. A tárcsa rovátkolt peremén optokapu világít át. A tárcsa egyik oldalán fényforrás, a másikon fényérzékeny elem. Miközben a tárcsa forog, a rovátkák az optokapu nyalábját vagy eltakarják, vagy nem. A kapu vevő részében ennek megfelelő elektromos jel jön létre.

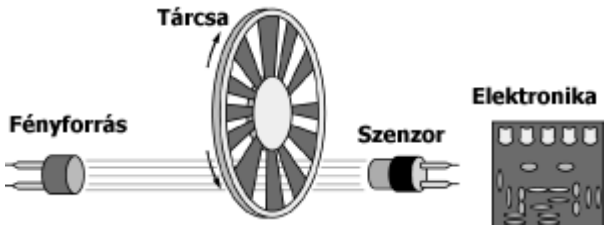
Alapvetően kétféle alaptípust lehet megkülönböztetni: inkrementális és abszolút. Mi Maxon Tacho enc 22 incrementális encodert használunk.

Az inkrementális encoder

Az inkrementális (növekményes) jeladóban olyan tárcsa van, amelyiken egyforma távolságra egyforma méretű rovátkák vannak. A rovátkákat két db optokapu figyeli. A két kapu úgy van elhelyezve, hogy egymáshoz képest 90 fokkal eltolt fázisú jelet adjanak a tárcsa forgásakor. Ez a két jel az "A" és a "B" fázis.

Az encoder tartalmaz egy elektronikát, ami gondoskodik az optokapu fényforrásának táplálásáról és a kapu vevőjéről érkező jelet valamilyen szabványos jellé alakítja. Ez leggyakrabban TTL, nyitott kollektoros, esetleg 24V-os jel, bizonyos esetekben a jelek inverze is ki van vezetve.

Az inkrementális jeladókban van egy harmadik optokapu is, ami a tárcsa egy olyan részén világít át, ahol csak egy rovátka van. Így ez a jel a tengely teljes körülfordulásakor ad egyetlen egy impulzust.

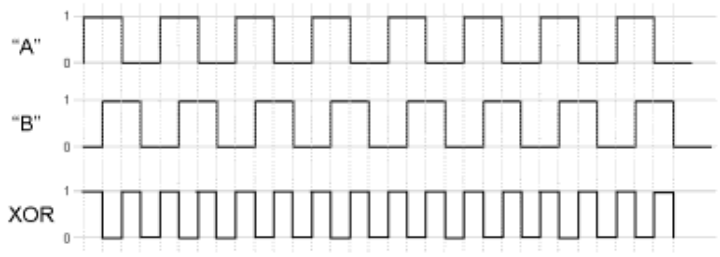


Az "A" és a "B" jel tehát 50% kitöltési tényezőjű négyszög jel, melyek között 90 fokos fáziskülönbség van. A két jelre azért van szükség, mert segítségükkel az encoder tengely forgásának iránya detektálható.

Ezt a fajta jeladót azért hívják inkrementálisnak, mert a tengely elfordulásával arányos jel (impulzus sorozat) a tengely helyzetéhez relatív. Ez azt jelenti, hogy az álló encoder abszolút szöghelyzetéről maga az encoder nem ad információt. A tényleges helyzetet a vezérlésnek kell nyilvántartania, amelyhez az encoder csatlakozik. Ezt az encoder tengelyének relatív elmozdulási távolságából tudja megtenni.

Ezt könnyebb megérteni, ha tudjuk, hogyan kezeli a vezérlés az inkrementális jeladó jeleit. A dolog rendkívül egyszerű. Először is van egy kapukkal felépített logika, ami az "A" és "B" jel fázishelyzetéből megállapítja a forgás irányát. Ezután az "A" vagy a "B" jelet egy le/fel számláló bemenetére vezetik. A számlálás irányát a forgási irányt megállapító logika kimenete vezérli. Ha az encoder tengelye előre forog, akkor az impulzusok a számláló tartalmát növelik, ha hátra, akkor csökkentik.

Valójában a számlálót nem az "A" vagy "B" jel lépteti, hanem beiktatnak egy EXOR kaput, ami összehasonlítja az "A" és "B" jelét.



A és B jel és a kizáró vagy műveletük

Az EXOR kapu kimenetén az "A" vagy "B" jel frekvenciájának duplája jelenik meg, ami egyúttal az encoder tárcsára felvitt rovátkák által meghatározott felbontást megduplázza. A számlálót tehát az EXOR kimenete lépteti le vagy fel.

Bizonyos megoldásoknál még tovább mennek, és a dupla sebességű jel minden élénél (szint átmeneténél) léptetik a számlálót, így négyszeres felbontás is elérhető (quadratic count). Az inkrementális encoder impulzusai egy számláló tartalmát csökkentik, vagy növelik. A számláló tartalma tehát arányos a tengely elfordulásával. A számláló tartalma nincs kapcsolatban az encoder tengelyének abszolút szöghelyzetével, a berendezés áramtalanítása során a számláló értéke elvész, illetve a kikapcsolt berendezés mozgó részeinek elmozdulásáról a vezérlés bekapcsolás után nem "tud".

Forrás: [15] http://szirty.uw.hu/Alapfokon/Encoder/encoder.html

Összefoglalás

A szakdolgozatom témája a Kétdimenziós pozicionálás NI eszközökkel. Szakdolgozatom készítése során a DE TTK Szilárdtest Fizika tanszékén működő optikai vizsgálóállomás két dimenzióban mozgatható mintatartóasztalának pontos (tized-milliméteres) mozgatását kellett megvalósítanom. Ezt NI eszközökkel végeztem el, pontosabban NI c-RIO eszközzel és a motormozgatáshoz kifejlesztett NI 9505-ös modullal.

A kódolást Labview fejlesztői környezetben végeztem. A munkám során kielégítő szinten megtanultam az NI c-RIO eszköz, valamint néhány moduljának programozását.

A diplomamunkám része volt az X-Y asztal építése is. A motorok egy-egy csigaműves tengelyhez csatlakoznak egy-egy kuplung segítségével. Az X irányú mozgatóegység a berendezés állványához van rögzítve, míg az Y irányú csigaműves meghajtóegység az előbbi asztalához van rögzítve. Így valósul meg a két egymásra merőleges irányú mozgás és pozicionálás. A biztonság és a nullhelyzet meghatározása miatt az asztalokat mindkét végükön végállás kapcsolókkal láttuk el. A motorok pontos pozicionálását PID szabályzással oldottam meg.

A munka egyik lehetséges fejlesztési iránya a végálláskapcsolók rendszerbe integrálása. A másik fontos irány a csigaművek lecserélése. Az így összeállított rendszer ugyan alkalmas arra, hogy az asztalra helyezett mintát pontosan pozícionáljuk, de még maximális motorsebességnél is igen lassú (ld. a videót YouTube-on).A csigaműves meghajtást más rendszerre kell cserélni a sebesség növeléséhez.

Irodalomjegyzék

- https://inf-moodle.duf.hu/mod/resource/view.php?id=2927
- https://inf-moodle.duf.hu/mod/resource/view.php?id=2928
- https://inf-moodle.duf.hu/mod/resource/view.php?id=2926
- http://hu.wikipedia.org/wiki/FPGA_%28Field-programmable_gate_array%29
- ftp://ftp.ni.com/pub/devzone/tut/fpga_training_slides.zip
- http://www.nowires.com.br/blogger/uploaded_images/FPGA-708477.jpg
- http://www.ni.com/fpga/rio.htm
- http://www.cegnyilvantarto.hu/cikkek/62/uj-compactrio-rendszer/
- http://sine.ni.com/np/app/main/p/ap/global/lang/en/pg/1/sn/n24:cRIO/fmid/102
- http://sine.ni.com/nips/cds/view/p/lang/en/nid/202711
- http://hu.wikipedia.org/wiki/PID_szab%C3%A1lyoz%C3%B3
- http://www.ms.sapientia.ro/~martonl/Docs/Labs/IRI_L7.pdf
- http://szirty.uw.hu/Alapfokon/Encoder/encoder.html

- Kevizky László, Bars Ruth, Hetthéssy Jenő, Barta András, Bányász Csilla – Szabályozástechnika 2006-os kiadás

Köszönetnyilvánítás

Köszönetet szeretnék mondani elsősorban, témavezetőmnek dr. Cserháti Csaba Tanár Úrnak, aki útmutatásaival és tanácsaival segítette a szakdolgozatom elkészítését.

Köszönet Szabó István tanszékvezető úrnak és a tanszéknek, hogy biztosította számomra a hardvereket és a szoftvereket.

Köszönet a LabView program campus licencéért az NI Hungary kft.-nek.

Köszönet még a tanszék minden dolgozójának, akik különböző módon járultak hozzá a munka folyamatosságához és a végeredmény elkészültéhez.

Plágium - Nyilatkozat

Szakdolgozat készítésére vonatkozó szabályok betartásáról nyilatkozat.

Alulírott ***Jándi Tibor*** (Neptunkód: ***G1JP0R***) jelen nyilatkozat aláírásával kijelentem, hogy a ***Kétdimenziós pozicionálás NI eszközökkel*** című szakdolgozat/diplomamunka

(a továbbiakban: dolgozat) önálló munkám, a dolgozat készítése során betartottam a szerzői jogról szóló 1999. évi LXXVI. tv. szabályait, valamint az egyetem által előírt, a dolgozat készítésére vonatkozó szabályokat, különösen a hivatkozások és idézések tekintetében.

Kijelentem továbbá, hogy a dolgozat készítése során az önálló munka kitétel tekintetében a konzulenst, illetve a feladatot kiadó oktatót nem tévesztettem meg.

Jelen nyilatkozat aláírásával tudomásul veszem, hogy amennyiben bizonyítható, hogy a dolgozatot nem magam készítettem vagy a dolgozattal kapcsolatban szerzői jogsértés ténye merül fel, a Debreceni Egyetem megtagadja a dolgozat befogadását és ellenem fegyelmi eljárást indíthat.

A dolgozat befogadásának megtagadása és a fegyelmi eljárás indítása nem érinti a szerzői jogsértés miatti egyéb (polgári jogi, szabálysértési jogi, büntetőjogi) jogkövetkezményeket.

hallgató

Debrecen, 2010. november 26. Jándi Tibor