ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR
ELEKTRONIKAI ÉS KOMMUNIKÁCIÓS RENDSZEREK INTÉZET
MŰSZERTECHNIKAI ÉS AUTOMATIZÁLÁSI TANSZÉK

SZAKDOLGOZAT

OE-KVK
2022.

Patócs Áron
T006780/FI12904/K

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Műszertechnikai és Automatizálási Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: Patócs Áron
Szakdolgozat száma: SZD21060114218156
Törzskönyvi száma: T006780/FI12904/K **Neptun kódja:** [redacted]

Szak: villamosmérnöki (magyar nyelven) (Budapest)
Specializáció: Műszer-automatika specializáció
Számítógépes folyamatautomatizálás modul
Automatizált gyártórendszerek modul
Elektronikai - Orvostechnikai műszerek és tesztelés modul
Felügyeleti informatika és elektronikus vagyonvédelem modul

A dolgozat címe: Fordulatszám szabályozás szemléltetését szolgáló demonstrációs eszköz fejlesztése
A dolgozat címe angolul: Development of a demonstration tool to exemplify motor speed control
A feladat részletezése: Örvényáramos fékezés a zavaró hatás bevitelére, a jellemzők mérése és a motor szabályozása Siemens PLC-n kerül megvalósításra.

Intézményi konzulens neve: Varga Árpád

A kiadott téma elévülési határideje: 2024. június 30.
Beadási határidő: 2022. 05. 15.

A szakdolgozat: Nem titkos.

Óbudai Egyetem Kandó Kálmán Villamosmérnöki Kar Elektronikai és Kommunikációs Rendszerek Intézet PH

Kiadva: Budapest, 2022. 03. 21.

.....
Intézetigazgató

A dolgozatot beadásra alkalmasnak találom:

Varga Árpád
.....
belső konzulens

.....
külső konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR
ELEKTRONIKAI ÉS KOMMUNIKÁCIÓS RENDSZEREK INTÉZETI
MŰSZERTECHNIKAI ÉS AUTOMATIZÁLÁSI TANSZÉK

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja, a titkosításra vonatkozó esetleges megkötések mellett.

Budapest, 2022. május 14.

Pataics Áron
.....
hallgató aláírása

Tartalomjegyzék

1 Bevezetés

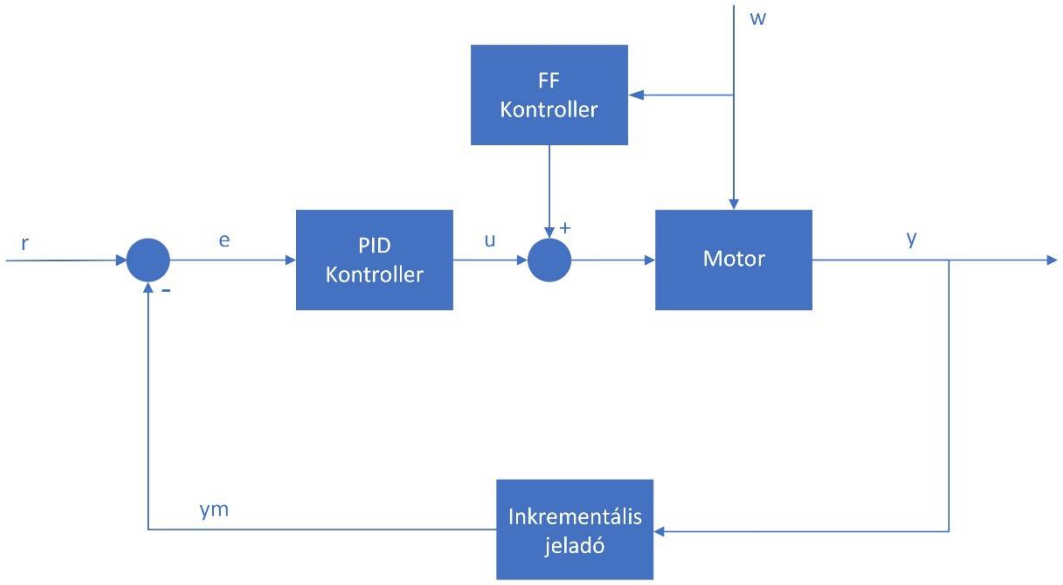
A laborban található eszközök alapján, valamint tanulmányaim során azt a következtetést vontam le, hogy nem igazán kerül szóba a PLC gyártók által kínált automatikus paraméterezésű intelligens PID blokkok használata. Az okok érthetőek, ugyanis használatuk egyszerűbben megtanulható a szabályozáselmélet tudományának elsajátításával szemben. Ennek ellenére álláspontom szerint, nem válik az érdeklődő hallgatók hátrányára, ha betekintést nyernek ezen szoftvereszközök használatába, a Siemens PID blokkján keresztül.

Célom, hogy egy olyan demonstrációs eszközt készítsek, ami kellően szemléletes és nem túlságosan összetett, csak annyira, hogy a gyártó szabályozójának minden lényeges funkciója bemutatható és kipróbálható legyen. A laborban rendelkezésre álló eszközök alapján a fordulatszám szabályozás témája megfelelőnek bizonyult ennek szemléltetésére. Az erre szolgáló próbapad összeállítás, egy DC motorból, inkrementális jeladóból és fékerő hatásának bevitelére szolgáló eszközből tevődik össze. A dolgozat váza alapvetően e szabályozás fejlesztésének menete köré épül.

Bízom benne, hogy beszámolóm segítséggel fog szolgálni az automatikus hangolású PID szabályozási megoldások iránt érdeklődőknek, de mindemellett nem szeretném azt a benyomást kelteni az olvasóban, hogy ezekkel az eszközökkel kiváltható lenne a szabályozók tervezésének ismerete minden folyamatautomatizálási eszköz megvalósítása esetén.

2 A szabályozandó szakasz fizikai felépítése

2.1 Zárt szabályozási kör és zavaró hatás felvázolása



1. ábra – Szabályozási kör blokkdiagrammja

Jelölések:
- r: referencia jel
- e: hibajel
- u: beavatkozó jel
- y: kimenet
- ym: visszacsatolt jel
- w: zavarjel

A blokkvázlaton [1. ábra] szemléltetett szabályozó, szabályozott szakasz és zavartényező fizikai megfelelői a PLC, mint PID és Feed Forward kontroller (FF kontroller), a serleges DC motor, egy induktív közelség érzékelő és 3 db, a motor tengelyére rögzített alumínium tárcsán elhelyezkedő fém henger, mint inkrementális jeladó és ugyanerre a tárcsára ható örvényáramú fékezés, mint zavaró jellemző. Ezeket az eszközöket fogom most részletesebben ismertetni.

2.2 Szabályozási kört megvalósító eszközök

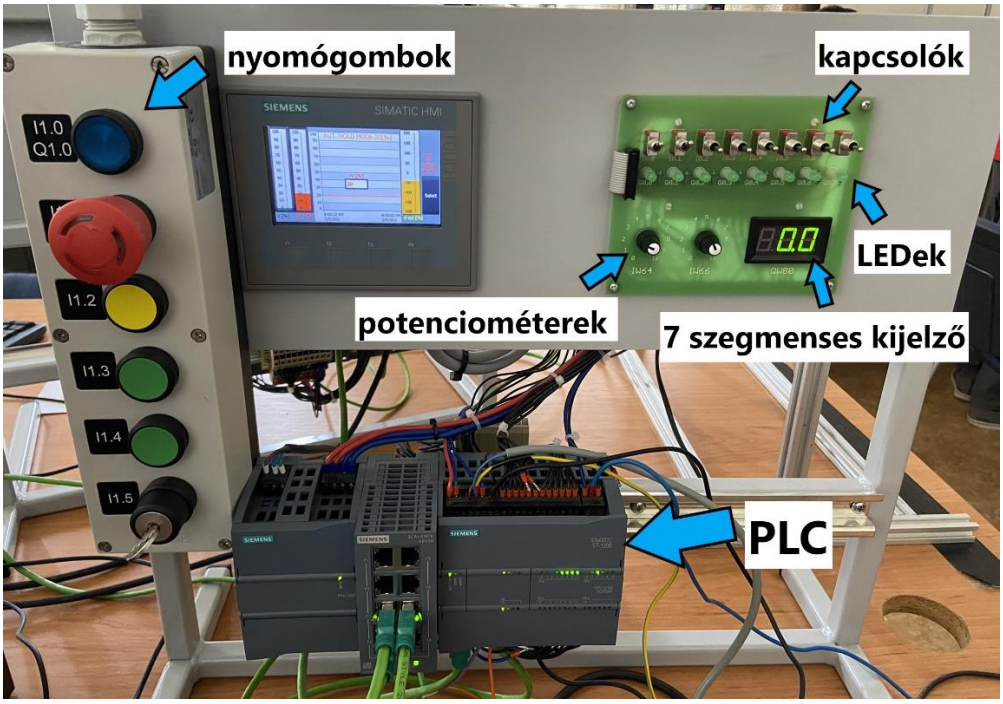
2.2.1 PLC

Az általam használt PLC a Siemens SIMATIC S7-1200 típusú modell, pontosabban a CPU 1214C DC/DC/DC (Táp/Digital In/Digital Out).
Gyári elnevezése a modellnek: 6ES7214-1AG40-0XB0
Felhasználás szempontjából a legfontosabb jellemzőit kigyűjtöttem az eszköz gyártói adatlapjáról.[1]

Tápfeszültség	24 V (20,4-28,8 V)
Maximális áramfelvétel	1,5 A
Digitális bemenetek száma	14 db (ebből 6 db High Speed Counter)
Digitális kimenetek száma	10 db (ebből 4 db 100kHz PWM[1])
Analóg bemenetek	2 db (10 bit felbontás, ±10 V)
Analóg kimenetek	0 db
Kommunikációs port	RJ 45 (Ethernet)
Támogatott Ethernet protokollok	TCP/IP, SNMP, DCP, LLDP
Támogatott irányítástechnikai protokollok	PROFINET I/O, PROFIBUS, OPC UA, AS-Interface
Kiegészítők maximális száma	3 db kommunikációs modul, 1 db bővítő kártya, 8db bővítő modul

A PLC egy 6ES7 232-4HA30-0XB0 típusú bővítőkártyával van felszerelve, ami egy ±10 V-os, 0-20 mA-es analóg kimenetet biztosít 12 bites felbontással.

[1] Pulse Width Modulation

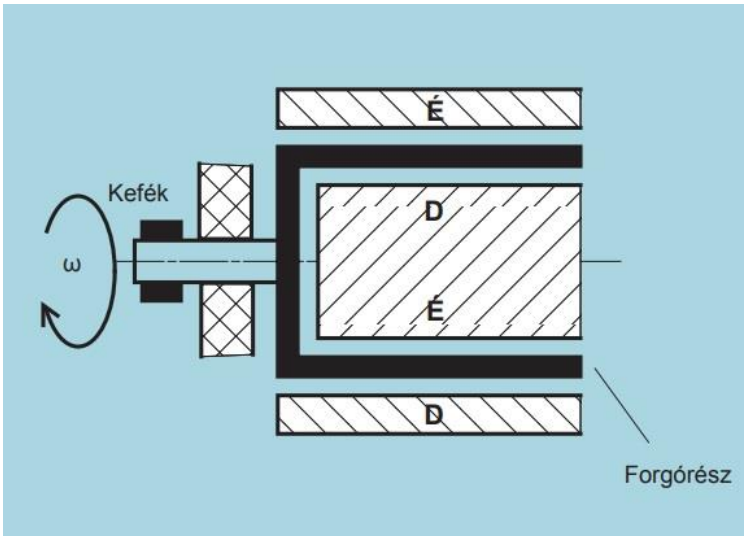


2. ábra – PLC

A képen látható PLC konfiguráció [2. ábra] I/O eszközeiből (kapcsolók, potenciométerek, nyomógombok, 7 szegmenses kijelző), csak egy darab potenciométert használtam fel, ezért a későbbiekben a nem használt I/O eszközökről nem fogok említést tenni. A PLC még két modullal van kiegészítve (az analóg kimeneti bővítőkártya mellett). Egy Siemens SCALANCE XB008 típusú Ethernet switch, és egy Siemens SIMATIC PM 1207 típusú tápegység látható még a processzormodul bal oldalára rögzítve. Ez a tápegység látja el az állványra felszerelt összes elektronikai eszközt 24 V-os egyenfeszültséggel.

2.2.2 Motor

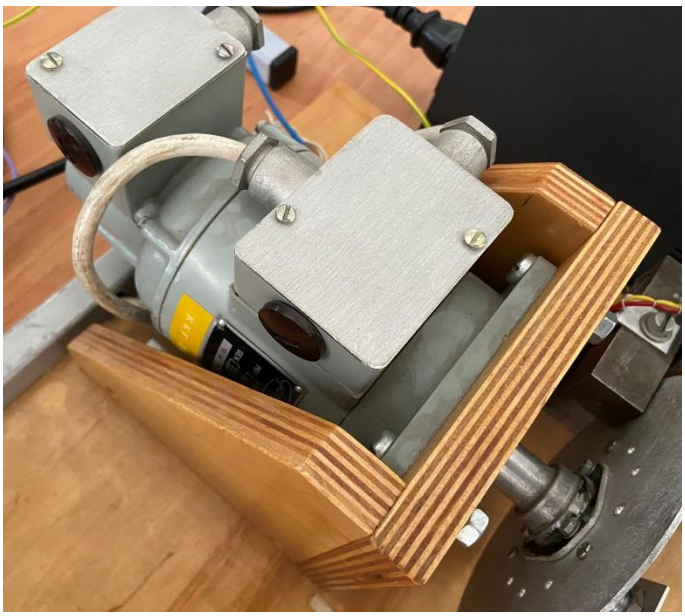
Az irányítandó beavatkozószerv egy GAMMA MST-464-10 típusú serleges egyenáramú szervomotor. A serleges motorok „elsősorban szervomotorként használatosak, mivel hatásfokuk kedvezőtlen, azonban a forgórész tehetetlenségi nyomatéka alacsony, mert a forgórész nem tartalmaz vasat. A serleges forgórész a kalickás forgórész speciális esetének tekinthető, amikor a kalicka egy teljes hengerfelület, amely az állórészen kialakított légrésben forog.” [2]



3. ábra – Serleges szervomotor felépítése [3]

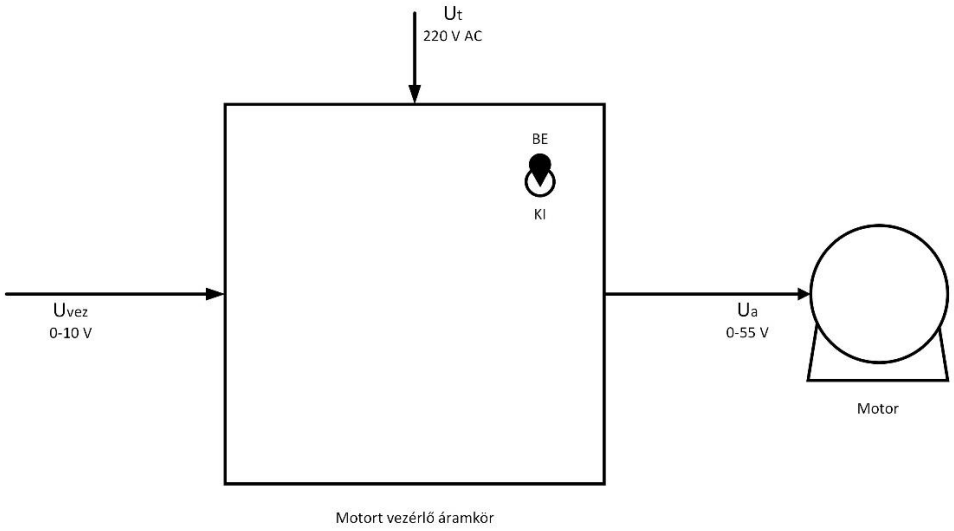
A motornak a forgórésze kefés kommutátoros, melyen maximum 420 mA armatúra áram folyik 55 V-os kapocsfeszültség mellett. Az állórész 220 V-os gerjesztést igényel, melynek áramfelvétele 160 mA. A motor maximális fordulatszámát a gyártó 1500 fordulat/perc-ként határozta meg.[4]

A motor ugyan tartalmaz egy tachométergenerátort is, melynek pontossága megfelelő lenne fordulatszámmérésre, de én mégis külső eszközt használtam erre a célra annak érdekében, hogy minél több tapasztaltot szerezhessek az inkrementális jeladók működése terén.



4. ábra – Gamma MST-464-10 típusú motor

A motort vezérlő áramkör megvalósítása számomra ismeretlen, ezért csak a be- és kimeneteivel foglalkoztam. A motor vezérlését 0-10 V közötti feszültséggel (U_{vez}) tudjuk elvégezni, míg a kapcsolás kimenetén ezzel arányos, a motornak megfelelő armatúrafeszültség (U_a) fog megjelenni. Az áramkör egy fémtokba lett helyezve, aminek elején 2 db kétállású kapcsoló és egy forgókapcsoló van. Ezek közül csak az egyik kétállású kapcsoló használatára van szükség, amivel tápfeszültség adható az áramkörnek.



5. ábra – Motort vezérlő áramkör feketedoboz modellje

2.2.3 Inkrementális jeladó

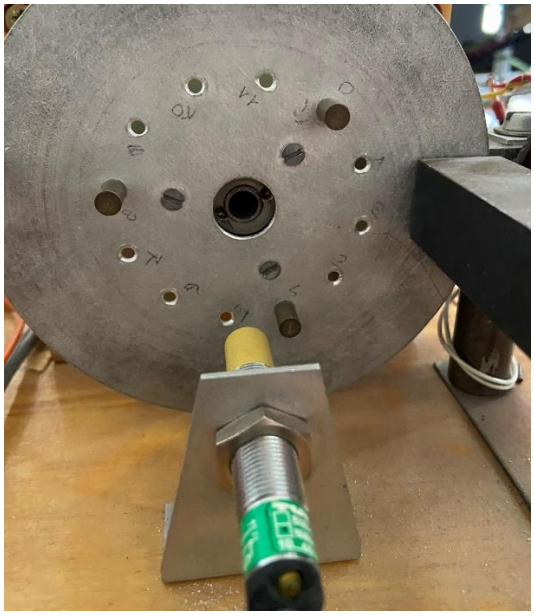
Az eszköz egy induktív közelség érzékelőből és 3 db a motor tengelyére rögzített tárcsával koncentrikus kör kerülete mentén egymástól egységnyi távolságra lévő fém hengerből áll. A jeladó TURCK Ni8U-M12-AP4X típusú hengeres kialakítású induktív szenzor. A termékcsalád gyártói adatlapja szerint és a cikkszáma alapján a következő termékjellemzők állapíthatók meg:

N - nem beágyazható
i - induktív
8 - 8 mm-es érzékelési távolság
U - Uprox® szenzor (érzékelési karakterisztikára utal)
M - tokozása részben menetes, krómmal galvanizált sárgaréz
12 - tokozása 12 mm átmérőjű
A - záró érintkező (Normally Open)

P - PNP tranzisztoros

4 - 10-65 V_{DC} polaritás-, rövidzár- és túláram védelemmel

X - 1 db beépített visszajelző LED



6. ábra – Inkrementális enkóder

2.3 Zavarjel előállítását szolgáló eszközök

2.3.1 Örvényáramú fékezés

A fék megvalósítása egy téglalap kialakítású vasmag [7. ábra, 1], melynek egyik oldalán egy tekercs [7. ábra, 3] helyezkedik el, azzal ellentétes oldalán pedig légrés található. Ebbe a légrésbe van élével belehelyezve a tárcsa [7. ábra, 2].

7. ábra – Örvényáramú fék koncepció [5] (bal) és valós eszköz (jobb)

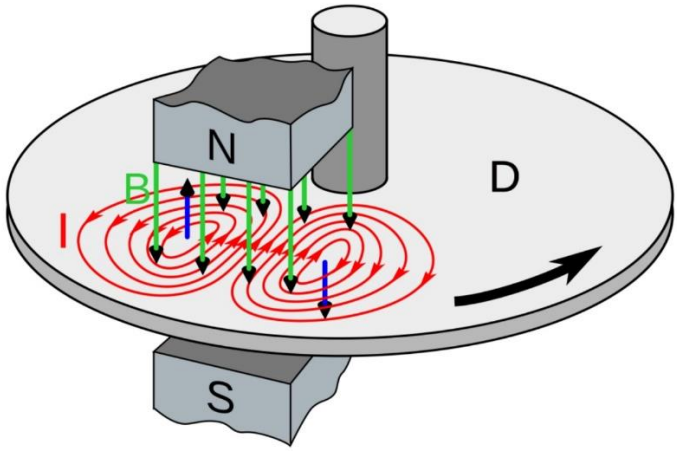
A tekercsre feszültséget kapcsolva a vasmagon áthaladó mágneses fluxus hatására a légrésben elektromágneses tér alakul ki.

$$rot\ E = -\frac{\partial B}{\partial t}$$

Ez az elektromágneses tér a benne nagy sebességgel áthaladó alumínium tárcsa egy adott pontjában folyamatosan változó feszültséget fog indukálni.

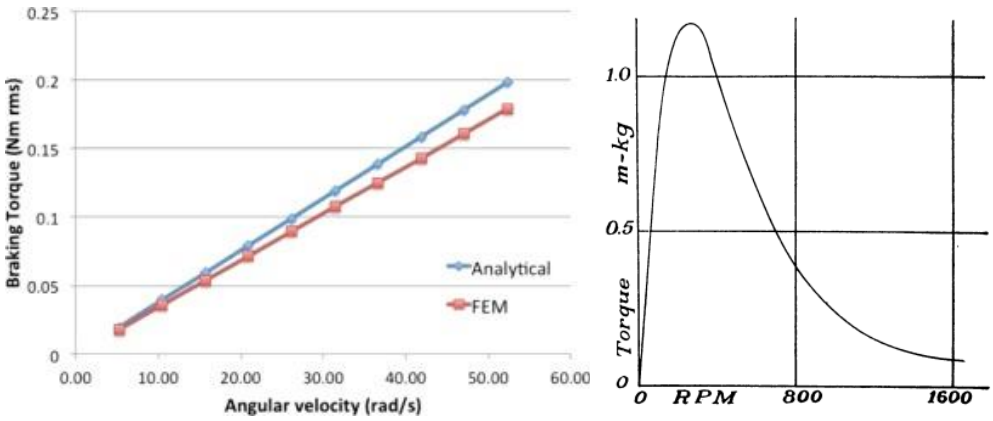
$$U_i = B * l * v$$

Ennek a váltakozó feszültségnek a hatására a fognak megindulni a tárcsában az örvényáramok [8. ábra]. Ezek az áramok saját mágneses teret alkotnak, melyek fékező hatást gyakorolnak a tárcsára a vasmag által létrehozott elektromágneses térrel való kölcsönhatásuk következtében.



8. ábra – Örvényáramok és mágneses erővonalai [6]

Az örvényáramú féknek számos előnye van a mechanikus elven működő társaival szemben. Mivel működése nem igényel fizikai kontaktust éppen ezért nincs kopó alkatrész és halk. Viszont hátrányai is vannak, mégpedig az áramok következtében keletkező hő és a fékerő nyomatékának fordulatszámfüggése. Bizonyos fordulatszámértékig ez a nyomaték lineárisan változik annak növelésével.



9. ábra – fékerő nyomatékának változása a fordulatszám függvényében

A [9. ábra] bal oldali koordinátarendszerének FEM (Finite Element Model) függvényén látszik, hogy kb. 55 rad/s szögsebességig (kb. 525 fordulat/perc) végezték el a modellezést [7]. A jobb oldali grafikonon [8] viszont jól látszódik, hogy egy bizonyos fordulatszám fölött ez a nyomaték jelentősen elkezd csökkenni. Ebből azt a következtetést vontam le, hogy a zavarás mértéke nem lesz lineáris a fordulatszám függvényében.

2.3.2 Labortápegység

Az örvényáramú fék meghajtására egy AXIO AX-3010DS típusú tápegységet használok. Fontosabb jellemzői: egycsatornás, kapcsolóüzemű, az áramkorlát 0-10 A között, a feszültségszabályozás 0-30 V között állítható. Kimeneti feszültségszint tartási képessége (feszültségforrás üzem) $U_{hiba} \leq 1\% +10$ mV és a kimenő áram stabilitása $I_{hiba} \leq 1\% +5$ mA (áramgenerátoros üzem)

2.3.3 HMI (Human-Machine Interface)

Egy Siemens SIMATIC KTP400 Basic típusú, 6AV2 123-2DB03-0AX0 gyártói cikkszámmal ellátott HMI-t használtam fel a PLC program vezérlésére és a mért adatok vizualizálására. Legfontosabb jellemzői [9]:

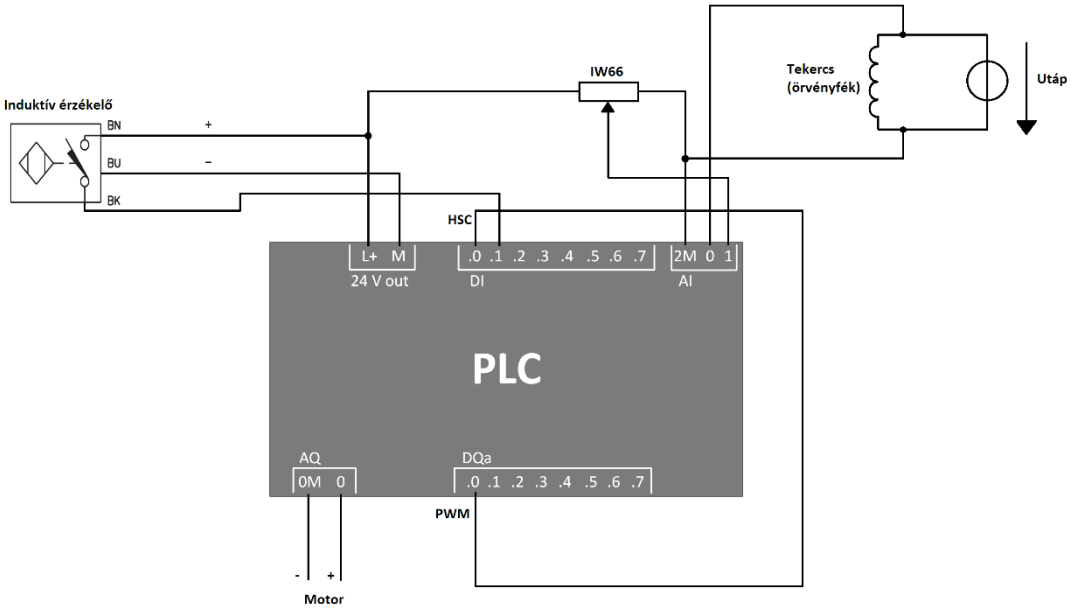
Kijelző típusa	4,3” TFT, érintésérzékeny kijelző
Kijelző felbontása	480x272
Funkcióbillentyűk száma	4 db
Tápfeszültség igény	24 V (19,2 – 28.8 V) DC

Interfész	1 db USB, 1 db Ethernet
Protokollok (Ethernet)	TCP/IP, DHCP, SNMP, DCP, LLDP
Protokollok	PROFINET, EtherNet/IP
Belső memória mérete	10 MB

A HMI Ethernet porton keresztül csatlakozik az Ethernet switch-hez, amihez a PLC és a programozására használt PC is kapcsolódik.

2.4 Kapcsolási rajz

A kapcsolási rajzon csak a projekt szempontjából releváns eszközöket tüntetem fel.



10. ábra – PLC analóg és digitális ki-/bemeneteinek kapcsolási rajza

A DQa.0 kimenet 10 kHz-es, 50% kitöltési tényezőjű PWM jelet generál, amit a DI.0 bemeneten High Speed Counterrel olvasok vissza. Ennek okát a PLC szoftver külső megszakításának (3.3. fejezet) tárgyalásánál fogom részletesebben kifejteni.

3 PLC program bemutatása

Siemens PLC-k esetében az OB1-be elhelyezett kódrészek futnak le minden ciklusban. Ennek futását megszakíthatják az interruptok. Két olyan interruptot is használok a programmegvalósításomban, amik rendszeres időnkénti, illetve esemény vezérelt megszakítást iktatnak közbe az OB1 futásába. Ezek a Cyclic Interrupt [OB30] és a Hardware Interrupt [OB40].

3.1 Globális változók

Name	Path	Data Type	Logical Address	Comment	Hmi Visible	Hmi Accessible
HSC_measure	Global_external	Bool	%I0.0		True	True
inductive_switch	Global_external	Bool	%I0.1		True	True
motor_control_an	Global_external	Word	%QW80		True	True
HSC_Value	Global_external	DWord	%ID1000		True	True
AN_disturbance_meas	Global_external	Word	%IW64		True	True
AN_pot2	Global_external	Word	%IW66		True	True
PWM_OutForMeasurement	Global_external	Bool	%Q0.0		True	True
r_reference	Global_internal	Real	%MD12		True	True
u_actuating	Global_internal	Real	%MD16		True	True
ym_feedback	Global_internal	Real	%MD20		True	True
motor_RPM	Global_internal	DWord	%MD8		True	True
HSC_interrupt	Global_internal	DWord	%MD4		True	True
w_disturbance	Global_internal	Real	%MD34		True	True
u_manual	Global_internal	Real	%MD42		True	True
PID_ModeActivate	Global_internal	Bool	%M24.6		True	True
r_prog_n+1	Global_internal	Real	%MD50		True	True
FFc_out	Global_internal	Real	%MD54		True	True
StartRecording_Button	HMI_variables	Bool	%M24.2		True	True
StopRecording_Button	HMI_variables	Bool	%M24.3		True	True
Programmed_mode_Button	HMI_variables	Bool	%M24.4		True	True
RecordingFinished_signal	HMI_variables	Bool	%M24.5		True	True
PID_Mode	HMI_variables	Int	%MW25		True	True
PID_Error	HMI_variables	Bool	%M28.0		True	True
SamplingScaler	HMI_variables	Int	%MW29		True	True
FeedForward_Switch	HMI_variables	Bool	%M24.7		True	True
PID_ErrorAck_Button	HMI_variables	Bool	%M27.0		True	True
PID_Reset_Button	HMI_variables	Bool	%M27.1		True	True
PID_ErrorBits	HMI_variables	DWord	%MD38		True	True
SamplingTime	HMI_variables	Int	%MW46		True	True
RecordInProgress	HMI_variables	Bool	%M27.2		True	True
PID_State	HMI_variables	Int	%MW48		True	True
recordfin_mem	Bit_memory_vars	Bool	%M27.5		True	True
inSw_mem	Bit_memory_vars	Bool	%M24.1		True	True

1. táblázat – A PLC program globális változói

A „Path” oszlopban látható, hogy melyik könyvtár eleme az adott változó. A könnyebb eligazodás érdekében osztottam fel több részre őket. A *Global_internal*-ban találhatók a

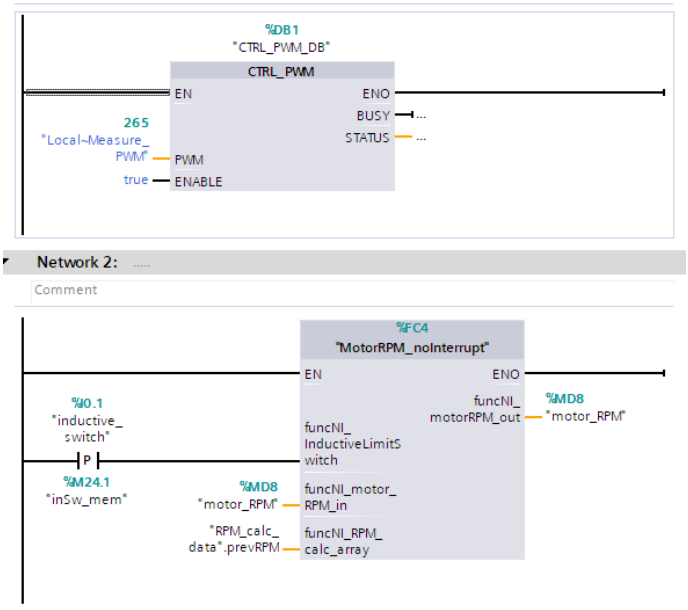
fizikai ki- és bemenetektől közvetlen formában függetlenek, a *Global_external*-ba a fizikai ki- és bemenetek. A *HMI_variables* tartalmazza a HMI-n felhasznált változókat és a *Bit_memory_vars* a felfutó, vagy lefutó él detektálására szolgáló előző ciklus értékét tároló változókat.

DB Name	Variable Name	Data type
r_Programmed [DB13]	r_Data	Array [1..2000] of Real
RPM_calc_data [DB4]	prevRPM	Array [0..9] of Dint
r_Saved [DB10]	r_Data	Array [1..2000] of Real
u_Saved [DB12]	u_Data	Array [1..2000] of Real
w_Saved [DB9]	w_Data	Array [1..2000] of Real
ym_Saved [DB8]	ym_Data	Array [1..2000] of Real

2. táblázat – Adat blokkok

Az adatblokkokban tárolom az adatrögzítések eredményeit, mérési előzményeket és a referencia jelet szolgáltató tömböt.

3.2 Fő program/Main [OB1]



11. ábra – Main [OB1]

A fő program megírása LAD programnyelven történt, összesen két ágat tartalmaz. Az elsőben a PWM kimenet engedélyezése látható, aminek konfigurálását TIA Portalban (Siemens S7-1200 programozói felülete) a *PLC→Properties* menüben végeztem el. PTO1/PWM1 konfiguráció:

Signal type: PWM

Cycle time: 100 µs

Initial pulse duration: 50 Hundredths

Pulse output: %Q0.0

A második ágban egy funkciókód van, ami azért felelős, hogy lenullázza a mért fordulatszámot, mert a motor forgásának leállása esetén, újabb impulzus hiányában nem kerül újbóli meghívásra a Hardware Interrupt-ban elhelyezett fordulatszám számításáért felelős függvény és az utoljára mért érték bennmarad.

3.2.1 MorotRPM_noInterrupt funkciókód

Name	Data type	Default value
Input		
funcNI_InductiveLimitSwitch	Bool	
funcNI_motor_RPM_in	DWord	
Output		
funcNI_motorRPM_out	DWord	
InOut		
funcNI_RPM_calc_array	Array[0..9] of DInt	
Temp		
funcNI_elapsedTime	Time	
funcNI_ii	Int	
Constant		
<Add new>		
Return		
MotorRPM_noInterrupt	Void	

12. ábra – programrész változói

```
1  "IEC_Timer_0_DB".TOF(IN:=#funcNI_InductiveLimitSwitch,
2                       PT:=t#21s,
3                       ET=>#funcNI_elapsedTime);
4  IF #funcNI_motor_RPM_in > 0 THEN
5      IF #funcNI_elapsedTime > t#20s THEN
6          #funcNI_motorRPM_out := 0;
7      ELSIF #funcNI_elapsedTime > t#10s THEN
8          #funcNI_motorRPM_out := 1;
9      ELSIF #funcNI_elapsedTime > t#8s THEN
10         #funcNI_motorRPM_out := 2;
11     ELSIF #funcNI_elapsedTime > t#5s THEN
12         #funcNI_motorRPM_out := 3;
13     ELSIF #funcNI_elapsedTime > t#4s THEN
14         #funcNI_motorRPM_out := 4;
15     ELSIF #funcNI_elapsedTime > t#3s THEN
16         #funcNI_motorRPM_out := 6;
17     ELSIF #funcNI_elapsedTime > t#2s THEN
18         #funcNI_motorRPM_out := 10;
19         FOR #funcNI_ii := 0 TO 9 DO
20             #funcNI_RPM_calc_array[#funcNI_ii] := 0;
21         END_FOR;
22     ELSE
23         #funcNI_motorRPM_out := #funcNI_motor_RPM_in;
24     END_IF;
25 ELSE
26     #funcNI_motorRPM_out := #funcNI_motor_RPM_in;
27 END_IF;
```

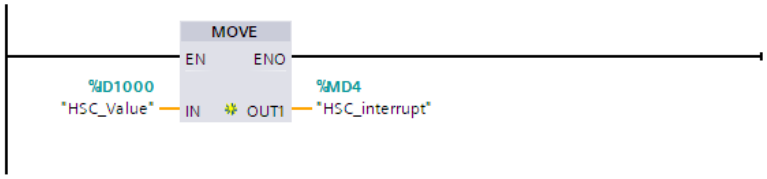
13. ábra – program megvalósítás SCL nyelven

Működésének lényege, hogy a *funcNI_InductiveLimitSwitch* változónak 1→0 átmenetére elindul az 1. sorban lévő időzítő, ami 21 másodpercig számlál és pillanatnyi értékét átadja a *funcNI_elapsedTime* változónak. A 4. sorban lévő feltételre azért van szükség, hogy ne

változtassa meg a fordulatszám értékét, akkor, amikor a motor még el sem indult. A 4-18. sorig az eltelt idő alapján számított fordulatszám értékek kerülnek a *funcNI_motorRPM_out* változóba. A 19-21. sorokban lévő FOR ciklus azért felelős, hogy az első ilyen programozott értékadásnál nullázza ki az RPM_calc funkciókódban (3.3.1. fejezet) számításra használt tömb elemeit, mert a motor újbóli beindításánál a leállást megelőző fordulatszám értékekkel átlagolná az újraindítást követő első 9 mérést és ez hibás eredményre vezetne.

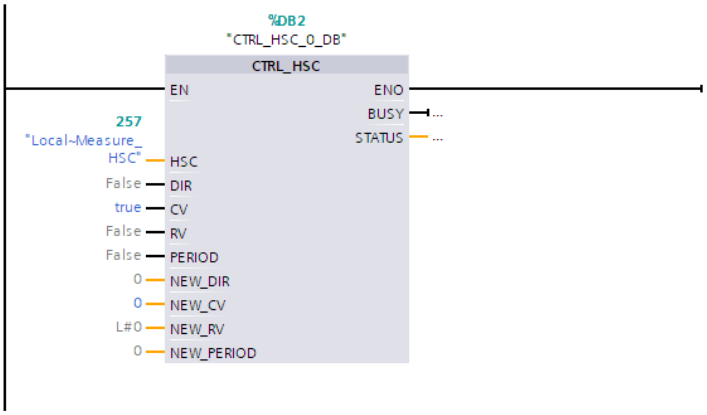
3.3 Külső megszakítás/HW Interrupt [OB40]

A külső megszakításban elhelyezett program akkor kerül futtatásra, amikor az induktív érzékelő kimenete és így a PLC *inductive_switch* változója 0-ról 1 értékre vált. A megszakításnak 3 ága van.



14. ábra – HW Interrupt 1. ága

A HSC-ben (High Speed Counterben) tárolt érték kerül átadásra a *HSC_interrupt* változóba, annak későbbi felhasználásra a megszakításban.



15. ábra – HW Interrupt 2. ága

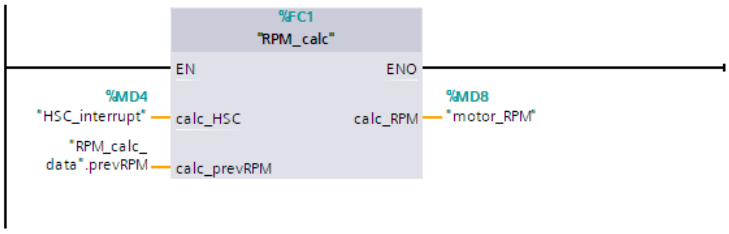
A HSC érték nullázása történik. A HSC konfigurálását a TIA Portal, PLC→Properties menüjében végeztem el. HSC1 konfiguráció:

Type of counting: *Count*

Operating phase: *Single phase*

Counting direction: *Count up*

Clock generator input: *%I0.0*



16. ábra – HW Interrupt 3. ága

Meghívásra kerül az RPM_calc funkciókód, aminek bemenete a HSC számlálójának megszakításkori értéke és az *RPM_calc_data.prevRPM* tömb, amiben az előző 10 ilyen megszakításkor számított fordulatszámérték van lementve. Kimenetén a szabályozónak átadott visszacsatolt jellemző még dimenzióval rendelkező (1/perc) értéke van átadva a *motor_RPM* változónak.

3.3.1 RPM_calc funkciókód

	Name	Data type	Default value	Comment
	Input			
	calc_HSC	DWord		1 unit = 0.1ms (10kHz PWM)
	Output			
	calc_RPM	DInt		
	InOut			
	calc_prevRPM	Array[0..9] of DInt		
	Temp			
	momentary_RPM	DInt		
	ii	Int		
	RPM_sum	DInt		
	notNull	Int		
	Constant			
	<Add new>			
	Return			
	RPM_calc	Void		

17. ábra – RPM_calc változói

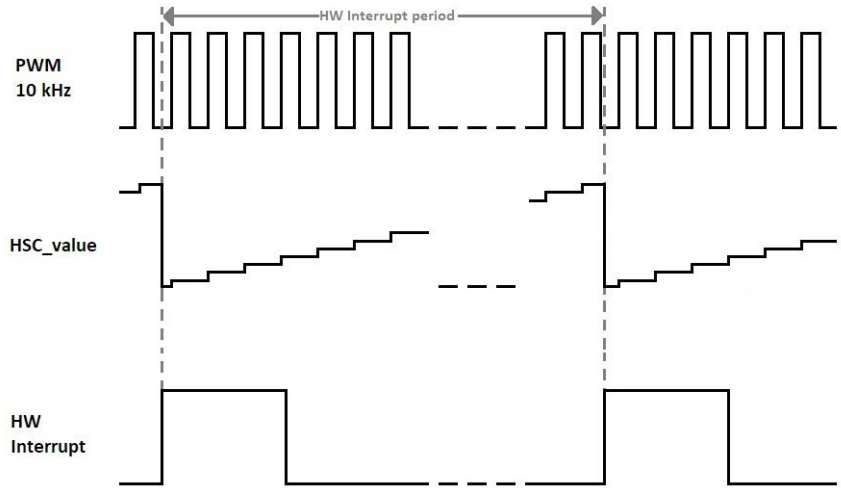
```
#momentary_RPM := 200000 / DWORD_TO_DINT(#calc_HSC);    //60 sec = 600000 * 0.1 ms, 3 PPR encoder -> 600000 / 3 = 200000

FOR #ii := 0 TO 9 DO                                    //FIFO
    IF #ii = 9 THEN
        #calc_prevRPM[#ii] := #momentary_RPM;
    ELSE
        #calc_prevRPM[#ii] := #calc_prevRPM[#ii + 1];
    END_IF;
END_FOR;

FOR #ii := 0 TO 9 DO                                    //drop out zero values from avarageing
    IF #calc_prevRPM[#ii] <> 0 THEN
        #notNull := #notNull + 1;
        #RPM_sum := #RPM_sum + #calc_prevRPM[#ii];
    END_IF;
END_FOR;

#calc_RPM := #RPM_sum / #notNull;
```

18. ábra – Fordulatszám számítást végző kód

A kód első sorában végzem el a fordulatszám számítását az alábbi egyenlet alapján:

$$RPM = \frac{10\ kHz}{PPR * calc\_HSC} * 600000$$

Ahol a 10 kHz a PWM jel frekvenciája, a PPR (Pulse Per Revolution) az „inkrementális enkóder" felbontása (egy körbefordulás alatt hány impulzus), a *calc_HSC* a HSC által megszámlált felfutó élek száma a PWM jelen és végül a 600.000-el való szorzás az 1/100µs mértékegységről 1/perc mértékegységre átszámítása.



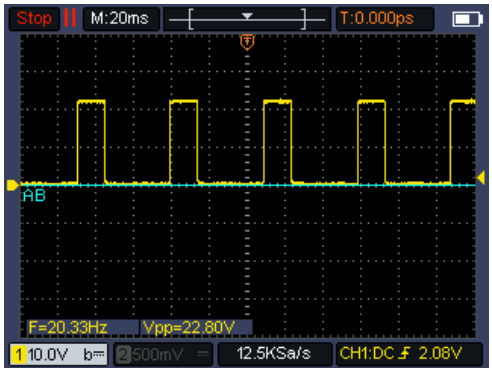
19. ábra – mérési elvet megvalósító jelek

Tehát a 19. ábra alapján a PWM jel felfutó éleit megszámlálja a High Speed Counter és letárolja a *HSC_Value* változóban, amit külső megszakításra másolok és törlöm ezt az értéket.

A rutin 3. sorában lévő FOR ciklus az előző 9 mérési eredményt mozgatja a *calc_prevRPM* tömbben egy helyjel alacsonyabb pozícióba, míg a 0. elem törlésre kerül. A tömb 9. eleme lesz a pillanatnyi mért érték.

A 11. sorban lefutó FOR ciklus azt számolja össze, hogy a tömb mely elemei nem nullák és számukat menti a *notNull* változóba. Emellett a nem nulla mérési értékeket összegzi az *RPM_sum* változóba. A mindenkori legfrissebb 10 mért érték folytonos tárolásával és a két változó osztásával ($\frac{RPM\_sum}{notNull}$) egy 10-es mozgó átlagolást végeztem el.

A mérés helyességének hitelesítése érdekében megmértem oszcilloszkóppal [10] az induktív jeladó kimenetén lévő jelet egy alacsony és egy magasabb fordulatszámon is.

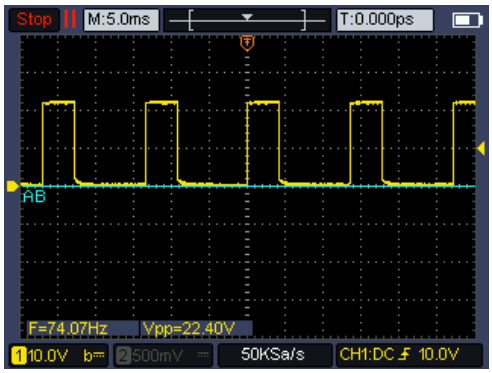


20. ábra – mérés alacsony fordulatszámon

A műszer kijelzőjén látható, hogy a mért jel 20,33 Hz frekvenciájú, itt a PLC-re feltöltött rutinom 396 fordulat/percet jelzett vissza.

$$\frac{f_{mért}}{PPR} * 60\,\text{sec} = \frac{20{,}33\ Hz}{3} * 60\,\text{sec} = 406{,}6\ \frac{ford.}{perc}$$

Az eltérés kb. 11 fordulat percenként, ami -2,7%-os relatív hibát jelent.



21. ábra – mérés magas fordulatszámon

A műszerről 74,07 Hz olvasható le, itt a rutin 1532 fordulat/perc értéket jelezte ki.

$$\frac{f_{mért}}{PPR} * 60\,\text{sec} = \frac{74{,}07\ Hz}{3} * 60\,\text{sec} = 1481{,}4\ \frac{ford.}{perc}$$

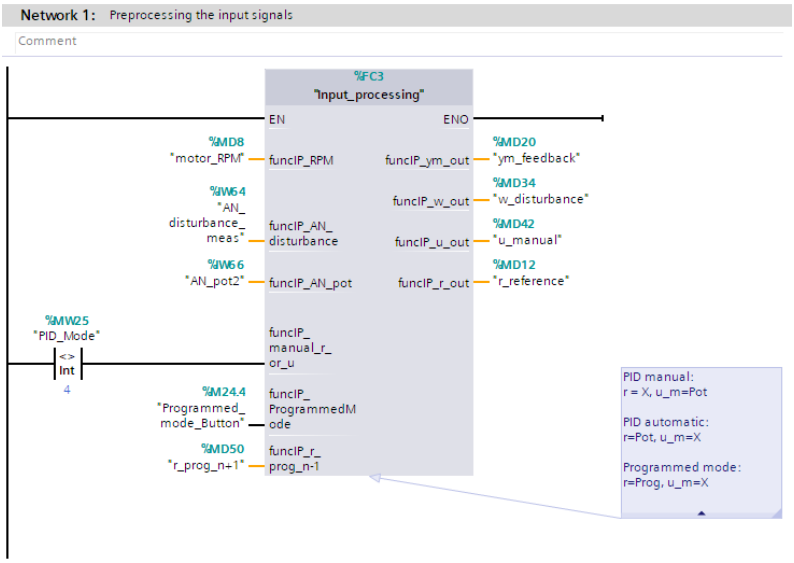
Az eltérés kb. 51 fordulat percenként, ami +3,3%-os relatív hiba. Ezek alapján azt állapítom meg, hogy a PLC-n megvalósított mérési eredmény kijelzése ±3,5% relatív hibahatáron belül történik, amit én elfogadhatónak találtam.

3.4 Ciklikus megszakítás/Cyclic Interrupt [OB30]

A projekt szempontjából legfontosabb programrész, ahol a szabályozó is szerepel, az a Cyclic Interrupt blokk. A fő program megszakításának gyakoriságát 20 milliszekundumra állítottam be, ezt az értéket gondoltam kellően gyakorinak, a mérés

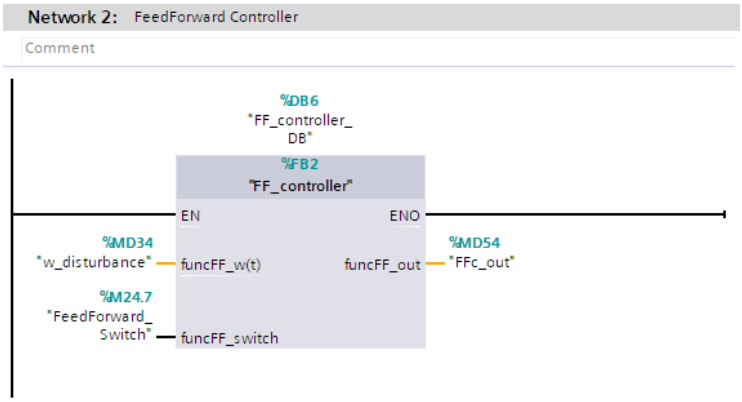
mintavételezése nincs ezzel szinkronban, mert az a fordulatszám függvényében változik. Mintavétel 1500 fordulat/perc estén (motor maximuma) kb. 13 milliszekundumonként történik, ami azt jelenti, hogy abban a tartományban a szabályozási ciklus 1-2 mintavételenként fog lefutni. A megszakítást LAD programnyelvben állítottam össze, ami öt ágból tevődik össze.

Az első ágban kapott helyet az Input_processing funkciókód meghívása. És egy egyenlőtlenség vizsgálat, miszerint, ha a PID_Compact manuális módban van, akkor az Input_processing *funcIP_manual_r_or_u* bemenetére FALSE érték kerül.



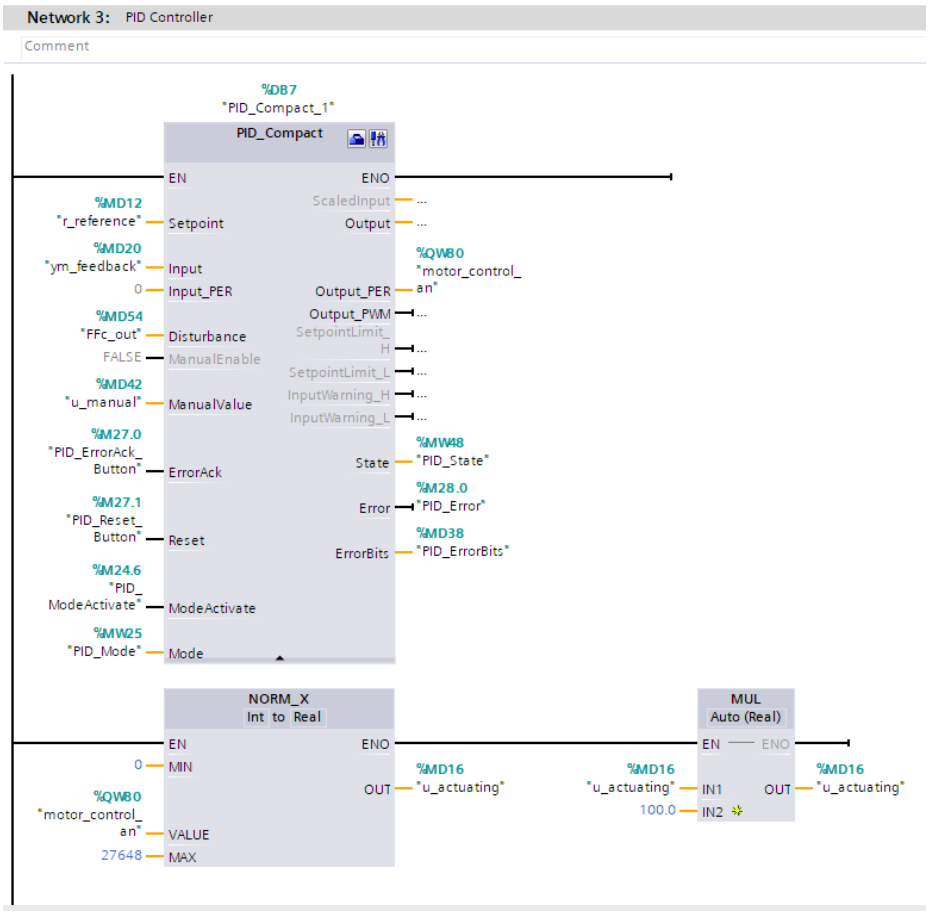
22. ábra – 1. ág, bemenő jelek feldolgozása

A második ág tartalmazza a zavarjel előre csatolást megvalósító FF_controller funkcióblokkot. Kimenő jelét a PID_Compact blokk összegzi a beavatkozó jellel.



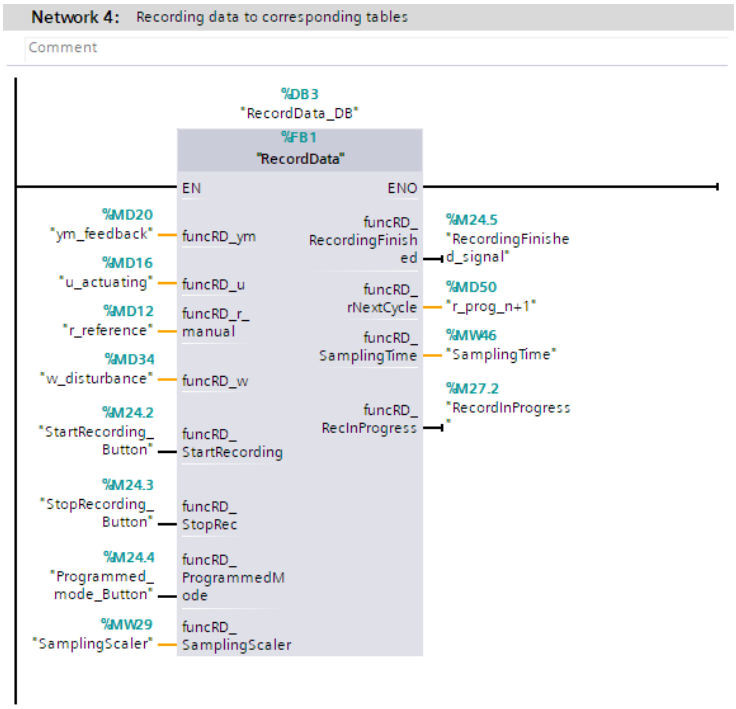
23. ábra – 2. ág, FFc_out jel előállítása

A harmadik ágban szerepel a szabályozó és annak kimenő jele 0-100% közé való transzformálása HMI kijelzés számára. Az ágban szereplő PID_Compact funkciói és felhasználásra szánt változói ennek a fejezetnek a mélységi keretein túllógnak, így ezeket, a jobb áttekinthetőség reményében még a HMI szoftverének bemutatása előtt részletezni fogom.



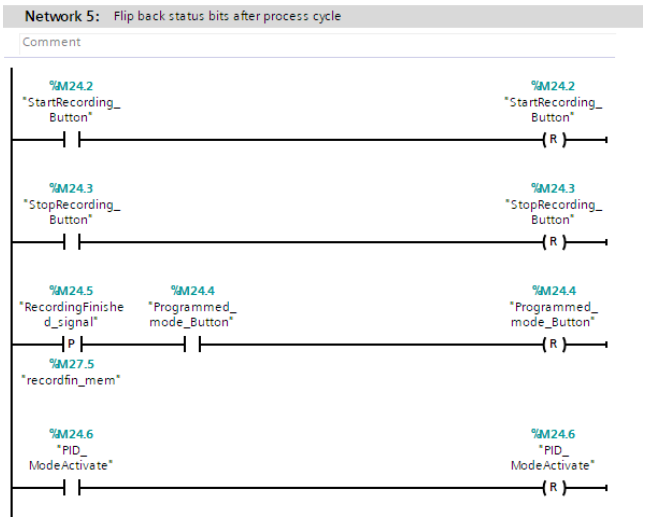
24. ábra – 3. ág, beavatkozó jel előállítása

A negyedik ágban fut le a mérési eredmények mentéséért, az automatikus referenciajel és a HMI számára visszajelzésnek szánt változók előállításáért felelős kód. Ezt a feladatot a RecordData funkcióblokk látja el.



25. ábra - 4. ág, mérési adatok mentése

Az ötödik ágban szereplő LAD logika a HMI kezeléséről információt szolgáltató változók alapállapotba való helyezését valósítja meg azok felhasználása után. Olyan változókat kezelek itt, amiket csak a Cyclic Interruptban használok fel.



26. ábra – változók visszaállítása

3.4.1 Input_processing funciókód

Name	Data type	Default value	Comment
Input			
funcIP_RPM	DInt		
funcIP_AN_disturbance	UInt		
funcIP_AN_pot	UInt		
funcIP_manual_r_or_u	Bool		TRUE: r FALSE: u
funcIP_ProgrammedMode	Bool		
funcIP_r_prog_n-1	Real		
Output			
funcIP_ym_out	Real		
funcIP_w_out	Real		
funcIP_u_out	Real		
funcIP_r_out	Real		

27. ábra – Input_processing változói

A bementén analóg, vagy dimenzióval rendelkező jelek, funkcióválasztó változók és a *funcIP_r_prog_n-1* real típusú érték szerepel, ami az előző ciklikus megszakítás során a RecordData funcióblokkban kapta értékét. Kimenő jelei már 0-100% közé normalizált értékek.

```
1  #funcIP_ym_out := 100 * NORM_X(MIN := 0, VALUE := #funcIP_RPM, MAX := 1500);
2  #funcIP_w_out := 100 * NORM_X(MIN := 0, VALUE := #funcIP_AN_disturbance, MAX := 10000);
3  IF #funcIP_ProgrammedMode = FALSE THEN
4      IF #funcIP_manual_r_or_u = 1 THEN
5          #funcIP_r_out := 100 * NORM_X(MIN := 0, VALUE := #funcIP_AN_pot, MAX := 27850);
6      ELSE
7          #funcIP_u_out := 100 * NORM_X(MIN := 0, VALUE := #funcIP_AN_pot, MAX := 27850);
8      END_IF;
9  ELSE
10     #funcIP_r_out := #"funcIP_r_prog_n-1";
11 END_IF;
```

28. ábra – Input_processing funkcióban lefutó kód

A kód 1. sorában normalizálom a visszacsatolt jellemzőt, gyártói megjelölés alapján a 100%-nak megfelelő terhelést 1500 fordulat/percre állítottam be. A 2. sorban a zavaró jellemző analóg mérésének a normalizálása történik 100%-nak a 10.000-es A/D átalakítóból kiolvasott értéket adtam meg, ami kb. 3,617 V-nak felel meg (27.648-as érték felel meg 10 V-nak). A 3-11. sorig a HMI-től érkező *funcIP_ProgrammedMode* „FALSE” értéke esetén állítom be a referencia jelet (PID_Compact automatic módjában), vagy beavatkozó jelet (PID_Compact blokk manual módjában) a potenciométerrel azonos nagyságúra. Viszont, ha a *funcIP_ProgrammedMode* „TRUE” értékű, akkor az erre a ciklusra érvényes referencia jelet az *r_Programmed.r_Data* tömb alapján szolgáltatja a RecordData funcióblokk.

3.4.2 FF_controller funkcióblokk

Bemeneti változói a *funcFF_switch*, ami engedélyezi az előre csatolt kompenzálást és a *funcFF_w(t)*, ami pedig a zavaró jellemző pillanatnyi mért értéke.

Name	Data type	Default value	Retain	Accessible f...	Visible in ...	Setpoint	Comment
Input				☐	☐	☐	
funcFF_w(t)	Real	0.0	Non-ret...	☑	☑	☐	
funcFF_switch	Bool	false	Non-retain	☑	☑	☐	
Output				☐	☐	☐	
funcFF_out	Real	0.0	Non-retain	☑	☑	☐	
InOut				☐	☐	☐	
Static				☐	☐	☐	
GE_in_prev	Real	0.0	Non-retain	☑	☑	☐	
GW_out_prev	Real	0.0	Non-retain	☑	☑	☐	
Temp				☐	☐	☐	
GE_inc	Real			☐	☐	☐	
GW_inc	Real			☐	☐	☐	
GE_out	Real			☐	☐	☐	
GW_out	Real			☐	☐	☐	
GW_2_in	Real			☐	☐	☐	
Constant				☐	☐	☐	
delta_t	Real	0.02		☐	☐	☐	
GE_TD	Real	20.0		☐	☐	☐	10.608
GW_T1	Real	2.9133		☐	☐	☐	
KC	Real	0.02716138		☐	☐	☐	GW_Kp/GE_Kp; 0.096996/3.5711

29. ábra – FF_controller funkcióblokk változói

A kimenetre az átviteli függvény eredménye kerül (bekapcsolt állapotban). A konstans változókban vannak letárolva az átviteli tagok időállandói, erősítései és a kód futásának ciklusideje.

```
1 IF #funcFF_switch = 0 THEN
2     #funcFF_out := 0;
3 ELSE
4     #GW_inc := #delta_t / #GW_T1 * (#"funcFF_w(t)" - #GW_out_prev);       //GW
5     #GW_out := #GW_inc + #GW_out_prev;
6     #GE_out := #GE_TD * (#GW_out - #GW_out_prev) / #delta_t;              //GE
7     #funcFF_out := #KC * #GE_out;
8     #GW_out_prev := #GW_out;
9 END_IF;
```

30. ábra – FF_controller funcióblokk programkódja

A kód 1. sorában lévő feltétel eredményeképp a bemeneti switch állapotától függően 0, vagy az átviteli függvények eredménye kerül a kimenetre. A 4. és 5. sor tartalma az egytárolós tag (zavarjel átvitelét kompenzáló függvény), míg a 6. és 7. sor a differenciáló tag (eredő szakasz átvitelét kompenzáló függvény). Ezek az összefüggések az előre csatolt kompenzáló tag kiszámított átviteli függvényéből (6.7. fejezet) kerültek átalakításra.[11]

3.4.3 RecordData funkcióblokk

A blokk bemenetére a rögzítésre kerülő jelek (ym, u, r, w), a felvételt indító és megállító vezérlőbitek (*funcRD_StartRecording* és *funcRD_StopRecording*), a

mintavételezési idejének beállítását szolgáló szorzó (*funcRD_SamplingScaler*) és a referencia jel szolgálást eldöntő bit szerepelnek (*funcRD_ProgrammedMode*).

Name	Data type	Default value	Retain	Accessible f...	Visible in ...	Setpoint
Input						
funcRD_ym	Real	0.0	Non-ret...	☑	☑	☐
funcRD_u	Real	0.0	Non-retain	☑	☑	☐
funcRD_r_manual	Real	0.0	Non-retain	☑	☑	☐
funcRD_w	Real	0.0	Non-retain	☑	☑	☐
funcRD_StartRecording	Bool	false	Non-retain	☑	☑	☐
funcRD_StopRec	Bool	false	Non-retain	☑	☑	☐
funcRD_ProgrammedMode	Bool	false	Non-retain	☑	☑	☐
funcRD_SamplingScaler	Int	0	Non-retain	☑	☑	☐
Output						
funcRD_RecordingFinished	Bool	false	Non-retain	☑	☑	☐
funcRD_rNextCycle	Real	0.0	Non-retain	☑	☑	☐
funcRD_SamplingTime	Int	0	Non-retain	☑	☑	☐
funcRD_RecInProgress	Bool	false	Non-retain	☑	☑	☐
InOut						
<Add new>						
Static						
funcRD_iTime	Int	0	Non-retain	☑	☑	☐
funcRD_inProgress	Bool	false	Non-retain	☑	☑	☐
funcRD_Prescaler	Int	0	Non-retain	☑	☑	☐
Temp						
Constant						

31. ábra – RecordData funcióblokk változói

Amennyiben ennek értéke TRUE, akkor továbbadásra kerül a referencia jel a következő ciklikus megszakításra (*funcRD_rNextCycle*), ha a HMI-n a „Programmed mode with data recording” gombbal mérést indítunk. A felvétel végét jelző bit (*funcRD_RecordingFinished*), a milliszekundumra átszámított mintavételezési idő (*funcRD_SamplingTime*) és a felvétel futását jelző bit (*funcRD_RecInProgress*) kerülnek még a kimenetre. A függvény belső változói a kód működéséhez kellenek.

```
IF #funcRD_StartRecording = TRUE THEN                    //Start recording routine
    #funcRD_inProgress := TRUE;
    #funcRD_RecordingFinished := FALSE;
END_IF;

IF #funcRD_iTime = 2000 OR #funcRD_StopRec THEN          //Stop recording routine
    #funcRD_inProgress := FALSE;
    #funcRD_iTime := 0;
    #funcRD_RecordingFinished := TRUE;
    #funcRD_rNextCycle := 0.0;
END_IF;

IF #funcRD_inProgress = TRUE THEN                        //Data saving routine
    #funcRD_Prescaler := #funcRD_Prescaler - 1;
    IF #funcRD_Prescaler <= 0 THEN                        //Sampling time scaler (OB30 cycle time: 20ms)
        #funcRD_Prescaler := #funcRD_SamplingScaler;      //Sample time = x*Cycle time
        #funcRD_iTime := #funcRD_iTime + 1;
        IF #funcRD_ProgrammedMode THEN
            IF #funcRD_iTime = 1 THEN
                "r_Saved".r_Data[#funcRD_iTime] := "r_Programmed".r_Data[#funcRD_iTime];       //1..2000 table, there is no 0th element of the array
            ELSE
                "r_Saved".r_Data[#funcRD_iTime] := "r_Programmed".r_Data[#funcRD_iTime - 1];   //reference signal of this cycle, becouse the iTime is already incremented
            END_IF;
            #funcRD_rNextCycle := "r_Programmed".r_Data[#funcRD_iTime];      //this will be the reference signal in the next cycle
        ELSE
            "r_Saved".r_Data[#funcRD_iTime] := #funcRD_r_manual;
        END_IF;
        "u_Saved".u_Data[#funcRD_iTime] := #funcRD_u;
        "w_Saved".w_Data[#funcRD_iTime] := #funcRD_w;
        "ym_Saved".ym_Data[#funcRD_iTime] := #funcRD_ym;

    END_IF;
END_IF;
#funcRD_RecInProgress := #funcRD_inProgress;
#funcRD_SamplingTime := #funcRD_SamplingScaler * 20;
```

32. ábra – RecordData funcióblokkban szereplő programkód

A 11. sorig az indításhoz és leállításhoz tartozó változóbeállítások vannak. A 13. sortól kezdődik az adatrögzítési folyamat programkódja. A *funcRD_Prescaler* ciklusonkénti dekrementálásával oldom meg, ha nem szeretnék minden 20 ms-ban mintavételt (14. sor). A *funcRD_iTime* változóban van a folyamat aktuális pozíciója, aminek 1-2000-ig növelem az értékét. Ez a változó jelöli ki továbbá, hogy a tömbök hányadik eleme legyen az éppen írott. Az alapjel az *r_Saved.r_Data* tömbben, a beavatkozójel az *u_Saved.u_Data* tömbben, a zavarjel a *w_Saved.w_Data* tömbben, míg a visszacsatolt jel az *ym_Saved.ym_Data* tömbben kerül rögzítésre.

4 PID_Compact blokk

A fejezet tartalma a Siemens által közzétett PID_Compact blokkot bemutató dokumentáció [12] alapján került összeállításra.

4.1 Kiválasztásának oka

A TIA Portalban többféle, a szabályozás tervezésének menetét megkönnyítő technológiai blokk érhető el:

- PID_Compact
- PID_3Step
- PID_Temp

A PID_3Step és PID_Temp blokkok speciális felhasználásúak és egyik sem motor fordulatszám szabályozásra használatos, ezért ezeket kevésbé részletezem. A **PID_3Step** háromállású, integráló jellegű beavatkozók és szelepek irányítására lett kitalálva. A PID_Compact funkciói mellett van lehetőség benne a beavatkozó pozíciójának visszacsatolásra is. A **PID_Temp** hűtő és fűtő szakaszok szabályozására ajánlott. Külön kimenettel rendelkezik a hűtő és külön a fűtő egység számára, emellett lehetőség van holt zóna (mekkora legyen az a legkisebb hibajel, amikor a szabályozó beavatkozik) és kaszkád szabályozás megvalósítására két PID_Temp blokk megfelelő konfigurálásának esetén.

A **PID_Compact** egy általános célú, univerzális felhasználásra kialakított PID blokk. A felsoroltak közül ez a legegyszerűbb, de kellően sok funkcióval rendelkezik. A szabályozó el van látva bemeneti- és kimeneti szaturációs védelemmel, integrál anti windup-al, lökésmentesítéssel a szabályozó bekapcsolása esetén. Mindezek mellett a leghasznosabb funkciója az automatikus hangolás, ami a szabályozó paramétereinek az automatikus meghatározását jelenti.

4.2 Felépítése

Átviteli függvénye:

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

y: kimeneti változó

K_P: erősítési tényező

b: erősítési tényező súlyozása (csak a referenciajelre van hatással)

w: referenciajel

x: bemenet (visszacsatolt jel)

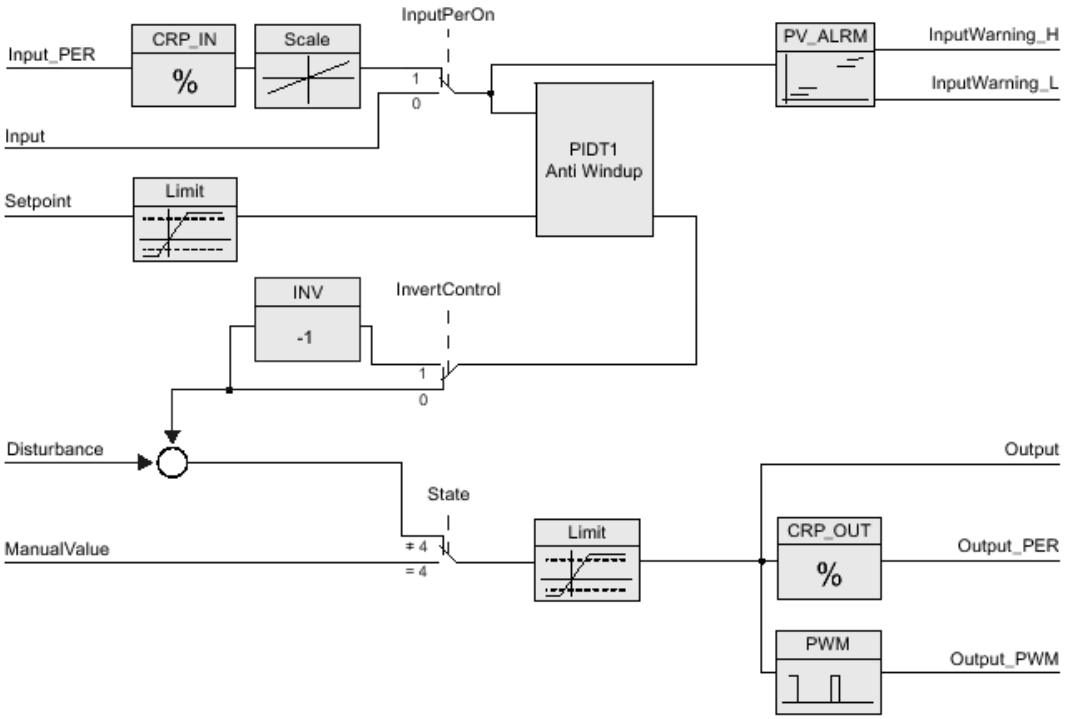
T_I: integráló tag időállandója

T_D: differenciáló tag időállandója

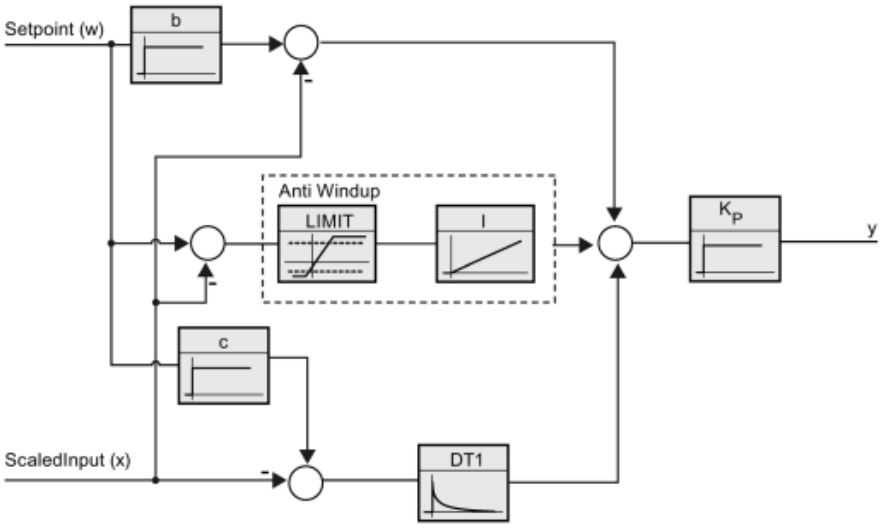
a: differenciáló tag késleltetési együtthatója

c: differenciáló tényező súlyozása (csak a referenciajelre van hatással)

s: Laplace operátor



33. ábra – PID_Compact blokk struktúrája [13]



34. ábra - PIDT1 Anti Windup struktúrája [14]

4.3 PID_Compact működési módjai

Hat működési módját különböztethetjük meg a technológiai blokknak, amelyek közötti váltás a blokk bemenetein vezérelhető.

Inaktív: (Inactive) A szabályozó kikapcsolt állapotban van és a blokk kimenete minden esetben 0.0 értéket vesz fel.

Előhangolás: (Pretuning) Automatikus hangolás első fázisa, szabályozó paramétereinek meghatározása történik egységugrásra adott válasz alapján. A mód elindítása feltételekhez kötött, amit a későbbiekben, a hangolás meneténél fogok tárgyalni.

Finomhangolás: (Fine tuning) Automatikus hangolás második fázisa. A szabályozó működéséhez nem szükséges lefuttatni, de robusztusabb szabályozási folyamat érhető el vele. Ennek aktiválása is feltételekhez kötött, amit szintén a hangolás meneténél fogok részletezni.

Automatikus mód: (Automatic mode) A szabályozó a beállított/meghatározott paraméterekkel működik.

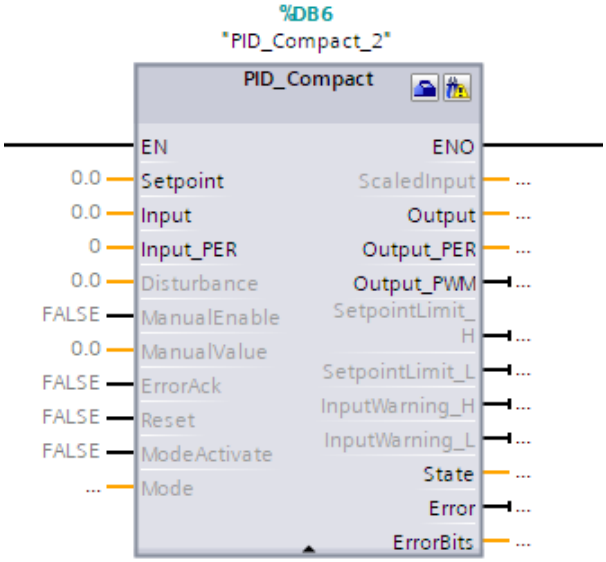
Manuális mód: (Manual mode) A mód aktiválásával a felhasználó határozhatja meg a rendelkező jelet, olyan módon, hogy a blokk *ManualValue* bemenetére adott jel változtatás nélkül kerül a blokk kimenetére.

Helyettesítő kimeneti érték: (Substitute output value) A szabályozó leállását előidéző hiba esetén milyen manuálisan megadott érték kerüljön annak kimenetére. Ez egy

biztonsági állapot, melyben lehetőség van a blokk hibamonitorozására. A mód kikapcsolható, aktiválódása konfigurációfüggő.

4.4 PID_Compact változói és paraméterei

A blokk megfelelő működéséhez azt egy ciklikus megszakítású szervezeti blokkba kell elhelyezni (Cyclic Interrupt [OB30]). A fejlesztői környezet beilleszthető blokkján a szabályozási körhöz szükséges csatlakozási pontokat, illetve a blokk viselkedését meghatározó és állapotának visszajelzését szolgáló legfontosabb paramétereket láthatjuk a bemeneti és kimeneti oldalakon. Ezeknek ismerete elengedhetetlen a szabályozó blokk megfelelő beállításához.



35. ábra

4.4.1 Bemeneti változók

Setpoint (real): szabályozási kör referencia jele

Input (real): szabályozási kör visszacsatolt jele

Input_PER (int): visszacsatolt jel, az analóg távadó közvetlen (a PLC analóg bemenetén előállt) jele.

Disturbance (real): a szabályozott jellemző zavarásának mért értéke. A blokk struktúráján [33. ábra] látható, hogy az itt megadott érték csak összeadódik a szabályozó kimenetével,

ezért a zavarjel előre csatolásához előzetes feldolgozása szükséges a mért zavarjellemzőnek.

ManualEnable (bool): Manuális mód aktiválása bemenetre adott felfutó él esetén.

ManualValue (real): A blokk manuális módja esetén ez az érték lesz a szabályozási kör beavatkozó jele.

ErrorAck (bool): Hiba megerősítés felfutó él esetén. Tárolt hibák és figyelmeztetések törlését eredményezi.

Reset (bool): újraindítja a PID kontrollert. Felfutó élre inaktív módba kerül, tárolt hibák és figyelmeztetések törlődnek, integrátor szummáját üríti. Lefutó élre az inaktív módot megszünteti és a *Mode* paraméterben meghatározott állapotba kerül.

ModeActivate (bool): Felfutó élre a blokk a *Mode* változóban tárolt értéknek megfelelő állapotba kerül.

4.4.2 Be-/kimeneti változók

Mode (int): Ez a paraméter dönti el, hogy milyen módba kerüljön a PID_Compact blokk a következő események bekövetkezése esetén:

ModeActivate	0→1 átmenet
Reset	1→0 átmenet
ManualEnable	1→0 átmenet

Továbbá a PLC indítását követően, ha a *RunModeByStartup* változó „TRUE” értéket vesz fel.

A különböző módoknak megfelelő változó értékek:

0 = Inaktív
1 = Előhangolás
2 = Finomhangolás
3 = Automatikus mód
4 = Manuális mód

4.4.3 Kimeneti változók

ScaledInput (real): *Input_PER* használata esetén, az analóg távadó jelének 0-100% közé normalizált dimenzió nélküli megfelelője.

Output (real): szabályozási kör rendelkező jele.

Output_PER (int): a PLC analóg kimenete számára előállított skálázott jel. Analóg vezérlésű beavatkozók számára.

Output_PWM (bool): PWM vezérlésű beavatkozók számára előállított jel.
SetpointLimit_H (bool): a szabályozási kör rendelkező jele elérte a beállított felső határt. A beállított határ fog feldolgozásra kerülni.
SetpointLimit_L (bool): a szabályozási kör rendelkező jele elérte a beállított alsó határt. A beállított határ fog feldolgozásra kerülni.
InputWarning_H (bool): a szabályozott jellemző elérte vagy meghaladta a beállított felső figyelmeztetési határt.
InputWarning_L (bool): a szabályozott jellemző elérte vagy meghaladta a beállított alsó figyelmeztetési határt.
State (int): visszajelzés a PID_Compact blokk aktuálisan érvényben lévő módjáról. A változó értékeihez tartozó módok megfelelnek a *Mode* be-/kimeneti változóéval, azzal a kiegészítéssel, hogy ez felvehet *State* = 5 (Helyettesítő kimeneti érték) állapotot is.
Error (bool): „TRUE" értéke azt jelzi, hogy legalább egy hiba van a PID_Compact blokk aktuális ciklusában.
ErrorBits (dword): megmutatja, hogy mely hiba van, vagy volt jelen a PID_Compact blokkban. Tartalmát törölni a *Reset,* vagy *ErrorAck* bitek felfutó élével lehet.

4.4.4 Belső változók/paraméterek

Itt a meglehetősen nagy számú változó miatt csak azokat fogom részletezni, amik relevánsak az általam megvalósítani kívánt szabályozási kört tekintve.
IntegralResetMode (int): meghatározza, hogy az integráló tag összege (*IntegralSum*) milyen értéket vegyen fel a blokk inaktív módból automatikus módba való átkapcsolása esetén. A változó számszerű értékei és a hozzájuk tartozó módok a

0: simítás, lökésmentes váltás
1: *IntegralSum* törlése
2: *IntegralSum* megtartása
3: Előre meghatározott érték az *IntegralSum*-ban

OverwriteInitialOutputValue (real): az *IntegralResetMode* = 3 esetén átkapcsolásnál úgy számol *IntegralSum* értéket, hogy a következő ciklusra ez az érték kerüljön a blokk kimenetére (ez legyen a szabályozási kör beavatkozó jele).

RunModeByStartup (bool):

TRUE: a PLC indítását követően a *Mode* változóban meghatározott állapotba kerül a blokk.

FALSE: a PLC indítást követően inaktív módba kerül a blokk.

LoadBackUp (bool): „TRUE” értékre állítása esetén a jelenlegit megelőző utolsó érvényes PID paraméterek kerülnek újra beállításra. Beállítás után a változó értéke automatikusan visszaáll „FALSE” értékre.

ActivateRecoverMode (bool):

TRUE: automatikus módban a blokkban keletkező bizonyos hibák esetén az a helyettesítő kimeneti érték módot veszi fel, vagy ha a hiba súlyossága megengedi, akkor automatikus módban marad.

FALSE: bármilyen hiba esetén a blokk inaktív módba kerül.

Warning (dword): fennálló és tárolt figyelmeztetések.

Progress (real): a hangolás folyamatának állapota %-os értékben.

CurrentSetpoint (real): mindig a pillanatnyi referencia értéket mutatja, hangolás esetén az utolsó -hangolás előtti- érték tárolódik el benne.

CancelTuningLevel (real): hangolás közben ez a megengedett legnagyobb referencia érték eltérés a *CurrentSetpoint*-ban tárolttól. Ezt az eltérést meghaladva a hangolás megszakad.

SubstituteOutput (real): helyettesítő kimeneti érték módban ennek megfelelő lesz a kimeneten megjelenő jel.

SetSubstituteOutput (bool):

TRUE: hiba esetén a blokk helyettesítő kimeneti érték módba kerül.

FALSE: hiba esetén a blokk inaktív módba kerül.

Config (struktúra): 16 darab változót tartalmazó struktúra. Ezekkel állítható be:

- melyik bemenetet használja a blokk
- a szabályozó invertáló jellege
- visszacsatolt jel felső és alsó korlátja, illetve figyelmeztetése
- referencia jel felső és alsó korlátja
- beavatkozó jel felső és alsó korlátja
- PWM kimenet minimum impulzusszélesség külön-külön a 0 és 1 állapotokra

- analóg bemenet skálázásának határértékei

CycleTime (struktúra): a PID_Compact blokk lefutásának első 10 ciklusából átlagot képezve kerül meghatározásra a szabályozó Δt, mintavételezési ideje. Ebben a struktúrában négy változó szerepel. Kikapcsolható a funkció és felügyelhető a mintavételezési idő. Felügyelet esetén, ha túl sok idő telik el a blokk következő meghívásáig, akkor a PID_Compact hibaüzenettel inaktív állapotba kerül. A struktúra *CycleTime.Value* változója tartalmazza a jelenleg érvényben lévő mintavételezési időt.

CtrlParamsBackUp (struktúra): *LoadBackUp* aktiválása esetén ebből a struktúrából lesznek kimásolva a PID szabályozó paraméterei a *Retain* struktúrába.

PIDSelfTune (struktúra): két struktúrát tartalmaz, egyet az előhangolás (*SUT*) és egy másikat a finomhangolás (*TIR*) számára. A struktúrák elemei információkat szolgáltatnak a hangolás menetéről.

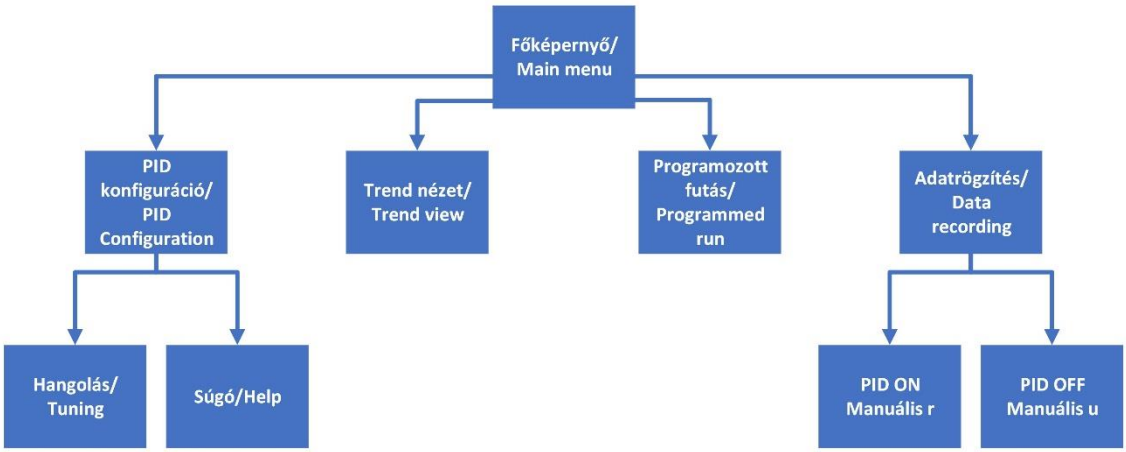
PIDCtrl.IntegralSum (real): a PID szabályozó integráló tag integrálási összegének pillanatnyi értékét tartalmazza.

Retain.CtrlParams (struktúra): a szabályozó jelenleg érvényben lévő paramétereit találhatók benne, mint az erősítés, integráló tag időállandója, differenciáló tag időállandója, differenciáló tag késleltetési együtthatója, erősítés súlyozás, differenciáló tag súlyozás és a hangolás során számított mintavételezési idő.

A PID_Compact blokknak van egy felhasználóbarát felülete, ahol ezek a változók beállíthatók. Ennek megnyitásához a blokk jobb felső sarkában lévő „Configuration” gombra [35. ábra] kell kattintani.

5 HMI program bemutatása

Az eszközön menüs elrendezésben valósítottam meg a kezelőfelületet. Az alábbi ábra [36. ábra] összefoglalja, hogy milyen hierarchia szerint épül fel a menü. Ennek alapján fogom bemutatni az elkészített felhasználói interfészt.



36. ábra – HMI menürendszer

5.1 HMI változói

A jobb átláthatóság kedvéért a TIA Portal-ban több táblázatban hoztam létre a változókat felhasználási helyük szerint. Azonban vannak általános, a HMI majdnem minden menüjében felhasznált változók, azokat a „Default tag table” elnevezésű könyvtárba soroltam.

Elnevezés	Kapcsolat	PLC címke	Adat típus	Frissítési mód	Frissítési ciklus
HMI_r(t)	HMI_Connection_1	r_reference	Real	Cyclic in operation	100 ms
HMI_u(t)	HMI_Connection_1	u_actuating	Real	Cyclic in operation	100 ms
HMI_ym(t)	HMI_Connection_1	ym_feedback	Real	Cyclic in operation	100 ms
HMI_RPM	HMI_Connection_1	motor_RPM	DWord	Cyclic in operation	100 ms
HMI_FF_switch	HMI_Connection_1	FeedForward_Switch	Bool	Cyclic in operation	100 ms
HMI_PID_Help_Page	<Internal Tag>	<No Value>	Bool	Cyclic in operation	1 s
HMI_w(t)	HMI_Connection_1	w_disturbance	Real	Cyclic in operation	100 ms

3. táblázat – Default tag table

Ebben a felsorolásban van egy változó, ami nem kerül átadásra a PLC számára, a *HMI_PID_Help_Page* egy belső HMI változó, aminek felhasználását a Súgó/Help képernyő alfejezetben magyarázom.

A PID szabályozó konfigurációjához és vezérléséhez használt változókat a „PID_Config” könyvtárban helyeztem el.

Elnevezés	Kapcsolat	PLC címke	Adat típus	Frissítési mód	Frissítési ciklus
HMI_PID_Gain	HMI_Connection_1	PID_Compact_1.Retain.CtrlParams.Gain	Real	Cyclic in operation	100 ms
HMI_PID_IntegralTime	HMI_Connection_1	PID_Compact_1.Retain.CtrlParams.Ti	Real	Cyclic in operation	100 ms
HMI_PID_DerivativeTime	HMI_Connection_1	PID_Compact_1.Retain.CtrlParams.Td	Real	Cyclic in operation	100 ms
HMI_PID_TdFilterRatio	HMI_Connection_1	PID_Compact_1.Retain.CtrlParams.TdFiltRatio	Real	Cyclic in operation	100 ms
HMI_PID_ProportionalWeight	HMI_Connection_1	PID_Compact_1.Retain.CtrlParams.PWeighting	Real	Cyclic in operation	100 ms
HMI_PID_DerivativeWeight	HMI_Connection_1	PID_Compact_1.Retain.CtrlParams.DWeighting	Real	Cyclic in operation	100 ms
HMI_PID_LoadBackUp	HMI_Connection_1	PID_Compact_1.LoadBackUp	Bool	Cyclic in operation	100 ms
HMI_PID_State	HMI_Connection_1	PID_State	Int	Cyclic in operation	100 ms
HMI_PID_Mode	HMI_Connection_1	PID_Mode	Int	Cyclic in operation	100 ms
HMI_PID_ModeActivate	HMI_Connection_1	PID_ModeActivate	Bool	Cyclic in operation	100 ms
HMI_PID_ErrorAck	HMI_Connection_1	PID_ErrorAck_Button	Bool	Cyclic in operation	100 ms
HMI_PID_Reset	HMI_Connection_1	PID_Reset_Button	Bool	Cyclic in operation	100 ms
HMI_PID_Error	HMI_Connection_1	PID_Error	Bool	Cyclic in operation	100 ms
HMI_PID_ErrorBits	HMI_Connection_1	PID_ErrorBits	DWord	Cyclic in operation	100 ms
HMI_PID_IntSum	HMI_Connection_1	PID_Compact_1.PIDCtrl.IntegralSum	Real	Cyclic in operation	100 ms

4. táblázat – PID_Config tag table

A PID hangolásához szükséges változókat a PID_Tuning könyvtárban összesítettem.

Elnevezés	Kapcsolat	PLC címke	Adat típus	Frissítési mód	Frissítési ciklus
HMI_Tuning_Progress	HMI_Connection_1	PID_Compact_1.Progress	Real	Cyclic in operation	100 ms
HMI_Pretuning_State	HMI_Connection_1	PID_Compact_1.PIDSelfTune.SUT.State	Int	Cyclic in operation	100 ms
HMI_FineTuning_State	HMI_Connection_1	PID_Compact_1.PIDSelfTune.TIR.State	Int	Cyclic in operation	100 ms
HMI_TuneRule_SUT	HMI_Connection_1	PID_Compact_1.PIDSelfTune.SUT.TuneRule	Int	Cyclic in operation	100 ms
HMI_TuneRule_TIR	HMI_Connection_1	PID_Compact_1.PIDSelfTune.TIR.TuneRule	Int	Cyclic in operation	100 ms

5. táblázat – PID_Tuning tag table

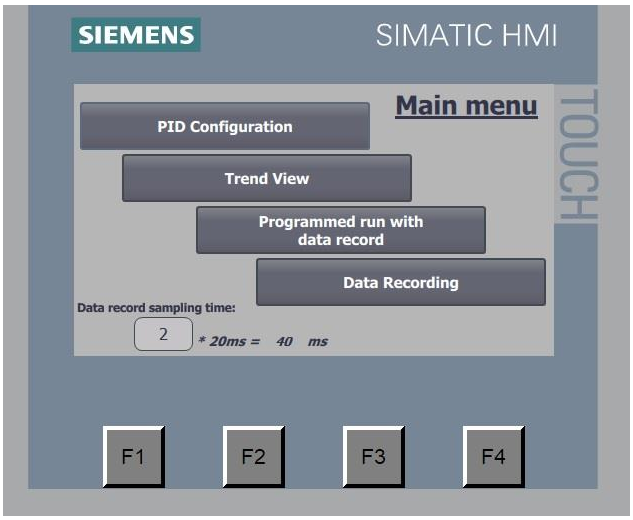
Az adatrögzítési folyamathoz szükséges változók a „RecordData” könyvtárba kerültek.

Elnevezés	Kapcsolat	PLC címke	Adat típus	Frissítési mód	Frissítési ciklus
HMI_RecSampleMultiplier	HMI_Connection_1	SamplingScaler	Int	Cyclic in operation	100 ms
HMI_RecSampleTime	HMI_Connection_1	SamplingTime	Int	Cyclic in operation	100 ms
HMI_StartRecording	HMI_Connection_1	StartRecording_Button	Bool	Cyclic in operation	100 ms
HMI_RecFinished	HMI_Connection_1	RecordingFinished_signal	Bool	Cyclic in operation	100 ms
HMI_ModeIsProgrammed	HMI_Connection_1	Programmed_mode_Button	Bool	Cyclic in operation	100 ms
HMI_RecordInProgress	HMI_Connection_1	RecordInProgress	Bool	Cyclic in operation	100 ms
HMI_StopRecording	HMI_Connection_1	StopRecording_Button	Bool	Cyclic in operation	100 ms
HMI_Rec_iTime	HMI_Connection_1	RecordData_DB.funcRD_iTime	Int	Cyclic in operation	100 ms

6. táblázat – RecordData tag table

Amely változókat nem ismertettem már a PLC program bemutatása, annak globális változói, vagy a PID_Compact blokk fejezeteiben, azokat a HMI-ban való felhasználás helyén fogom ismertetni. A „PLC címke” oszlopban megadott PLC változóval az értékük megegyező, vagy maximum 100 milliszekundum időeltéréssel egyező lesz.

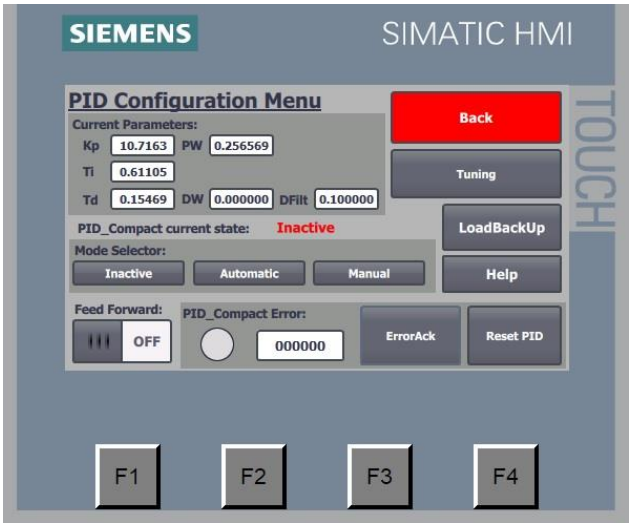
5.2 Főképernyő



37. ábra – Main_screen

Itt láthatóak az almenükbe lépéshez szükséges gombok és lehetőség van a „**Data record sampling time**” beállítására, amivel megadhatjuk, hogy milyen időközönként mentse el a szabályozási körben található jelek pillanatnyi értékeit. A 20 milliszekundumos szorzó azért van előre megadva, mert a rögzítési folyamat a ciklikus megszakításban fut, aminek a ciklusideje 20 ms. Az **F4** gomb megnyomásával a rendszer minden képernyőjéről a PID konfiguráció képernyőre jutunk, melyről visszaléphetünk az F4 újbóli megnyomásával.

5.3 PID konfiguráció képernyő



38. ábra – PID_Config

A „**Current Parameters**” területen megadhatók és visszaolvashatók a PID szabályozó paraméterei. Ezek a változók a *PID_Compact_1.Retain.CtrlParams* struktúrából vannak kiolvasva, tehát ezek az aktuálisan érvényesek. A **LoadBackUp** gombbal lehetőség van az utolsó tuning folyamat előtti érvényes paramétereket visszatölteni a *PID_Compact_1.CtrlParamsBackUp* struktúrából a *Retain* struktúrába. A LoadBackUp gomb megnyomásra a *HMI_PID_LoadBackUp* változót helyezi „TRUE” állapotba, amit a PID_Compact blokk a változók áttöltése után automatikusan visszahelyez „FALSE” állapotba. A *CtrlParamsBackUp* struktúrát a tuningolás első lépéseként a PID_Compact szoftvere tölti fel a *Retain* struktúrából.

A „**Tuning**” gomb megnyomásával a Hangolás képernyőre navigálhatunk tovább.

A „**PID_Compact current state**” sorban olvasható le a PID blokk aktuálisan érvényes állapota (*HMI_PID_State* változó).

A „**Mode selector**” részen szereplő gombokkal állítható be a blokk módja. A gombok úgy működnek, hogy megnyomásra beállítják a *HMI_PID_Mode* változót a megfelelő értékre (4.4.2. fejezet) és a *HMI_PID_ModeActivate* változót „TRUE” értékre állítják, aminek felfutására a PID_Compact blokk a kívánt állapotot veszi fel, amiről megbizonyosodhatunk a *HMI_PID_State* változó kiolvasásával.

A „**FeedForward**” csúszkával be-, illetve kikapcsolható a zavarjel előre csatolásáért felelős kontroller.

A „**PID_Compact Error**” részen megtekinthetők a blokk „hibabitjeinek” aktuális állapotai, aminek jelentéséről a részletesebb információkat kaphatunk a hibakódra nyomva. Több hiba fennállása esetén a hibának megfelelő értékek összeadásra kerülnek a *HMI_PID_ErrorBits* változóban. Annak bal oldalán elhelyezkedő kör a *HMI_PID_Error* változó aktuális értékéről ad visszajelzést (TRUE: piros, FALSE: üres). A terület jobb oldalán lévő „**ErrorAck**” és „**Reset**” gombokkal rendre a hiba nyugtázása (ErrorBits törlése) és a PID_Compact blokk újraindítása végezhető el. Ez a két gomb úgy működik, hogy benyomásra logikai 1 értékre állítják a hozzájuk tartozó bitet, majd felengedésre logikai 0-ba írják azokat vissza.

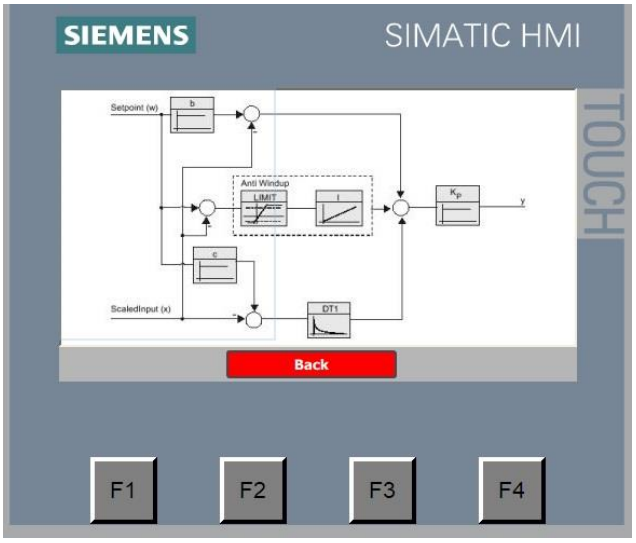
A PID_Compact blokk újraindulási folyamat:

0 → 1 átmenetre: A blokk inaktív státuszba kerül, *ErrorBits*, *Warnings* és *PIDCtrl.IntegralSum* változók törlése megtörténik.

1 → 0 átmenetre: A blokk *Mode* bemenet által meghatározott státuszba áll.

A „**Help**” gomb megnyomásával a Súgó képernyő hozható elő.

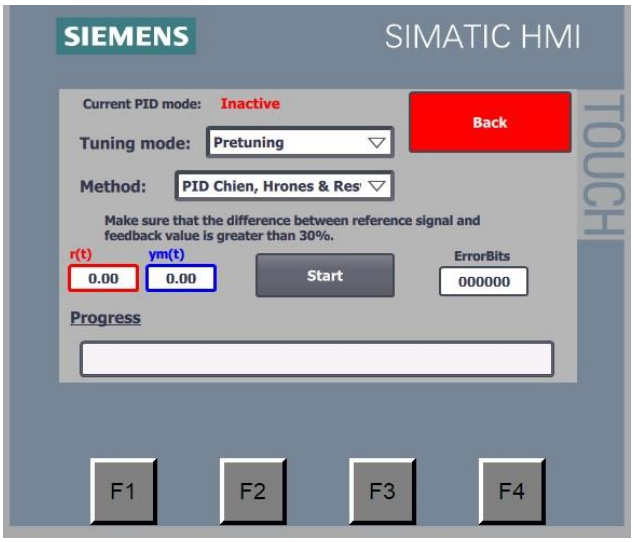
5.4 Súgó képernyő



39. ábra – PID_Help

Két oldalt tartalmazó lapozható képernyő, amin megtekinthető a PID_Compact PIDT1 részének a blokkvázlata [34. ábra], illetve paramétereinek leírása. Ezen a képernyőn használom fel a HMI egyetlen általam megadott belső változóját, a *HMI_PID_Help_Page* változót. Ezt arra használom fel, hogy letároljam a képernyő aktuális állapotát. Az ábra két szélén egy-egy láthatatlan gombot helyeztem el, amire rányomva változtatom ennek a változónak az értékét és ez alapján vált képet, válik lapozhatóvá az oldal.

5.5 Hangolás képernyő

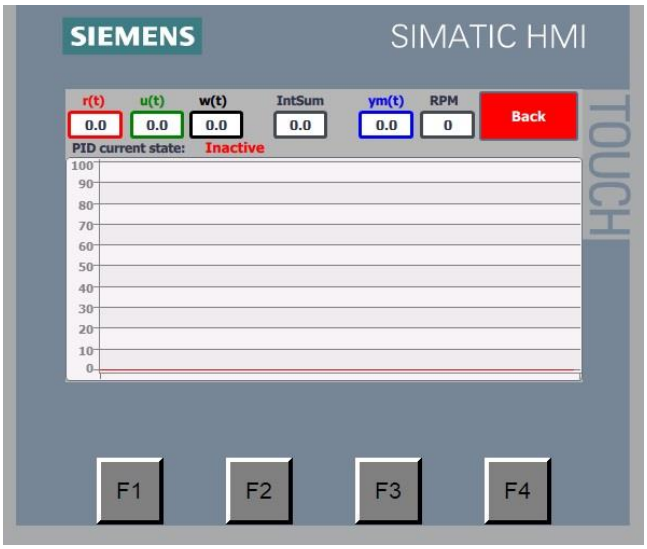


40. ábra – PID_Tuning

A „**Tuning mode**” lenyíló listában a Pretuning és Fine tuning módok közül választhatunk. Fine tuning csak Pretuning elvégzése után, vagy a *PID_Compact_1.PIDSelfTune.TIR.RunIn* változó „TRUE” értéke esetén kezdeményezhető. A „**Method**” lista előhangolás esetén PID CHR (Chien, Hrones and Reswick) és PI CHR számítási metódust, míg finomhangolásnál PID automatic, PID rapid, PID slow, Ziegler-Nichols PID, Ziegler-Nichols PI és Ziegler-Nichols P metódusokat kínál. Szöveges segítséget is találunk az indításhoz teljesítendő feltételekhez, aminek elégtételéhez segítségül szolgálhat a referencia (**rt**) és visszacsatolt jellemző (**ymt**) megjelenített értéke. A „**Start**” gomb megnyomásával elindul a hangolási folyamat, aminek stádiumairól a „**Progress**” címke alatt kapunk információkat.

A PLC program szemszögéből a hangolási folyamat elindításához *HMI_PID_Mode* (1=Pretuning, 2=Fine tuning) és *HMI_PID_ModeActivate* (0→1 átmenet) változókat használom. A számítási módszerek beállításához a *HMI_TuneRule_SUT* (pretuning) és *HMI_TuneRule_TIR* (fine tuning) változóknak kell a metódushoz tartozó értéket megadni. Az oszlopdiagrammon kijelzett folyamat százalékos értéke a *HMI_Tuning_Progress* változóban tárolt, míg fölötte szövegesen megjelenik (hangolás közben), hogy az milyen stádiumban van. Ez a kiírás *HMI_Pretuning_State* és *HMI_FineTuning_State* alapján kerül meghatározásra egy hozzájuk tartozó szöveges listából.

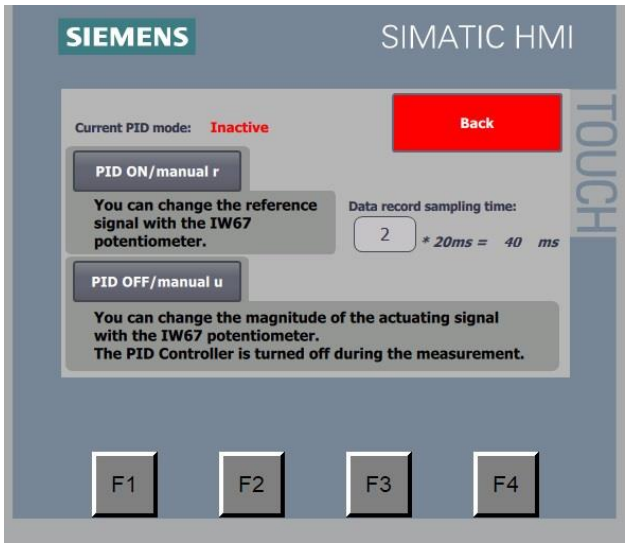
5.6 Trend nézet képernyő/Programozott futás képernyő



41. ábra – Trend_view

Az 41. ábrán látható jelek kerülnek kiírásra és grafikonos megjelenítésre. Hozzájuk tartozó változók balról jobbra haladva: *HMI_r(t)*, *HMI_u(t)*, *HMI_w(t)*, *HMI_PID_IntSum*, *HMI_ym(t)* és *HMI_RPM*. Ha a „Programmed run with data recording” gomb megnyomásával navigálunk ide a főképernyőről, akkor automatikusan elindul egy mérési értékeket regisztráló program, ami a PLC *r_Programmed.r_Data* tömb elemeinek felhasználásával fog referencia jelet beállítani a PID szabályozó számára. Továbbá megjelenik egy sáv, ami felvételből hátralévő időt mutatja.

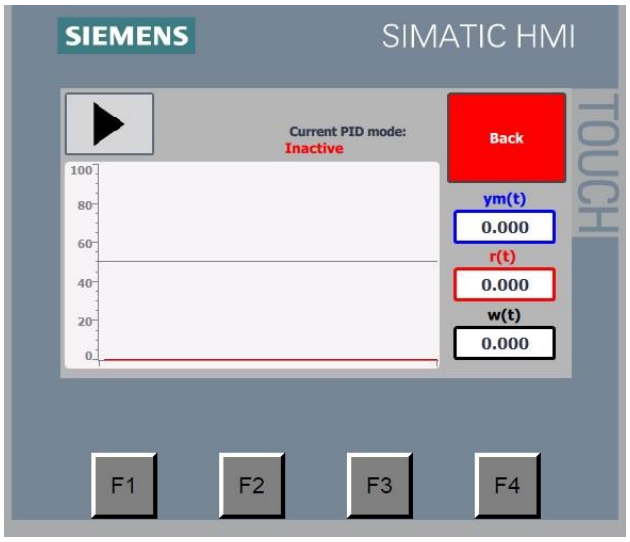
5.7 Adatrögzítés képernyő



42. ábra – Data_record

Itt is beállítható a mentett adatok mintavételezési ideje, megtekinthető a PID_Compact aktuális státusza. A „**PID ON/manual r**” gomb megnyomásával a Recording_rManual, míg a „**PID OFF/manual u**” a Recording_uManual képernyőre juthatunk. Az egyik módban a külső potenciométerrel a referenciajelet, míg a másikban a beavatkozójelet tudjuk módosítani.

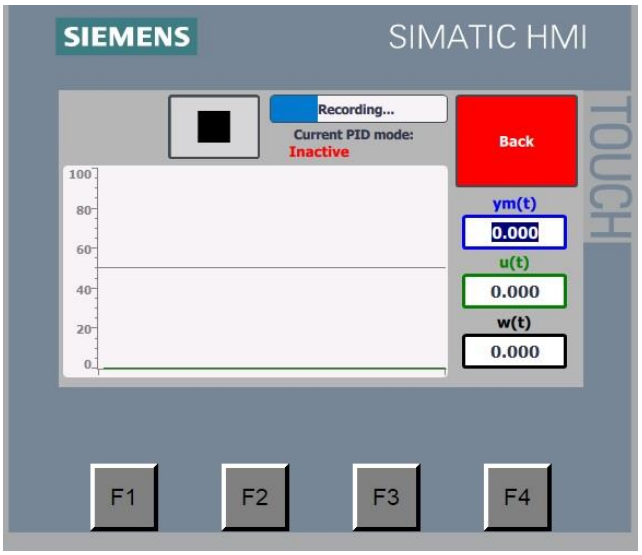
5.8 Manuális r és manuális u képernyők



43. ábra – Recording_rManual

Az rManual és uManual képernyők majdnem mindenben megegyeznek, annyi a különbség, hogy az itt megjelenített r(t) jel helyett, ott az u(t) jelet látjuk és ezzel azonosan a grafikonos megjelenítés is változik.

A háromszöget ábrázoló „**Play**” gomb megnyomásával elindíthatjuk az adatrögzítést, aminek folyamatáról a „**Recording…**” feliratú sáv ad visszajelzést. Annak betöltődésével az adatmentési folyamat automatikusan megáll, amit, ha nem szeretnénk kivárni, kézileg is megállíthatjuk a négyzetet ábrázoló „**Stop**” gombbal.



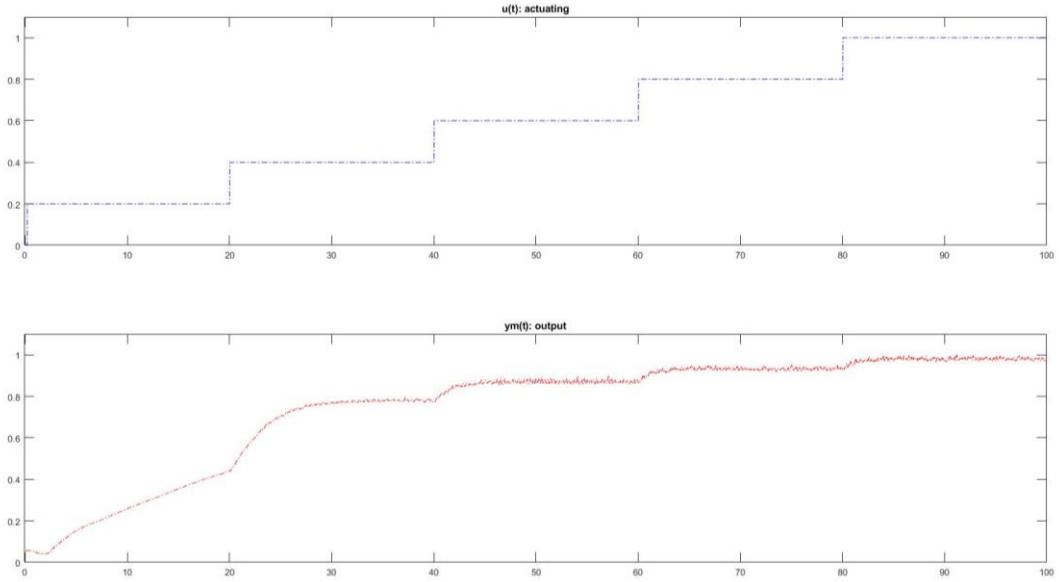
44. ábra – Recording_uManual

Ezen a két képernyőn a *HMI_StartRecording* és *HMI_StopRecording* változókat használtam fel a mérés elindításához, illetve megállításához. A „**Recordig…**” sáv a *HMI_Rec_iTime* változóból olvassa ki az értéket. Az is elmondható mindkét képernyő működéséről, hogy a „**Back**” gomb megnyomásával az adatrögzítés leáll.

6 A PID szabályozó és az előre csatolt zavarjel kompenzálás

6.1 Szakaszerősítés vizsgálata

Első lépésként a szakasz viselkedését vizsgáltam meg a beavatkozójel 0-100%-ig való növelésével, 20%-os lépésekkel. A mért adatokat minden esetben MATLAB szoftverrel rajzoltattam ki (plot függvény).

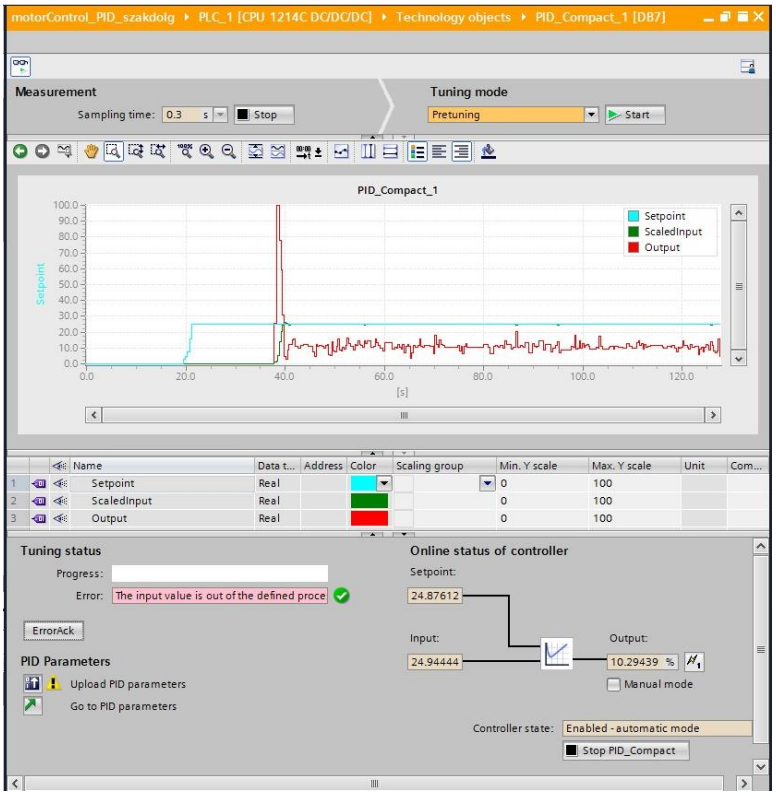


45. ábra – szakasz mérési eredmény

A grafikonokról látható, hogy a szakasz nem lineáris, 20%-os beavatkozó jelre az időállandója nagyobb, mint a többi léptetési esetben és az erősítés mértéke is csökken a motor fordulatszámának növekedésével. Ennek okán úgy döntöttem, hogy a PID paramétereket egy munkapont körüli szabályozásra fogom beállítani.

6.2 PID szabályozó paraméterezése PC-vel

A PID_Compact blokk PID szabályozónak felparaméterezése többféleképpen is történhet. A szabályozó paraméterei megadható kézi bevitellel is, vagy használhatók a blokk pretuning és fine tuning funkciói. Én a blokk beépített hangolási lehetőségét fogom használni. Ezt viszonylag egyszerűen elvégezhető a TIA Portal V13-as verziójában, a PID_Compact blokk [35. ábra] bal felső sarkában lévő „Commisioning” gombra kattintva előugró ablakban [46. ábra].



46. ábra – Commisioning ablak felülete

Fontos, hogy a tuning funkciók csak a blokk inaktív, automatikus, vagy manuális módjában indíthatók el. Miután megbizonyosodtam róla, hogy ez a feltétel teljesül, beállítottam a lehető legalacsonyabb mintavételezési időt (0,3 másodperc). Ez a mintavételezési sűrűség csak a PC-n való megjelenítésre vonatkozik. A jobb alsó sarokban lévő „Start PID_Compact” gombbal engedélyeztem a blokk futását és kiválasztottam a legördülő menüből a „pretuning” lehetőséget, majd a „start” gombbal elindítottam a hangolási procedúrát. Miután ezzel végzett, a „fine tuning” lehetőséggel is lefuttattam a műveletet. A hangolási folyamatok előtt még arra kellett odafigyelnem, hogy a pretuning esetében a rendelkezőjel és a visszacsatolt jel közötti különbség több legyen, mint 30%, a fine tuning esetében meg közel azonos. Ha ezek a feltételek nem teljesülnek, akkor a hangolási eljárás nem fog elindulni és hibára fut, aminek az okát az „Error” feliratú szövegmegjelenítő mezőből kiolvashatjuk. Az első tuning folyamat után kapott paraméterek a következők voltak:

Gain	10.71634
Ti	0.6110475
Td	0.1546869

TdFiltRatio	0.1
PWeighting	0.2565695
DWeighting	0.0
Cycle	0.0199947

Megjegyzem, hogy a szoftver és a mérések állapota itt még nem volt az előző fejezetekben leírtak szerinti állapotban, ezért a véglegesen használt PID paraméterek eltérnek ezektől az értékektől.

Mint látszik számítógépes online kapcsolat esetén kompromisszumok nélkül elvégezhető a hangolási folyamat, ennek ellenére szándékomban állt a hangolás és a PID paraméterek kézi bevitelének gyors és offline is elvégezhető megvalósítása a HMI segítségével. Erre szolgál a HMI PID_Config képernyője (5.3. fejezet). A későbbi hangolási folyamatokat már csak a HMI felületén végeztem.

6.3 PID szabályozó paramétereinek meghatározása

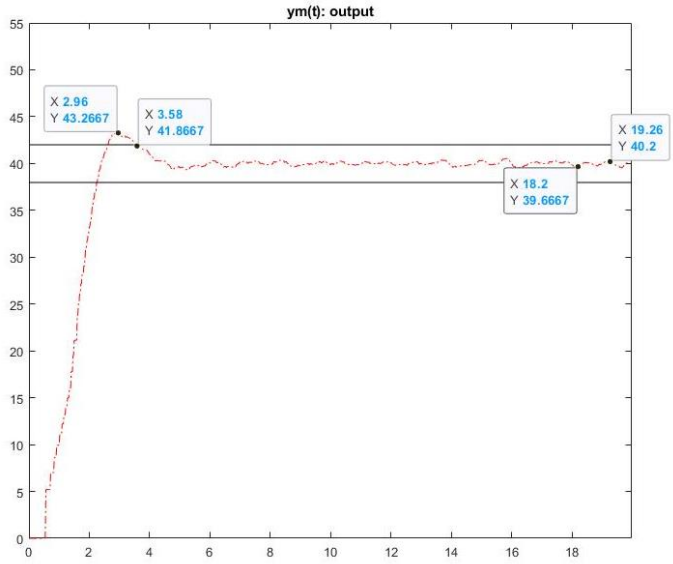
A szakasz nem-lineáris jellege miatt kijelöltem egy munkapontot, ami alapján a szabályozó paramétereit állítottam. Ez a munkapont a 675 ford./perc (45%). A HMI felületén a PID Configuration → Tuning → Pretuning → PID CHR → Start gombok megnyomása után el is indult az előhangolási folyamat, majd a potenciométerrel 45%-os alapjelet beállítva elindítottam a finomhangolási folyamatot is (Fine tuning → PID automatic → Start). Eredményként a következő PID paramétereket kaptam:

Gain	6,419511
Ti	0,6808932
Td	0,1533931
TdFiltRatio	0,1
PWeighting	0,5753454
DWeighting	0,0

6.4 Szabályozás minőségi jellemzői

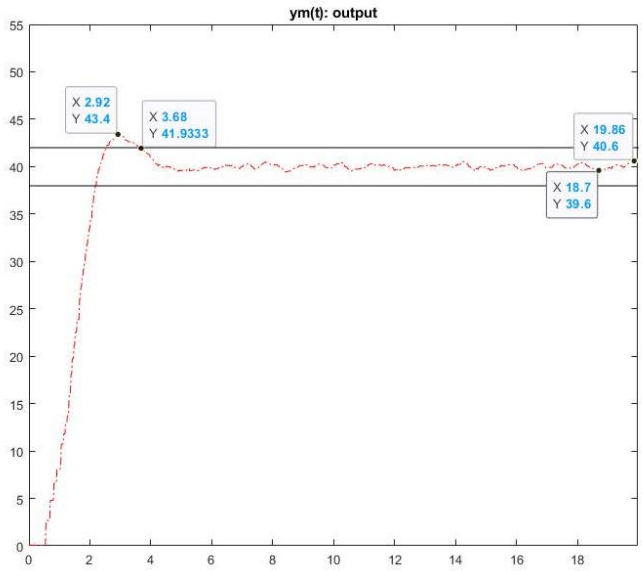
Ezen paraméterek használatával vizsgáltam meg a szabályozás minőségét. A HMI Data recording → PID ON/Manual r képernyőn elindítottam az adatrögzítési folyamatot, és a referenciajelet ugrásszerűen 40%-ra állítva vizsgáltam a visszacsatolt

jelet. Ezeket az adatokat aztán MATLAB szoftver segítségével kirajzoltattam és megvizsgáltam a minőségi jellemzőket.



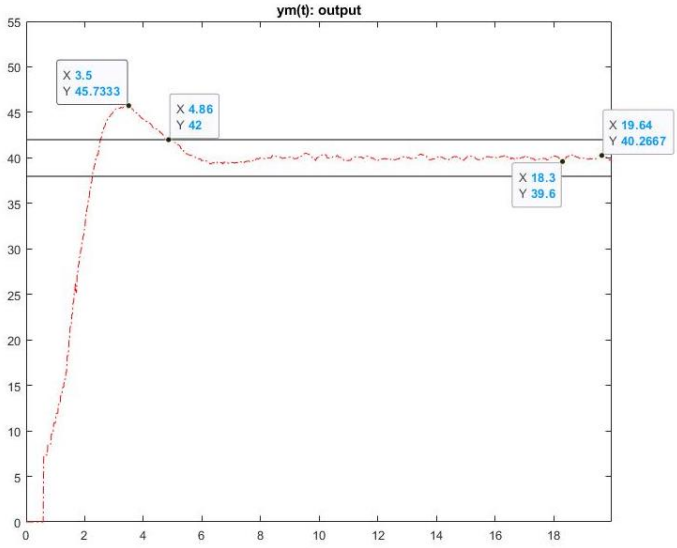
47. ábra – Minőségi jell. automatikus hangolás

A grafikonról azt olvastam le, hogy a szabályozott jellemző 43,27%-os értékig növekszik, majd csökken vissza, ami **8,175% túllendülés**t jelent. A **beállási idő 3,58 másodperc**, 5%-os toleranciasáv esetén. Maradó hiba kevésbé figyelhető meg, inkább egy folytonos 1% relatív nagyságú lengés látható, amit részben a mérési zaj tesz ki. Mivel a motornak nincs kifejezetten kijelölt szerepe, vagy feladata, hanem csak szabadon forog, ezért így nem igazán lehet meghatározni, hogy mely minőségi jellemzők szorulnak még javításra, emiatt úgy döntöttem, hogy a túllendülés mértékét csökkentem, de úgy, hogy a beállási idő ne növekedjen. Úgy gondoltam, hogy differenciáló tag hatásának növelésével ezt elérhetem, ezért annak hatását kiterjesztettem az alapjel változására is azzal, hogy a DWeighting paraméter [34.ábra] értékét megnöveltem a jelenlegi 0 értékről. Először arra voltam kíváncsi, hogy milyen hatása van, ha 1.0 értéket adok neki.



48. ábra – Minőségi jell. megnövelt DW-vel

A relatív **túllendülés** értéke **8,5%**, a **beállási idő 3,68 másodperc**, beállást követően a lengések relatív értéke 1,5% körül alakul. Meglepődve tapasztaltam, hogy nem igazán volt hatása a DW paraméter változtatásának, ezért megnöveltem a Td értékét 0,4-re, hogy nagyobb legyen a differenciáló hatás.



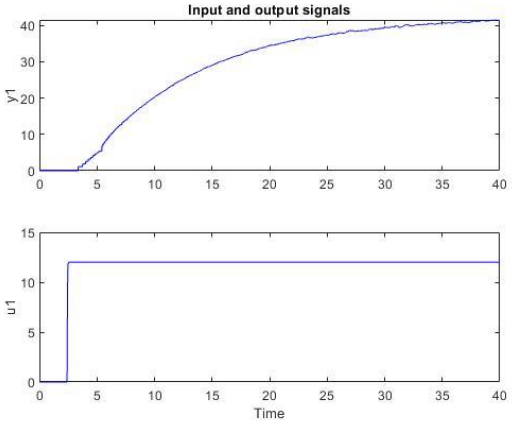
49. ábra – Minőségi jell. megnövelt Td

A relatív **túllendülés** értéke **14,3%**, a **beállási idő 4,86 másodperc**, beállást követően a lengések relatív értéke 1% körüli. Azt tapasztaltam, hogy a differenciáló hatás növelésével pont ellentétes hatást értem el, ezért a minőségi jellemzők javításának

próbálkozásával felhagytam, és az automatikus hangolás által beállított paramétereket írtam vissza a PID_Compact blokkba.

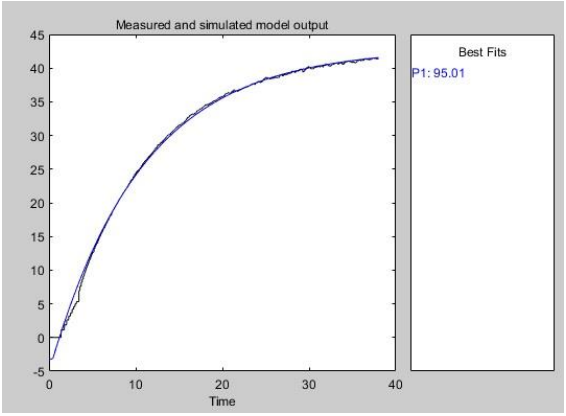
6.5 Szakasz modell közelítés

A szakasz munkapontját 40 és 50% között határoztam meg, ezért egy 40%-os kimeneti értéknek megfelelő beavatkozó jelet adtam a motort vezérlő áramkör bemenetére.



50. ábra – Szakasz ugrás-válasz függvénye

A grafikonokról azt állapítottam meg, hogy a kimeneti függvény arányos-egytárolós taggal jól közelíthető. A mérés elvéből kifolyólag alacsony fordulatszámokon a távadó nagy holtidővel rendelkezik, de ez a munkapont körüli fordulatszámnál már lecsökken, ezért úgy döntöttem, hogy a holtidős tagot kihagyom a zavarjel kompenzálásból. A közelítést MATLAB System Identification, process models tool-al készítettem.



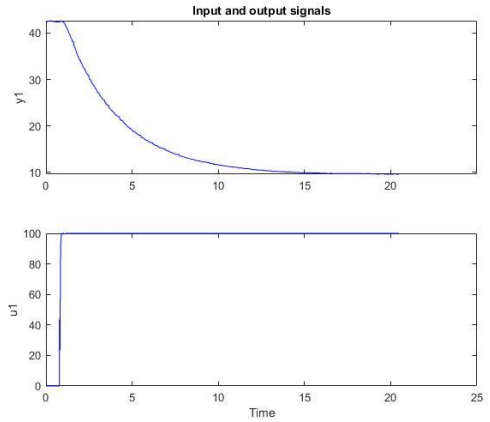
51. ábra – Szakasz identifikáció illeszkedése

Az identifikálás eredményeképp az alábbi átviteli függvényt kaptam:

$$G_e(s) = \frac{3{,}5711}{1 + 10{,}608s}$$

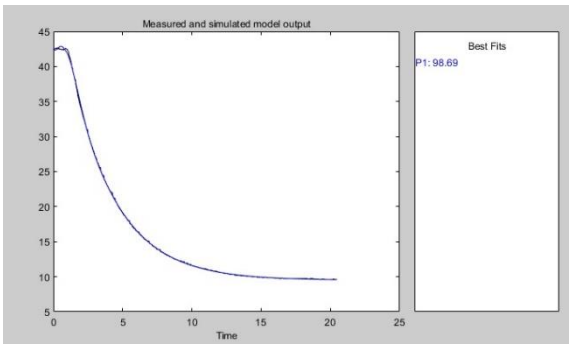
6.6 Zavarátvitel jellemzőinek közelítése

A mérés megkezdése előtt a beavatkozó jellel beállítottam 40%-os kimeneti értéknek megfelelő motor fordulatszámot, megvártam, míg beáll és csak aztán kezdtem neki a mérésnek. A labortápegységet 3,5 V-ra beállítva, annak kimenetére a mérés megkezdése után csatlakoztattam a féket, így 0-ról 100%-ra, ugrásszerűen próbáltam változtatni a zavarás mértékét.



52. ábra – Szakasz zavarjel ugrásra adott válasza

A grafikonokon azt láttam, hogy megint közelíthető a kimeneti függvény arányos--egytárolós taggal, de annak erősítése most negatív előjelű. Beolvastam az adatokat a System Identification tool-ba és ugyanúgy egy pólust jelöltem meg, de most a „Disturbance model”-t is beállítottam a folyamat indítása előtt.



53. ábra –T1 tag illeszkedése zavarás hatására adott válasz függvényére

Az identifikálás eredményeképp az alábbi átviteli függvényt kaptam:

$$G_W(s) = \frac{-0{,}096996}{1 + 2{,}9133s}$$

6.7 Előre csatolt zavarjel kompenzáló tag átviteli függvénye

Mivel a Feed Forward kontroller kimenete a beavatkozó jelhez fog hozzáadódni, ezért a szakasz időállandói a kimenő jelére hatással lesznek. Ennek kioltására a szakasz átviteli függvényének (G_e) reciprok értékével szoroztam meg a zavaró tényező átviteli függvényét (G_W).

$$G_{FF}(s) = -\frac{G_W(s)}{G_e(s)}$$

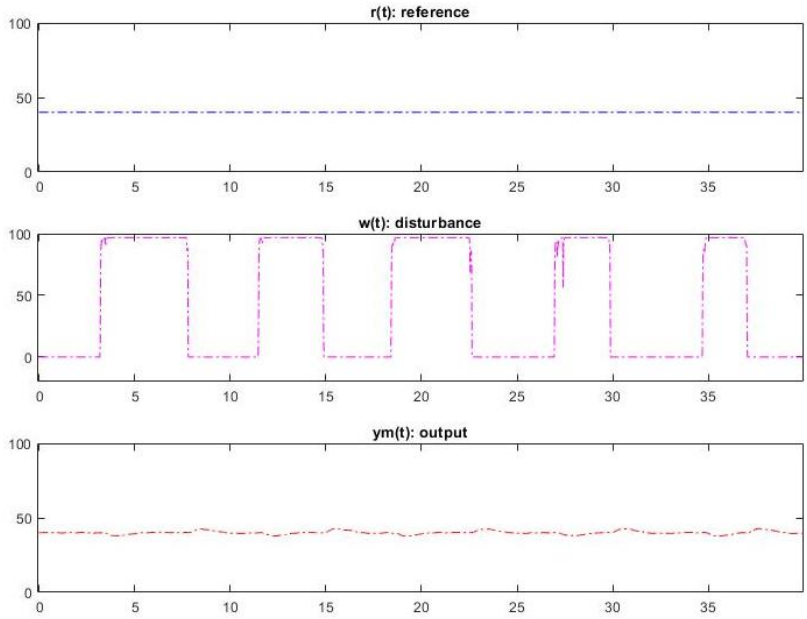
$$G_{FF}(s) = -\frac{K_W}{K_P} * \frac{1}{1 + sT_W} * \frac{1 + sT_e}{1}$$

Az függvény első tagja az erősítés, második tagja egytárolós tag ($\frac{1}{sT+1}$), harmadik tagja egy differenciáló tag (sT_D) és a bemenő jel összege. A kimenet kétszeres invertálása miatt az előjel változatlan marad, de ez érthető is, mivel a zavarás pozitív előjelű hatása esetén (fékerő növekedés) a beavatkozó jel növelésére lesz szükség, hogy a fordulatszám ne csökkenjen. Az alkalmazott átviteli függvény a következőképpen alakult:

$$G_{FF}(s) = -\frac{-0{,}096996}{3{,}5711} * \frac{1}{1 + 2{,}9133s} * \frac{1 + 10{,}608s}{1}$$

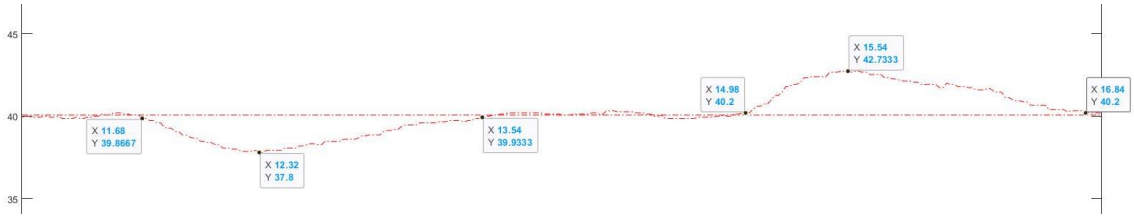
6.8 Előre csatolt zavarjel kompenzálás eredménye

Első lépésként végeztem egy mérést a Feed Forward kontroller működése nélkül, hogy legyen összehasonlítási alapom. A mérés során a zavaró hatás kb. 0-100% és 100-0% közötti ugrásszerű változásának hatására létrejövő szabályozott jellemzőben bekövetkező különbségek kerültek vizsgálatra.



54. ábra – zavarás hatása a kimenetre 40% alapjel mellett

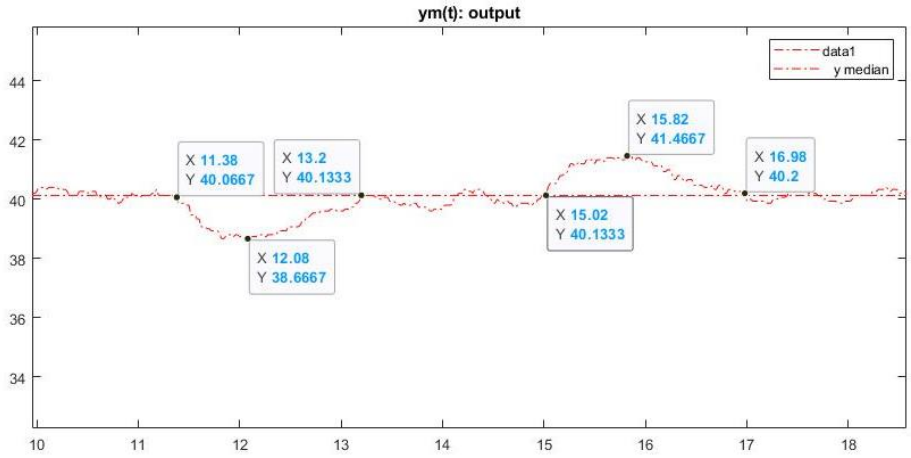
A grafikonokon látható, hogy változatlan alapjel mellett a zavarásnak milyen hatása van a szabályozott jellemzőre. A zavarjel mértéke a mérés során 97% nagyságú volt, amit nem változtattam a további mérések során (a labortápegység feszültségszabályozójának maximum 1% +10mV hibájától eltekintve). A 10 és 15 másodperc közötti zavarás ugrásszerű megjelenését és megszűnését alaposabban megvizsgálva,



55. ábra- zavarás hatása a visszacsatolt jelre (kinagyítva)

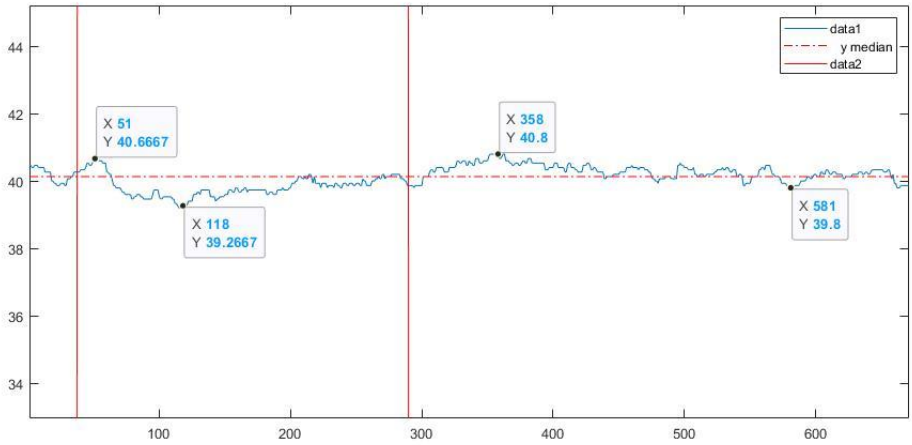
azt olvastam le, hogy **5,5-7%-os kitérés** keletkezett a zavaró hatás ráadása és elvétele után és a PID szabályozó integráló tagja **1,89 másodperc** alatt korrigálta az eltérést a referencia jeltől.

Ezután a TIA Portalban a létrehozott FF_controller blokk kimenő jelét a PID_Compact blokk „Disturbance” bementére kötöttem és engedélyeztem a HMI PID_Config képernyőjén az előre csatolt zavarjel kompenzálást. Így ismételtem meg a mérést.



56. ábra – zavarás és kompenzációjának hatása a ym jelre (kinagyítva)

Azt tapasztaltam, hogy zavarás hatására az eltérés maximális értéke lecsökkent **3,3-3,7%**-ra, az eltérés **1,82-1,96 másodperc** alatt megszűnt, ami az előzővel azonosnak mondható. A remélt javulás érdekében megpróbáltam növelni a Feed Forward kontroller D tagjának az időállandóját 10,608-ről 20 másodpercre és megismételtem a mérést még egy alkalommal.



57. ábra – FF működése
zavarás (vörös) és szabályozott jellemző (kék)

Már az előző esetben sem voltak nagyobbak az eltérések a zavarás hatására, mint a mérőeszköz ±3,5%-os hibája, de ott még különbséget lehetett tenni a zavarás hatása és a mérési pontatlanság okozta kilengés között míg itt már kevésbé lehet ezt megállapítani, mivel a **legnagyobb eltérés 2%**-os volt. A fentiek alapján arra a következtetésre jutottam,

hogy a zavarjel előre csatolása pozitív hatással volt a szabályozott jellemző zavarérzékenységére.

7 Fejlesztési lehetőségek

A megvalósított szoftveres eszközt megfelelően alkalmasnak találtam a PID_Compact blokk részletes ismertetéséhez, mert annak konfigurálása jól bemutatható a létrehozott PLC és HMI program alapján.

A szabályozás minősége hagy még lehetőséget annak további fejlesztésére, a minőségi jellemzők javításával és linearizálással. Úgy vélem, hogy a Feed Forward kontroller paraméterezésének folyamatát fel lehetne gyorsítani egy külön HMI képernyő hozzáadásával, amin specifikusan megadhatók az alaptagok és időállandók, így a szakasztól függetlenítve azt. Ez viszont igényelné az ezt támogató PLC-n futó FF_controller fonkcióblokk fejlesztését is. A visszacsatolt jel minőségi jellemzőinek kijelzése a HMI képernyőjén további időt takaríthat meg a hangolási folyamat során.

8 Összefoglalás

Az első ránézésre egyszerűnek tűnő szakasz vizsgálata során hamar igazolást nyert, hogy izgalmas kihívásokat rejt a megvalósítási folyamat. Már a mérés kialakítása sem bizonyult egyszerű feladatnak, mely során kompromisszumot kellett kötnöm (vagy alacsony felbontás, vagy hosszú mintavételezési idő, vagy változó mintavételi gyakoriság). A PID_Compact blokk szoftveres környezetének kialakítása és működésének feltérképezése meglepően időigényes folyamat volt.

A beszámolóban vannak fejezetek, amelyeknek egy kezelési útmutatóra hasonlíthat a szerkezete, de ez nem véletlen, ugyanis a demonstrációs eszköz megvalósításához szükségesnek tartom a működésének és a használatának részletes bemutatását.

Véleményem szerint az egyszerűen használható automatikus PID blokk kognitív erőforrást és időbeli kereteket szabadíthat fel hallgatóknak és tanároknak egyaránt, így a szabályozás javításának más részeire, például a szakasz linearizálási folyamatára, vagy előre csatolt kompenzálás kialakítására fordíthatnak nagyobb figyelmet. Akár arra is alkalmas az eszköz, hogy összevessük az automatikus hangolás eredményeként kapott és az általunk számított PID paramétereket a minőségi jellemzők tekintetében.

A bevezetésben megfogalmazott célokat megvalósítottam azzal, hogy a fordulatszám szabályozáson keresztül a gyártó szabályozójának minden lényeges funkciója bemutatható és kipróbálható a dolgozatom alapján.

9 Summary

An inspection of what at first glance appears to be a simple process quickly confirmed that the implementation procedure is an exciting challenge. Even the design of the measurement proved that it won't be an easy task because I had to make trade-offs (either low resolution, or long sampling time, or variable sampling frequency). Designing the software environment and mapping the operation of the PID_Compact block was a surprisingly time-consuming process.

There are chapters in the report that may resemble the structure of an instruction manual, because I think it is necessary to describe its operation and use in detail to implement the demonstration tool.

In my opinion, the easy-to-use automatic PID block can free up cognitive resources and time for students and teachers, so that they can devote more attention to other parts of the control improvement, such as the phase linearization process or the design of Feed Forward compensation. It can even be used to compare the PID parameters obtained as a result of automatic tuning and the manually calculated ones in terms of quality characteristics.

I have achieved the objectives set out in the introduction by demonstrating and testing all the essential functions of the manufacturer's PID controller through the motor speed control based on my study.

10 Irodalomjegyzék

[1]:https://mall.industry.siemens.com/mall/hu/hu/Catalog/DatasheetDownload?downloadUrl=teddatasheet%2F%3Fformat%3DPDF%26caller%3DMall%26mlfbs%3D6ES7214-1AG40-0XB0%26language%3Den – 2022. május 08.

[2]:https://hu.wikipedia.org/wiki/Aszinkron_g%C3%A9p – 2022. május 08.

[3]:https://www.magyar-elektronika.hu/images/stories/downloads/Telkes/me_2010_1-2_telkes_6.pdf – 2022. május 08.

[4]:Beleznai Károlyné - Automatika elemek katalógusa, Közgazdasági és Jogi Könyvkiadó, Budapest, 1967. – 472. oldal

[5]:https://www.sciencedirect.com/science/article/pii/S0307904X15004114 – 2022. május 09.

[6]:https://www.econengineering.com/konferencia/2018/Orvenyaramu_fek_szimulacioja_ANSYS_Maxwell_segitsegevel_BME_Gep_es_Termektervezes_Tanszek_Juhasz_Roland.pdf - 4.dia, 2022. május 09.

[7]:https://www.sciencedirect.com/science/article/pii/S0307904X15004114 - Fig.7, 2022. május 09.

[8]:https://authors.library.caltech.edu/81101/1/06436528.pdf - Figure 6, 2022. május 09.

[9]:https://mall.industry.siemens.com/mall/en/ww/Catalog/DatasheetDownload?downloadUrl=teddatasheet%2F%3Fformat%3DPDF%26caller%3DMall%26mlfbs%3D6AV2123-2DB03-0AX0%26language%3Den – 2022. május 10.

[10]: Owon HDS242S kézi oszcilloszkóp Adatlap:https://www.tme.eu/Document/780f26acde7c4d71fa2fdbacdc512155/HDS200.pdf – 2022. május 14.

[11]: Varga Árpád - Projekt II +FeedForward III resz (online tananyag) https://www.youtube.com/watch?v=04z3R9uukEc – 2022. május 14.

[12]:https://cache.industry.siemens.com/dl/files/036/108210036/att_74025/v1/s71500_pid_control_function_manual_enUS_en-US.pdf - 2022. május 06.

[13]:https://cache.industry.siemens.com/dl/files/036/108210036/att_74025/v1/s71500_pid_control_function_manual_enUS_en-US.pdf - 249.oldal, 2022. május 06.

[14]:https://cache.industry.siemens.com/dl/files/036/108210036/att_74025/v1/s71500_pid_control_function_manual_enUS_en-US.pdf - 249.oldal, 2022. május 06.