ÓBUDAI EGYETEM

Kandó Kálmán Villamosmérnöki Kar
Automatika Intézet

SZAKDOLGOZAT

OE-KVK
2021.

Markó Bálint
EBBD3K
T007627/FI12904/K

ÓBUDAI EGYETEM

Kandó Kálmán Villamosmérnöki Kar
Automatika Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve:	Markó Bálint	**Szakdolgozat száma:**	1
Törzskönyvi száma:	T007627/FI12904/K	**Neptun kódja:**	EBBD3K
Szak, specializáció:	villamosmérnök, automatizálás, programozható irányítások		

A dolgozat címe: Mérőeszköz és PLC kommunikációjának programozása

A dolgozat címe angolul: Programming of measuring system and PLC communication

A feladat részletezése:

Az iparban számtalan helyen használnak gyártás közbeni ellenőrzésre lézeres távolságmérő eszközöket, amely alkalmas beépített alkatrész helyes pozíciójának ellenőrzésére. A mérést nem elég csak elvégezni, de dokumentálni is kell a mért eredményeket, minőség nyomon követhetőség miatt. Tervezze meg és készítse el egy ilyen eszköznek a kommunikációs és vezérlő programját. Készítsen egy program blokkot, amely összeállít egy adatstruktúrát, amelyet elküld egy adatbázis szervernek. A feladat megvalósításához használjon Siemens S7-1511C típusú PLC-t a vezérlés elvégzéséhez és használjon Keyence IX mérőeszközt a mérési feladat elvégzéséhez. A kommunikációs blokkot készítse el SCL programnyelven.

Intézményi konzulens neve:	Lamár Krisztián
Külső konzulens neve:	Tamás-Kiss Ádám
Munkahelye:	Flextronics International Kft.
A kiadott téma elévülési határideje:	2 év
Beadási határidő:	2021. május 15.
A záróvizsga tárgyai:	Programozható irányítások Járműelektronika

A szakdolgozat titkos / nem titkos.

Kiadva: Budapest, 2021. március 31. PH ……
Intézetigazgató

A szakdolgozatot beadásra alkalmasnak találom:

20……. hó …… nap 20…….hó …….nap

…….. ……
belső konzulens külső konzulens

ÓBUDAI EGYETEM

Kandó Kálmán Villamosmérnöki Kar
Automatika Intézet

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, 2021. ..05..13.....

.....
Hallgató aláírása

TARTALOMJEGYZÉK

1. BEVEZETÉS

Napjainkban az ipari környezetben nélkülözhetetlen eszközökké váltak az ellenőrző berendezések. Ezért választottam szakdolgozatom témájául egy ilyen eszköz vezérlésének a programozását.

Egy ellenőrző berendezés programjának elkészítése megköveteli a logikus gondolkodást, a mérnöki precizitást, illetve a villamosmérnöki és informatikus szakmák számos tudományágának ismeretét. A program elkészítéséhez meg kellett ismerkednem az ipari berendezésekben használatos építőelemekkel, az egyes részegységek működésével, valamint a programozás tanórán már megszerzett tudásomat is szükséges volt bővítenem. Az ellenőrzés által a gyártás során keletkezett hibákról kapunk visszacsatolást. Ezeket a hibákat valós időben képesek vagyunk detektálni, ezáltal csökkentve a selejt és a termelési költségeket.

Az első fejezetekben bemutatom az iparban használatos eszközöket és azok felépítéseit. A következő részben ismertetek egy ipari kommunikációs buszt. Ezek után részletezem egy PLC és egy külső eszköz kommunikációjának elkészítését, illetve azt, hogy az ellenőrző eszköz vezérlőprogramjának megírása milyen módon valósítható meg. Az iparban nem csak az ellenőrzés fontos, hanem a kiértékelés eredményének letárolása is. Ezért leírom, milyen módon tudunk gyártási adatokat menteni adatbázisba. Bemutatásra kerül még az ellenőrző program elkészítésének módja és az aktuális mérési eredmények kijelzése a felhasználónak.

A gyakorlatban egy ilyen ellenőrző berendezés elkészítését egy csapat készíti. Külön ember végzi el a mechanikus tervezést és szerelést, külön ember az elektromos tervezést, majd egy harmadik ember a programozást. Egy ilyen gép legyártása tehát egy csapatmunka, amihez nélkülözhetetlen a sokrétű ipari ismeret. A munkahelyemen van egy belső automatizálási csoport, amelyben én a gépek programozásáért vagyok felelős és ha szükséges villamos terveket is készítek.

2. PROGRAMOZHATÓ LOGIKAI VEZÉRLŐ

2.1. Történeti áttekintés

A mai modern iparban az irányítástechnika már elképzelhetetlen korszerű félvezetőn alapuló vezérlések nélkül. Ezek közül a legelterjedtebb a PLC (Programmable Logic Control), magyarul programozható logikai vezérlő.

Az első PLC-k az 1960-as években jelenek meg Amerikában. A General Motors készített egy pályázatot egy olyan vezérlő egységre, amely ötvözi a számítógépes, relés és félvezetős technológiát.
Ezzel a megoldással ki tudták váltani a huzalozott vezérlést. Az effajta vezérlés felépíthető pneumatikus, hidraulikus és elektromechanikus elemekből. Azonban meg lehet valósítani digitális kapukból is (ÉS,VAGY,NEM). Ezeknek a rendszereknek a működését a huzalozás határozza meg. Vezérlések bővítése, átalakítása nehézkes, mivel a kapcsolást újra kell huzalozni. A huzalozott vezérlés relékből, végállásérzékelőkből, szelepekből és egyéb elemekből álló logikai vázlat vagy áram-útterv alapján készül. A bemeneteken történő változás azonnal megjelenik a kimeneten. Ilyenkor beszélünk párhuzamos jelfeldolgozásról. PLC-s vezérlés soros feldolgozású, vagyis a vezérlőutasítások egymás után program soronként hajtódnak végre. Ha nagyobb terjedelmű a program, a lefutás lassabb lehet, mint a huzalozott vezérlésé. Változtatás esetén nem kell a vezérlést megbontani, elegendő a PLC-ben tárolt programot módosítani. Könnyebb megoldani az üzem közbeni paraméter állítást, visszajelzéseket kapunk diagnosztikához. Sokkal bonyolultabb vezérlések hozhatók létre ezzel a módszerrel.

Néhány ismertebb gyártó a teljesség igénye nélkül:

- Omron,
- Siemens,
- Rockwell(Allen-Bradley),
- Festo,
- Mitsubishi,
- Beckhoff.

Az első PLC-knek még nem volt szabványosított programozás nyelvük. A legtöbb PLC még lyukkártyás módszerrel volt programozható. A lyukkártyán kívül még az assembly és a gépi kód volt fontos. A számítógépek fejlődésével fejlődtek a PLC-k is. A 70-es években kezdett megjelenni a létradiagramos programozási nyelv, amelyet napjainkban is előszeretettel használnak.

A 80-as években továbbfejlődött az iparág, igy nem csak diszkrét jeleket volt képes kezelni a PLC, hanem más időben folytonost, analógot is. Ezeket a jeleket fel lehetett használni sebességmérésre, hőmérsékletmérésre vagy akár szintmérésre. Ez a fajta fejlődés már nagyobb ugrás volt a PLC-k előtti vezérlésekhez képest, mivel ezeket a jeleket már lehetett digitalizálni (0-10V, 4-20mA). Ezeket az értékeket már nem lehetett egy biten ábrázolni, ezáltal megjelentek a különböző adatábrázolási formák is. A jelfeldolgozás már igényelte a nagyobb hardvert, így a vezérlők is tovább fejlődtek.

A hardver növekedésével lehetőség nyílt mért adatokat tárolni a PLC memóriájában. Ezeket az adatokat a statisztika és termelési hatékonyság növelése érdekében egyre gyakrabban kellett kiolvasni. Ez idézte elő az igényt a portok és protokollok használatára. Ezen modulok megjelenése következtében került a PLC-re a mai napig használatos soros kommunikációs port.

A gyártási kapacitás és sebesség növekedése érdekében egyre több dolgot kellett automatizálni. A részegységek összehangolása és a teljes automatikus berendezések felügyelete egyre nagyobb számítási igényt emésztett fel. Mérföldkövet jelentettek a 90-es években megjelent nagysebességű processzorok, és a magas szintű matematikai funkciókkal és adatkapcsolati hálózattal bővült vezérlők.

A vezérlések szekrényei és a PLC-k is egyre kisebbek lettek. A kábelezés gazdaságossá tételéhez egyre több kommunikációs lehetőség jelent meg. Így a távoli érzékelők jelei is kevesebb kábelezéssel a vezérlőbe kerültek.

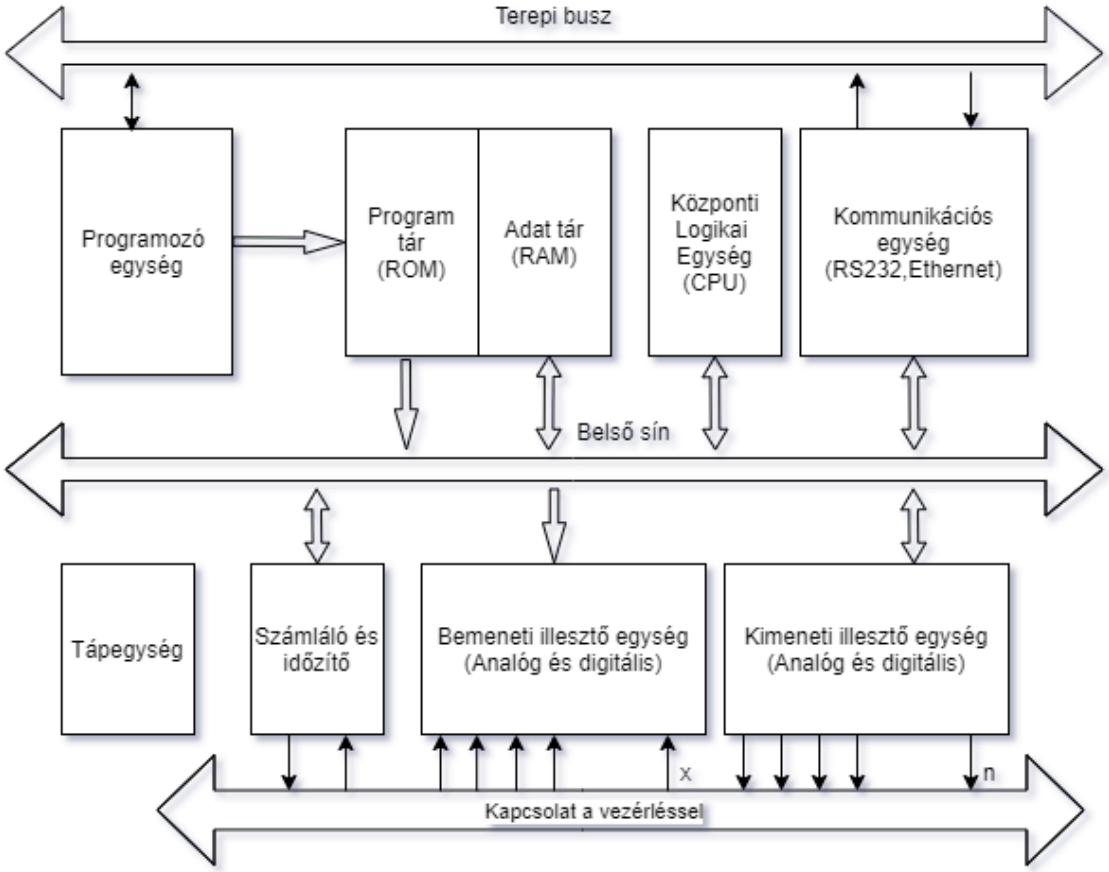
A PLC-k használata már túllépett az ipari használaton. Napjainkban már ott van az épület és közmű automatizálásban, közlekedés automatizálásban. A modern vezérlésekhez már helyi megjelenítés is társul. Ezen nyomon lehet követni a gép állapotát, és be lehet avatkozni a gép működésébe. Módosítható róla akár egy motor fordulatszáma, vagy akár hőmérséklet változtatás is lehetséges. Ma már nem csak egyszerű vezérlésekről beszélünk, hanem komplex automatizált rendszerekről. Ezek a rendszerek egy nagy hálózat alá tartoznak az I/O kezeléstől az adattárolásig.

2.2. PLC-k felépítése

Megkülönböztetünk kompakt és moduláris PLC-ket. A kompakt PLC-n található be- és kimeneti modul az, amely bővíthető. A moduláris változatán nincs be- és kimeneti modul, de tudjuk bővíteni.

Főbb egységei (1.ábra).

- Tápegység
- Számláló és időzítő
- Bemeneti illesztő egység
- Kimeneti illesztő egység
- Kommunikációs egység
- Központi logikai egység (CPU)
- Memória
    - programmemória (EPROM,FlashROM)
    - adatmemória (RAM)



1. ábra PLC felépítési vázlata

A CPU feladata a rajta lévő vezérlő program valós időben való futtatása, ami a programtárban helyezkedik el. Ciklikusan történik meg a program futtatása, egy ciklus pár 10 ms alatt fut le. Beolvassa a bemeneti egységen lévő állapot jeleket, és lépésről-lépésre végigfut a programozó által megírt programon, majd kiírja az eredményeket a kimeneti egységbe. A beolvasott és kiírandó adatok adatmemóriába kerülnek be. A ciklus futása alatt a bemeneteken lévő állapotváltozásokat a PLC ilyenkor nem veszi figyelembe, csakis az elején. A CPU-nak továbbá feladata még a futásához szükséges jelek előállítása. A működésért a program memóriában található PLC „operációs rendszere” felel. Ezt a gyártó tölti fel, és módosítása nem szükséges, így ez EPROM vagy ROM típusú memóriában helyezkedik el. A program memóriában találjuk ezen kívül még a felhasználó programját. A felhasználói program feltöltésére kapcsolatot kell biztosítanunk a memória és a felhasználó között. Ezt manapság egy laptoppal oldhatjuk meg, ami valamilyen interfészen keresztül kapcsolódik hozzá. Régen direkt a programozásra voltak külön eszközök kifejlesztve. Ha letöltöttük és teszteltük a

programunkat, akkor a programozó interfészünket el is távolíthatjuk a PLC-től. A letöltött programunk FlashROM-ba vagy MMC kártyára töltődik.
A be- és kimeneti illesztő egységeken keresztül csatlakozik a vezérlés a PLC-hez. Minden csatornának külön meghatározott címe van, ami alapján eléri a CPU. A korszerű PLC-k túlnyomó részén található valamilyen kommunikációs port, aminek a segítségével össze tudjuk kapcsolni másik eszközökkel, akár ipari buszon vagy Etherneten keresztül.

2.3. PLC-k programozása

A PLC-k programozási nyelvét az IEC 61131-3:2003 vagy az MSZ-EN 61131-3:2003 határozza meg. Ezeket a programozási nyelveket az (1. táblázat) láthatjuk.

Leírás	Rövidítés	Megjelenítés	Megjegyzés
Utasításlista	IL	karakteres	Egyszerű utasításokat, címkéket, konstansokat tartalmaz (Assembly)
Strukturált szöveg	ST	karakteres	Hasonlít a magas szintű programozási nyelvekhez (C, Basic)
Létradiagram	LD	grafikus	Egy 90° elfordított kapcsolási rajz
Sorrendi folyamatábra	SFC	grafikus	Összetett folyamatábra (Grafcet)
Funkcióblokk diagram	FBD	grafikus	Logikai szimbólumokat tartalmaz

1. táblázat Programozási nyelvek

2.3.1. Utasításlista

Utasításlistás programozásnál a parancsokat előírt formában soronként kell megadni. A programot be lehet vinni számítógépről, vagy programozó konzol segítségével. Az utasításkészlet nagyban függ a PLC processzorától. Az utasítások a PLC memóriájának egy-egy részén helyezkedtek el. Így kialakultak 8, 16 és 32 utasításos PLC-k.

Az utasítás egy műveleti kódból és egy operandusból épül fel.

	Műveleti kód	Operandus	
Példa:	A	I	0.0
	AN	M	10.0
	S	Q	0.0

A műveleti kód azt jelzi, hogy a processzornak milyen műveletet kell elvégeznie. A parancsokat általában az utasítás nevének rövidítésével jelöljük.

Az operandus azt mutatja meg, hogy melyik adat területtel kell elvégezni a műveletet. Ezeket megadhatjuk abszolút és szimbolikus címzéssel is.

Az utasítások programmá szervezése szabályokhoz van kötve, ezeknek a szabályoknak egy részét az adott PLC típusa határozza meg, de vannak általános szabályok is.

Néhány ilyen szabály:

- Lehetőleg kerüljük a program hurkokat,
- A ciklus végrehajtásához biztosítani kell a kezdőcímre valló visszatérést,
- Szubrutinból a főprogramba a visszatérést biztosítani kell,
- Az egymásba ágyazott szubrutinok hívását a PLC korlátozhatja.

Korábban az utasítások kizárólag bitműveleteket tudtak végrehajtani, ám manapság már a bájt- és szóműveletek is természetesek. A programozó létrehozhat saját blokkokat, de használhatja a fejlesztő szoftverbe beépítetteket is.

Ezt a programozási nyelvet azoknak tudnám ajánlani, akik rendelkeznek valamilyen assembly ismerettel.

2.3.2. Strukturált szöveg

A strukturált szöveg használata leginkább valamelyik magasabb szintű nyelvhez hasonlít (C, Pascal, Basic). Fejlesztésekhez kiváló, illetve olyan folyamatokhoz, amihez nincsen létra jel vagy funkcióblokk jel. A szabvány meghatározza az adattípusokat. A 2. táblázatban ezeket mutatnám be.

Adattípus	Jelentés
INT	Egész szám (-32768-tól +32767-ig)
UDINT	Előjel nélküli dupla egész (0-2^{32}-1
REAL	Valós vagy lebegőpontos szám
DATE	Naptári dátum
STRING	Karakterekből álló információ
F-EDGE	Felfutó él

2. táblázat Adattípusok

A szabvány megengedi a RAM közvetlen címzését is, amit a % jellel tudunk megtenni. A memória általában három részre van felosztva: a kimenetekhez rendelt RAM (Q), a bemenetekhez rendelt (I) és a belső memóriához rendelt RAM (M). A memóriába írhatunk bites adatokat (X), bájtosat (B), szavasat (W) vagy duplaszó hosszúságút (D). Meghatározunk még globális és lokális változókat. A globális változókat a programhoz készítjük, amíg a lokális változókat egy-egy funkcióblokkhoz.

Adatok deklarálása egy funkció blokkhoz.

```
VAR_INPUT
(Kimeneti változók deklarálása)
END_VAR
VAR_OUTPUT
(Kimeneti változók deklarálása)
END_VAR
VAR
(Ide kerülnek a belsőleg használni kívánt változóink)
END_VAR
```

3. táblázat IEC szimbólumok

Kulcsszó	**Leírás**
ACTION/END – ACTION	Akció kezdetének (start) és végének (end) deklarálása. Használt rövidítések: N nem változik S set (beírás) R reset (törlés) L korlátozott idejű akció D késleltetett akció P impulzus jellegű akció SD tárolt és késletetett akció DS késleltetett és tárolt akció SL korlátozott ideig tárolt akció
ARRAY OF	Adattípus deklarálása
AT	Memóriahely hozzárendelése változóhoz. Például CRI AT %QX400
CASE OF/ELSE/END_CASE	CASE típusú struktúra deklarálása
CONFIGURATION/END_CONFIGURATION	Konfigurációs fájl kezdetének és végének definiálása
CONSTANT	Konstans értékek deklarálása. Például VAR CONSTANT PI: REAL:=3,14 END_VAR
EN/END	Végrehajtás engedélyezése létradiagramon
EXIT	Kilépés a hurokból
FOR/TO/BY/DO/END_FOR	Ciklusszervezés definiálása
FUNCTION/END_FUNCTION	A függvényfunkció kezdetének és végének deklarálása. Például Y:= SIN(X)*SIN(X)

FUNCTION_BLOCK/END_FUNCTION_BLOCK	Funkcióblokk kezdetének és végének deklarálása.
IF/THEN/ELSIF/ELSE/END_IF	IF/THEN/ELSE struktúra definiálása
INITIAL_STEP/END_STEP	Egy kezdeti lépés definiálása
PROGRAM/END/_PROGRAM	Egy program kezdetének és végének deklarálása
REPEAT/UNTIL_END_REPEAT	Ismétlési funkció definiálása
RETAIN	Egy változó értékének definiálása a PLC tápfeszültségének hibája után (power interrupt) VAR_OT RETIN Status:INT END_VAR
RETURN	Funkcióblokk (szubrutin) vége deklarálása
STEP/END STEP	Lépés kezdetének és végének deklarálása
TASK	Taszk deklarálása
TRANSITION/FROM/TO/END_TRANSITION	Átmenet a kezdetének és végének deklarálása
TYPE/END_TYPE	Adattípus megváltozásának deklarálása
VAR/END_VAR	Belső változó beállításának deklarálása
VAR_INPUT/END_VAR	Bemeneti változó beállításának deklarálása
VAR_IN_OUT/END_VAR	Bemeneti és kimeneti változó beállításának deklarálása
VAR_OUTPUT/END_VAR	Kimeneti változó beállításának deklarálása
VAR_EXTERNAL/END_VAR	Külső változó beállításának deklarálása
VAR_GLOBAL/END_VAR	Globális változó beállításának deklarálása, amelyet az összes elemmel lehet címezni a programon belül

WHILE/DO/END_WHILE	WHILE (AMÍG) struktúra definiálása
WITH	Program-vagy funkcióblokk közötti kapcsolódáshoz használt kulcsszó

Az ilyen módon megírt program sorokat könnyedén át lehet másolni másik PLC típusba, ha megvannak a hordozhatóság feltételei a két fejlesztő környezet között. Gyakoribb utasításokat láthatjuk (3. táblázat).

Ennek a programozási nyelvnek az előnye a nagy rugalmasság, viszont használata magasszintű programozás ismereteket és gyakorlatot igényel.

2.3.3. Létradiagram

A létradiagram leinkább az áramút tervnek a PLC technikában alkalmazott egyszerűsített változta. Annyiban különböznek egymástól, hogy az áramutakat vízszintesen rendezi és más karakterekből épülnek fel a kapcsolások, amik jobban megfelelnek a számítógépes ábrázoláshoz.

A létradiagrammos programot számítógépen szerkesztjük meg grafikus módon. Ezek olyan módon épülnek fel, hogy bal oldalon egy referencia feszültségtől indulva jönnek az érintkezők, jobb oldalon pedig a kimenetek helyezkednek el. A program fő alkotóelemei a huzalozás, kontaktusok, időzítők, számlálók és tekercsek (4. táblázat). A létra programozásnál hasonló szabályok vannak, mint az áramútterv készítésénél.

4. táblázat Létradiagram főbb szimbólumai

IEC grafikus szimbólumok	Leírás
-| |-	Záró- (munka)- érintkező
-|/|-	Bontó- (nyugalmi) érintkező
-|P|-	0 ➔1 átmenetet adó érintkező
-|N|-	1 ➔0 átmenetet adó érintkező
-( )-	Tekercs
-(/)-	Negált működésű tekercs
-(S)-	RS FF beírótekercs
-(R)-	RS FF törlőtekercs
-(M)-	Tápfeszültség- kimaradásakor állapotát megtartó tekercs
-(SM)-	Tápfeszültség- kimaradásakor állapotát megőrző RS FF beírótekercs
-(RM)-	Tápfeszültség- kimaradásakor állapotát megőrző RS FF törlőtekercs
-(P)-	0➔1 élre működő (ON) tekercs
-(N)-	1➔0 élre működő (OFF) tekercs

Egy létradiagramos program elkészítése:

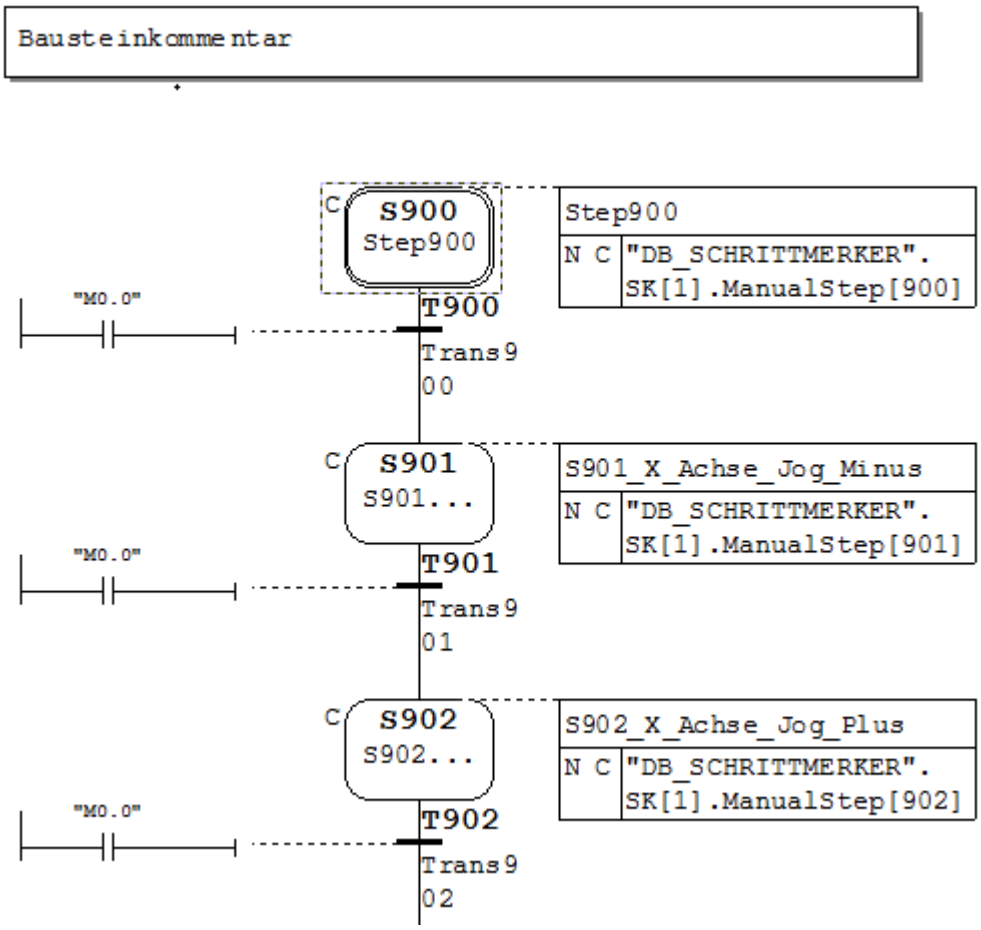
Először létre kell hozni egy Tag táblát, ahol elnevezzük a be- és kimeneteket. Miután ezt megtettük meg kell határoznunk a folyamatot. A folyamat meghatározása után már csak meg kell szerkesztenünk a programunkat és leszimulálni azt.

Ezt a programozási nyelvet a legegyszerűbb elsajátítani egy kezdő számára, de jó megoldás lehet egy érintkezős vezérlésben jártas szakember számára is.

2.3.4. Sorrendi folyamatábra

A programozni kívánt folyamatot lépésről lépésre grafikus módon írjuk le. Minden lépéshez (Step) tartozik valamilyen feltétel sorozat (Translations). A lépésekhez még kapcsolódik egy akció (Actions), amely akkor lép működésbe, amikor a feltétel teljesül, ennek a hatására lép előre a program is. A feltételeket és akciókat is külön kell programoznunk, amelyeket a négy másik nyelv valamelyikével kell megírnunk. Feltételeknek általában bemeneteket adunk meg, míg akciónak kimeneteket írunk be. Ezen a nyelven megírt programot mutatnám be (2. ábra). A programozási nyelv nagyon érzékeny a szintaktikára.

A folyamat végén a visszaugrásról a programozónak kell gondoskodnia. Ezt a nyelvet közvetlenül nem lehet betölteni a PLC-be, ezért ezt a segéd szoftver átfordítja valamelyik másik nyelvre.



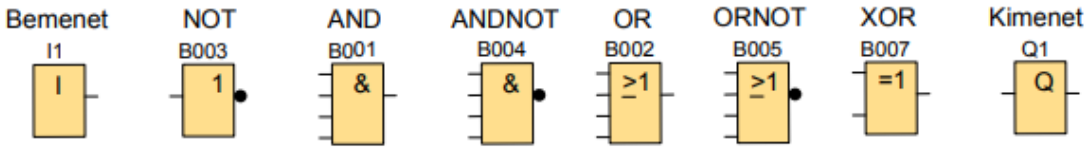
2. ábra Folyamatábra

A folyamatábrába tudunk programozni „ÉS” „VAGY” elágazásokat is, „VAGY” eseménynél megadható melyik feltételt vizsgálja meg a program legelőször.

2.3.5. Funkcióblokk diagram

A funkcióblokkos programozás a digitális technikából ismert kapukból alakult ki (3.ábra). A blokkokon be- és kimenetek találhatók. Bal oldalon a bemenetek, míg a jobb oldalon a kimenetek találhatóak. Ebből adódóan a jel az előző blokk kimenetétől a bemenet felé halad, azaz balról jobbra.

Néhány alapjel:



3. ábra Funkcióblokk főbb parancsai

Általában a funkcióblokkos programozás és a létradiagramos programozás átváltható egyikből a másikba. A szintaktikai szabályok a huzalozott, feszültséglogikájú hálózat hardver kialakítási szabályaival egyeznek meg pár kivétellel. Ezt a programozási nyelvet annak lehet ajánlani, aki jártas a feszültség logikájú hálózat tervezésben.[1][5][6]

3. GÉPI LÁTÁS AZ IPARBAN

A gépi látás a XXI. században kiemelt fontosságú a nagy méretű automatizálásnak köszönhetően. Azonban a digitális képfeldolgozást nem csak az iparban használják, de az űrkutatásnál és az orvostudománynál is. Az iparban számunkra a felhasználás igen sokrétű. Alkatrészek válogatásához, minőségellenőrzéshez vagy akár robot pozíció meghatározásához is használható.

Általában három fő részre lehet felosztani a gépi képfeldolgozást. Az első lépésben a kép előfeldolgozása történik meg. Itt még a bemenet és a kimenet is egy-egy kép. Ezután következik a szegmentálás és a kiemelés. A bemenetünk is egy kép, de a kimenet már olyan tulajdonságok alkotják, mint az objektumok, élek és kontúrok. Az utolsó lépés már az objektum felismerés.[7][11]

3.1. Fizikai felépítés

Általánosságban egy gépi látás öt fő komponensből tevődik össze:

- Jelforrás, pl. megvilágítás,
- Képalkotó egység
- Jelfeldolgozó
- Szoftverkomponens
- Kommunikációs interfész

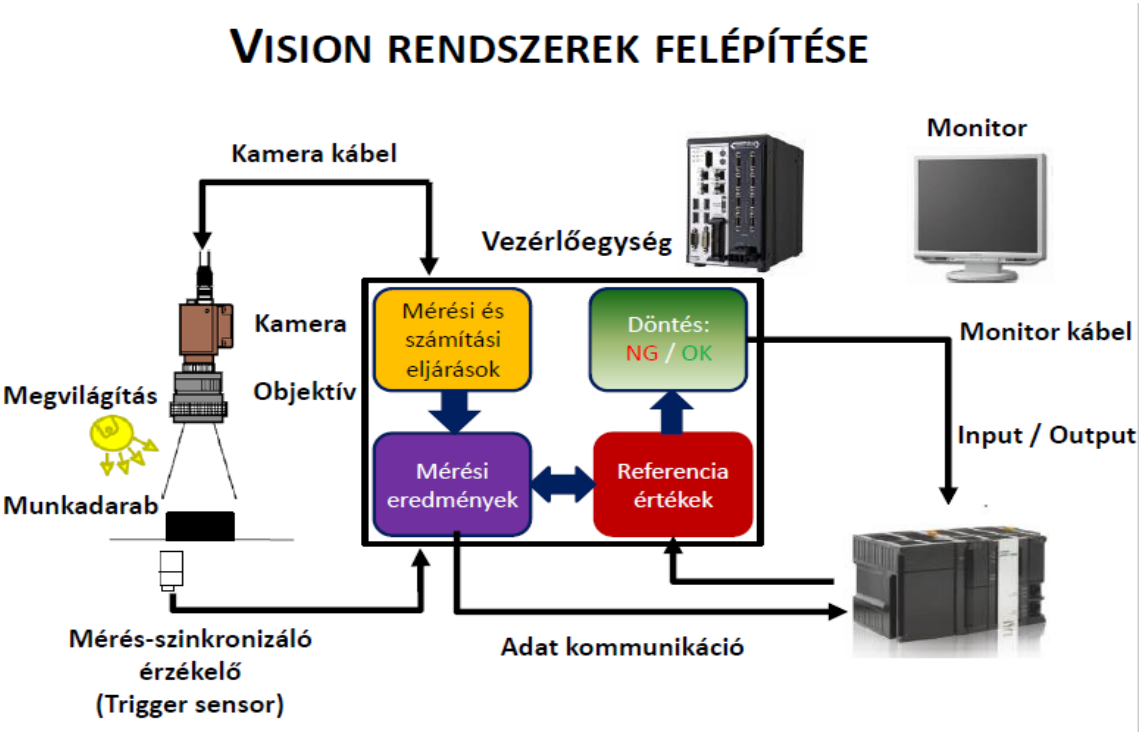
A gépi látás szempontjából az egyik legfontosabb dolog a megvilágítás. Ezáltal nem csak a természetes fényre tudunk hagyatkozni a lényeges információk kiemeléséhez, de a megfelelő információ eléréséhez elengedhetetlen egy jó megvilágítás.

Képalkotó berendezés olyan eszköz, amellyel a mért adatokból képesek vagyunk képet vagy kép sorozatot előállítani. A képalkotó rendszerek igen különbözőek, mivel sokféle fizikai jellemzőből lehet képet előállítani.

A jelfeldolgozó egység a hardver és a szoftver összesége, amely elvégzi a feldolgozási és elemzési folyamatot.

A szoftverkomponens az összegyűjtött jeleket képes értelmezni és döntéseket hozni. Ezekre a programokra nincsen semmilyen megkötése sem. A gyártó dönti el, hogy biztosít-e előre megírt programsorokat, vagy a felhasználónak kell elkészítenie ezt.

A kommunikációs interfészen keresztül tudjuk vezérelni a képfeldolgozást, és a visszakapott döntéseket is itt kapjuk meg. A (4. ábra) láthatunk egy általános felépítést.



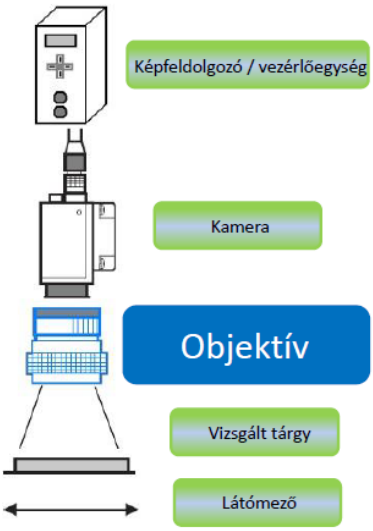
4. ábra Vision rendszer felépítése

3.2. Optika

Az objektív (5.ábra) feladata a vizsgálandó tárgy felől érkező fény átkonvertálása a szenzor által értékelhető formába. Ez csatolófelületként működik a tárgy felől érkező fény és a képalkotó között.

Az optika meghatározása:

- A látómező méretét
- A mélységélességet
- Az optikai torzítást
- A kép fényességét

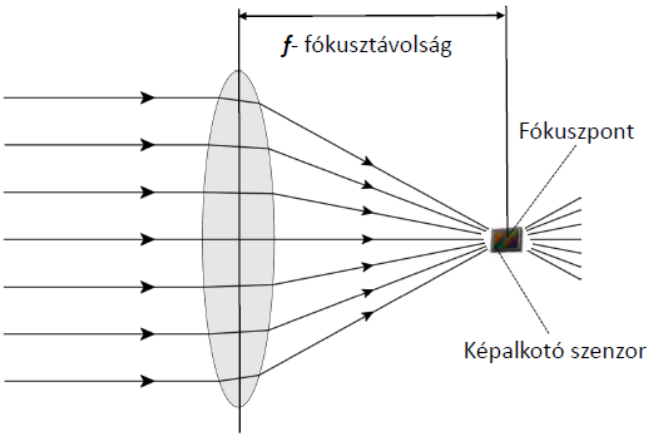


5. ábra Objektív

Az objektív nem csak egy lencséből áll, hanem több lencse helyezkedik el egy objektívben. A lencséket egymáshoz képest lehet változtatni, így a továbbított kép is változik. Az objektíven még lehet állítani apertúrát, amivel az objektíven áthaladó fény mennyiségét tudjuk változtatni. Ezen felül még lehet fókusztávolságot állítani, mivel nem képes teljes tartományban éles képet visszaadni. A csatlakozási lehetőségre többféle szabvány van, így lehet váltogatni egy-egy kamerán az objektíveket.

3.2.1. Fókusztávolság

Megadja a lencse és a fókuszpont távolságát, ez a méret látható (6.ábra).



6. ábra Fókusztávolság

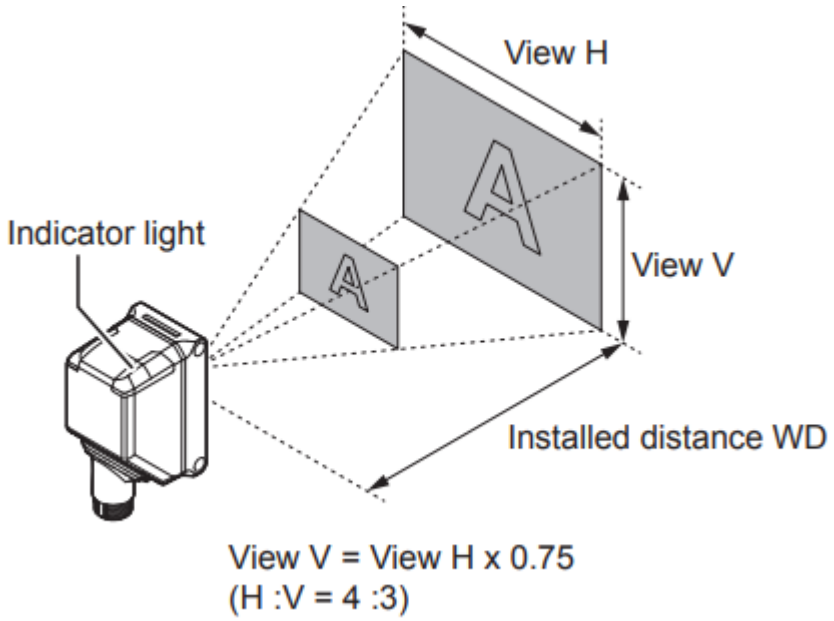
Kisebb méretű nagyítás esetén egy képen több elemvizsgálat végezhető el, de az elemek felbontása kisebb, így rosszabb a képminőség.
Nagyobb mértékű nagyítás esetén kevesebb elemet tudunk megvizsgálni egyszerre, de az elemek felbontása nagyobb.

3.2.2. Mélységélesség

A mélységélesség a kamera-tárgy tengelyen vett fókuszban lévő tartománya. A vizsgált tárgy előtt és mögött elhelyezkedő részt is magában foglalja. Nagyságát lehet befolyásolni például közgyűrűvel. Egy hosszabb közgyűrűvel kisebb lesz a mélységélesség.

3.2.3. Felbontás

A felbontás a képalkotó rendszertől és a látómezőtől függ (7.ábra). A pixelszám a pixelek maximálás száma egy képen. Ebből adódóan a pixel a kép legkisebb alkotóeleme. A felbontást a látómező és a pixelszám hányadosaként írhatjuk fel.
A látómező az a rész, amit a lencsén keresztül látni szeretnénk. A (5. táblázat) láthatjuk hogyan változik a kép tartománya a felszerelés távolságától.



7. ábra Felszerelési távolság

Felszerelési távolság WD(mm)	**Látómező szélessége H(mm)**	**Látómező magassága V (mm)**
40	8	6
50	11	8
100	22	16
150	34	24

5. táblázat Felszerelési távolság táblázat

3.3. Optikai szövegfelismerés OCR

Az OCR (Optical Character Recognition) egy olyan technológia, amely lehetővé teszi a szkennelt vagy digitálisan fotózott dokumentumok szerkeszthető és szövegesen kereshető formátum létrehozását. Automatizálásban az OCR technológiát az alkatrészekre vagy termékekre nyomtatott karakterek felismerésére, ellenőrzésére vagy meghatározására használjuk. Ilyen vizsgálat lehet egy élelmiszeres doboz lejáratának vizsgálata vagy akár egy motorblokk számának vizsgálata. A karakterláncok nélkülözhetetlenek az olyan információk megjelenítéséhez, mint a gyógyszerek lejárata vagy a gyártásban készült alkatrészek szerelési sorszáma. A karakterek felismerése sok problémát okozhat. Ha a megrendeléseket és a leltárt papír alapon kezelik, az emberi hibával járhat. Ezért az emberi hibák kiküszöbölése mellett arra használják a technológiát, hogy megbízhatóbbá tegyék a kezeléseket, csökkentsék a munkaerőt és a költségeket.

A karaktervizsgálat során a rögzített karaktereket összehasonlítják az adatbázisban szereplő karakterekkel és a legközelebbi alakú karakterként ismeri fel. A karaktereket összehasonlítja a referenciaként beállított karakterlánccal, aminek a végén egy döntés születik. Az ilyen típusú eljárás megakadályozza azokat a hibákat, amelyeket kézírással okozunk.

A karaktervizsgálatnak több típusa van, például a jelölés megléte, jelölés minősége, optikai karakter ellenőrzés és az optikai karakterfelismerés. A vizsgálat alapvető művelete a karakterek egyenkénti kinyerése a készített képből, hogy összehasonlíthassuk a regisztrált karakterekkel, és azonosítsuk őket.[9]

3.4. Lézeres távolságmérés

Az első, ember által készített látórendszerek a klasszikus fénykép készítését vették alapul. Ezeket intenzitásképnek nevezzük, tehát a kép fényessége attól függ, hogy mennyi fény jut vissza a lencsébe. A fényforrást és az érzékelőt is igen nehézkes modellezni, az objektumról pedig nem áll rendelkezésünkre információ. Ezért egy egyszerű képből nehéz meghatározni egy objektum térbeli elhelyezkedését és formáját. Ezért használnak manapság széles körben olyan eszközöket, amelyek közvetlenül alkalmasak távolság mérésre. A mérési távolságkép készítésének többféle megoldása van, ezeket a (6. táblázat)-ban mutatom be.[4]

6. táblázat Távolságkép módszerek

Módszer	**Felbontás**	**Korlátozó tényezők**	**Sebesség**	**Mozgó tárgyak mérése**	**Vizsgálható tárgyakra vonatkozó megkötések**
Pontonkénti letapogatás	Nagyon magas	Mechanika	Nagyon lassú	Nem lehetséges	Kevés
Fényes	Közepes	Mechanika és a kamera felbontása	Lassú	Korlátozásokkal	Kevés
Kódolt fény	Közepes	Megvilágítás és a kamera felbontása	Közepes	Nem lehetséges	Közepes
Fázistolás	Magas	Szürkeárnyalatok képzése és a kamera felbontása	Gyors	Nem lehetséges	Közepes
Színkódolt fázistolás	Magas (többértelműség előfordulhat)	Színek előállítása és a kamera felbontása	Nagyon gyors	Lehetséges	Nagyon szigorú (a tárgy felülete nem tartalmazhat lyukakat, árkokat)

Összetett fény	Alacsony	Megvilágítás és a kamera felbontása	Nagyon gyors	Lehetséges	Szigorú
Színkódolás	Közepes	Színek előállítása és a kamera felbontása	Nagyon gyors	Lehetséges	Szigorú (a tárgy felülete egyszínű kell legyen)

4. PROFINET KOMMUNIKÁCIÓ

A Profinet (IEC 61158 és IEC 61784[7]) egy nyílt ipari ethernet szabvány az automatizálás világában. Ez a kommunikációs forma a Profibus utódja, amely több mint 20 éve van jelen az ipari buszok piacán. A profinetnek köszönhetően biztonságos és gyors adatkapcsolat hozható létre a decentralizált terepi moduloktól kezdve a gyors válaszidőt igénylő motorhajtásokig. Lehetőségünk van a lassabb buszrendszerek kiváltására a meglévő rendszer lecserélése nélkül. Nagy előnye az osztott automatizált gyártási rendszereknél van, ahol ezáltal valós idejű adatokkal tudunk dolgozni.

Előnyei a rendszernek:

- szabványos csatlakozókat (RJ45/M12) és hálózati elemeket használ,
- támogatja majdnem az összes hálózati topológiát,
- hálózaton keresztüli diagnosztikai lehetőség,
- biztonságos kommunikáció,
- intelligens érzékelőket is könnyen tudunk csatlakoztatni,
- gyakorlatilag korlátlan állomásszám.

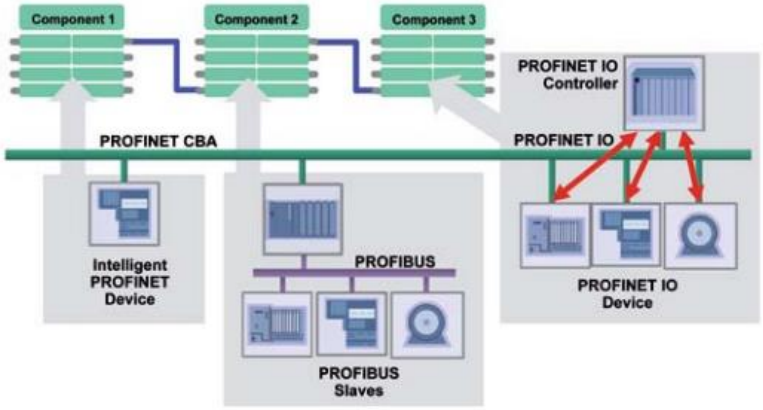
Három teljesítményszint áll a fejlesztők rendelkezésére:

- IRT(Isochronous Real Time): gyors és szinkronátvitelt igénylő rendszerhez(hajtások),
- RT(Real Time): valós idejű kommunikáció,
- TCP/IP: nem valós idejű átvitel.

Több profillal is rendelkezik a profinet, amely közül néhány speciális egységekhez és applikációkhoz használható fel.

4.1. Profinet CBA

Profinet CBA: Profinet Component Based Automation, magyarul Profinet komponens bázisú automatizálás.



8. ábra Profinet CBA

Ez a rendszer egy nem valós idejű kommunikációt valósít meg TCP/IP alapokra építve. A felépítését láthatjuk a (8. ábra). Erőssége a PLC-k közötti kommunikáció. Ez a rendszer a profibus technológiájánál használt FMS (Fieldbus Message Specification) technikára hajaz. A CBA rendszer lényege, hogy a különálló PLC-k egy nagy rendszert alkotnak, amelyek kommunikálnak egymással. A profinet komponenseket véges bemenettel lehet irányítani. Az alrendszerekben futnak a felhasználói programok, amelyek a közvetlen vezérlést látják el. A profinet CBA esetén 50-100 ms közötti kommunikációs időről beszélhetünk. Párhuzamosan, ezzel a technikával együtt viszont alkalmazhatunk profinet IO-is, ahol már milliszekundumos kommunikációs idő érhető el.[12]

4.2. Profinet IO

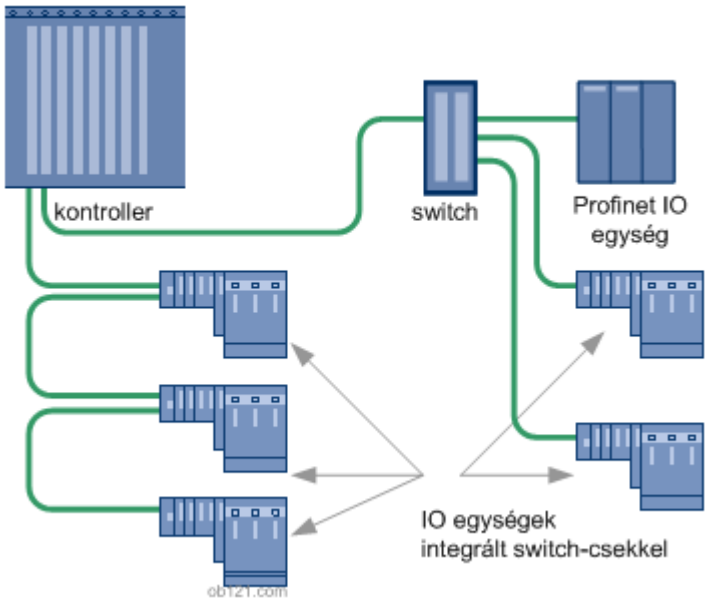
Profinet IO: Profinet Input / Output magyarul Profinet bemenet / kimenet.

A Profinet IO egy gyors, decentrális perifériákkal való adatcserére készített kommunikáció. A vezérlő és az egységek közötti teljes kommunikációt írja le a Profinet IO. Ez a rendszer megfelel a Profibus DP felépítésének, itt viszont a vezérlőket még Master-nek, amíg a csatlakozott egységeket Slave-nek nevezzük. A diagnosztika és paraméterezés itt is elérhető. A kommunikációs ciklusidő milliszekundumos tartományba esik bele az IO kommunikációnál.

A profinet IO hálózat a következő elemekből épül fel:

- IO Supervisor: egy PC-n futó szoftver, amelynek segítségével az eszközök paraméterezhetők és diagnosztizálhatók,
- IO Device: terepi egységek, amelyeket a IO vezérlő irányít. Ez tartalmazhat több egységet és sub-modulokat is,
- IO Controller: általában egy PLC, ami felügyeli a folyamatokat.

Az eszközök címzése MAC és IP címekkel történik. Ha előzetesen a MAC címhez társítunk egy szimbolikus nevet, akkor azt már használhatjuk Profinet címzésnél. A projekt letöltése után bekerülnek a kommunikációhoz nélkülözhetetlen adatok az IO contollerekbe. Az egységek csak switch-eken keresztül csatlakoztathatóak a hálózat más résztvevőihez. Ez megvalósítható csillag topológiába több portos switch-eken keresztül, vagy a terepi modulokba integrált két portos switch-en keresztül vonal topológiába. Mindkét megoldást láthatjuk (9. ábra).



9. ábra Profinet IO

A Profinet telegramokkal nem az egységet címzi meg, hanem annak valamelyik portját MAC cím alapján. Ezért egy egységnek nem egy MAC címe van, hanem három, mivel mindkét portnak van és magának az egységnek is van egy MAC címe. Az egységek között

pont-pont kapcsolatot hoz létre, ezért ha valamelyik egységek közötti kapcsolat megszűnik, az utána lévő egységekkel is megszűnik a kapcsolat.[8]

4.3. Profinet hálózat

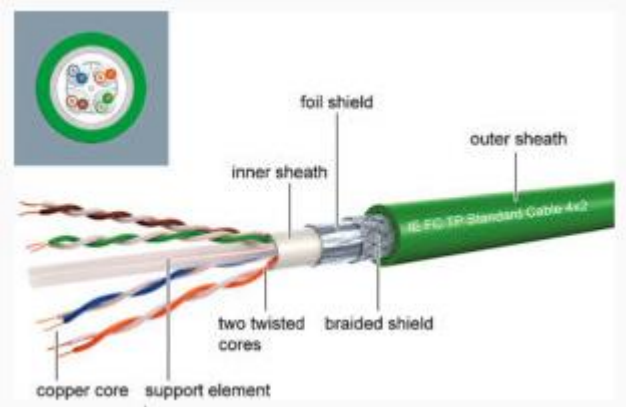
A Profinet szabványos Ethernet IEEE 802.3 fizikai rétegen alapul. Működik Ethernet kábellel, száloptika (FO), Power over Ethernet (PoE) kábellel és vezeték nélküli hálózaton is. Különböző körülmények között is teljesítenie kell a Profinet-nek. Irodai környezetben a standard Ethernet kábelek megfelelnek, de egy gyári területen már nem. Gyári területen több zavaró tényezővel is meg kell birkóznia. Egy Cat-5 kábel nem biztos, hogy használható egy irodai környezetben, de a standard Cat-5e kábel már igen. Az átviteli sebesség nagyban függ a kábel típusától. A (7.táblázat) -ban látható néhány kábel típus átviteli sebességekkel.

7. táblázat Kábel típusok

	Hossz (méter	**Átviteli sebesség 10 Mb/s**	**Átviteli sebesség 100 Mb/s**	**Átviteli sebesség 1 Gb/s**	**Átviteli sebesség 10 Gb/s**	**Teljesítmény Ethernet (POE)**
Cat-5	100	x	x			x
Cat-5e	100	x	x	x		x
Cat-6	100 55 10 Gb/s sebességgel	x	x	x	x	x
Cat-6a	100	x	x	x	x	x

A standard Ethernet kábelek sodrott érpár technikát használnak (10.ábra) a keresztbeszélgetések kiküszöbölésére. Termelési közegben a kábelnek több mechanikai behatásnak is ellen kell állnia. Azonban termelésben jobban kell védeni az

elektromágneses zavaroktól is (EMC), ezért itt plusz árnyékolási rétegek kerülnek rá a kábelre.[13]



10. ábra Kábel keresztmetszet

4.4. Diagnosztika

A terepi buszok és az ipari Ethernet protokollok és különösen a Profibus és Profinet egységek egyik nagy erőssége a diagnosztika. Ezek a diagnosztikai információk felgyorsíthatják a hibaelhárítást, és lecsökkentheti az állásidőt.

A Profibus és Profinet International (PI) létrehozta és támogatja a Profibus és Profinet szabványokat. Minkét szabvány megtalálható az IEC 61158-ban. A Profibus két protokollon alapuló rendszer. Az egyik az RS-485, a Profibus DP részére diszkrét alkalmazásokhoz, a másik az MBP a Profibus PA számára folyamatalkalmazásokhoz. A Profibus proxy segítségével integrálódik a Profinet-be. A Profibus és Profinet rendszer több módot használ a leállások elkerülésére. Ilyenek a riasztások, amiket ki lehet tenni egy HMI-re. Ezek a riasztások lehetnek

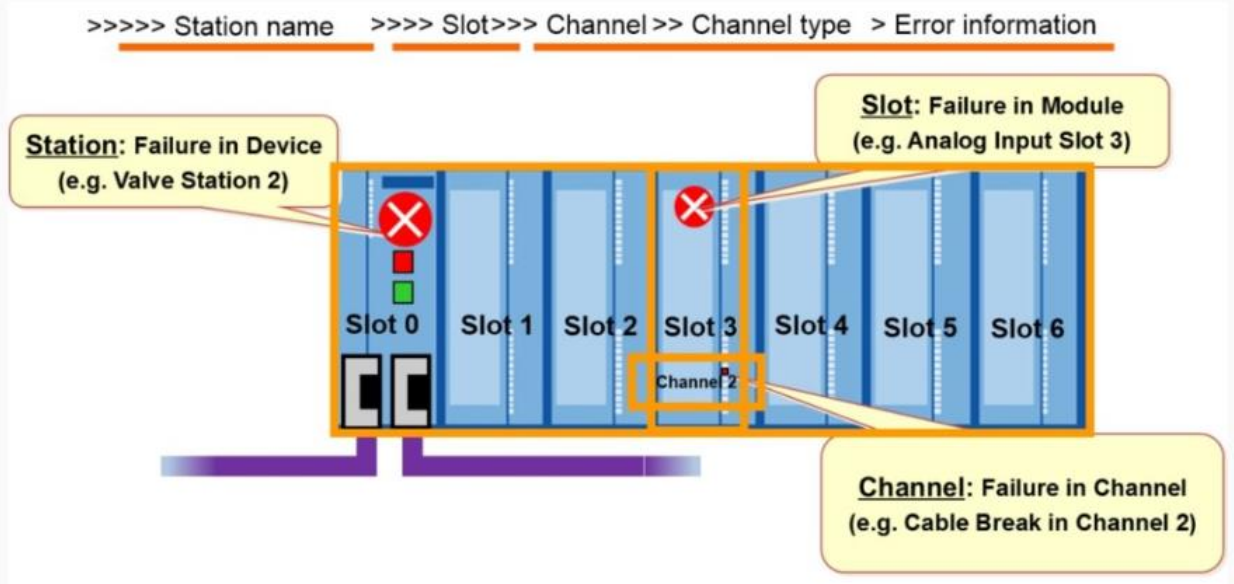
- vezérlőhibák (3. Rack kiesett)
- csatlakoztatott hardver hiba (Vezetékszakadás egy kimeneten)
- folyamat változó hiba (magas hőmérséklet)

A protokollok meghatározzák és támogatják a riasztásokat, és diagnosztikai információkat szolgáltatnak. Ezenkívül olyan információkat is szolgáltatnak, amelyek

felhasználhatók karbantartás előrejelzésére is. De az információk nem csak az eszközökről, hanem a hálózatról is a felhasználó rendelkezésére állnak.

Amikor az eszközt cserélni kell, Profinetnek az az elve, hogy minél egyszerűbb legyen az eszközcsere. Így egy eszköz cseréjekor csak a címét kell beállítani elektronikusan a Profinet-nél.

A Profinet esetében az első diagnosztikai pont a riasztás. Riasztást lehet generálni folyamatból vagy hardverből.



11. ábra Diagnosztika

A statikus fizikai réteg teszttől sokféle hibát képesek detektálni, ilyen például a törött kábel. Run-time módban képes meghatározni az összes aktív állomást a buszon, indítási problémát is felismer és statisztikát is képes gyűjteni a kieső modulokról. LLDP támogatásnak köszönhetően a vezérlő fel tudja ismerni milyen eszközök és milyen topológiában vannak hozzá kapcsolva. A (11. ábra) látható egy hiba diagnosztikája.

A Profinet rendelkezik a leginkább átfogó diagnosztikai csomaggal, amelyek az ipari Ethernek rendszere közül elérhető.[12]

5. PROGRAMOZÁS

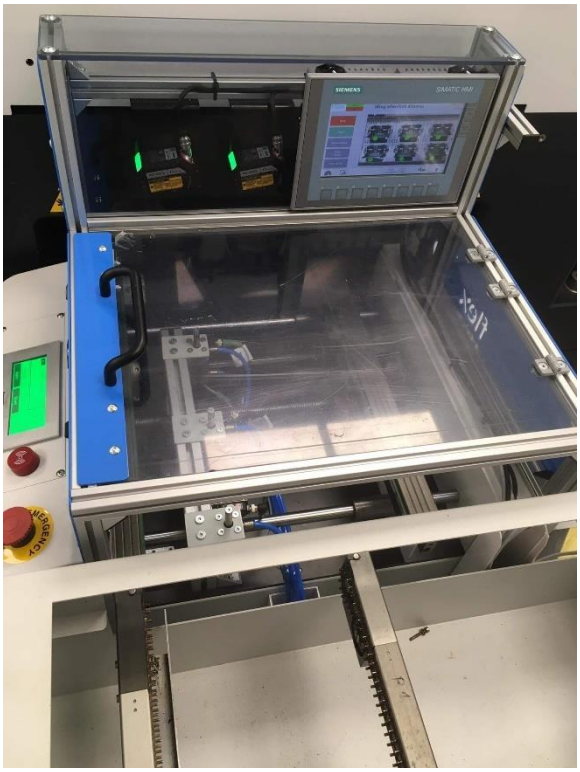
A vezérlés elkészítéséhez egy Siemens gyártmányú S7-1511C típusú PLC-t választottam. A választásom azért erre a vezérlőre esett, mivel ezzel a gyártóval van a legtöbb tapasztalatom, és nem találtam másik vezérlőt, amely ilyen felépítésű adatot tud küldeni az adatbázis felé, és a megrendelő cég el is fogadja. A beépítésre került mérőeszközök kettő darab Keyence IX-150 sensor, amelyeket egy Keyence IX-1000 és egy IX-1050-es vezérel. A választás azért erre esett, mivel más gyártónál nem találtam hasonló felépítésű lézeres távolságmérőt, és nem elhanyagolható a szervíztámogatottsága sem.

A PLC programot a Siemens által készített Tia Portal V16 program segítségével készítettem el. Ez a program a Simatic Managernek a továbbfejlesztett változata. Az új típusú PLC-ket már ebben a fejlesztő környezetben lehet programozni. A Tia Portal leginkábban abban fejlettebb az előző generációs szoftvernél, hogy minden eleme egybe van építve. Ezáltal, ha HMI-t vagy hajtást szeretnénk vezérelni és beállítani, ahhoz nem kell már más program. A HMI kezeléshez Wincc Flexible szoftvert lehetett régebben használni, amíg a hajtás paraméterezéshez a Starter nevű programot. Ezzel megkönnyítették a programozók dolgát, mivel minden hardvert elérnek egy adott programból.

A kamera programot a Keyence IX-Navigator nevű szoftverével készítettem el. A Keyence-nek termékcsaládonként más felhasználói programja van. Ezek a programok álltalában nagyon felhasználóbarátok. Felépítésük egyszerű, könnyű megtanulni a használatát.

A 12. ábrán látható az elkészült berendezés, amely egy beültetőgép és egy kemence közé került beépítésre. A termékek egy palettán érkeznek, amelyen 6 darab található. A termék bemutatására a cégtől nem kaptam engedélyt, így csak röviden ismertetném a terméket. Egy nyomtatottáramkörilap közepén található egy darab foglalat, amelybe később kerül bele négy darab kondenzátor. Ez mellé mindkét oldalra fel van rakva egy három érintkezővel ellátott alkatrész. Az utóbbi alkatrészek forrasztva lesznek rögzítve a termékhez. Ha a beültetőgép hibásan helyezi rá az alkatrészeket a nyomtatott áramköri

lapra nem lesz tökéletes a forrasztás, ezáltal sok selejt keletkezik. A selejt csökkentése miatt került beüzemelésre a gép.



12. ábra Elkészült berendezés

5.1. Hardver konfiguráció

A programozás első lépéseként a Tia Portalban létre kell hozni egy új projektet. A projekt létrehozása után két lehetőségünk van összeállítani a hardver konfigurációt. Új eszközt hozzáadhatunk típus specifikusan, amikor a programozó választja ki, hogy melyik adott PLC-vel és hozzá kapcsolódó modulokkal dolgozik. Megadhatjuk úgy is, hogy belerakunk a projektünkbe egy nem specifikus PLC-t, és ez alapján a PLC-ről töltjük vissza a hardvert.

Személyes véleményem szerint az előbb említett lehetőség kényelmesebb és jobban alkalmazható, azonban az utóbbi is számos előnnyel jár. Kiindulásként elkezdtem összeállítani a hardver konfigurációt, ami a Tia Portalban egyszerűen megoldható. Fizikálisan meg kell néznünk a konkrét PLC-k típusát, majd egy legördülő listából azt ki kell választanunk. Ezután a kiegészítő I/O modult kell hozzáadnunk a hardverhez, ami

hasonló módon megoldható. Az összeállítás után még be kell állítani a PLC IP címét, amivel ezután szeretnénk kommunikálni.

A PLC előkészítése után a következő elem a HMI, amit hozzá kell adnunk a projektünkhöz. A HMI hozzáadását elvégezhetjük egy varázslón keresztül, de hozzáadhatjuk ugyan csak egy legördülő listából. A varázslóval való hozzárendelésnél már ki kell választanunk melyik PLC-hez szeretnénk csatlakozni, és milyen képernyőket szeretnénk még pluszba felvenni. Ha nem varázslón keresztül adjuk hozzá a HMI-t, akkor nekünk kell gondoskodni a PLC-HMI kapcsolatról.

A Keyence távolságmérő szenzort is hozzá kell adnunk a hardver konfigurációnkhoz. Ezt a hozzárendelést már nem tudjuk ilyen egyszerű módon megcsinálni. Mielőtt hozzá lehetne adni a hardverhez, először a gyártó oldaláról le kell töltenünk a kameránkhoz tartozó GSD fájlt, amit be kell illesztenünk programunkba. A GSD betöltése után már hozzá lehet adni a konfigurációnkhoz. A kapcsolat menüpont alatt még össze kell kötnünk virtuálisan a PLC-t és a kamerát, ezzel létrehozva a kommunikációt. Az IP cím beállítása itt sem maradhat el. Az IP konfigurációt nem csak a Tia Portalban kell elvégeznünk, hanem a kamerához tartozó IX-Navigator nevű szoftverben is. Miután ezekkel végeztünk, a kezdeti lépések megvoltak.

A kezdeti lépések után fel tudjuk tölteni a konfigurációt a rendszerünkre, amit előzőleg már fizikailag megvalósítottunk. Ehhez a programozó készülékünket ugyanabba az IP tartományba kell állítanunk, mint amire az eszközöket állítottuk. Ezután le tudjuk tölteni a konfigot a rendszerünkre. Először a PLC-re töltsük le. A PLC-t kiválasztva elérhetővé válik a letöltés opció. Ezután a HMI-t töltjük fel, azt is kiválasztva külön. Ha mindent jól állítottunk be előzőleg, akkor a rendszerünk hiba nélkül elindul. Ha a rendszerünkbe hiba van, akkor azt meg tudjuk nézni a Tia portalon belüli Online & diagnosztika menüpont alatt.

5.2. PLC programozása

Minden PLC-ben fut egy gyárilag előkészített operációs rendszer, amely lekezeli az interrupokat, kommunikációt. Miután felnyitjuk a program blokk könyvtárat, láthatjuk, hogy itt már előre be van rakva a Main (OB1) blokkunk. Ez a fő blokkunk, innen kerülnek meghívásra a programozó által írott blokkok. A Siemens PLC-k több OB-val (organisation blocks) rendelkeznek, amiknek különböző feladataik vannak. Az OB1 a legalapvetőbb, és ennek van a legkisebb prioritása a többivel szemben. Amíg más OB-k valamilyen interrupt hatására kezdenek lefutni, addig itt ciklikus lefutást láthatunk.

Vannak olyan OB-k, amelyek különböző időközönként futnak le. Van olyan, ami a bekapcsolás után egyszer fut le, és azután sosem; ez a Startup nevű OB. De léteznek olyanok, amelyek valamelyik hardver meghibásodására aktiválódnak. Ezeket előszeretettel használják diagnosztizáláshoz. Itt a programozó dönthet, melyikeket használja, és melyiket nem, kivéve az OB1-et, amit mindenképpen tartalmazni kell a projektünknek.

A PLC-ben nem csak OB-kat lehet létrehozni, hanem Function (FC), Function block (FB), Data block (DB) és Instance Data Block (IDB)-ot is ezeket rövid leírással a (8. táblázat) láthatjuk.

8. táblázat Programmodulok

Programmodul	**Rövid leírása**
Szervezőmodul (OB)	A felhasználói program szerkezetét határozzák meg
Rendszer-FC-k (SFC) és rendszer FB-k (SFB)	Az SFB-k és SFC-k a CPU-ba beépített függvények és rendszerműveletek, illetve gyakran használt egyéb műveletek végrehajtására szolgálnak.
Funkciómodul (FB)	A felhasználó által programozott program modulok, amelyeknél a paraméterek automatikusan adatterületet kapnak.
Függvények (FC)	A felhasználó által programozott program modulok, amelyek nem rendelkeznek a

	paramétereik számára automatikusan adatterülettel
Instant adatmodulok (Instant-DB)	A fejlesztő program minden FB-híváshoz Insant-DB-t rendel, és azt automatikusan létre is hozza.
Adatmodulok (DB)	Adatok tárolására szolgáló területek a felhasználói tárban. Szemben az FB-khez rendelt instant-adatmodulokkal ezek úm. globális adatterületek

A Funkció (FC) egyik alapeleme a strukturált programozásnak. Egy FC rendelkezhet be- és kimenetekkel, illetve belső változókkal is. A funkció végrehajtása után a lokális változók értékei elvesznek. Ha a kimenetre olyan lokális változót másolunk át, amit előtte nem inicializáltunk, a kimenetünk bizonytalan lesz. Egy FC hívása megtörténhet OB-ból, FB-ből és egy másik FC-ből is. Ha megadtunk be- és kimeneti paramétereket, akkor ezt híváskor kezelnünk kell.

Egy másik programkezelő blokk az úgynevezett Function blok (FB). Ez egy „emlékező funkció”, ugyanis minden FB-hez tartozik egy instance DB. Az itt létrehozott lokális változók a programlefutás után megmaradnak. Itt kivételnek számítanak a TEMP változók, amik hasonlóan az FC-hez lefutás után elveszítik értéküket. A globális DB-ket rendszerint a programozó hozza létre, ezeket minden célra fel lehet használni. Az instance DB-k egy FB hívásakor generálódnak ebből adódóan az instance DB az adott FB-hez köthető. A saját programon elkészítésekor ezeket vettem figyelembe.

Az IX szenzort lehet vezérelni digitális be- és kimenetek segítségével. Ha ezt a megoldást választom, akkor a PLC és kamera kapcsolatot egy 20 eres szivárvány kábel segítségével oldhatom meg. Ebben az esetben a kamerában található funkciókat limitált számban érhetem el. A legfőbbek így is elérhetőek, a kamerának tudunk küldeni jelet program váltásához. Hiba törlő bitet tudunk küldeni felé és trigger jelet. Visszakapott jelek közül is a legfontosabbakat megkapjuk kamera üzemben (RUN), elfoglalt jel (BUSY), és megkapjuk az ellenőrzés sikerességének eredményét. Amit ezen a kapcsolaton nem kapunk vissza, az az egyes ellenőrző részek toleranciái és tényleges

eredményei. Mivel számomra fontos volt, hogy minden lehetséges adatot visszakapjak a kamerától, így a hálózaton való kommunikáció mellett döntöttem.

A szenzor vezérlő magában nem tudja a PROFINET-es kommunikációt, ezért egy hozzá tartozó kiegészítő modult kell hozzá kapcsolni. A köztes átjárást egy Keyence DL-PN1 biztosítja.

A programozást azzal kezdtem, hogy létrehoztam egy saját adattípust a DL-PN1 felhasználói kézikönyv segítségével. Ebben fel van tüntetve, hogy mely adatterületeken helyezkednek el. Erre azért volt szükségem, hogy későbbi programfelhasználásnál ne kelljen újra elkészíteni ezt az adattáblát. Létrehoztam az FC_Keyence_IX blokkomat, amelyben megvalósítottam az adatok írását és olvasását.

5.2.1. FC Keyence IX

A blokk elkészítésének elején létrehoztam egy változó táblát az átadni kívánt adatoknak. Amint láthatjuk, a bemenő paraméter a „Slot_No”, ezzel adom meg, hogy a program hány Slot helyet olvasson. Azt, hogy melyik eszköznek az adatait olvassa, a „Station_No” paraméterrel tudom megadni.

```
VAR_INPUT
    Slot_No : UInt;
    Station_No : UInt;
END_VAR
```

Kimenetként ugyan csak két paramétert hoztam létre, amely közül az „Error” nevű jelzi, ha hiba történt az adat olvasás során és a „SlotNr” változó megadja, melyik Slot-on lévő eszközzel van gond.

```
VAR_OUTPUT
    Error : Bool;
    SlotNr : UInt;
END_VAR
```

Be- és kimeneti paraméterként a fentebb már említett adat típust adtam meg.

```
VAR_IN_OUT
    IX_Data : "Keyence_IX";
END_VAR
```

Először a „Station_No” és a „Slot_No” segítségével egy ciklussal kiolvasom a hardver címeit.

```
0003 FOR #tmp_Slot_No := 1 TO #Slot_No DO
0004 #Geo2Log.AREA := 1;
0005 #Geo2Log.HWTYPE := 4;
0006 #Geo2Log.IOSYSTEM := 100;
0007 #Geo2Log.SLOT := #tmp_Slot_No;
0008 #Geo2Log.STATION := #Station_No;
0009 #Geo2Log.SUBSLOT := 0;
0010 #Ret_Geo2Log := GEO2LOG(GEOADDR := #Geo2Log, LADDR =>
#GEO_LADDR[#tmp_Slot_No]);
0011 END_FOR;
```

Amint látható, FOR ciklussal elszámoltatok egytől a „Slot_No”-ban megadott értékig. Minden ciklus során a GEO2LOG parancs segítségével a Slot számokból visszakaptam a hardver azonosítókat, és ezeket letároltam egy ARRAY típusú változóba. A blokk további részén ezen címek alapján olvasok és írok adatokat.

Adatok beolvasására a DPRD_DAT funkciót használtam. Ez a blokk konzisztens adatok olvasására szolgál PROFINET és PROFIBUS-os hálózatokon.

```
#RetVal_read[1]:=DPRD_DAT(LADDR:=#GEO_LADDR[1],RECORD=>
#IX_Data.Recive.Slot0);
```

„RetVal_Read[1]” változóba kerül be a funkció visszatérési értéke, amely 0, ha sikeres az olvasás vagy hiba esetén egy hexadecimális szám. „LADDR” bemenetre az előzőleg kiolvasott hardver cím kerül beírásra. „RECORD” lábára írom azt a címet, ahová szeretném eltárolni a változók értékét.

Az adatok kiírására DPWR_DAT funkciót használom.

```
#RetVal_write[1]:=DPWR_DAT(LADDR:=#GEO_LADDR[1],RECORD:=
#IX_Data.Send.Slot0);
```

Ennek a blokknak a felépítése hasonló az előzőhöz, csak itt a „RECORD” lábára az az adat kerül, amit szeretnénk elküldeni a másik eszköznek.

A funkció blokk végén ismét egy „FOR” ciklust alkalmaztam az esetleges hibák vizsgálatához.

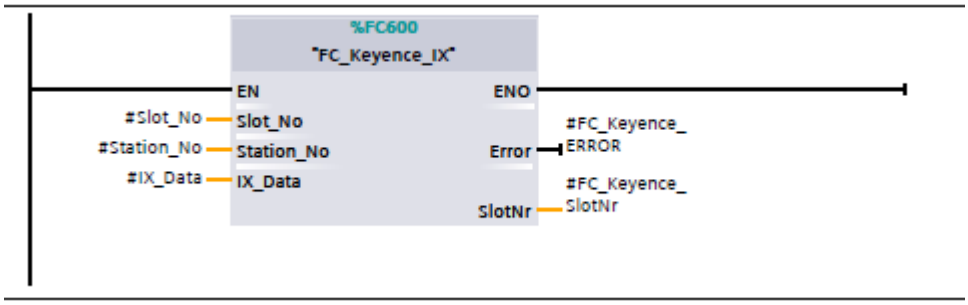
```
0003 FOR #tmp_Slot_No := 1 TO #Slot_No DO
0004 #Geo2Log.AREA := 1;
```

```
0005 #Geo2Log.HWTYPE := 4;
0006 #Geo2Log.IOSYSTEM := 100;
0007 #Geo2Log.SLOT := #tmp_Slot_No;
0008 #Geo2Log.STATION := #Station_No;
0009 #Geo2Log.SUBSLOT := 0;
0010 #Ret_Geo2Log := GEO2LOG(GEOADDR := #Geo2Log, LADDR =>
#GEO_LADDR[#tmp_Slot_No]);
0011 END_FOR;
```

Itt ismételten elszámolok a ciklussal egytől addig az értékig, ami a „Slot_No” bejövő paraméterben meg van adva. Vizsgálom az adatok olvasás és írási sikerességét, ha valamelyik blokk nem 0-val térne vissza, akkor az „Error” bit egybe állítom, és a „SlotNr” változóba betöltöm a hibás eszköz számát.

5.2.2. FB Keyence IX

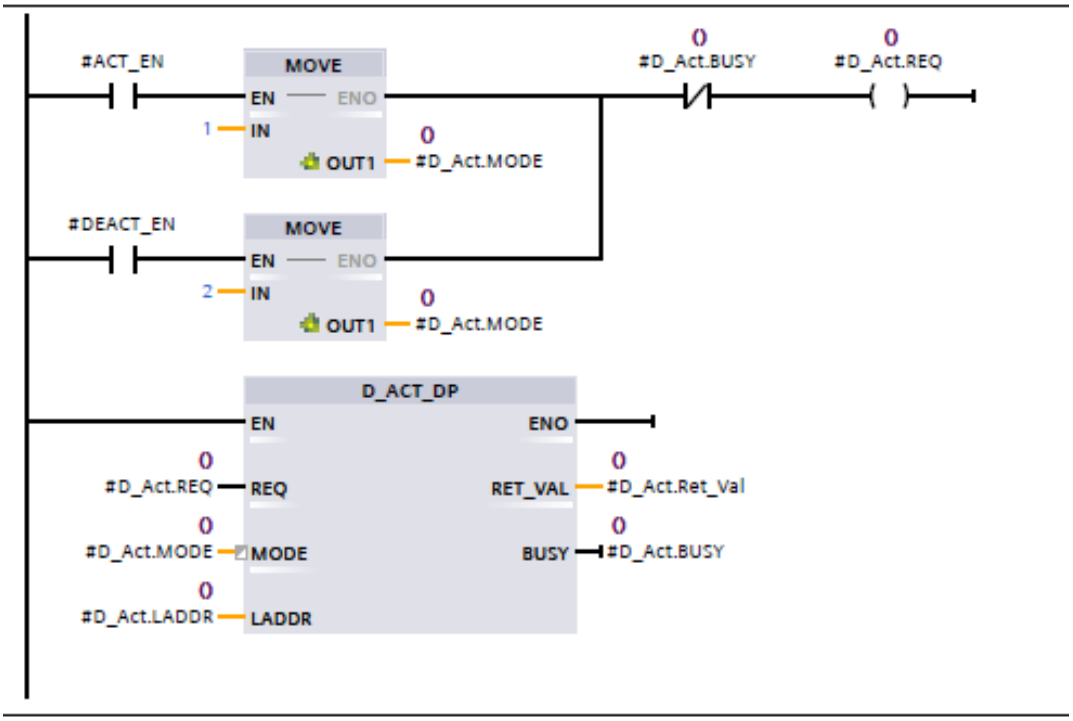
Az FB Keyence IX funkció blokkban hívtam fel az előzőleg bemutatott FC Keyence IX (13.ábra) blokkot.



13. ábra FC Keyence IX meghívása

Paraméterezésénél itt is bemeneti paramétereket használtam, ami majd az FB blokkján kapja meg az értékét. Bementi paraméter még „ACT_EN” és „DEACT_EN” bit-ek. Ezeknek a segítségével le lehet választani az IX szenzorunkat (14. ábra).
A D_ACT_DP funkcióval aktiválni és deaktiválni lehet azt az eszközt, amelynek a címét megkapja a „LADDR” lábán. Ezzel a funkcióval meg lehet tenni azt, hogy deaktiválom az eszközt és lekapcsolom a hálózatról. A hardver konfigurációban ez az eszköz bent marad, de a PLC már nem keresi. Ha ilyen blokkot használok, megtehetem azt, hogy a PLC programban előkészítek több IX szenzort, és konfigurációtól függően ezeket be tudom később is kapcsolni a program módosítás nélkül is.

További bemeneti változó Pr_No_Main, Pr_No_Exp, Mode, Trigger_Main, Trigger_Exp, Reset és Error_Time. Ezeket a blokk további bemutatása során magyarázom el.



14. ábra Modul aktiválása

A DL-PN1 eszközhöz két darab IX szenzort lehet csatlakoztatni ennek a kiválasztásához a „Mode” változó alkalmas. Ha logikai 1 állapotba kapcsoljuk, akkor az azt jelenti, hogy a második szenzor is csatlakoztatva van. 0 állapotban csak egy fej van csatlakoztatva.

Ennek megfelelően külön vezérlő bitek vannak a Main és a Expansion eszközökhöz. Ezért áll rendelkezésünkre két darab programszám bemeneti változó (Pr_No_Main, Pr_No_Exp).

Nálam ezek a változók integer típusban vannak megadva, de a szenzor 4 biten megadva várja a programszámot. Ezért konvertálnom kellett az integert BCD16-ra. Ezt megtettem mind a két programszámmal. Látható, hogy az átkonvertált adatok egyes bitjeit áttöltöm az IX bemeneti címeire.

```
0001 #s_Pr_No_Main :=INT_TO_BCD16(#Pr_No_Main);
0002 #s_Pr_No_Exp := INT_TO_BCD16(#Pr_No_Exp);
0003
0004 #IX_Data.Send.Slot2."Change program number bit 0 (main unit)" :=
#s_Pr_No_Main.%X0;
```

```
0005 #IX_Data.Send.Slot2."Change program number bit 1 (main unit)" :=
#s_Pr_No_Main.%X1;
0006 #IX_Data.Send.Slot2."Change program number bit 2 (main unit)" :=
#s_Pr_No_Main.%X2;
0007 #IX_Data.Send.Slot2."Change program number bit 3 (main unit)" :=
#s_Pr_No_Main.%X3;
0008
0009 #IX_Data.Send.Slot5."Change program number bit 0 (expansion unit)" :=
#s_Pr_No_Exp.%X0;
0010 #IX_Data.Send.Slot5."Change program number bit 1 (expansion unit)" :=
#s_Pr_No_Exp.%X1;
0011 #IX_Data.Send.Slot5."Change program number bit 2 (expansion unit)" :=
#s_Pr_No_Exp.%X2;
0012 #IX_Data.Send.Slot5."Change program number bit 3 (expansion unit)" :=
#s_Pr_No_Exp.%X3;
0013
```

Magát a szenzor vezérlését CASE funkcióval oldottam meg mind a két fej esetében.

```
0001 CASE #Main_Act_Step OF
0002 0:
0003 #IX_Data.Send.Slot1."Reset request (main unit)" := true;
0004 #Main_Act_Step := 1;
0005
0006 1:
0007 IF #s_Trigger_Main AND NOT #IX_Data.Recive.Slot1."Program change
response (main unit)" AND NOT #IX_Data.Recive.Slot1."Trigger/timing response
(main unit)" THEN
0008 #Main_Act_Step := 2;
0009 #Main_Busy := TRUE;
0010 #IX_Data.Send.Slot1."Reset request (main unit)" := FALSE;
0011 ELSE
0012 #IX_Data.Send.Slot1."Reset request (main unit)" := TRUE;
0013 END_IF;
0014
0015 2:
0016 IF #IX_Data.Recive.Slot2."External output 3 (main unit)" AND NOT
#IX_Data.Recive.Slot2."External output 2
(main unit)" THEN
0017 #IX_Data.Send.Slot2."Program change request (main unit)" := TRUE;
0018 END_IF;
0019
0020 IF #IX_Data.Recive.Slot1."Program change response (main unit)" = TRUE THEN
0021 #IX_Data.Send.Slot2."Program change request (main unit)" := FALSE;
0022 #Main_Act_Step := 3;
0023 END_IF;
0024
0025
```

```
0026 3:
0027 #IX_Data.Send.Slot1."Trigger/timing request (main unit)" := TRUE;
0028
0029 #R_TRIG_Main_Resp(CLK:=#IX_Data.Recive.Slot1."Trigger/timing response
(main unit)");
0030
0031 IF (#R_TRIG_Main_Resp.Q = TRUE) AND
#IX_Data.Recive.Slot1."Measurement value result update complete (main
unit)" OR
0032 NOT #IX_Data.Recive.Slot1."Measurement value result update complete (main
unit)" THEN
0033 #Main_Act_Step := 4;
0034 #IX_Data.Send.Slot1."Trigger/timing request (main unit)" := FALSE;
0035 END_IF;
0036
0037
0038 4:
0039 #R_TRIG_Upd_Data_Main(CLK:=#IX_Data.Recive.Slot1."Measurement value
result update complete (main unit)");
0040
0041 #F_TRIG_Upd_Data_Main(CLK:=#IX_Data.Recive.Slot1."Measurement value
result update complete (main unit)");
0042
0043 IF #F_TRIG_Upd_Data_Main.Q OR #R_TRIG_Upd_Data_Main.Q THEN
0044 #Main_Act_Step := 5;
0045
0046 END_IF;
0047
0048 5:
0049 IF #Done_Datacopy = TRUE THEN
0050 #Main_Act_Step := 0;
0051 #Main_Busy := FALSE;
0052 END_IF;
0053
0054 ELSE // Statement section ELSE
0055 ;
0056 END_CASE;
```

Amint láthatjuk a vezérlés 6 lépésből áll. Nullás lépésen kérek egy Reset-et a szenzortól, majd rögtön tovább is ugrik az első lépésre. Ennél a lépésnél várom a Trigger jelet, hogy elindíthassam a folyamatot, és figyelem a szenzor állapotát. Következő lépésben, ha elérhető állapotban van, akkor kérek tőle egy programszám váltást, majd ugrok a következő lépésre. Harmadik lépésben kapja meg a jelet a szenzor a képkészítéshez. Majd várom, hogy frissítse az adatait, és lépek tovább. Negyedik lépésnél figyelem az adat

frissítés bitet egy felfutó és egy lefutó él vizsgálattal. Minden sikeres ellenőrzés után ez a bit invertálódik, ezért szükséges a két vizsgálat. Következő lépésnél várok az adatmentés bit-emre, amelyet a programban lentebb írok majd.

Ha mind a két fej az 5-ös lépésben várakozik két fejes vezérlés esetén, akkor a fukció blokk kimenetére írom át a visszakapott eredményeket.

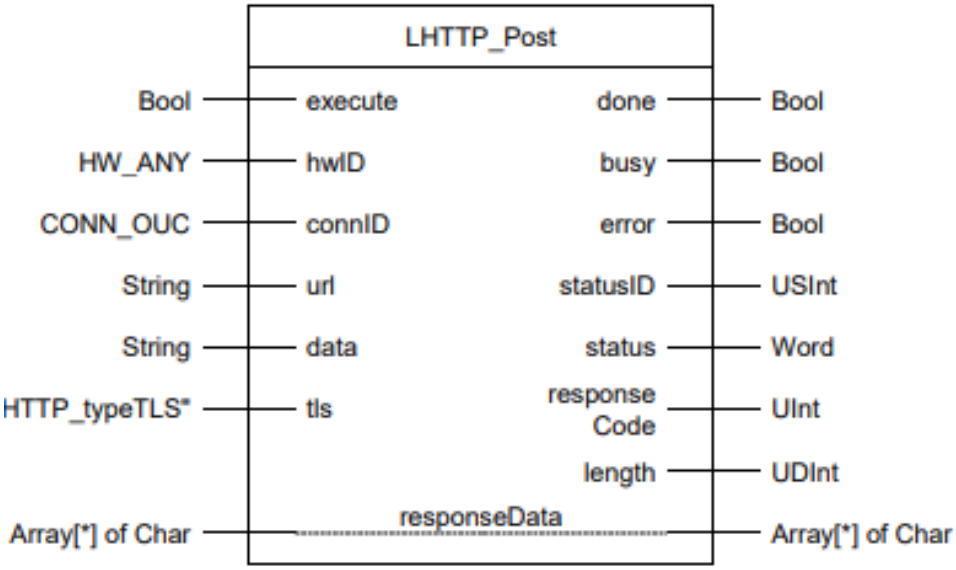
```
0001 IF (#Main_Act_Step = 5 AND (#Expansion_Act_Step = 5 OR NOT #Mode))
THEN
0002 REGION DInt Data
0003 //Data1
0004 #Data[1].%B0 := #IX_Data.Recive.Slot1."Measurement value 1*3";
0005 #Data[1].%B1 := #IX_Data.Recive.Slot1."Measurement value 1*4";
0006 #Data[1].%B2 := #IX_Data.Recive.Slot1."Measurement value 1*5";
0007 #Data[1].%B3 := #IX_Data.Recive.Slot1."Measurement value 1*6";
//Data1
0082 #Data_Bool[1].%X0 := #IX_Data.Recive.Slot1."Measurement value 1 invalid";
0083 #Data_Bool[1].%X1 := #IX_Data.Recive.Slot1."Measurement value 1 under
range";
0084 #Data_Bool[1].%X2 := #IX_Data.Recive.Slot1."Measurement value 1 over
range";

0085 //
```

Ezt az átírást meg kell tennem 15 darab Dint adat esetében, amelyet egy ARRAY tömbbe helyezek el. Az adatmásolást meg kell tennem még 15-ször 3 bit esetén, amelyet egy 15 elemű ARRAY byte tömbbe helyezek el. Ha kész az adatmentés, akkor nyugtázom és visszaléptetem a lépésláncokat 0-ba. A blokk legvégén generálok lépéslánc hibákat, ha túl sokáig várakozna egyes lépésekben. A lépések közötti hibagenerálási idő bejövő paraméterként állítható. Továbbá a szenzoroktól kapott hibabiteket is továbbítom a blokk kimenetére. Hibás működés esetén nyugtázni tudom bejövő paraméteren keresztül az aktuális hibákat.

5.2.3. Mérési adatok mentése

Minden mérési eredményt menteni kell egy adatbázisba. Az adatbázis TCP kapcsolaton keresztül érhető el a PLC-től. Az adat küldéséhez létre kell hozni egy előre meghatározott adatstruktúrát, amit a szerver fel tud dolgozni. Esetemben az adatok JSON-ban vannak összeállítva. A kommunikációt a Siemens által készített LHTTP_Post (15.ábra) blokkon keresztül oldom meg. A blokk módosítva lett általam, mivel a maximális elküldhető adat jóval kevesebb, mint amit nekem küldenem kell. Ennek a



15. ábra LHTTP_Post blokk

feladatnak a megoldása nem volt a legegyszerűbb feladat. Eddig az adatküldés egy köztes PC-n keresztül történt. Így az adatstruktúra egyszerűbb volt mivel nem kellett megfelelnie a JSON felépítésének, ezt a PC elvégezte.

A szerver eléréséhez a címet a fenti ábrán az „URL” bemeneti lábra kell írni, amely string típusú. A küldhető adat a „data” lábra kerül szintén string típusban. Én a „data” bemenetet módosítottam egy 3000 darabból álló karakter tömbre. Majd hozzáadtam még egy bemenő paramétert „DataLenght”-et, amely integer típusú. Ezt azért tehettem meg, mert a blokkon belül használt TSEND_C funkció képes 8192 byte-nyi adat hosszúságot is kezelni.[10]

Az adat összeállításhoz készítettem a funkciót, amely elkészíti az adattömböt. A funkciónak egy kimenő paramétere van, ez a „o_Length”, amely integer típusú változó

és megmondja milyen hosszú a tömb. Ezen kívül van egy be- és kimeneti paraméter, amely egy ismertlen hosszúságú karakter tömb. A blokk a folyamat elején kiüríti a belső változók előző adatait.

Az adatok sting típusban vannak tárolva, így ezeket a változókat össze kell fűzni. Erre a funkcióra való a „CONCAT_STRING” parancs, amely két string-et tud egymás mögé helyezni. Az IN1 paraméter után helyezi az IN2 paraméteren lévő adatot, és beletölti a „Header” nevű string változóba. Itt olyan adatok kerülnek összefűzésre, mint a termék egyedi azonosítója, státusza, mikor és ki készítette. Miután összeállítottam ezeket az adatokat, meg kell számolnom hány karakter van benne. Ezt a régió végén a LEN paranccsal tudom megtenni. Az eredményét egy integer változóba helyezem.

```
0017 REGION Adatok összefűzése
0018 ////////////////////FACTORY//////////////////////////////
0019 #Header := CONCAT_STRING(IN1 := '{"factory":{"tester":"', IN2 :=
"DB_FF_Data".Factory.Tester);
0020 #Header := CONCAT_STRING(IN1 := #Header, IN2 := '","user":"');
0021 #Header := CONCAT_STRING(IN1 := #Header, IN2 :=
"DB_FF_Data".Factory.User);
0022 ////////////////////PANEL//////////////////////////
0023 #Header := CONCAT_STRING(IN1 := #Header, IN2 :=
'"},"panel":{"dut":{"id":"');
0024 #Header := CONCAT_STRING(IN1 := #Header, IN2 :=
"DB_FF_Data".Panel.Dut.ID);
0025 #Header := CONCAT_STRING(IN1 := #Header, IN2 := '","timestamp":"');
0026 #Header := CONCAT_STRING(IN1 := #Header, IN2 :=
"DB_FF_Data".Panel.Dut.Timestamp);
0027 #Header := CONCAT_STRING(IN1 := #Header, IN2 := '","testTime":');
0028 #Header := CONCAT_STRING(IN1 := #Header, IN2 := #s_TestTime);
0029 #Header := CONCAT_STRING(IN1 := #Header, IN2 := ',"status":"');
0030 #Header := CONCAT_STRING(IN1 := #Header, IN2 :=
"DB_FF_Data".Panel.Dut.Status);
0031 #Header := CONCAT_STRING(IN1 := #Header, IN2 := '","group":{"name":"');
0032 #Header := CONCAT_STRING(IN1 := #Header, IN2 :=
"DB_FF_Data".Panel.Dut.Group.Name);
0033 #Header := CONCAT_STRING(IN1 := #Header, IN2 := '","timestamp":"');
0034 #Header := CONCAT_STRING(IN1 := #Header, IN2 :=
"DB_FF_Data".Panel.Dut.Group.Timestamp);
0035 #Header := CONCAT_STRING(IN1 := #Header, IN2 := '","status":"');
0036 #Header := CONCAT_STRING(IN1 := #Header, IN2 :=
"DB_FF_Data".Panel.Dut.Group.Status);
0037 #Header := CONCAT_STRING(IN1 := #Header, IN2 := '","groups":[');
0038
0039
```

```
0040
0041 #Lenght_Header := LEN(#Header);
0042
0043 END_REGION
```

Ebben a részben állítottam össze a termék általános adatait, és ezek után mutatom be a mérési eredmények összeállítását. Itt négy darab mérési eredményt állítok össze, amelyek közül csak az elsőt mutatnám be. Az összeállítás ugyanazzal a metódussal történik, mint az általános adatoknál. Ennél a résznél olyan adatok kerülnek összeállításra, mint hogy mikor történt a feladat, annak az eredménye. Bekerül még a mért eredmény, illetve az alsó és felső határértékek.

```
REGION Elso
0048 #POS1 := CONCAT_STRING(IN1 := '{"name":"', IN2 :=
"DB_FF_Data".Panel.Dut.Group.Groups.Inspection.Name);
0049 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 := '","timestamp":"');
0050 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 :=
"DB_FF_Data".Panel.Dut.Group.Groups.Inspection.Timestamp);
0051 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 := '","status":"');
0052 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 :=
"DB_FF_Data".Panel.Dut.Group.Groups.Inspection.Status);
0053 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 := '","tests":[{"name":"');
0054 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 :=
"DB_FF_Data".Panel.Dut.Group.Groups.Inspection.Tests.POS1.Name);
0055 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 := '","lolim":"');
0056 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 :=
"DB_FF_Data".Panel.Dut.Group.Groups.Inspection.Tests.POS1.Lolim);
0057 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 := '","hilim":"');
0058 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 :=
"DB_FF_Data".Panel.Dut.Group.Groups.Inspection.Tests.POS1.Hilim);
0059 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 := '","value":"');
0060 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 :=
"DB_FF_Data".Panel.Dut.Group.Groups.Inspection.Tests.POS1.Value);
0061 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 := '","status": "');
0062 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 :=
"DB_FF_Data".Panel.Dut.Group.Groups.Inspection.Tests.POS1.Status);
0063 #POS1 := CONCAT_STRING(IN1 := #POS1, IN2 := '},');
0064
0065 #POS1_int := LEN(#POS1);
0066 END_REGION
```

Miután minden eredményt összeállítottam, el kell készítenem belőle a kimeneti karaktertömböt. Ehhez az összeállításhoz a „Strg_TO_Chars” funkciót használom. A parancsnak a bemenetre kell helyezni azt a string változót, amelyet szeretnénk

elhelyezni a tömbbe. Meg kell adni neki még, hogy melyik helytől hány darab karaktert helyezzen el.

```
0118 Strg_TO_Chars(Strg := #Header,
0119 pChars := 0,
0120 Cnt => #Lenght_Header,
0121 Chars := #DataString);
0122
0123 Strg_TO_Chars(Strg := #POS1,
0124 pChars := #Lenght_Header,
0125 Cnt => #POS1_int,
0126 Chars := #DataString);
0127
0128 Strg_TO_Chars(Strg := #POS2,
0129 pChars := #Lenght_Header+ #POS1_int,
0130 Cnt => #POS2_int,
0131 Chars := #DataString);
0132
0133
0134 Strg_TO_Chars(Strg := #POS3,
0135 pChars := #Lenght_Header + #POS1_int + #POS2_int,
0136 Cnt => #POS3_int,
0137 Chars := #DataString);
0138
0139 Strg_TO_Chars(Strg := #POS4,
0140 pChars := #Lenght_Header + #POS1_int + #POS2_int + #POS3_int,
0141 Cnt => #POS4_int,
0142 Chars := #DataString);
```

Miután minden elkészített string változót beletöltöttük a karakter tömbbe, még meg kell adnunk milyen hosszú is az elkészített tömb. Mivel minden string összeállításánál megszámoltam hány karakter van benne, itt már csak összegeznem kell ezeket, és kiraknom a kimentre. Ha az adat elkészült, akkor átadhatom az értékeket a szerver kommunikációs blokkjának.

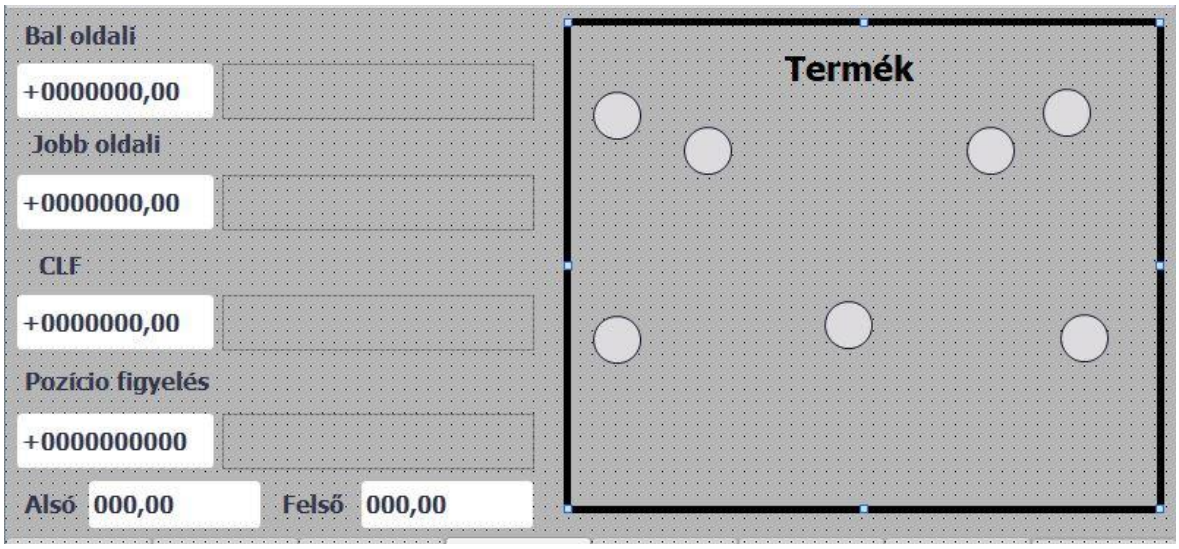
5.3. A mérőszenzor programja

Keyence IX szenzor tanításához az IX Navigator nevű programot használtam. Első lépésként beállítottam a használni kívánt IP címét a vezérlőnek és a vezérlési formát. Majd létrehoztam a nullás programot. Az IX szenzor nem csak egy sima távolságmérő eszköz, hanem egybe van integrálva egy kis felbontású kamerával is. Az integrált kamerának az a nagy előnye, hogy beállítás során láthatjuk melyik pontokon is mérünk. A program első lépése egy referencia sík felvétele. Ezt megtehetjük kétféleképpen egy mester kép készítésével vagy minden mérés előtti sík felvételével. Én a mesterkép felvételével készítettem el a tanítást. Három mérési pontot kell elhelyeznünk a felületen, amiből megkapjuk a síkunkat.

A sík felvétele után először egy pozíció figyelést helyeztem el a terméken, így ha minimálisan el van mozdulva a mérőeszköz, akkor el tudja tolni a mérési pontokat. A pozíció keresés után elhelyeztem 6 darab magasságfigyelést. Ezeket úgy állítottam be, hogy adott területen a legmagasabb értéket adja vissza. A magasság figyelések párban vannak és arra vagyok kíváncsi, hogy milyen magasság különbség van a két pont között. Két pont magasságát kivonom egymásból, és ezt az eredményt fogom elküldeni a PLC felé. Így lesz három darab magasság különbség értékem, és egy pozíció felismerési értékem. Az értékeket megjelenítem egy HMI-n is, amelyet később mutatok be. Az ellenőrző program végén még lehetőségem van különböző bool változókat átadni a PLC-nek. Én itt a hiba és a foglalat jeleket küldtem el.

5.4. HMI felület

A HMI felületen létrehoztam egy oldalt, ahol a mért eredmények tekinthetők meg (16. ábra), és egy másik oldalt, ahol a heti mérési statisztika látszik (17. ábra).



16. ábra Termék mérési eredményei

A fenti képen láthatjuk a mérési értékeket, tűréshatár beállítását és visszajelzést a mérő pozíciókról. A mérőpontokat jelző szürke körök úgy vannak beállítva, hogy ha az adott mérési eredmény hibás, akkor az pirosra vált, és ha jó, akkor zöldre. Mivel egyszerre hat darab termék érkezik az állomásra, így mind a hat mérési eredményt külön kell megjeleníteni. Ehhez én nem szerettem volna hat darab képernyőt létrehozni. Az adatokat úgy jelenítem meg, hogy nem direktbe hivatkozok az eredményekre, hanem azoknak a memória helyét adom meg. A memória terület betöltését a PLC-ben írtam meg.

```
0002 // Palettan az elso ketto darab
0003 IF "DB_HMI".Keyence_IX.Address_Data = 16#4 THEN
0004 "DB_HMI".Keyence_IX.Address_Data_1 := 0;
0005 "DB_HMI".Keyence_IX.Address_Data_2 := 4;
0006 "DB_HMI".Keyence_IX.Address_Data_3 := 8;
0007 "DB_HMI".Keyence_IX.Address_Data_4 := 12;
0008
0009 "DB_HMI".Keyence_IX.Address_Data_5 := 60;
0010 "DB_HMI".Keyence_IX.Address_Data_6 := 61;
0011 "DB_HMI".Keyence_IX.Address_Data_7 := 62;
0012 "DB_HMI".Keyence_IX.Address_Data_8 := 63;
0013 // Statusz
0014 "DB_HMI".Termek_Addr.Addr0 := 277;
```

```
0015 "DB_HMI".Termek_Addr.Addr1 := 278;
0016 "DB_HMI".Termek_Addr.Addr2 := 279;
0017 "DB_HMI".Termek_Addr.Addr3 := 280;
0018 END_IF;
0019 IF "DB_HMI".Keyence_IX.Address_Data = 16#20 THEN
0020 "DB_HMI".Keyence_IX.Address_Data_1 := 16;
0021 "DB_HMI".Keyence_IX.Address_Data_2 := 20;
0022 "DB_HMI".Keyence_IX.Address_Data_3 := 24;
0023 "DB_HMI".Keyence_IX.Address_Data_4 := 28;
0024
0025 "DB_HMI".Keyence_IX.Address_Data_5 := 64;
0026 "DB_HMI".Keyence_IX.Address_Data_6 := 65;
0027 "DB_HMI".Keyence_IX.Address_Data_7 := 66;
0028 "DB_HMI".Keyence_IX.Address_Data_8 := 67;
0029 // Statusz
0030 "DB_HMI".Termek_Addr.Addr0 := 283;
0031 "DB_HMI".Termek_Addr.Addr1 := 284;
0032 "DB_HMI".Termek_Addr.Addr2 := 285;
0033 "DB_HMI".Termek_Addr.Addr3 := 286;

0034 END_IF;
```

Amint látható, először ellenőrzöm, hogy melyik terméket szeretné megnézni a kezelő, és ahhoz társítom a memória területet. Ezeket a címeket megkapja a HMI, és onnét olvassa ki az aktuális adatokat.

Termelés hatékonyság miatt meg kell jelenítenem a jó és a rossz termékek darabszámát. Ezt a statisztikát egy hétre visszamenőleg lehet megnézni. Kijelzi hány darab termék ment keresztül az állomáson, és ezeknek hány százalékuk kapott jó eredmény és hány rosszat (17.ábra).

17. ábra Termelési statisztika

5.5. Statisztika

A 9. táblázatban olvashatóak le a bevezetés előtti és utáni selejtezési darabszámok és százalékok. Amint láthatjuk bevezetés elött 2% felett voltak a selejtszámok amit sikerült leredukálni már a bevezetést követő héten 1,2%-ra.

9. táblázat Selejtezési táblázat

	Bevezetés elött		Bevezetés után			
	Darab/hét	Selejt%	Darab/hét	Selejt %	Darab/hét	Selejt %
HP-alkatrész túl magas - Jobb	41	1,12%	6	0,41	19	0,31%
HP- alkatrész túl magas – Bal	15	0,41%	8	0,55%	20	0,33%
HP- alkatrész túl magas - Középső	21	0,57%	4	0,27%	4	0,07%
Össz.	3667	2,09%	1460	1,23%	6102	0,7%

6. ÖSSZEFOGLALÁS

A program felépítését úgy próbáltam elkészíteni, hogy a programozásban nem annyira jártas kollegák és karbantartók is megértsék. Figyeltem arra, hogy a felhasználó a lehető legtöbb információt megkapja a gépről, gondolok itt arra, hogy látható melyik szekvenciákban várakozik a program. Figyeltem arra, hogy a kezelőfelület használata ne legyen túl bonyolult, és a felhasználó könnyen el tudja sajátítani a kezelését.

Készítettünk a gépről egy folyamatképesség vizsgálatot is, amely azt takarta, hogy egy palettán érkező hat darab termék mindegyikét harmincszor lemértük. Ezzel megtudtuk, hogy milyen a rendszer szórása és átlaga. A termékeket egy dimenzió méréssel foglalkozó csapat lemérte ATOS Scanbox-szal, ezzel is alátámasztva a mérési eredményeinket. Így a lehető legszűkebb határok közé tudtuk beállítani a toleranciákat.

Mivel a kieső termékek selejtezése igen nagy bevételkiesést eredményez, ezért a gép költsége összemérhető a bevételkieséssel.

Amint már dolgozatomban említettem, a gép letelepítése után csökkentek a selejtezési darabszámok. Tehát a vele szemben támasztott követelményeknek megfelel.

Szeretném az általam készített Keyence IX vezérlő programját továbbfejleszteni olyan módon, hogy későbbiekben csak fel kelljen paraméterezni egy blokkot, ezzel megkönnyítve későbbi munkáimat, ha hasonló mérőeszközt szeretnék használni. Továbbá a későbbi fejlesztések közé tartozik még az adatbázissal történő kommunikációs blokk frissítése, hogyha nagyobb mennyiségű adatot kell eltárolni, akkor azt is képes legyen elküldeni.

TARTALMI KIVONAT

A szakdolgozatom célja az volt, hogy elkészítsek egy PLC programot, amely végrehajtja a kommunikációt és vezérli a Keyence IX mérőeszközt, majd a mérési adatokat összeállítja, végül letárolja egy adatbázisba. Tartalmát tekintve három részre osztottam fel.

Az első részben ismertetem a PLC-k tulajdonságait, fizikai felépítését. Bemutatásra kerülnek a szabványos programnyelvek és sajátosságaik. Következőkben áttekintésre kerülnek az iparban használt kamerák és lézeres távolságmérő eszközök felépítése.

A harmadik részében mutatom be az elkészített programkódot. Megírásra került egy funkció, amelyben megállapítottam az IO modulok címeit. A visszakapott értékek alapján belső adatterületre helyeződnek a bejövő jelek, és kimeneti tárra kerülnek a belső memóriában letárolt állapotok. Elkészítettem egy blokkot, amelyben megírtam a szenzor vezérlőfolyamatát. A mérési eredményeket a megírt blokk kimenetén olvashatjuk. Ezen adatok összeállítását egy külön funkcióban készítettem el. Az itt visszakapott struktúrát közvetlenül továbbítani tudom egy szerveren elhelyezett adatbázisba. A mérési eredményeket és a statisztikát nem csak az adatbázis felé továbbítom, de megjelenítem egy felhasználói interfészen is.

A statisztai adatokból látható, hogy az ellenőrzés bevezetése után csökkent a selejt aránya, amivel sikerült növelni a jó termékek kihozatali darabszámokat.

SUMMARY

The aim of my thesis is to create a PLC program that performs the communication and controls the Keyence IX laser sensor, then compiles the measurement data and finally stores it in a database. In terms of content, I divided it into three parts.

In the first part, I am describing the properties and physical structure of PLCs. Standard programming languages and their features are presented. In the following, an overview of the design of cameras and the structure of laser rangefinders used in industry are provided.

In the third part, I am presenting the prepared program code. A function has been written in which the addresses of the IO modules are determined. Based on the returned values, the input signals are placed in the internal memory and the statuses stored in the internal memory are moved into the output memory. I made a block in which I wrote the control process of the sensor. The measurement results can be read at the output of the written block. I compiled this data in a separate function. I can transfer the structure here directly to a database hosted on a server. Not only do I transmit the measurement results and statistics to the database, but also display them on a user interface.

Statistics show that after the introduction of the control, the proportion of scrap decreased, which succeeded in increasing the yields of good products.

ÁBRAJEGYZÉK

TÁBLÁZATOK JEGYZÉKE

IRODALOMJEGYZÉK

[1]. Maczik Mihály A. (2015): PLC ismeretek és példatár. Budapest: Műszaki könyvkiadó

[2]. Czúni László, Tanács Attila (2011): Képi információ mérése

[3]. Keyence, Image based laser sensor IX-Series Instruction Manual

Elektronikus hivatkozások

[4]. Lassó András (2000):Lézeres távolságmérés mérési útmutató- IIT MoMic labor

[5]. Dr. Ferenczi István (2018): PLC programozási alapismeretek – Nyíregyházi Egyetem

[6]. Juhász Róbert: Ismerkedés a PLC-vel

[7]. Gépi látás, https://hu.wikipedia.org/wiki/G%C3%A9pi_l%C3%A1t%C3%A1s (Letöltve: 2019. 12. 05.)

[8]. PROFINET, https://www.ob121.com/doku.php?id=hu:comm:bus_profinet (Letöltve:2020.02.10.)

[9]. Character Inspection/OCR, https://www.keyence.com/ss/products/vision/visionbasics/use/inspection05/ (Letöltve: 2021.03.11.)

[10]. Library for HTTP Communication (LHTTP), https://support.industry.siemens.com/cs/document/109763879/library-for-http-communication-(lhttp)?dti=0&lc=en-WW (Letöltve 2021.01.15.)

[11]. Budapesti Műszaki és Gazdaságtudományi Egyetem Gépészkari lapjának szerzője, Gépi látás az ipari gyakorlatban, https://www.cnc.hu/2018/08/gepi-latas-az-ipari-gyakorlatban/ (Letöltve: 2019.12.05.)

[12]. PROFIBUS Neutzerorganisation , PROFINET System Description , https://docplayer.net/39751409-Profinet-system-description-technology-and-application-open-solutions-for-the-world-of-automation.html (Letöltve 2020.05.11.)

[13]. PROFINET Infrastructure -CAT-5 CABLE, https://profinetuniversity.com/profinet-basics/profinet-infrastructure-cat-5-cable/ (Letöltve 2020.05.11.)