



**Budapesti Műszaki és Gazdaságtudományi Egyetem
Gépészmérnöki Kar**

Gyártásközi termékellenőrzés és -kezelés intelligens kamerával és ipari robottal

TDK dolgozat

Készítette: Fehér Zoltán

Neptun kód: GT03SO

Tanszéki konzulens: Tóth András

Tudományos Munkatárs

Budapest, 2013.

Tartalom

1.	Bevezetés.....	5
2.	Alakfelismerő rendszerek.....	6
2.1.	Képfeldolgozó alkalmazások típusai	6
2.2.	Képfeldolgozó rendszerek az autóiparban [14]	6
2.3.	Korszerű képfeldolgozási alkalmazások az ipari robottechnikában.....	8
2.4.	Szállítószalagos adagolás.....	9
2.5.	Ládából való adagolás.....	10
3.	Rendszer előkészítése.....	12
3.1.	Kommunikációs program	12
3.1.1.	Perifériák kapcsolódása.....	13
3.1.2.	Program kezelőfunkciói	13
3.2.	Robot program.....	15
3.2.1.	Szimuláció elkészítése.....	15
3.2.2.	1. feladat robotprogramja	15
3.2.3.	2. feladat robotprogramja	16
4.	Rendszer megépítése	19
4.1.	Kommunikációs program tesztelése.....	20
4.2.	Állvány tervezése	21
4.3.	Kamera tesztelése	23
4.4.	Szállítószalag sebességének meghatározása.....	25
4.5.	A rendszer kalibrálása és próbája	27
5.	A gyakorlati feladatok vizsgálata.....	28
5.1.	Első gyakorlati feladat: LED-ek beépítése és ellenőrzés.....	28
5.1.1.	ArUco jelek	28
5.1.2.	Feladat előkészítése.....	28

5.1.3.	Feladat vizsgálata	31
5.1.4.	Második gyakorlati feladat: Optimalizálás	32
6.	Összefoglal.....	34
7.	Summary	35
8.	Irodalomjegyzék.....	36
9.	Melléklet.....	37

Melléklet jegyzéke

- 1) 1. feladat robotprogramja
- 2) 2. feladat robotprogramja hengerekre
- 3) DVD melléklet

DVD melléklet tartalma:

{1} 3D modellek

Állványra szerelt verzió Pack'n'Go fájllai.rar

Asztalra szerelt verzió Pack'n'Go fájllai.rar

Labor Pack'n'Go fájllai.rar

{2} RobotStudio fájlok

Szimuláció.rspag

Megfogó.rslib

Adagolás.rslib

{3} Videók

1. feladat.avi

2. feladat_henger.avi

2. feladat_panel.avi

Szimuláció.avi

{4} Robotprogramok

1. feladat.prg

2. feladat.prg

{5} Perifériák összekapcsolásához szükséges fájlok

- Szállítószalag kapcsolat

BCS_Setup.exe

- Kommunikációs program

Kommunikációs program.exe

Kommunikációs program.vbp

Kommunikációs program.vbw

{6} Labor útmutató.docx

1. Bevezetés

A dolgozatomban célja egy a BME-n oktatható laboratóriumi gyakorlat elkészítése, melynek témája az „Gyártásközi termékellenőrzés és -kezelés intelligens kamerával és ipari robottal”. A mai automatizálási trendekben teret nyernek a képfeldolgozó kamerák, amelyeket így véleményem szerint hasznos az oktatásban is megjeleníteni. A hallgatók két gyakorlati feladat révén megismerhetik, miként is működnek ezek a folyamatok.

Feladatomban a rendszer architektúrájának megtervezésével kezdtem el. A már meglévő –mások között a szakdolgozatomban általam is fejlesztett- [2] [3]robot-kamera programot kiegészítettem robotos szállítószalag vezérléssel. A gyakorlat során alapvető információkkal tud szolgálni a program, mint például a felismert alkatrész neve, vagy a szállítószalag előírt sebessége. A program megírása során igyekeztem felhasználóbarát alkalmazást készíteni.

A gyakorlatot a Gyártástudomány- és Technológiai Tanszék robotlaborjában készítettem el. Megvizsgáltam a robot környezetét, telepített kamera elrendezés mellett döntöttem, és kamera állvány prototípusokat terveztem. Ezeket megépítve, kiválasztottam az ideális kivitelt. A kamera képességeit is megvizsgáltam, így hosszabb távú következtetésekre jutottam a képfeldolgozási feladatok körét illetően.

A laborgyakorlat során két feladatot kell majd megoldani a hallgatóknak. Az első feladat során LED-ekből egyedi ArUco 2D kódmintákat kell megépíteni minden egyes hallgatónak. Ezeket a szállítószalagra helyezve, a rendszer minősíti, selejt esetén megállítja a szállítószalagot. Itt megvizsgálható a hiba oka. A második feladat során egy nagysorozatban gyártott ipari elektronikai alkatrészt kell a hallgatóknak betanítani a kamerának. A betanítás ellenőrzése 10 db helyes 1 db selejt alkatrésszel történik. A véletlenszerűen érkező darabokból a robot a selejttárolóba rakja a selejtet. Itt is az ellenőrzés után vizsgálják meg a hallgatók a selejteket.

Készítettem egy labor útmutatót, amely a hallgatók számára tömören leírja a modern képfeldolgozási trendeket a robotalkalmazásokban. Az útmutató kitér a rendszer részeire, és a betanítási lehetőségekre, kiemelve azokat a részeket, amelyek tervezéskor fontos szempontot képezhetnek. Leírom a második feladat során használt SIEMENS kamera betanításának lépéseit is.

Az elkészült munkámat remélem a hallgatók is érdekesnek találják majd. Megpróbáltam minél szívesebb és gyakorlatiasabb órát összeállítani.

2. Alakfelismerő rendszerek

A dolgozatom célja, hogy egy olyan gyakorlatot készítsék, amelyben hallgatók megismerkednek az iparban használt képfeldolgozási rendszerekkel, különös tekintettel a robotos alkalmazásokra.

2.1. Képfeldolgozó alkalmazások típusai

Léteznek egyszerű szenzorok (pl.: erő-, nyomásérzékelő szenzorok). Általában 1 dimenziós feldolgozó képességgel rendelkeznek. Bináris jelet (igen-nem) küldenek, így befolyásolva a robotprogram függvényeit.

Az alakfelismerő szenzorok összetettebb szenzorok. Ezek már területet figyelnek, amit egy algoritmus értékel ki. Ilyenek például vonalkód leolvasók, a DataMatrix vagy a QRCode.

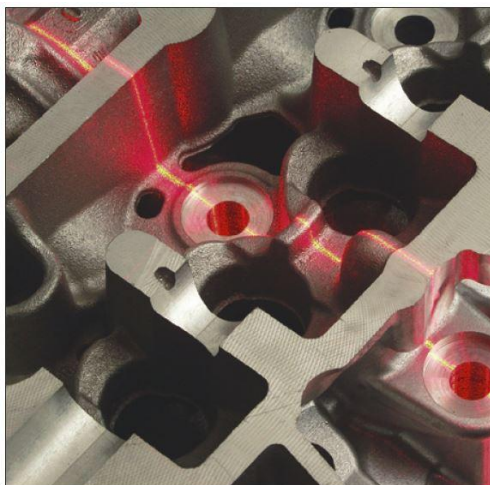
Léteznek kompakt rendszerek, vagy más néven intelligens kamerák is. Ezek képfeldolgozó processzor ellátott kamerák. Zárt rendszerek, kevésbé rugalmasak, de vírusok ellen védettek. Ezzel ellentétben léteznek PC alapú kamerák is. Ezek a rendszerek testre szabhatóak, viszont vírusos fertőzéssel szemben kevésbé védettek. [1]

2.2. Képfeldolgozó rendszerek az autóiparban [14]

Képfeldolgozó rendszerekkel találkozhatunk a gyártás legalapvetőbb lépéseitől (például: fémalkatrész vágása) a rész- vagy végtermék ellenőrzésig. Az autóipari beszállítókra vonatkozó legújabb követelmények megkövetelik, hogy 1 millió leszállított részegységből legfeljebb 20 darab lehet hibás. Intelligens kamerák használatával selejtes darabokat tudunk kiválogatni, illetve statisztikákat is készíthetünk. A hibás darab gyors azonosítása elősegíti azok megelőzését, így optimalizálva a gyártást.

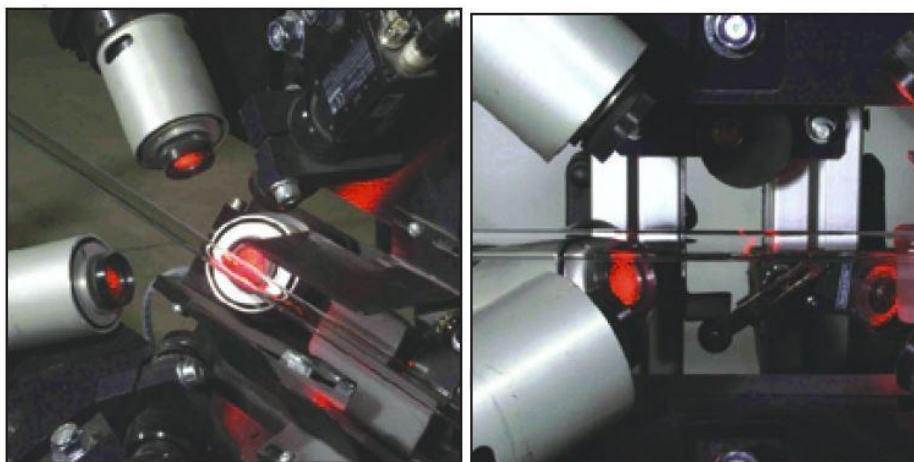
Az autóiparban bonyolult öntvényekkel is találkozhatunk. Ezek gyártás során bekerülnek egy mérőhelyre, ahol kameraképek készülnek. Ezen képek feldolgozása után, számos paraméter ellenőrizhető. A kamerák és fényforrások száma az alkatrész bonyolultságától függ. Lézeres sávmegvilágítással háromdimenziós mérések is elvégezhetők. Egy ilyen berendezés például 750 paraméter mérést tudja elvégezni közvetve vagy közvetlenül 24 másodperc alatt. Ellenőrzött paraméterek lehetnek: alak, elhelyezkedés, mélység, terület, felületminőség. Kontúrok esetén $\pm 0,3\text{mm}$ -es tűréssel, de indexfuratok esetén a $\pm 0,014\text{mm}$ tűrés is elérhető. Nyílások területének $\pm 1\%$ -os mérési pontosságtól, azok elhelyezkedésének $\pm 0,3\%$ -os pontossáig elvégezhetőek a mérések [15]. A mérési adatok összesített eredményét tintasugaras nyomtatóval vagy lézergravírozással rá lehet vinni az

ellenőrzött alkatrészekre. Ennek alapján a sorvezérlés továbbengedheti vagy kiszűrheti a darabot.



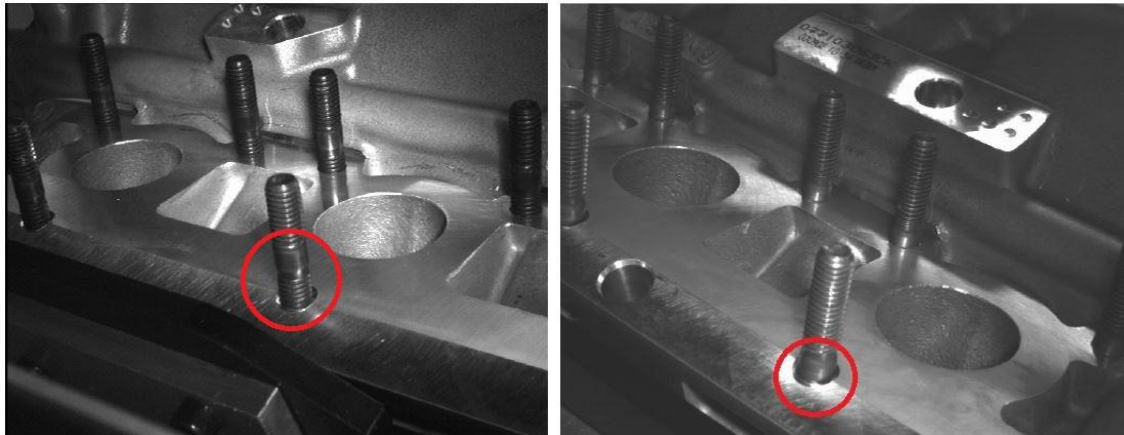
1. ábra Öntvény ellenőrzés[15]

Az autók lámpáinak gyártásánál is alkalmaznak ilyen rendszereket. Ezek jó-rossz jelzést tudnak adni a világítótestről. Helyzetet, irányt tudnak mérni, sőt a felmelegedés és lehűlés hatását is tudják vizsgálni. A kamerák a gyártásban vissza is csatolhatnak, a mérések eredményeivel vezérelhetik a léptetőmotorokat, amelyek a részegységeket a tűréshatáron belüli pozícióba mozgatják. A nap 24 órájában dolgozik egy ilyen rendszer, óránként 1500-2000 darabot is meg tud vizsgálni. Üvegcső gyártáskor is használhatóak ezek a rendszerek, ilyenkor falvastagságot, átmérőt, keresztmetszet kör alakját, illetve a belső és külső fal középpontja közötti távolságot is tudják mérni. Az üvegcső maximum 3m/s sebességgel tud mozogni, ± 5 mikronos pontosságú a mérés [15].



2. ábra Üvegcső gyártás[15]

Hengerfejek szerelésénél is alkalmaznak ilyen rendszereket. Például a kipufogó csonkot a hengerfejhez 12 csavar rögzíti. Ezeket kézzel kell behelyezni a hengerfejen lévő furatokba. Ha rossz az irány a csavarmenet helyén sima felület látható. A helyes irány az, ha a sima felület közvetlen a hengerfej felszíne felett van. Ez látható a 3. ábraán, amelynek bal oldala a helytelen, jobb oldala a helyes irányt mutatja. Ez kamerával könnyen ellenőrizhető.

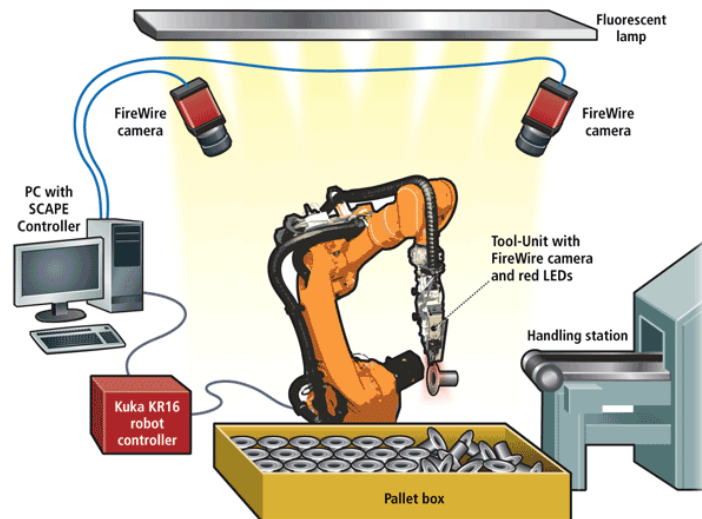


3. ábra Hengerfej csavarjainak ellenőrzése[14]

2.3. Korszerű képfeldolgozási alkalmazások az ipari robottechnikában

A modern gyárak törekednek a termék ellenőrzés gyorsítására, a ciklusidő csökkentésre úgy, hogy az minél kevesebb emberi erőforrást emésszen fel, ezt leginkább robotosítással érhetik el. A mai világban egyre inkább teret nyernek képfeldolgozási alkalmazások. A cél, hogy emberi beavatkozás nélkül a robot el tudja dönteni, hogy a termék selejt-e, és ennek megfelelően cselekedjen. A kamera több feladatot is el tud látni: jelenlétet érzékelés, kódleolvasás, alakfelismerés.

A hagyományos robotos rendszerekben problémát jelent az alkatrészek orientálása. Az alkatrész elhelyezkedését kell a robothoz igazítani, nem pedig fordítva. Ennek rossz kalibrálása ütközésekhez, törésekhez vezethet. Sok esetben olcsóbb telepíteni egy képfeldolgozó rendszert a robot mellé, mint különböző adagolókkal és készülékekkel a megfelelő helyzetbe állítsuk a darabot a robotnak. [8]

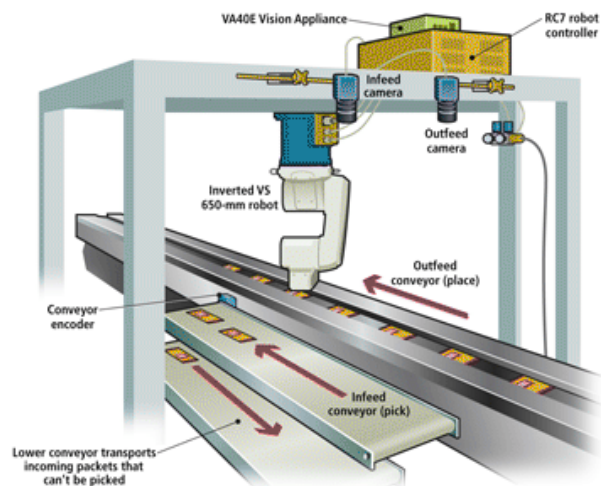


4. ábra Cella elemei [11]

2.4. Szállítószalagos adagolás

A laborban található SIEMENS intelligens kamerával kapcsolatos egyik tanulmánya egy villamos motor állórész vizsgálat volt. A telepített kamera 3,5 percről 35-40 másodpercre csökkentette a ciklus időt. A szállítószalagon érkező alkatrészeket a kamera megvizsgálta, ha helyes darabról volt szó, folytatódott a szerelés. A selejteknél jelzést adott az irányító személynek, aki megvizsgálta a hiba okát. [9]

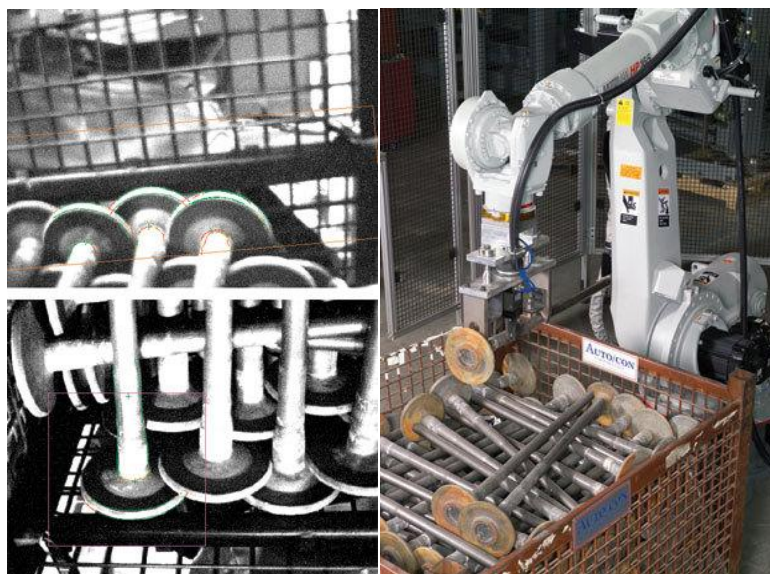
Szállítószalagos adagolásnál nem elég a megfelelő kamerát használni, külön algoritmust is kell használni a mozgás miatt. Itt a kamera/kamerák több képet készítenek (általában 10-20at), még az alkatrész elhalad a látómezejükben. Erre azért van szükség, hogy pontos képet kapjon a robot cella arról, hogy milyen pozícióban van az alkatrész, mivel ez a szállítószalagon változhat. Betanításkor elég egy 360°-os párosító algoritmus használata, amivel így az összes lehetséges pozíciót le is fedjük. Előfordulhat, hogy a két alkatrész egymáshoz túl közel kerül, a vezérlő egyszerűen kihagyja a 2. elemet, mivel azt a robot nem tudná felvenni. [10]



5. ábra Szállítószalagos rendszer [10]

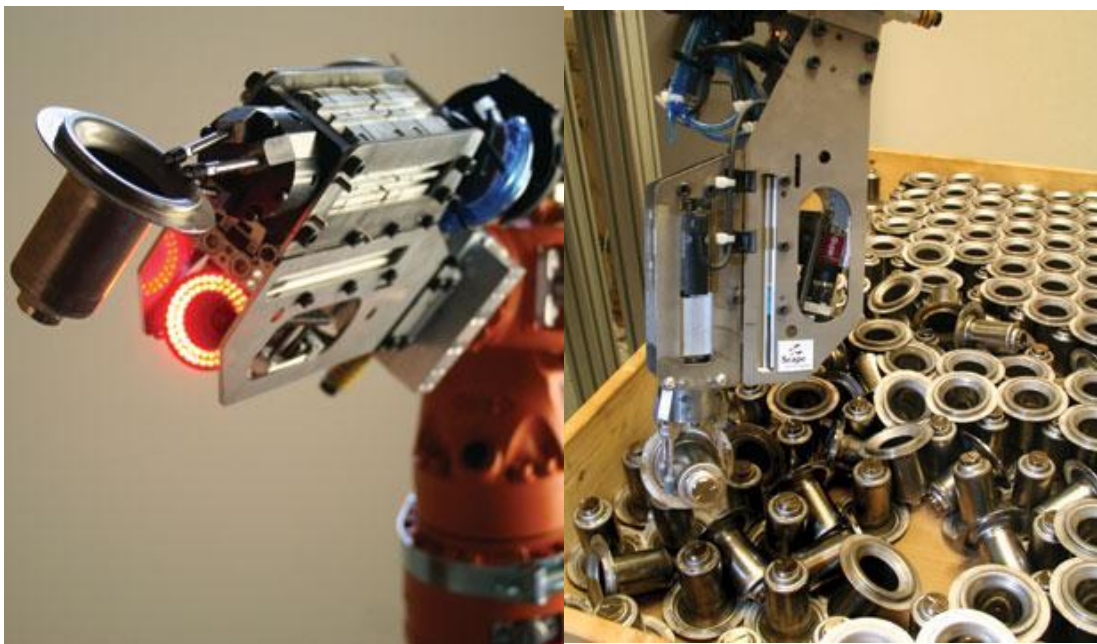
2.5. Ládából való adagolás

A kamera 2D-ben lát, viszont sok esetben, amit a robot megfog az egy térbeli elem. Az iparban erre általában a következő megoldást alkalmazzák: elkészítik az elem 3 dimenziós modelljét, ez lehet CAD file is. Egy program segítségével ezt tanítják be a kamera vezérlőjének, ami kulcs fontosságú feature-ket különít el, ez alapján megfogási stratégiákat készíthet a robot. A kamera ezután 2D-s kép alapján képes társítani a 3D-s modell valamely orientációjának képével. A különböző elhelyezkedéseket ezután a robot képes kezelni. Természetesen fontos a megfelelő megvilágítás, különben nem vagy tévesen kivehető kontúr elemeket találunk meg, ez igaz a szállítószalagos esetekre is. [11]



6. ábra Kontúr felismerés [12]

Gyakran előfordul réteges elrendezés a tárolóban, ahonnan a robot kiveszi az elemeket. Ilyen esetben a leghosszabb felismert éllel rendelkező elem az, amelyik biztosan legfelül van. A pontos felvétel érdekében minimum 2 kamerára van szükség. Az első kamera belátja az egész területet, és egy alkatrész felé irányítja a robotot. A második kamera ennek az alkatrésznek a pontos orientációját határozza meg. Általában egy egyedi megfogó szükséges ezekhez a feladatokhoz, hiszen a megfogási stratégiák különböző alkatrészeknél nagyon eltérőek lehetnek. A felismerés kisebb pontatlanságait figyelembe véve esetleges kis tájoló elemek is szükségesek lehetnek a megfogón, ami pontosan bevezeti az alkatrészt a rögzítő részhez. [12]

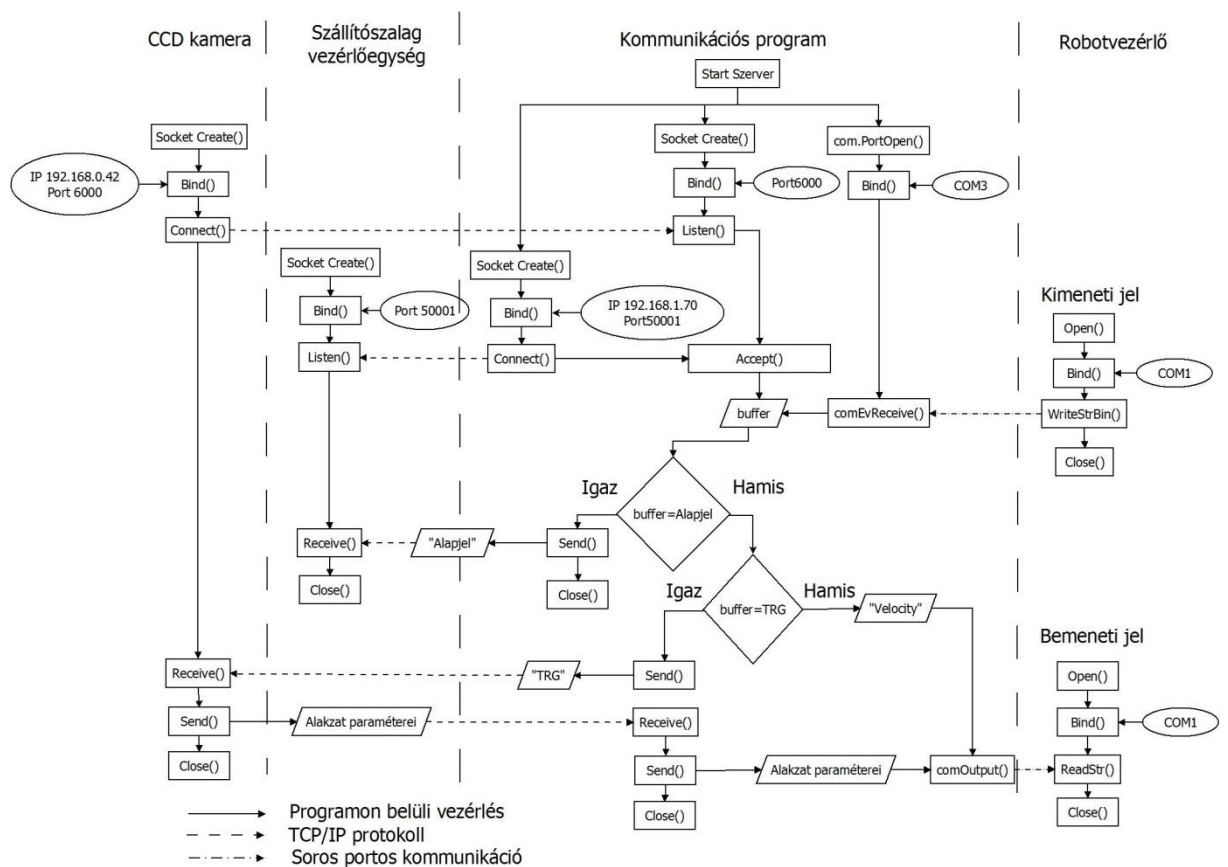


7. ábra Robotra felszerelt kamera a pontos helyzetfelismerésért [11]

3. Rendszer előkészítése

A minőségellenőrzési rendszer megépítését megelőzően szükség volt a szoftveres támogatás elkészítésére. Készítettem egy programot, amely mind a három eszközt összeköti: a robotot, a kamerát és a szállítószalagot. Ügyeltem arra, hogy az alkalmazásom biztosítsa a megfelelő információkat a felhasználónak. Egy, a feladatnak megfelelő-, robotprogramot is írtam, amely a selejtválogatás folyamatát el tudja látni.

3.1. Kommunikációs program



8. ábra Kommunikációs program architektúrája

A kommunikációs program alapja egy előző tanszéki projektben megtalálható [3]. A program fejlesztését Microsoft Visual Studio 6.0-ban Visual Basic nyelven végeztem el. Az alapprogram nem tartalmazta a szállítószalag vezérléssel való kapcsolatot. A szállítószalaggal kapcsolatos rutinok az én munkám eredményei. Az 8. ábrán látható a program architektúra vázlata. Ez a perifériák között futó jelek útját mutatja, az egyéb felhasználói opciókat nem jelölöm, az túlbonyolítaná az ábrát.

3.1.1. Perifériák kapcsolódása

A kamera beépített programja kliensként funkcionál. Ez azt jelenti, hogy a programom szerverként kell, hogy működjön. TCP/IP kapcsolatot Visual Basic 6.0-ban Winsock komponens használatával lehet programozni. Nyitni kell egy portot, ami a kamera esetében a 6000-es számú. Ezen kell figyelni a beérkező jeleket (Listen). Amint kapcsolódott, a program elfogadja ezt a kapcsolatot, és az adatforgalom szabaddá válik.

A robot a „TRG” jelet elküldi a számítógépnek, amit az továbbít a kamerának. Ennek hatására fényképek készülnek, amelyet az alakfelismerő program kiértékel. Így válaszul a vezérlés a sikeres vagy sikertelen találatok paramétereit küldi vissza a számítógépnek, amely azt továbbítja a robotvezérlő felé.

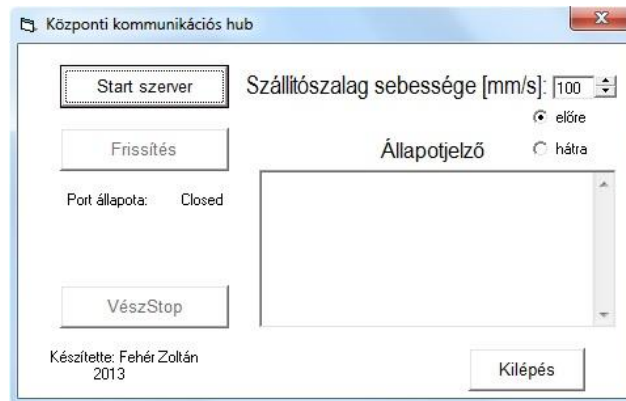
A szállítószalag vezérlőjének [2] programja szerverként fut. Itt is szintén TCP/IP protokoll programozást kell használni. A számítógép ehhez kliensként kapcsolódik a 192.168.1.70 IP cím 50001-es portján. Nincs szükség „Listen” funkcióra, mint a szerver oldalon. Itt a kliens sikeres vagy sikertelen kapcsolata azonnal látszik.

A szállítószalag vezérlő programja „figyelem;Speedtarget;checksum” jelet dolgoz fel, így változtatja a motor fordulatszámát. A „figyelem” egy # decimális értéke, ami 35-tel egyezik meg, a „Speedtarget” az „Alapjel” a robottól, a „checksum” a kettő összege. Ezeket „integer” értéként kell elküldeni. Az „Alapjel” a motor fordulatszáma ford/min mértékegységben. Az egyszerűség kedvéért, mivel csak a kívánt fordulatszám a változó, úgy készítettem el a programot, hogy a robotnak elegendő ezt az adatot kiküldenie, az adatcsomag többi részét a program kiegészíti a konstansokkal, és úgy küldi el a szállítószalag vezérlőnek. A szállítószalag vezérlése tudja az általa mért paramétereket visszaküldeni, de nekem ezekre nincs szükségem.

A robot nem képes TCP/IP kapcsolatra. Soros porti kommunikációval lett rákötve a számítógépre, amit a Visual Basic 6.0-ban MSComm komponenssel lehet programozni. Nyitni kell egy „COM3” csatornát, és meg kell adni a kapcsolathoz tartozó paramétereket. Ekkor a robot tud csatlakozni a számítógéphez „COM1” csatornán. Ezen keresztül küldi el a robot az „Alapjel”-et, a „TRG”-t és a „Velocity” jeleket.

3.1.2. Program kezelőfunkciói

A program grafikus kezelőfelületét a 9. ábra mutatja. Láthatóak a program gombjai és kijelzői. Ezeket fogom a következőkben bemutatni.



9. ábra Kommunikációs program

A program elindításához szükséges egy létező soros port kapcsolat. Jelen esetben ez a robot és a számítógép összekapcsolását és bekapcsolását jelenti. Miután a program elindult, lehetőség van a szállítószalag sebességének megadására, alapértékként 100 mm/s van megadva. Ez változatható a mellette lévő nyilakkal, illetve mi is adhatunk meg értéket, amelynek egész számnak kell lennie, és a (50,200) határon belül kell mozognia. A motor ezt a tartományt tudja kezelni. A szállítószalag vezérléshez az „Alapjel” 1/min mértékegységben van megadva, ezt a program a sebességből átszámolja. Majd leellenőrzi megfelelő értéket adtunk-e meg. Ha mindent rendben talál a „Start szerver” lenyomásával létrehozza a kapcsolatokat a perifériákkal. A szerver port állapotát is jelzi a program. Az Állapotjelzőben tájékoztat minket a program a történésekről: milyen jelek érkeznek a robottól és a kamerától, illetve, hogy a kamera melyik alkatrészt ismerte fel. „Stop” gombbal zárhatóak a kapcsolatok.

Miután elindult a program, a fordulatszám átírható, hiszen könnyebb így a szállítószalag sebességét beállítani. Miután átírtuk, nyomjuk meg a Frissítés gombot, különben nem küldi ki az új jelet a robotnak. A gomb lenyomásával a program egyből elküldi a szállítószalag vezérlőnek az „Alapjel”-et, kiegészítve a megfelelő adatokkal. Így biztosítva az azonnali sebességváltást. A robot, amint lekéri a jelet, hogy elindítsa a szállítószalagot, miután elvette onnan a selejt alkatrészt, már ezt a frissített értéket fogja majd megkapni. Ez természetesen csak akkor igaz, ha a szállítószalag fut, amikor a Frissítés gombot lenyomjuk. Ha áll a szalag, azonnali sebességváltást nem fog végrehajtani, csak az új jelet fogja majd kiküldeni a robotnak. Így a kísérletek kényelmesen lebonyolíthatók, bármikor is akarjuk a sebességet megváltoztatni.

Végül a biztonságra ügyelve, ha valami nem úgy működik, ahogy azt szeretnénk, leállíthatjuk a szállítószalag a VészStop gomb lenyomásával. Ez a gomb a kapcsolatot nem

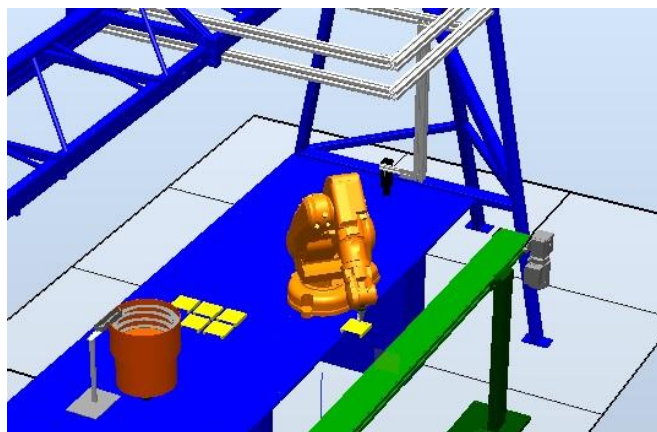
bontja fel a perifériák között, a Frissítés gomb újbóli használatával a szalag újra elindítható, és a robotprogram zavartanul futhat tovább.

3.2. Robot program

3.2.1. Szimuláció elkészítése

RobotStudióban [7] készítettem egy szimulációt. Mivel ez a program az ABB terméke, tartalmazta az általam használt ABB IRB 140 robotot. A szakdolgozatomban [1] készítettem már szimulációkat, az egyiket felhasználva készítettem el ennek a projektnek a feladatát. Az első gyakorlati feladatot lemodelleztem, ahol nyákok érkeznek.

Az 5.15.02 verziószámú RobotStudióban már van lehetőség alakfelismerő kamerával egybekötött rendszer programozására is. Nincs lehetőség viszont soros portos kapcsolatot létrehozni. Mivel a robotlaborban így valósul meg a kommunikáció, ezért csak az alkatrészmozgást és érzékelést, illetve a robotmozgást készítettem el. A 10. ábra a szimulációról mutat egy képet.



10. ábra Szimulációs környezet

3.2.2. 1. feladat robotprogramja

A program indulása után elküldi a „Velocity” jelet, amire a szállítószalag sebességből kiszámolt motorfordulatszámot kapja válaszul a számítógéptől. Ezután ezt „Alapjel”-ként visszaküldi, így elindítva a szállítószalagot. A szállítószalag sebesség a robotmozgástól függ, áll-e vagy elindul-e az adagolás, ezért egyszerűbb azt robot részről kezelni.

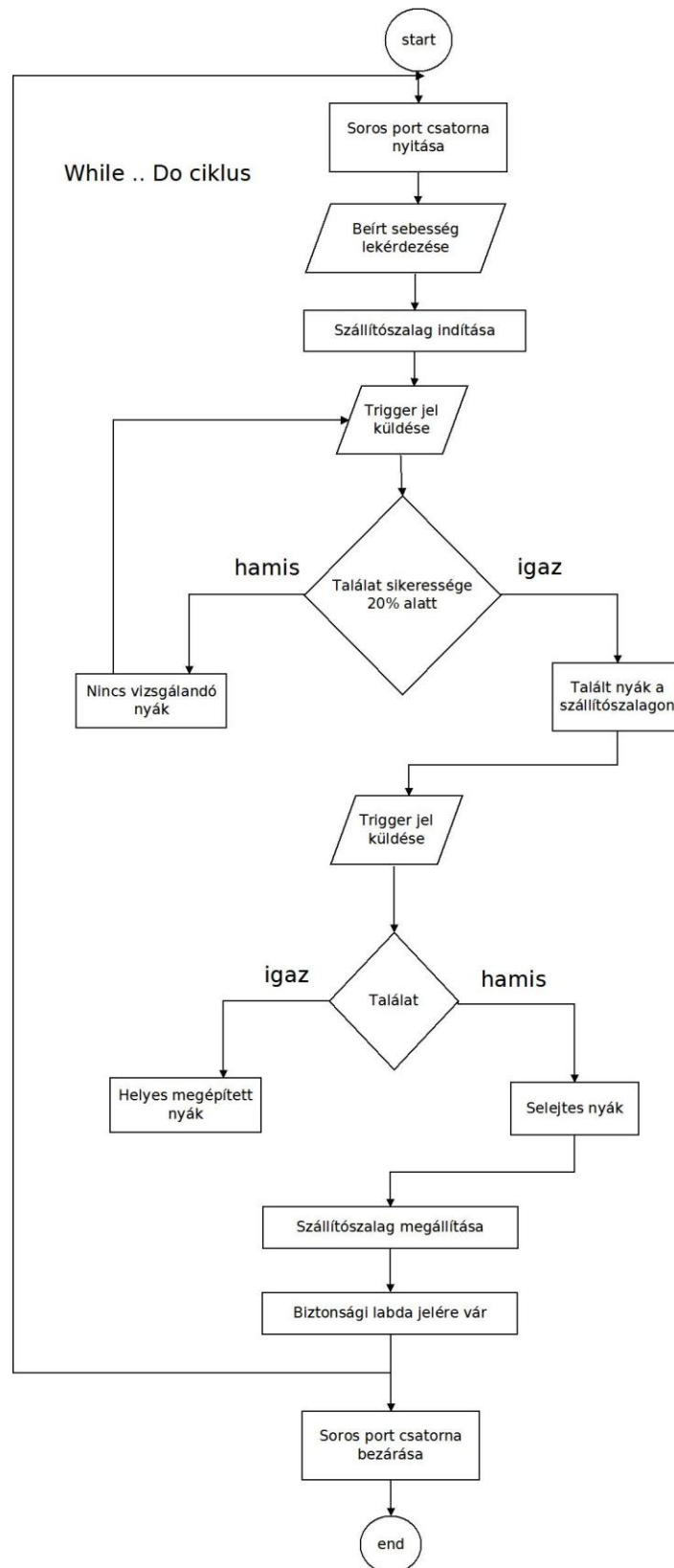
Ezután a robotvezérlő folyamatosan küldi a „TRG” jeleket, amelyre válaszul megkapja a képfeldolgozás eredményeit. Ebben szerepel, hány százalékos sikerrel találta meg az alkatrészt a kamera. Ezt feldolgozza a robotprogram. Kísérleteket végezve azt tapasztaltam, hogy a 20 és 75% közötti tartomány számítható selejtnek. Azért nem 0%, mert akkor a robot

mindig megállítaná a szállítószalagot, akár van rajta alkatrész, akár nincs. Az alakfelismerés azelőtt megtörtént sok esetben, mikor még nem ért be a kamera alá teljesen a darab. Ezért két lépésben oldottam meg a selejtszűrést. Ha 20% alatti a felismerés a szalag nem áll meg, folyamatosan küldi a „TRG” jeleket legfeljebb három másodpercig. Ha sikeresen felismeri az alkatrészt, a szalag nem áll meg. Ha nincs megfelelő mértékű egyezés, selejtes darabnak ismeri fel az alkatrészt, a szállítószalag megáll. Miután megvizsgáltuk a selejtes darabot, a biztonsági labda megnyomásával újra elindul a szállítószalag, esetleges frissített sebesség figyelembe vételével. A blokkdiagram a 11. ábra mutatja.

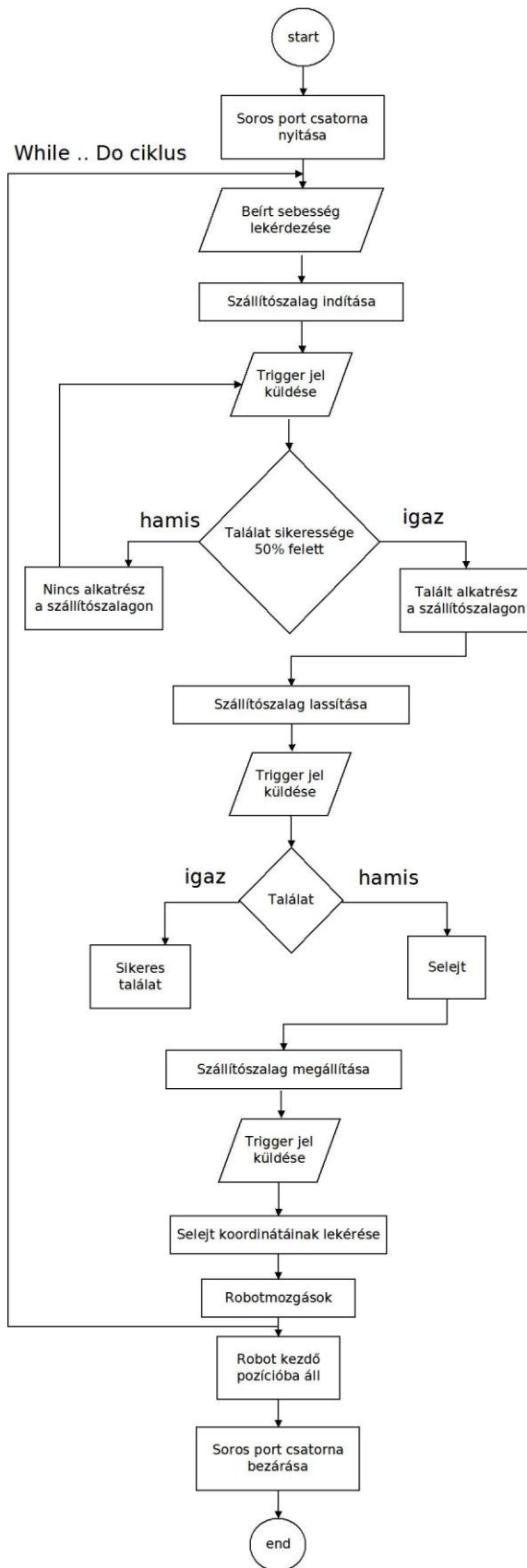
3.2.3.2. feladat robotprogramja

A program indulása után ismét lekérdezi a sebességet, és elindítja a szállítószalagot. A robotvezérlő folyamatosan küldi a „TRG” jeleket, majd a válaszokat feldolgozza. Hasonlóan az előző programhoz, itt is először 20%-os sikerességig folyamatosan küldi a robot a „TRG” jelet. Ezután lelassítja a szállítószalagot 100mm/s-ra, majd folyamatosan küldi a trigger jeleket legfeljebb három másodpercig. Sikertelen találat esetén a robot kiküldi a „0” jelet, ami megállítja a szállítószalagot. Miután megállt a szállítószalag, a pozíció frissítés céljából újra megismételi a képfeldolgozás ciklusát, hogy pontos paramétereket kapjon a vezérlés.

Ezután felveszi az alkatrészt a robot, majd a ciklus újraindul. Így biztosítva van, hogy sebesség frissítés esetén a robot mindig a legújabb paraméterrel dolgozzon. Ezután alaphelyzetbe visszaáll a robot, és lezáródik a soros port csatorna. A 12. ábrán látható a robotprogram blokk diagramja.



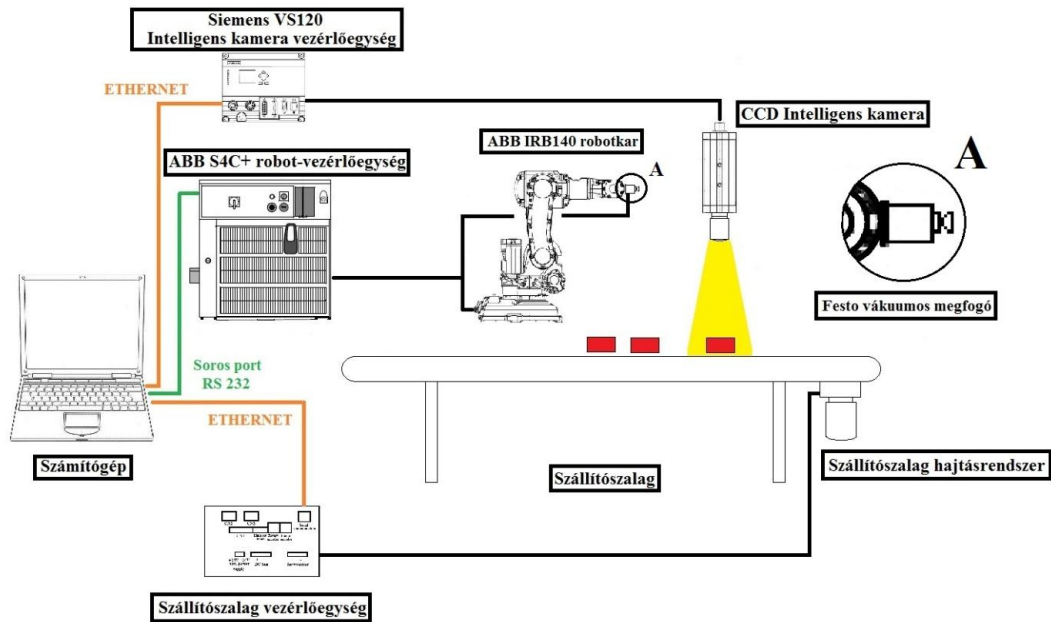
11. ábra 1. feladat robotprogram blokkdiagramja



12. ábra Robot program blokk diagramja

4. Rendszer megépítése

A BME Gyártástudomány és –Technológia Tanszék robot laborjában telepítettem a mintarendszert, ez látható a 14. ábra bal oldalán, a vezérlők a jobb oldalon láthatóak. Elkészítettem a cella sematikus rajzát, ami a 13. ábrán látható. Nem állt rendelkezésre megfelelő kamera állvány, így azt terveznem kellett. Beépítés előtt teszteltem a számítógépen futó alkalmazás adatforgalmát, illetve megvizsgáltam a kamera alakfelismerő algoritmusát.



13. ábra: A rendszer architektúrája



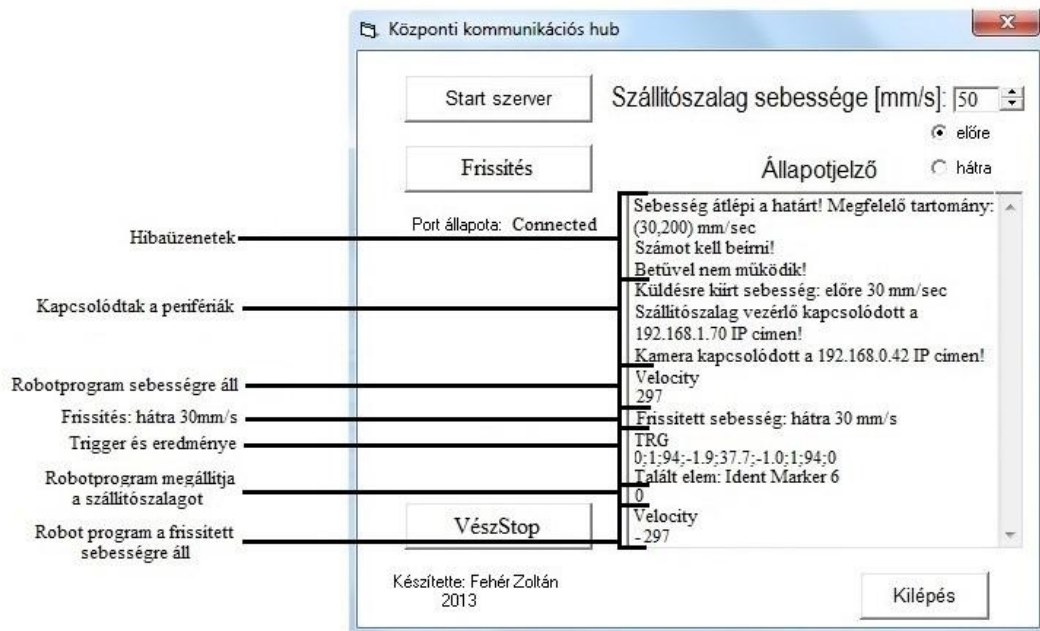
14. ábra A cella elemei

4.1. Kommunikációs program tesztelése

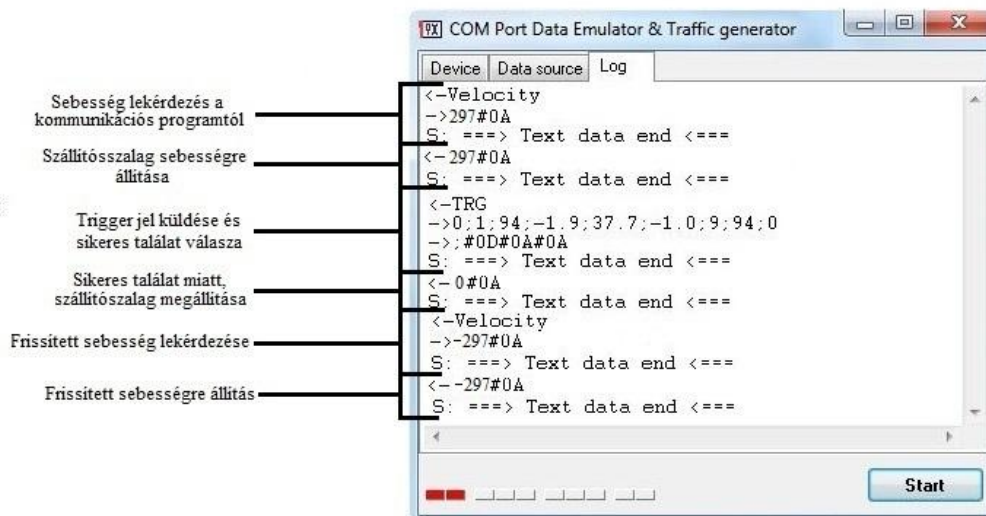
A kommunikációs programot teszteltem mielőtt beépítettem volna a rendszert a robot laborba. A robot és számítógép közötti adatforgalmat ellenőriztem.

Az AGGsoftware termékeit használtam. Virtual Null Modem-mel [4] szimuláltam egy COM1-COM3 kapcsolatot. COM1 a robot, COM3 pedig a kommunikációs program csatornája. COM Port Data Emulator [5] program segítségével pedig el tudtam küldeni a megfelelő jeleket a programnak, amik a robotprogramban is szerepelnek. A programom összes funkcióját teszteltem. Ezért biztos lehettem benne, hogy az a valós környezetben is megfelelően fog működni. A 15. ábrán látható tesztelés során, hogy futott a programot. Az ábra két részén ugyanaz az esemény látható felhasználói és robot oldalról. A kommunikációs programon kicsit változtattam, hogy a teljes szöveg látható legyen, a valóságban nem látható ekkora felület az állapotjelzőből.

Felhasználó tájékoztatása



Robot adatforgalma



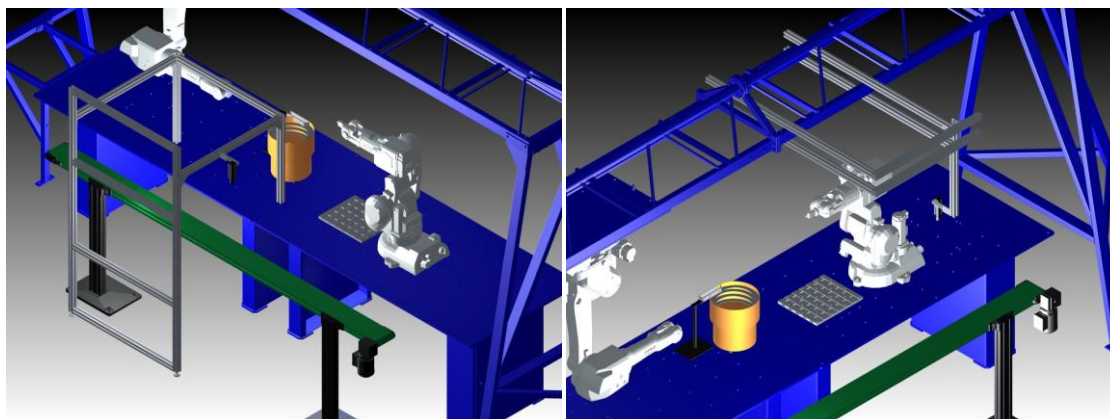
15. ábra Robot jelforgalmának tesztje és annak megjelenése a programban

4.2. Állvány tervezése

Szakdolgozatomban egy öntött vas alapú mobil állványt használtam, ami rendelkezésre állt a laborban. Kis behatásra is könnyen megbillen, elmozdul. Minden apró változás után újra kell kalibrálni a kamerát, ez nem megengedhető egy laborgyakorlat során. Ezért szükség volt

telepített állvány tervezésére. Két koncepciót készítettem el: egy asztal feletti állványra felszerelt változatot és egy padlóra-asztalra szerelt változatot.

A tanszéknél rendelkezésre álltak profil elemek még egy előző projektből. Ezeket használtam fel a tervezéshez. Elkészítettem az idomlécek 3D CAD modelljét, majd ezekből készítettem állvány összeállításokat. A 16. ábrán látható, hogy helyezkednek el ezek a labor modelljében.



16. ábra Két állvány változat modell térben

Miután elkészültek a tervek, megépítettem a prototípusokat a laborban (17. ábra). Látható, hogy az asztalon elhelyezkedő modell nagy teret foglal el. Az IRB 1400 robotkar mozgásterét nagymértékben korlátozza, és a használt IRB 140 robotkarnál is fenn áll az ütközés veszély. Későbbi vizsgálatom során kiderült, hogy a kamera sincs elég magasan, nem látja be az egész szállítószalagot. Megoldható lenne a kamera megfelelő elhelyezése, de az tovább növelné a méretet, instabilabb lenne. Az asztalon mozog a robot, a rezgések befolyásolják a kamera pontosságát.

A második verzió mellett döntöttem. Az állvány, amire fel van szerelve a szerkezet, független az IRB 140-től, így annak mozgása nem okoz rezgéseket a kamerának. Méretei kisebbek, négy ponton van rögzítve, így stabil. Nem befolyásolja a robotmozgását.

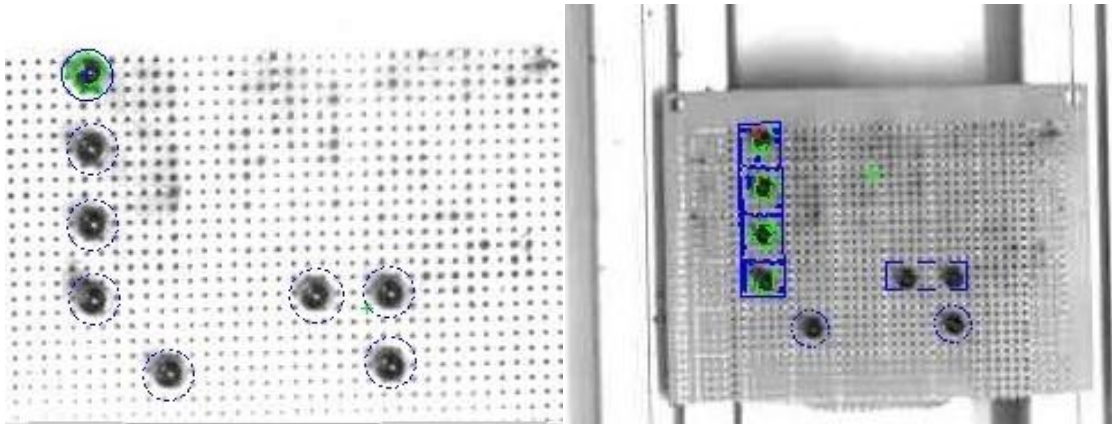


17. ábra Elkészült állványok

4.3. Kamera tesztelése

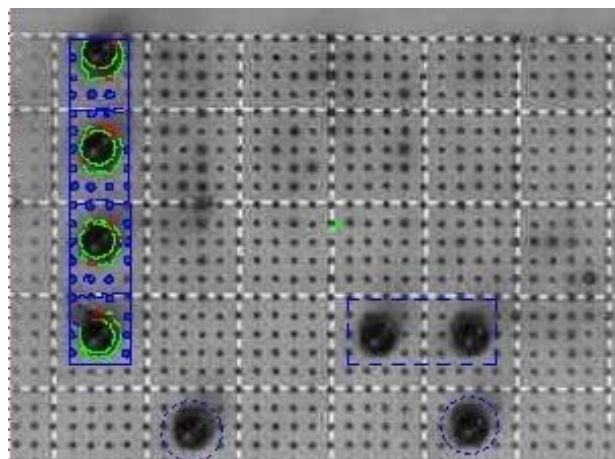
A kamera tesztelését egy általános vizsgálattal kezdtem, így más ezzel a kamerával kapcsolatos projekteknek nyújthatok segítséget. Különböző magasságokban különböző beállításokkal próbáltam ki álló helyzetből, milyen mértékben képes felismerni egy kis alkatrészt a kamera. Egy próbanyákon helyeztem el sötétkék infra LED-eket, igyekeztem a legnagyobb kontrasztot elérni, hogy könnyebb legyen az alakfelismerés.

Először 1500 mm magasról végeztem kísérleteket. Magas kontrasztot állítottam be, így emberi szemmel jól elkülönültek a LED-ek. Alacsonyabb kontrasztot is próbáltam, így a világítótestek nem látszódtak olyan jól, de a nyák egészét jobban lehetett látni. Az eredmények a 18. ábrán láthatóak. A bal oldali kép egy nagyított kép, hogy jobban kivehető legyen a felismert deformált alak. Alapvető problémát jelent, hogy egyik beállítási módnál se ismeri fel a LED-et körnek. Kör és négyzet alapú terület betanításával is próbálkoztam, de az ábrán látható, hogy nincs különbség a két mód között. Az eredeti betanított pozíciót ismeri fel helyesen, kismértékű forgatásra már nagymértékben esik a sikeres felismerés aránya.



18. ábra 1500 mm magas kamera beállítás

A sikertelen alakfelismerések miatt közelebb helyeztem a kamerát 700 mm-re (19. ábra). Itt nem volt gond a helyes alak felismerésével. Ezen a magasságon folytattam tovább a kísérleteket, amelyek a helyes betanítási stratégiára vonatkoztak.



19. ábra 700 mm magas kamera alakfelismerése

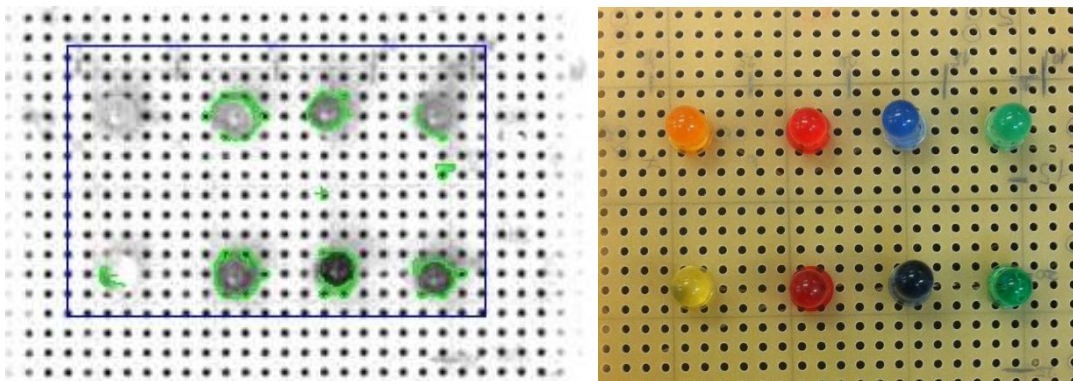
Több változatot is megvizsgáltam. Megpróbáltam a LED-eket egyesével betanítani. Ez a módszer túl érzékenynek bizonyult. Mozgás közben előfordulhatnak pontatlanságok, amik 1-1 LED-re levetítve lényegesek lehetnek. Annak ellenére, hogy nem volt tényleges hiba, selejtnek találta meg sok esetben a nyákot a kamera. Az összes LED-et egy ROI-ba (Range Of Interest) helyezve egy LED hibája túl ki súlylalt vett részt az alakfelismerésbe. Ezért egy hibás nyákot (1 világítótest hiányzott) is helyes alkatrésznek ismerte fel.

Így jutottam arra a következtetésre, hogy csoportokra bontva tanítom be a kamerának az alkatrészem. Többféle próbálkozás után a maximum hármas csoportok bizonyultak elfogadható megoldásnak. Három LED-nek 1 hiánya már 66%-ban helyes területet ad, hasonló rossz felismerést jelent az is, ha egy sorral elvétjük a világítótest elhelyezését. Ez jó,

hiszen nem kell az alakfelismerő programnak magas határt szabni, hogy mit ismerjen fel helyes darabnak. Így a mozgásból származó bizonytalanságot ki tudjuk szűrni. Ennek eredményeül, én a kísérletek során 75%-os hibahatárt használtam.

Az algoritmus sikeresen felismerte a különböző orientációjú és pozíciójú nyákat. Kipróbáltam hibásan összeállított elrendezést is. Rossz helyre, rossz irányba, ferde LEDeket is teszteltem. Egy blokkot elcsúsztatva helyeztem el, ekkor az elcsúsztatott csoportot helyesen ismerte fel, de a többi csoport helyét nem találta meg.

A piacon különböző fajtájú és színű LED-ek kaphatók. Én 5mm-es LED-eket használok a kísérletek során. A szín nagyban befolyásolja a sárga alapszínű nyákon a láthatóságot. Ezért a színekkel kapcsolatosan is végeztem vizsgálatot (20. ábra). Bal oldalt a kamera által látott kép, jobb oldalt a tényleges színek láthatók. A világosabb színek majdnem észrevehetetlenek a kamerának. Az infra-LED-ek sötétebb színeit könnyebben felismeri a program. Ezért a kék LED-ek mellett döntöttem a gyakorlati feladat kivitelezésekor.



20. ábra Szín felismerés

A kísérletek alapján nyilvánvalóvá vált, hogy ez a kamera nem használható nagy területek vizsgálatára. Az iparban sok esetben két kamerát alkalmaznak bin-picking rendszer telepítésekor. Ez a kamera a robotra szerelt, egy kiválasztott alkatrészt megvizsgáló feladatot tudna ellátni. A tároló egészét megvizsgáló kamera szerepére nem alkalmas. Szállítószalagos adagolás esetében alkalmas lehet a kamera. Széles szállítószalag esetén viszont szükség lehet terelőelemekre, hogy a kamerának megfelelő nagyságú területre korlátozódjanak az alkatrészek elhelyezkedései.

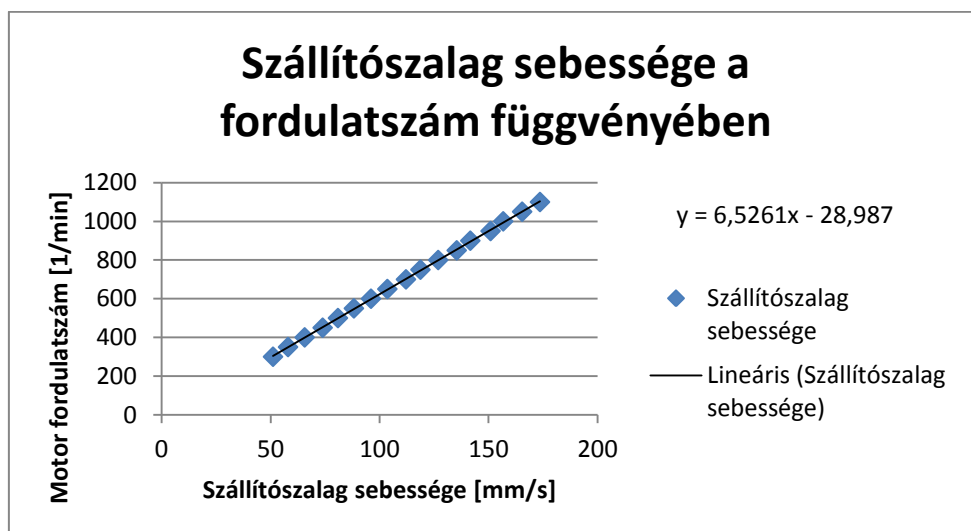
4.4. Szállítószalag sebességének meghatározása

A szállítószalag vezérléséhez szükséges „Alapjel” a fordulatszámot jelenti. Ha szállítószalagos adagolásról beszélünk, annak sebességét kell megadnunk. Ezért megmértem a

kettő közötti összefüggést. Kijelöltem két méteres szakaszt a szállítószalag hevederén, majd lemértem, mennyi idő alatt tette meg ezt a távot egy nyák, különböző fordulatszámokon. Mérésem eredményét az 1. táblázat tartalmazza. Ezt az 1. grafikonon is ábrázoltam, majd Excel program segítségével meghatároztam a hozzá tartozó függvényt.

	n [1/s]	t [s]	v [mm/s]
1	300	39,13	51
2	350	34,48	58
3	400	30,47	66
4	450	27,07	74
5	500	24,72	81
6	550	22,67	88
7	600	20,82	96
8	650	19,31	104
9	700	17,84	112
10	750	16,84	119
11	800	15,76	127
12	850	14,77	135
13	900	14,12	142
14	950	13,25	151
15	1000	12,76	157
16	1050	12,09	165
17	1100	11,52	174
18	1150	9,06	221
19	1200	8,95	223
20	1250	8,89	225
21	1300	8,96	223

1. táblázat: Szállítószalag sebességének mérése



1. grafikon: Szállítószalag sebessége a fordulatszám függvényében

Látható, hogy 1150 1/min fordulatszám felett állandósul az érték, és nagymértékben eltér a többi pont viselkedésétől, itt eléri a maximális fordulatszámot, ezeket a pontokat nem ábrázoltam, mert hibás függvényt kaptam volna. 50 és 200 mm/s között határoztam meg a program által engedélyezett sebességet. Ellenőrzésként az elkészült programmal visszafelé is elvégeztem a kísérletet. Sebességet állítottam be mérési pontoknak, és lemértem mennyi idő alatt tesz meg két métert egy nyák, az eredmények a 2. táblázatban láthatóak. A két mérés közel azonos eredményt ad, lassú sebességnél nagyobb a hiba, a mérési bizonytalanság ebben nagy szerepet játszik. Továbbá alacsony sebességnél a nagyobb a hibát a vezérlés hibája is okozza. A használt összefüggés elfogadható.

Program értéke [mm/s]	Idő [s]	Számolt sebesség [mm/s]	Hiba [mm/s]	Hiba [-]
200	10,17	196,66	3,34	0,02
190	10,79	185,36	4,64	0,02
180	11,35	176,21	3,79	0,02
170	11,93	167,64	2,36	0,01
160	12,63	158,35	1,65	0,01
150	13,53	147,82	2,18	0,01
140	14,26	140,25	-0,25	0,00
130	15,15	132,01	-2,01	-0,02
120	16,32	122,55	-2,55	-0,02
110	17,36	115,21	-5,21	-0,05
100	19,09	104,77	-4,77	-0,05
90	20,98	95,33	-5,33	-0,06
80	23,12	86,51	-6,51	-0,08
70	26,08	76,69	-6,69	-0,10
60	29,63	67,50	-7,50	-0,12
50	34,75	57,55	-7,55	-0,15

2. táblázat: Szállítószalag sebességének ellenőrzése

4.5. A rendszer kalibrálása és próbája

Vizsgálatom során a 700mm magasan elhelyezett kamera képen volt látható a teljes nyák. Itt minden apró hibát érzékelt az alakfelismerés. A laborban ez a távolság azt jelenti, hogy a VS120 nem látja be az egész szállítószalagot. Két lehetőség közül kellett választanom. A kamera jól kiszűri a hibát, de terelő elemeket kell használni a szállítószalagon, hogy minden nyákot meg tudjon vizsgálni a kamera, vagy magasabbra helyezett kamera kisebb pontossággal. A második opció mellett döntöttem, ebben az esetben az elhelyezkedést és

orientációt felismeri a kamera, viszont a ferdén behelyezett LED-eket nem. A 700mm magasság további hátránya megmutatkozott további vizsgálataim során is. Mivel a kamera egészben látja csak a nyákot, a trigger jel küldésének és a szállítószalag sebességének pontosan összehangoltnak kell lennie, különben két trigger jel küldése és feldolgozása között kiérthet a nyák a kamera látómezejéből. Ezért 1160mm magasan rögzítettem a kamerát.

A robot programot ehhez a magassághoz kellett igazítanom. Le kellett mérnem a megfelelő váltószámot pixel és mm között. Ezután a rendszer tökéletesen el tudta látni feladatát.

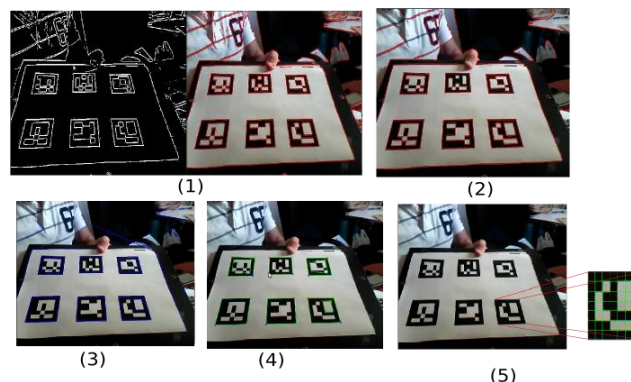
5. A gyakorlati feladatok vizsgálata

5.1. Első gyakorlati feladat: LED-ek beépítése és ellenőrzése

Ennek a feladatnak a megtervezésénél fontos szempont volt, hogy a hallgatók aktív résztvevői legyenek a gyakorlatnak. Egy adott mintázatot kell kirakni LED-ekből egy próba nyákon. A robot az előre betanított minták alapján megállítja a szállítószalagot hibás darab észlelése esetén. A selejt alkatrészeket a gyakorlatvezető segítségével megvizsgálják a hallgatók.

5.1.1. ArUco jelek

Mivel a gyakorlat ideje korlátozott, ezért egy egyszerű mintát kell majd a hallgatóknak lemásolni. Konzulensem az ArUco kódok használatában állapodtunk meg, amit már egy előző tanszéki projektben is használtak [6]. Ez egy képfeldolgozásban használt programkönyvtár. Egy 5x5-os négyzetrács által meghatározott marker, amely számokat kódol, 0-tól 1024-ig. Egy négyzet nagyjából 1 bitnek felel meg a Hamming kódban. Így a 0-át jelző marker fekete lenne, ezért az inverzét használják.

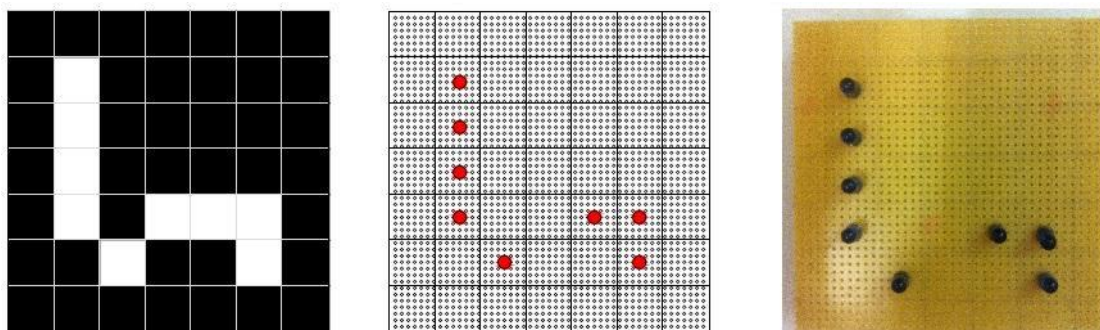


21. ábra ArUco markerek [6]

5.1.2. Feladat előkészítése

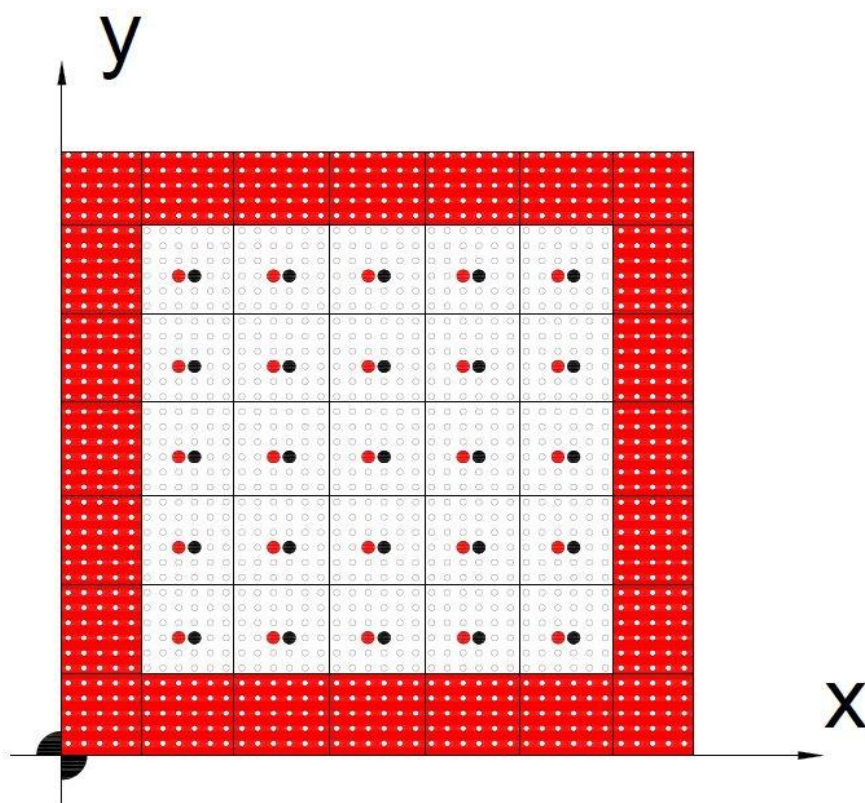
Megvizsgáltam mind az 1024 darab ArUcod markert, abból kiválasztottam 30 egymással csak kismértékben egyező mintát. Ezeket AutoCAD programban megrajoltam.

Készítettem egy próbanyák CAD rajzot is, hogy megmutassam, a hallgatóknak mit kell elkészíteni. A markerek egy 7x7 négyzetből álló területen helyezkednek el. A nyákterveket is felosztottam ekkora részekre, ahol fehér kód van, oda LED-et helyeztem.



22. ábra A feladat alapja és megoldása

A feladat végrehajtását az ID 6 markeren keresztül mutatom be (22. ábra). Kétféle megoldás került mérlegelésre, mindkét verzió abszolút koordináta rendszerben értendőek. Az első, hogy a hallgatók a nyák furatait megszámlálva tudja elhelyezni a kívánt koordinátára a LED-et. A második lehetőség mm-ben megadott koordinátákat lemérve végzik el az összeszerelést. A számoláshoz és méréshez szükséges koordinátarendszert a 23. ábra mutatja. Az ábrán egy mesterdarab látható, amelyben minden lehetséges helyen elhelyeztem egy LED-et. A hallgatóknak a feladata, hogy a gyakorlatvezető által kiosztott minták alapján megépítsék a kapott feladatot. A négyzetekre felosztott nyákon látható pirossal megjelölt száraznak (23. ábra) a koordinátái a hasonlóképpen felosztott 2. táblázatban láthatóak. A négyzetben lévő felső számok az (x,y) koordináták mm-ben, az alsók pedig a furat darab számaiban értendőek. A piros területre nem kell LED-eket helyezni, ott a nyák talpazatának ragasztása helyezkedik el. Úgy gondolom, hogy a mm-ben megadott koordináták közelebb állnak a mérnökséghez, ezért ezt a megoldást választottam. Próbák során körülbelül 5 perc alatt meg lehetett egy mintát építeni.



23. ábra Első feladat koordinátarendszerbe helyezve

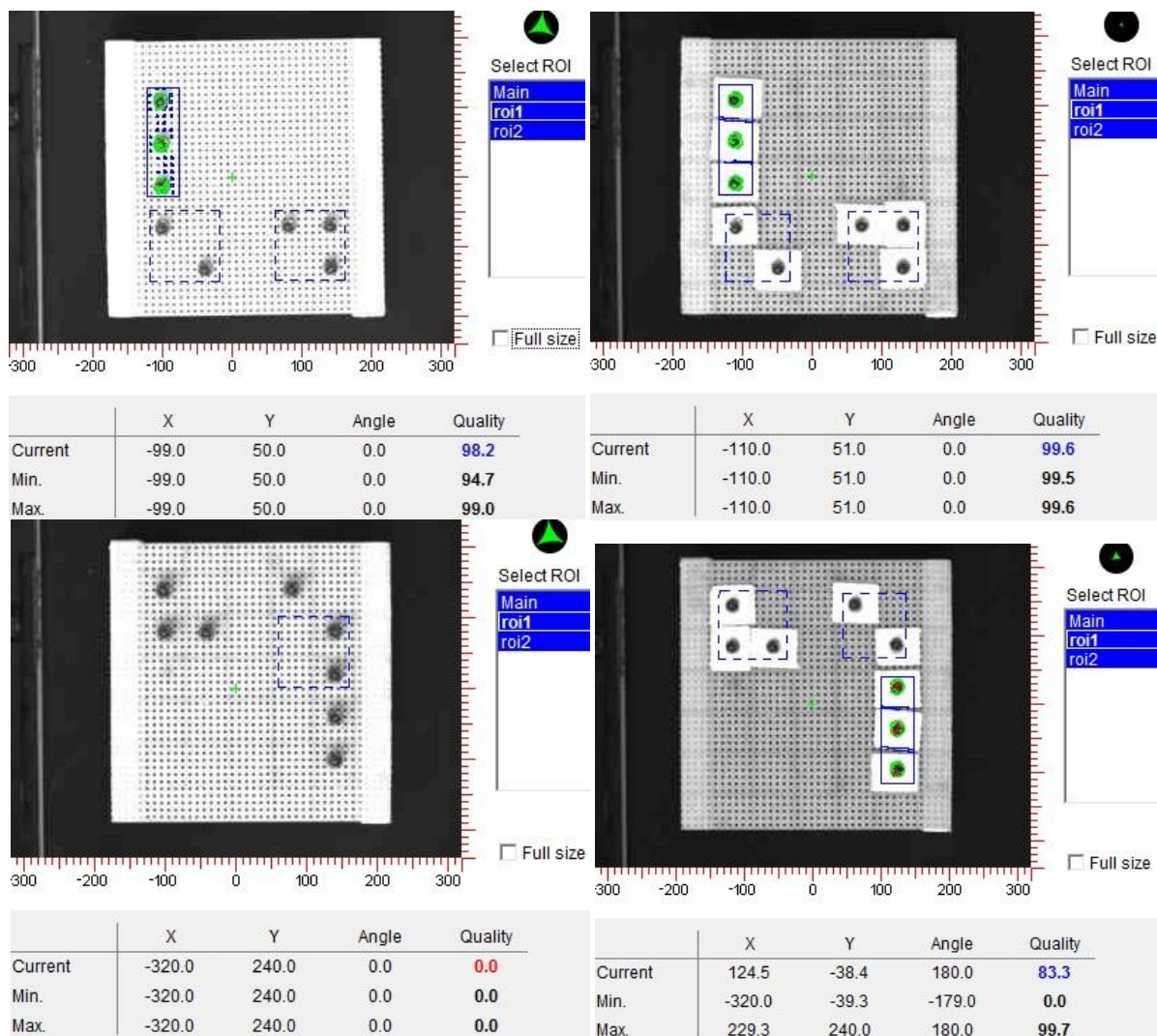
	18,78	33,78	48,78	63,78	78,78	
	8,31	13,31	19,31	25,31	31,31	
	18,63	33,63	48,63	63,63	78,63	
	8,25	13,25	19,25	25,25	31,25	
	18,48	33,48	48,48	63,48	78,48	
	8,19	13,19	19,19	25,19	31,19	
	18,33	33,33	48,33	63,33	78,33	
	8,13	13,13	19,13	25,13	31,13	
	18,18	33,18	48,18	63,18	78,18	
	8,7	13,7	19,7	25,7	31,7	

3. táblázat Első feladat koordinátái

5.1.3. Feladat vizsgálata

Vizsgálataim során kiderült, hogy a nyák betanítása problémás. A sok furat miatt magas a kép zaj aránya. Sok esetben forgatás hatására nem sikerült mindig sikeres eredményt kapnom, annak ellenére, hogy a darab nem volt selejtes. Mivel az eredmények zavarosak voltak, a robotprogram se volt képes minden esetben tudta megfelelően eldönteni az alkatrész helyességét. Ezért változtatnom kellett a nyákon.

A problémát a zaj szűrésével orvosoltam. Egy fehér lapokat helyeztem a LED-ek alá. A fehér színű háttér segítségével sikerült megfelelően definiálni a világítótestek körvonalát. Az összehasonlítást a 24. ábra mutatja. A betanítási stratégia és a kamera beállítása hasonló, csak a fehér fedésben tér el a két minta. Ugyanaz a ROI látható mindegyik ábrán. Látható, hogy a fehér háttér mennyire segíti a felismerést.



24. ábra A két megoldás közti különbség

5.1.4. Feladat tesztelése

A robotprogram a 3.2.2. fejezetben leírtak alapján készítettem el. Majd teszteltem is, milyen sikerrel találja meg a hibás darabokat. Megvizsgáltam továbbá, hogy eredetihez képesti elfordulási szög befolyásolja-e a mérést, vagy nem. Csoportos vizsgálatot végeztem, a kamerának két nyákot kellett megkülönböztetnie, és ellenőriznie. Minden szög esetén négyszer mértem le ugyanazt a nyákot. Selejtet és jó darabot is mértem. 100mm/s-es szállítószalag sebesség mellett végeztem el a kísérleteket. Ennek eredményei a 4. táblázatban láthatóak. A 64 mérés 90%-a helyes felismerés volt, ezért ez a program használható gyakorlat során.

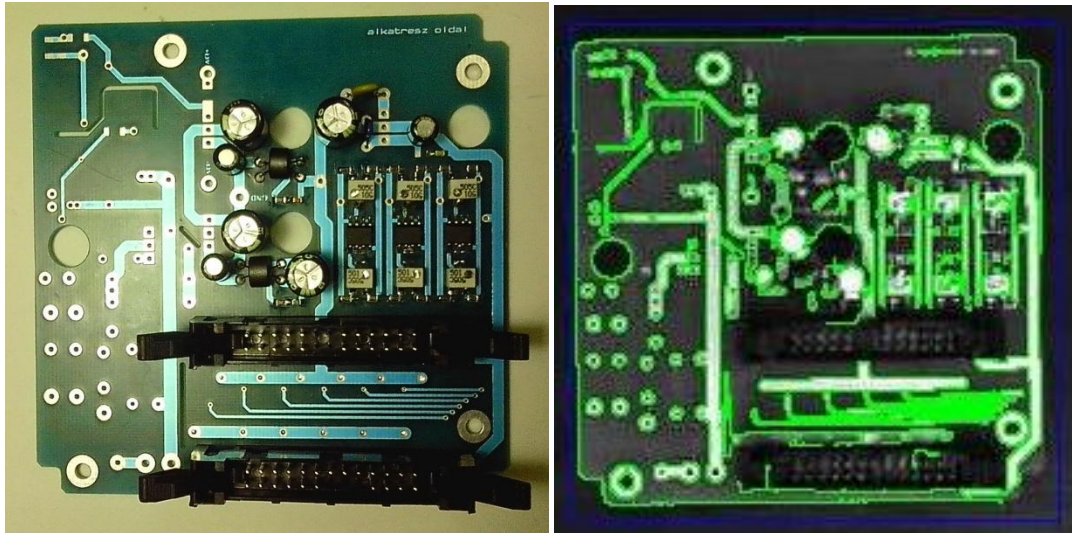
	Forgatás szöge [fok]	Vizsgált darab	Helyes felismerés Marker ID 6	Siker százalék Marker ID 23	Helyes felismerés Marker ID 89	Siker százalék Marker ID 395
jó	0	4	4	100	3	75
	45	4	4	100	4	100
	90	4	4	100	3	75
	135	4	2	50	3	75
	180	4	3	75	4	100
	225	4	4	100	3	75
	270	4	4	100	2	50
	315	4	3	75	3	75
selejt	0	4	4	100	3	75
	45	4	4	100	3	75
	90	4	4	100	3	75
	135	4	3	75	4	100
	180	4	4	100	4	100
	225	4	2	50	3	75
	270	4	4	100	2	50
	315	4	4	100	4	100

6. táblázat: 1. feladat vizsgálata

6.1.1. Második gyakorlati feladat: Optimalizálás

A második feladat egy robotos selejtszűrést mutat be. A hallgatóknak használniuk kell a kamera beépített programját. A biztonsági labdák segítségével tudják irányítani, mikor rakja le a selejtet a robot illetve, hogy mikor induljon az ellenőrzés. Be kell tanítaniuk egy nagysorozatban gyártott ipari elektronikai alkatrészt, egy teljes hidas erőmérő szenzorhoz gyártott erősítő panelt. Ezeket a CNC Part Bt. [16] gyártja. Tíz darab kerül ellenőrzésre,

amelyből egy hibás van. A panelból egy mintadarab készült el a dolgozat leadásáig. Megvizsgáltam a kamera mennyire részletesen ismeri fel a alkatrészt. A panel és az alakfelismerés eredménye a 24. ábrán látható.



25. ábra Második feladat során használt jelerősítő panel

A két darab fekete csatlakozó elemet nem ismeri fel, ennek oka az lehet, hogy ezek a részek magasabbak a többi elemnél, ezért a fókuszált részből kiesnek. A kamera nem ismerte fel továbbá a forrasztási pontokat. Érzekelte az összes többi elektronikai alkatrész kontúráját és az azokat összekötő pályákat is. Az ellenőrzés során használható a kamera ezen alkatrészek jelenlétének vizsgálatához.

A robotprogramot is teszteltem, de a panel helyett, hengereket használtam. A 3.2.3. fejezetben leírthoz hasonló a program, de ebben az esetben a sikeresen megtalált darabokat vettem fel, mivel a hengerekből nincs hibás darab. Így vizsgáltam meg, hogy a robotprogram megfelelően tudja kezelni a kamerát és a szállítószalagot. A kamera a szállítószalag maximális sebességekor (200mm/s) is képes helyesen felismerni az 1. feladat nyákjait. A panelek helyett nyákokkal vizsgálgódtam, mert ezek hasonló méretűek. Amint elkészülnek a panelek, a robotprogramot selejtkezelésre át fogom alakítani.

A sebesség automatikus szabályozás a kamera észleléseire alapoztam. Kísérleteim során kiderült, hogy a cella elemei között nem elég gyors a kommunikáció számítógépen keresztül. Az alkatrész érzékelés és a szállítószalag lassítása között annyi idő telik el, hogy az alkatrész kikerül a kamera látómezejéből. Ezt is nyákokkal teszteltem. Ezért a sebességszabályozást jelenlét érzékelő telepítésével fogom megoldani. Az optimalizálás feladat ezután kerül kivitelezésre.

7. Összefoglalás

Dolgozatom célja egy új laborgyakorlat tervezése volt, melynek címe: „Gyártásközi termékellenőrzés és -kezelés intelligens kamerával és ipari robottal”. Kutatásom az iparban alkalmazott képfeldolgozó rendszerek tanulmányozásával kezdtem. Kitértem autóiipari és robotos alkalmazásokra.

A robotos cella megépítését a szoftveres háttér elkészítésével kezdtem. Elkészítettem egy olyan programot, amely biztosítja a robot, a kamera és a szállítószalag közötti kommunikációt. Igyekeztem felhasználóbarát programot készíteni, amelyik segíti a kísérletezések könnyebb végrehajtását. A gyakorlat feladataihoz is elkészítettem a robotprogramokat, és egy szimulációt is.

A rendszer telepítéséhez terveznem kellett egy rögzített állványt, amely a kamerát tartja. Készítettem 3D modelleket prototípusokról, és ezeket meg is építettem. Végül az állványra szerelt kivitel mellett döntöttem, ez nem zavarja a robot mozgásterét. Teszteltem továbbá a kamerát, és hosszabb távú következtetéseket vontam le, amelyeket jövőbeli projektekhez fel lehet használni majd. Meghatároztam a szállítószalag sebessége és motor fordulatszáma közötti összefüggést is. A programomban 50 és 200 mm/s közötti tartományra korlátoztam a sebességet.

Miután elkészült a rendszer megvizsgáltam a hallgatók feladatait is. Első feladatuk LED-ekből 2D-s markerek megépítése próbanyákon. Ezt képfeldolgozás segítségével a szállítószalagon ellenőrzi a robot, selejt esetén megáll a szállítószalag, és a gyakorlatvezető segítségével megvizsgálható az alkatrész. A robotprogram ellenőrző képességét teszteltem. A kísérlet eredményeként megbizonyosodtam, hogy a program használható ennél a feladatnál, mivel 90%-ban sikeresen különítette el a selejtet és jó alkatrészeket.

Második feladat során egy nagy sorozatszámú gyártott elektronikai alkatrészt kell a robotnak ellenőrizni. 10 darabból 1 selejt, a hallgatók feladata a Siemens intelligens kamera betanítása. A panelek nem készültek el időben, de az előzetes vizsgálatok során megbizonyosodtam, hogy a robotos cella alkalmas lesz a feladatra.

8. Summary

The aim of my thesis is to construct a new laboratory course named: Inmanufacturing product monitor and –handling with intelligent camera and industrial robot. I started my research with studying the industrial vision systems. I discussed the automobile industry and robotic application.

I started the building of the robotic cell with creating the software background. I created a program, which can ensure the communication between the robot, camera and the conveyor. I tried to create a user friendly program, which helps performing the experiments easier. I also made robot programs and a simulation for the course tasks.

I had to plan a fixed frame for the system, which holds the camera. I created 3D models about the prototypes, and i built these. I decided at the frame-fixed type, it does not disturb the robot movement area. I tested the camera, and made long term conclusions, which can be used in projects in the future. I determined the function between the speed of the conveyor and the number of rotations of the motor. In my program I limited the speed from 50 to 200 mm/s.

After I have finished building the system I examined the tasks of the students. Their first task is to build 2D markers from LEDs into PCB test boards. The robot checks these on the conveyor with the help of the vision camera. If it is a scrap, then the conveyor stops, and with the help of the leader of the course the students examine the piece. I tested the ability of checking of the robot program. The result of the experiment was that the robot program can be used in this exercise, because it managed to separate the scrap and the good pieces in 90% of the cases.

During the second task the robot has to examine a electronic part manufactured in large quantity. From 10 pieces 1 is scrap, the task of the students is to train it to the Siemens intelligent camera. The panels have not been finished in time, but from the prior experiments it is verified that the robotic cell will be suitable for the task.

9. Irodalomjegyzék

- [1] Fehér Zoltán: Intelligens kamera alapú robotos alkatrész-adagoló fejlesztése; Szakdolgozat; Budapest; 2012
- [2] Bauer Márk Ottó: Szállítószalag sebességszabályozásának fejlesztése; Szakdolgozat; Budapest; 2013
- [3] Albrecht Kornél: Robotos szerelés támogatása CCD kamerás alakfelismerő rendszerrel; Diplomamunka; Budapest, 2011
- [4] <http://www.virtual-null-modem.com/>
- [5] <http://www.aggsoft.com/>
- [6] Kovács Dénes, Magyar Bence: ACROBOTER beltéri robot lengő egységének helyzetmeghatározása gépi képfeldolgozás és infra LED markerek alkalmazásával, Budapest, 2012
- [7] ABB Operation Manual Robotstudio; 2008-2011; Document ID: 3HAC032104-001; Svédország
- [8] http://www.fanucrobotics.com/robotics-articles/Part_Feeding_Solutions.aspx
- [9] Beez Maschinenbau: In the Eye of Progress, német verzió publikálva: inspect 1/06, Németország, 2008
- [10] Winn Hardin: Productive Line, <http://www.vision-system.com>, Május 1, 2009
- [11] Emma Clowting: Quick Pick, <http://www.vision-systems.com>, Szeptember 1, 2008
- [12] Winn Hardin: Vision Enables Freestyle picking, <http://www.vision-system.com>, Június 1, 2009
- [13] SIEMENS SIMATIC VS120 Image Processing Systems; Operating Instructions Edition 02/2006; A5E00757507-01
- [14] Kőhalmi József: Képfeldolgozás az autóiparban, Magyar Elektronika; 2003612
- [15] www.falkon.hu
- [16] <http://www.cncpart.hu/>
- [17] Dr. Mátyási Gyula, Tóth András, Gyimóthy Dávid: L13 Gyakorlat: Robotprogramozás, Budapest, 2012

10. Melléklet

1) 1. feladat robotprogramja:

```
%%%  
VERSION:1  
LANGUAGE:ENGLISH  
%%%  
  
MODULE Module1  
  
    VAR num X:=0;  
  
    VAR string received_string;  
        VAR bool bOK;  
    VAR bool bOK1;  
    VAR num TPAnswer;  
    VAR iodev ComChannel;  
    VAR string speed;  
    VAR num OK:=0;  
    VAR num success:=0;  
    Var num i:=0;  
  
PROC main()  
    velocity;  
    WHILE X<>1 DO  
        waittime 0.1;  
        Identification;  
        if success=0 then  
            WaitDI DI11_10,1;  
        endif  
        velocity;  
        Waittime 1;  
    ENDWHILE  
ENDPROC  
  
Proc Identification()  
  
    OK:=0;  
    While OK<40 Do  
        Open "COM1",ComChannel\Append\Bin;  
        ClearIOBuff ComChannel;  
        WriteStrBin ComChannel,"TRG";  
        Close ComChannel;  
        Open "COM1",ComChannel\Read;  
        received_string:=ReadStr(ComChannel);  
        ClearIOBuff ComChannel;  
        Close ComChannel;  
        TPErase;  
        bOK1:=StrToVal(StrPart(received_string,5,2),OK);  
  
    ENDwhile  
  
    success:=0;  
    i:=0;  
    While i<5 DO  
        if success=0 then  
            Open "COM1",ComChannel\Append\Bin;  
            ClearIOBuff ComChannel;  
            WriteStrBin ComChannel,"TRG";
```

```

        Close ComChannel;
        Open "COM1",ComChannel\Read;
        received_string:=ReadStr(ComChannel);
        ClearIOBuff ComChannel;
        Close ComChannel;
        TPErase;
        bOK:=StrToVal(StrPart(received_string,3,1),success);
        IF success=0 THEN
            TPWrite "Sikertelen alakfelismerés!";

            ELSE
            TPWrite "Sikeres alakfelismerés!";

        ENDIF
    endif
i:=i+1;
Endwhile

If success=0 then
    Open "COM1",ComChannel\Append\Bin;
    ClearIOBuff ComChannel;
    WriteStrBin ComChannel,"0";
    Close ComChannel;
    TPErase;
endif

ENDPROC

PROC velocity()
    Open "COM1",ComChannel\Append\Bin;
    ClearIOBuff ComChannel;
    WriteStrBin ComChannel,"Velocity";
    Close ComChannel;
    Open "COM1",ComChannel\Read;
    speed:=ReadStr(ComChannel);
    ClearIOBuff ComChannel;
    Close ComChannel;
    TPErase;

    Open "COM1",ComChannel\Append\Bin;
    ClearIOBuff ComChannel;
    WriteStrBin ComChannel,speed;
    Close ComChannel;
    TPErase;
ENDPROC

ENDMODULE

```

2) 2. feladat robotprogramja hengekre:

```

%%%
VERSION:1
LANGUAGE:ENGLISH
%%%

MODULE Module1
    PERS wobjdata object:=[FALSE,TRUE,"",[[500.8,452.2,22.3],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    PERS wobjdata target:=[FALSE,TRUE,"",[[452,-300,150],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    CONST ROBTARGET
    calib:=[[570,150,699],[0.5,0,0.866025404,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];

```

```

CONST robtarget object_near:=[[0,0,50],[0,0.707106781,0.707106781,0],[-2,-1,-
1,7],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget object_aim:=[[0,0,0],[0,0.707106781,0.707106781,0],[-2,-1,-
1,7],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget object_near_2:=[[0,0,100],[0,0.707106781,0.707106781,0],[-2,-1,-
1,7],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget target_near:=[[0,0,45],[0,0.707106781,0.707106781,0],[0,-
1,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
PERS tooldata vakuumos_megfogo=[TRUE,[[ -
6,0,71],[1,0,0,0],[0.1,[0,0,100],[0,1,0,0],0,0,0]];
CONST robtarget target_aim:=[[0,0,0],[0,0.707106781,0.707106781,0],[0,-
1,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
VAR jointarget kezdet:=[[0,0,0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

VAR num X:=0;
VAR string received_string;
VAR num success;
VAR bool bOK;
VAR num found{5};
VAR num len{4};
VAR num TPAnswer;
VAR iodev ComChannel;
VAR NUM Amount;
VAR num XOffs:=0;
VAR num YOffs:=0;
VAR num Angle:=0;
VAR num Ident:=0;
VAR string text;
VAR num Xf:=0;
VAR num Yf:=0;
VAR string speed;
VAR num Y:=0;

PROC main()
MoveL calib,v1000,z100,vakuumos_megfogo\WObj:=wobj0;
Confl\Off;
WHILE Y<>1 DO
    eltolas;
    WaitDI DI11_11,1;
    velocity;
    waittime 0.1;
    Identification;
    WaitTime 0.1;
    GetVisionData;
    Waittime 0.1;
    MoveL
Reltool(object_near_2,XOffs,YOffs,0),v500,z100,vakuumos_megfogo\WObj:=object;
Confl\Off;
MoveL
Reltool(object_near,XOffs,YOffs,0),v500,z100,vakuumos_megfogo\WObj:=object;
WaitDI DI11_10,1;
MoveL
Reltool(object_aim,XOffs,YOffs,0),v50,fine,vakuumos_megfogo\WObj:=object;
SetDO DO11_9,1;
WaitTime 1;
MoveL
Reltool(object_near,XOffs,YOffs,0),v50,z100,vakuumos_megfogo\WObj:=object;
MoveL
Reltool(object_near_2,XOffs,YOffs,0),v500,z100,vakuumos_megfogo\WObj:=object;
Confl\Off;

```

```

MoveL Reltool(target_near,Xf,Yf,0),v500,z100,vakuumos_megfogo\WObj:=target;
Confl\Off;
MoveL Reltool(target_aim,Xf,Yf,0),v50,fine,vakuumos_megfogo\WObj:=target;
SetDO DO11_9,0;
WaitTime 1;
MoveL Reltool(target_near,Xf,Yf,0),v50,z100,vakuumos_megfogo\WObj:=target;
X:=X+1;
ENDWHILE
MoveAbsJ kezdet,v100,z5,tool0;
ENDPROC

```

```

Proc Identification()
success:=0;
WHILE success=0 DO
    Open "COM1",ComChannel\Append\Bin;
    ClearIOBuff ComChannel;
    WriteStrBin ComChannel,"TRG";
    Close ComChannel;
    Open "COM1",ComChannel\Read;
    received_string:=ReadStr(ComChannel);
    ClearIOBuff ComChannel;
    Close ComChannel;
    TPErase;
    TPWrite "Eredmény:"+received_string;
    bOK:=StrToVal(StrPart(received_string,3,1),success);
    IF success=1 THEN
        TPWrite "Sikeres alakfelismerés!";
        Open "COM1",ComChannel\Append\Bin;
        ClearIOBuff ComChannel;
        WriteStrBin ComChannel,"0";
        Close ComChannel;
        TPErase;
    ELSE
        TPWrite "Sikertelen alakfelismerés!";
    ENDIF
ENDwhile
ENDPROC

```

```

Proc GetVisionData()
success:=0;
WHILE success=0 DO
    Open "COM1",ComChannel\Append\Bin;
    ClearIOBuff ComChannel;
    WriteStrBin ComChannel,"TRG";
    Close ComChannel;
    Open "COM1",ComChannel\Read;
    received_string:=ReadStr(ComChannel);
    ClearIOBuff ComChannel;
    Close ComChannel;
    TPErase;
    TPWrite "Eredmény:"+received_string;
    bOK:=StrToVal(StrPart(received_string,3,1),success);
    IF success=1 THEN
        TPWrite "Sikeres alakfelismerés!";
    ELSE
        TPWrite "Sikertelen alakfelismerés!";
    ENDIF
    found{1}:=StrMatch(received_string,5,";");

```

```

        found{2}:=StrMatch(received_string,found{1}+1,"");
        found{3}:=StrMatch(received_string,found{2}+1,"");
        found{4}:=StrMatch(received_string,found{3}+1,"");
        found{5}:=StrMatch(received_string,found{4}+1,"");
        len{1}:=found{2}-found{1}-1;
        len{2}:=found{3}-found{2}-1;
        len{3}:=found{4}-found{3}-1;
        len{4}:=found{5}-found{4}-1;
        bOK:=StrToVal(StrPart(received_string,found{1}+1,len{1}),XOffs);
        bOK:=StrToVal(StrPart(received_string,found{2}+1,len{2}),YOffs);
        bOK:=StrToVal(StrPart(received_string,found{3}+1,len{3}),Angle);
        bOK:=StrToVal(StrPart(received_string,found{4}+1,len{4}),Ident);
        XOffs:=(XOffs)*(-0.25);
        YOffs:=(YOffs)*(+0.25);
    endwhile

ENDPROC

PROC velocity()
    Open "COM1",ComChannel\Append\Bin;
    ClearIOBuff ComChannel;
    WriteStrBin ComChannel,"Velocity";
    Close ComChannel;
    Open "COM1",ComChannel\Read;
    speed:=ReadStr(ComChannel);
    ClearIOBuff ComChannel;
    Close ComChannel;
    TPErase;

    Open "COM1",ComChannel\Append\Bin;
    ClearIOBuff ComChannel;
    WriteStrBin ComChannel,speed;
    Close ComChannel;
    TPErase;

ENDPROC

PROC eltolas()

    If X<=1 THEN
        Yf:=X*40;
    ENDIF

    IF X>1 AND X<=3 THEN
        Yf:=(X-2)*40;
        Xf:=-40;
    ENDIF

    IF X>3 AND X<=5 THEN
        Yf:=(X-4)*40;
        Xf:=-80;
    ENDIF

ENDPROC

ENDMODULE

```