ROBOTTECHNIKA.

Dr. Csáki Tibor
egyetemi docens

Bevezetés

A tantárgy célkituzése:

- A robotok, mint mechatronikai egységek megismerése
- A robotokban alkalmazott alapveto egységek áttekintése
- A robotok rendszerekben való muködtetésének felvázolása
- A robotok programozása alapjainak elsajátítása
- Robot alkalmazások tanulmányozása
- Fejlodési tendenciák megismerése

Mi az a robot?

A szó eredete a szláv rabota, ami munkát jelent. A magyar nyelvben a robot kemény, unalmas, fárasztó munka, ebbol a szláv szóból ered.

A robot mai jelentései:

- (köznyelvi): fárasztó munka
- háztartási gép
- játék
- szoftver (pl.webcrawler)
- (irodalmi) technikai alkotás
  - Karel Capek 1921
  - Isaac Asimov 1950-
- ipari robot (Ennek a tantárgynak ez a témája)

Alapfogalmak

A robot

- ***mechatronikai*** szerkezet, amely
  - (nyílt) kinematikai láncú mechanizmust és
  - (intelligens) vezérlést tartalmaz,
  - irányított mozgásokra képes
  - automatikus muködésre képes
  - eloírt programozható feladatokat végez
  - együttmuködik a környezetével

Mechatronics: The synergistic integration of mechanical engineering with electronics and intelligent computer control in the design and manufacture of products and processes.

Azaz: A ***mechatronika*** a gépészmérnöki tevékenység szinergikus integrációja az elektronikával és az intelligens számítógépes vezérléssel a termékek és folyamatok tervezésében és a gyártásban.

A robotok felosztására sokféle szempont létezik. A japán és az európai terminológia eltér egymástól. Japánban minden robot, ami mozog és még valamit csinál.

Mozgatott mechanizmus:

- ipari manipulátor (“buta” vezérlés, kevés szabadság)
    - programozható manipulátor
        - egyszeru mozgás, kötött program
    - teleoperációs manipulátor
        - tetszoleges mozgás, nincs program
- ipari robot (okos vezérlés, szabadon programozható)
    - pont-szakasz vezérlésu robot
        - PS, point-straight line mozgás, a pálya paraméterei nem adhatók meg, csak a célpont programozható
    - pályavezérlésu robot
        - szervorobot, CP, contiunuous path robot, a pálya paraméterei megadhatók, programozhatók (pályatípus, sebesség, gyorsulás, stb.)
    - intelligens, szenzorvezérelt robot

- SC, sensor controlled, a programozott pályától eltéro pályán is mozoghat, programozható paraméterek, pl. ero függvényében.

A robotok osztályozhatók:

- mozgásuk
- munkaterük
- felépítésük
- vezérlésük
- feladatuk
- energiaforrásuk
- méretük
- stb. szerint

Robotok osztályozása mozgásuk szerint:

- csak a célpont programozható (pont-szakasz vezérlés)
- a pálya paraméterei is programozhatók (pályavezérlés)

Munkaterük szerint (nem igazán lényegi osztályozás, részletesebben a morfológiánál tárgyaljuk):

- derékszögu koordinátás (hasáb) munkateru robot
- hengerkoordinátás robot
- gömbkoordinátás robot

Vezérlésük szerint:

- alacsony költségu (PLC jellegu) vezérlés
- nagytudású (enhanced, CNC jellegu) vezérlés
- Intelligens (mesterséges intelligencia funkciókat alkalmazó) vezérlés

Feladatuk szerint:

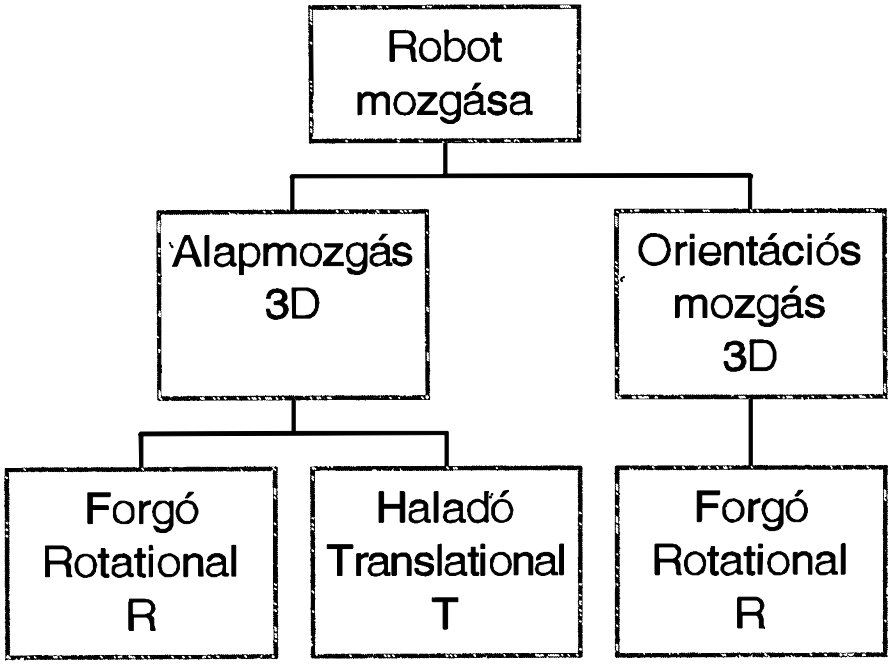
- anyagkezelo robot: a robot a munkadarabot manipulálja
- muveletvégzo robot: a robot a szerszámot kezeli, mozgatja
- szerelo robot: munkadarabot is, szerszámot is kezel, mozgat.

Energiaforrásuk szerint:

- villamos hajtású robot
- hidraulikus robot
- (pneumatikus robot)

Robotok morfológiai felépítése

Ha a tér (jelen esetben a robot munkatere) tetszoleges pontját tetszoleges irányítású robotkézzel, vagy a kézben levo szerszámmal el akarjuk érni, 6 függetlenül irányítható, mozgatható robotkoordinátára van szükségünk.



Alapmozgások

Az alapmozgások megvalósítása haladó vagy forgó mozgást lehetové tevo kényszerekkel történhet. Egy tetszoleges térbeli pont helyzetét 3 koordinátájával adhatjuk meg. Ez a pont elérheto 3 megfeleloen egymásra épített robotkarral. A robot karjait, tagjait összeköto kényszerek lehetnek haladó (H, vagy transzverzális T, vagy prizmatikus P) kényszerek, mozgások, csúszkák, vagy forgó (F, vagy rotációs R) mozgások, kényszerek, tengelyek, csuklók. Általánosan a robot tagokat összeköto kényszert robot csuklónak fogjuk nevezni, és T vagy R betuvel jelöljük.

A 3 koordinátát tehát 2 féleképpen valósíthatjuk meg, azaz

$2^3 = 8$ morfológiai alapváltozat képezheto.

Ezek a következok:

1. TTT derékszögu robot (rectangular), hasáb alakú munkatér.
2. TTR (nem gyakori típus)
3. TRT szerelo, festo robot (pl. ASEA)

4. TRR (nem gyakori típus)
5. RTT henger koordinátás robot (pl. FANUC, RB)
6. RTR (nem gyakori típus)
7. RRT UNIMATE nehézüzemi robot és SCARA szerelo robot
8. RRR PUMA, antropomorf robot.

Orientáció

Az orientáció 3 szögkoordináta megadását és a 3 forgó mozgás megvalósítását jelenti. Robotkezekben alkalmazott leggyakoribb megoldások:

Euler: z - y' - z'

Kardán: x - y' - z"

RPY (roll - pitch - yaw): y - z' - y'

Robotikai rendszer elemei

Egy robotos rendszer általában együttmuködo gépi és emberi eroforrásokat felhasználva épül fel.

Hardver

Fobb hardver összetevok:

- a robot mint gépészeti berendezés
- robotvezérlo
- technológiai berendezés
- a technológiai berendezés vezérlése
- társberendezés, segédberendezés
- társberendezés vezérlése PLC
- betanító egység
- szenzor
- szenzor processzor
- cella vezérlo
- tervezo és programozó munkaállomás
- kommunikációs kapcsolatok

- LAN

Szoftver

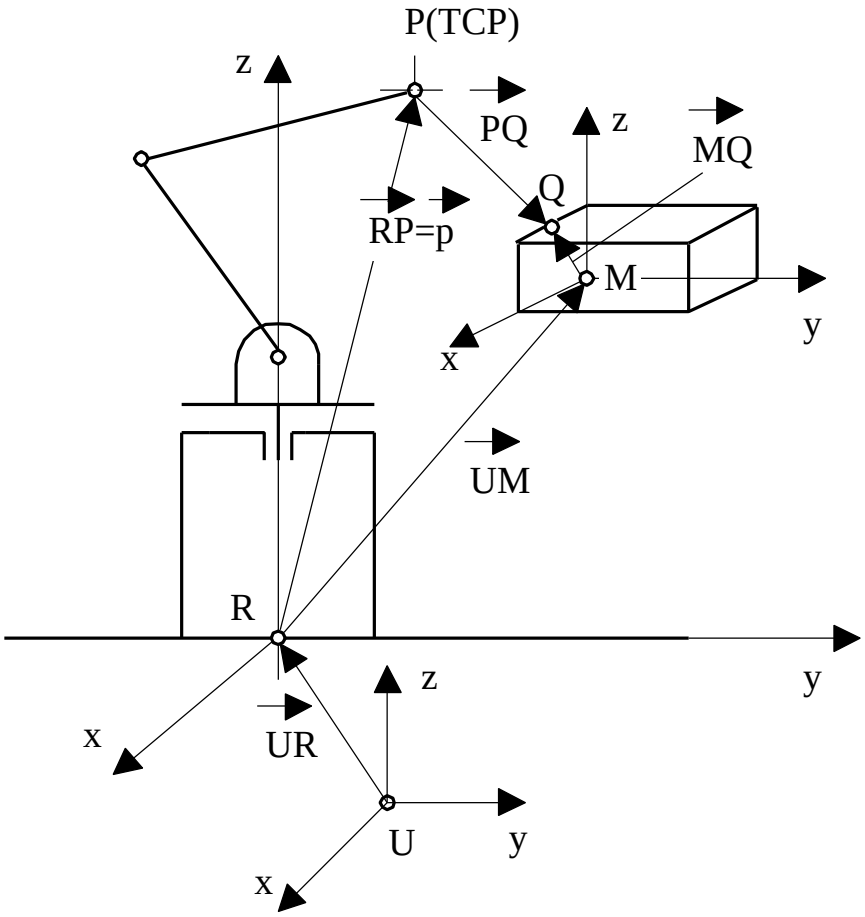
- tervezo sw.
    - UNIX op.r.
    - CAD
    - CAPP
    - CAM
    - hálózati sw.
- robot programozó rendszer
    - UNIX vagy PC-s op.r.
    - programozó
        - feladat tervezo
        - interpreter
        - szimulátor
        - compiler
        - kommunikációs sw.
- robot vezérlo rendszer
    - RT op.r.
    - robot program értelmezo
    - robot vezérlo taszkok
    - kommunikációs sw.

Robotok geometriai rendszerei

A robot mozgását célszeruen koordinátarendszerekben adhatjuk meg. A koordinátarendszerek kapcsolatát transzformációk írják le.

A robotot szemlélhetjük kívülrol és a robot belsejébol, minden pont a leírástól függetlenül azonos kell hogy legyen.

Külso koordinátarendszerek és nevezetes pontok



U universe világ

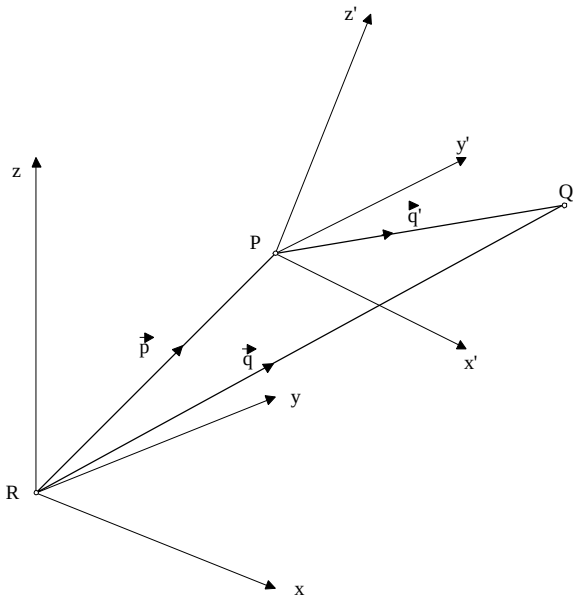
R robot

W workpiece munkadarab

P TCP programmed point, tool center point programozott pont

Q a megmunkálási pont

Homogén transzformáció



A robotkarok egymáshoz viszonyított helyzetének egzakt megadása érdekében minden robotkarhoz rögzítünk egy úgynevezett belso koordinátarendszert és a karok egymáshoz viszonyított helyzetét ezen koordinátarendszerek közötti transzformációval írjuk le.

$$T_{DH} = Rot(z_{i-1}, \Theta_i) \cdot Trans(0,0,d_i) \cdot Trans(a_i,0,0) \cdot Rot(x_i, \alpha_i)$$

Két koordinátarendszer közötti kapcsolat leírásának matematikai eszköze a homogén transzformáció. Általános esetben a két koordinátarendszer között eltolás és elfordulás is létezik.

Esetünkben az „R" kezdopontú koordinátarendszerben valamely „Q" pont helyzetét a **q** helyvektor írja le. Szeretnénk ismerni ugyanezen pont helyzetét a „P" kezdopontú koordinátarendszerben azáltal, hogy meghatározzuk a hozzá tartozó **q'** helyvektort.

$$\vec{q} = \underline{\underline{T}} \cdot \vec{q'}$$

A homogén transzformációs mátrix általánosságban a következo alakú:

$$\underline{\underline{T}} = \left[\begin{array}{ccc|c} & \underline{\underline{h}} & & \vec{p} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Ahol

$\underline{\underline{h}}$ 3x3 orientációs mátrix, $\vec{p}$ eltolási vektor.

Így a robot teljes térbeli helyzete megadható.

Denavit-Hartenberg transzformáció

Hogyan vehetjük fel célszeruen ezeket a koordinátarendszereket? Szabályok:

- A „z” tengelyek a kényszer (csukló, tengely, csúszka) geometriai tengelyei,
- A belso koordinátarendszerek kezdopontjai a „z_i”-re állított normáltranszverzálisok talppontjai,
    1. Kitéro z_i és z_{i+1} esetén egyértelmuen meghatározható.
    2. Metszo z_i és z_{i+1} esetén a metszéspont.
    3. Párhuzamos z_i és z_{i+1} esetén tetszés szerinti kezdopontot választunk.
    4. Egybeeso z_i és z_{i+1} esetén tetszés szerinti kezdopontot választunk.
- Az „x_i” tengelyek iránya megegyezik a normáltranszverzálisok irányaival
    1. Kitéro z_i és z_{i+1} esetén egyértelmuen meghatározható.
    2. Metszo z_i és z_{i+1} esetén a tetszoleges, de célszeruen választott x_i irány.
    3. Párhuzamos z_i és z_{i+1} esetén végtelen sok , de meghatározott irányú normáltranszverzális létezik, de a kezdopont tetszoleges.
    4. Egybeeso z_i és z_{i+1} esetén tetszés szerinti irányt és kezdopontot választunk.

Ilyen választás mellett a robottagok egymáshoz viszonyított helyzete az ún. Denavit-Hartenberg paraméterek (Θ, α, d, a) segítségével négy transzformációval megadható. A paraméterek közül „Θ” és „d” a robottagok mozgása során folyamatosan változik, „α” és „a” konstrukciósan rögzített állandó értékek, melyek az adott robotra jellemzok.

Több robotkar egymáshoz kapcsolódása esetén az egyes koordinátarendszerek transzformációját megvalósító DH mátrixok összeszorzódnak. A transzformációt az egymást követo tagpárokra elvégezve a robot programozott pontjának helyzete megadható.

$$T_{=H} = T_{=0,1} \cdot T_{=1,2} \cdot \ldots$$

$$\underline{\underline{T}}_{DH} = \begin{bmatrix} c\Theta & -s\Theta \cdot c\alpha & s\Theta \cdot s\alpha & a \cdot c\Theta \\ s\Theta & c\Theta \cdot c\alpha & -c\Theta \cdot s\alpha & a \cdot s\alpha \\ 0 & s\alpha & c\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A robotvezérlés belso alapfeladata

Robotok gépi funkciói

Pozícionálás

Pozícionáló rendszerek felépítése

A pozícionáló rendszer részei

Motor

Mozgásátalakító

Útméro

Irányító rendszer

Megfogás

A robotok megfogószerkezetei alkalmazkodnak az általuk kiszolgált technológiai folyamathoz. Mivel az emberi kéz munkavégzo képességét és mozgásait igyekszenek másolni, az emberi kéz mozgását leképzo szerkezet lenne a legkedvezobb, ez azonban nehezen valósítható meg. A különbözo tárgyak megfogása alakzáró vagy erozáró kapcsolattal realizálható. Megfogási elvek:

- szorítópofák (2, 3),
- ujjak,
- speciális készülékek

Feladatuk szerint:

- munkadarab megfogók,
- szerszám megfogók.

Érzékelés

A robotoknak muködésük egyes fázisairól és a kiszolgált technológiai folyamatról különbözo információkkal kell rendelkezniük. Ezeket az információkat a szenzorok szolgáltatják s a robot irányítórendszere értelmezi oket.

Szenzorok csoportosítási szempontjai:

A környezetbol való információszerzés módja szerint:

- érintkezéses,
- érintkezés nélküli.

Az információkat vagy a szenzor és a mérendo test közötti kölcsönhatáson alapuló elv, vagy pedig a szenzorhoz kapcsolt közeg pillanatnyi jellemzoi alapján kapjuk. A szenzorok struktúrája az alkalmazott méroátalakítótól és a fizikai hatáselvtol függ. A méroátalakítók lehetnek:

- aktívak,
- passzívak.

A passzív átalakítok fizikai mennyiségei pl. ellenállás, induktivitás, kapacitás, stb. Az aktív átalakítók jellemzoi pl. feszültség, áram, töltés, stb.

Alkalmazási területek.

Az érintkezéses szenzorok geometriai és fizikai jellemzokrol szolgáltatnak információt. Alkalmazási helyeik robotokon:

- A környezettel való kapcsolattartásban ero, nyomaték, út (helyzet) érzékelése,
- A megfogószerkezetben ero és elmozdulás lehatárolása,
- A karokon ero, nyomaték, elmozdulás értékének meghatározása,
- A hajtórendszerben ero, nyomaték, nyomás, áramerosség és feszültség mérése.

Az érintkezés nélküli szenzorok leggyakoribb alkalmazási területe az útmérés, újabban az alakfelismerés. Elonyük a tapintóero hatásának kiküszöbölése.

Elhelyezkedésük szerint megkülönböztetünk külso és belso szenzorokat.

Belso szenzorok

A belso szenzorok a robotmechanikára, a hajtórendszerre és a megfogószerkezetre vannak telepítve. Funkciójuk szerint útmérok, szögsebességmérok, ero-és nyomatékmérok.

Útmérok

A robotkarok csuklókoordinátáit realizáló szögelfordulások és elmozdulások pillanatnyi értékének meghatározására szolgálnak. Fajtái:

- Digitális, abszolút – kódolt méroléc, kódtárcsa,

- Digitális, növekményes – lineáris rács, forgóadó,
- …

Ero (nyomaték) érzékelo

Elsosorban a hajtórendszerek és a technológiai folyamattal közvetlen kapcsolatban álló megfogószerkezetek és szerszámok muködését szabályozzák. Muködési elvük a méroelem alakváltozásán alapul, amely a nyúlásméro bélyegek segítségével ellenállásváltozássá, illetve feszültségváltozássá alakítható. Mind egy-, mind többkomponenses ero-, illetve nyomatékérzékelok használatosak.

Külso szenzorok

A robot és a környezet közt teremtenek kapcsolatot. A robotok leggyakoribb alkalmazási területe a felmérések szerint:

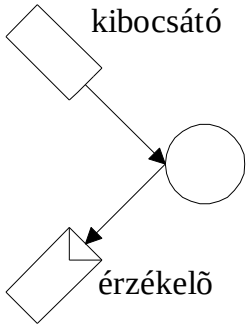
- öntés,
- kovácsolás,
- palettázás és egyszeru alkatrész összerakás,
- ponthegesztés,
- festékszórás,
- ívhegesztés,
- sorjátlanítás,
- automatikus ellenorzés,
- gyártócellában való alkalmazás,
- automatizált szerelés.

Szinte mindegyik területen rendkívül fontos a külso érzékelés, mert segítségével pl. a technológiai folyamat turéshatáron kívüli eltéréseit is kezelni lehet.
A külso szenzorok által szolgáltatott információk növelik a robot intelligencia szintjét, segítségükkel módosíthatók az eredeti mozgáspályákat megvalósító programok.
A külso érzékelés két nagy területe:

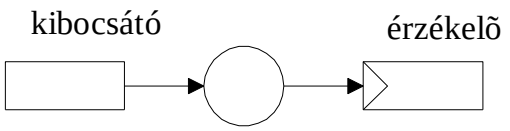
- tapintóérzékelés,
- látóérzékelés.

A tapintóérzékelés ún. bináris érzékelés, azaz a szenzor érzékeli, hogy a robot kapcsolatba kerül-e valamilyen tárggyal, de nem azonosítja azt. Az érzékelés történhet érintéssel, mikrokapcsoló, tapintós útméro, vagy tapintós induktív érzékelo által. Valamely tárgy jelenléte annak megérintése nélkül is érzékelheto. Ennek megoldásai:

Optikai



fényvisszaverõdésen alapuló



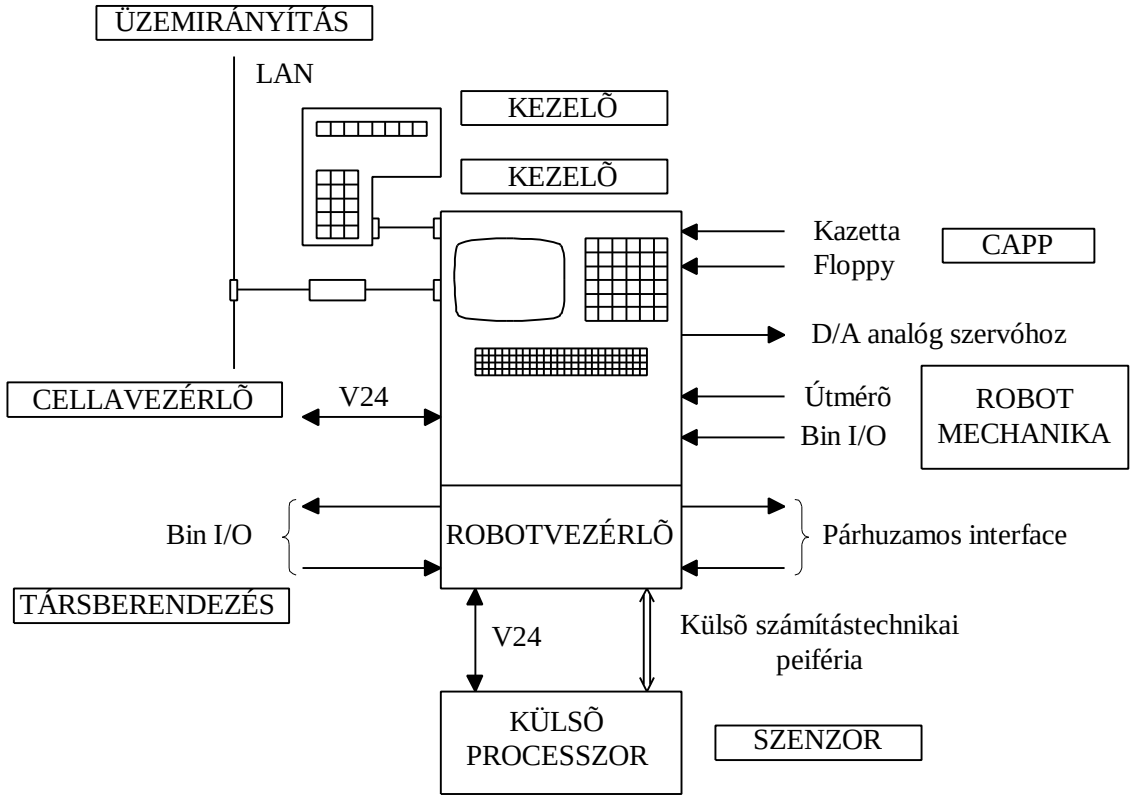
fényzáron alapuló

Induktív és kapacitív (ritkább) érintés nélküli érzékelok.

A látóérzékelok a robot legfejlettebb szenzorai. Tulajdonképpen a kamerát és a képfeldolgozó processzort értjük alatta.
Típusai:

- kétdimenziós látóérzékelo különálló tárgyak bináris érzékelésére,
- kétdimenziós látóérzékelo különálló tárgyak szürke árnyalatai szerinti érzékelésére,
- kétdimenziós látóérzékelo egymással érintkezo, vagy átfedésben lévo tárgyak érzékelésére,
- kétdimenziós ellenorzo készülékek,
- kétdimenziós vonalkövetok,
- különálló tárgyról háromdimenziós információk kiszurésére alkalmas rendszerek perspektívikus ábrázolás, sztereotechnika, strukturált megvilágítás vagy pásztázó keresés elvén,
- háromdimenziós információ kiszurése rendezetlen tárgyhalmazokról,
- térbeli helyszínelemzés mobil robotok navigációjához, útvonalkereséséhez és az akadályok elkerüléséhez.

Kommunikáció



A robot a kezelovel és további automatikusan muködo berendezésekkel interfészeken keresztül kommunikál. A kommunikáció a muködéshez és az együttmuködéshez szükséges információcsere megvalósítását jelenti.

A kommunikációkat megvalósító interfészek

Bináris logikai interfész
Bináris jeleket csatol a robotvezérlo és a robot, valamint a környezete között. A bináris jelekkel történo kommunikáció a legegyszerubben megvalósítható, programozása is a legegyszerubb. A bináris jelek lekezelése általában PLC-vel, vagy PLI-vel történik. A robotvezérlo szempontjából inputok pl. a kérés bitek, feltétel bitek, stb., outputok pl. a parancs bitek, nyugtázó bitek, stb.

Digitális interfész
A digitális interfész összetett információk gyakran kétirányú továbbítására szolgál. Az információk kódoltak. Az információcsere szabványos protokollok segítségével történik. A bitszintu, bináris logikai kommunikációnál intelligensebb partnereket feltételez (vezérlések, intelligens szenzorok, stb.) Az átvitel módja:

- Párhuzamos átvitel
  - IEEE busz
  - Centronics interfész
- Soros átvitel
  - RS 232
  - USB
- Átvitel bit buszokon

Analóg interfész
A robotvezérlo és környezete közötti analóg jelekkel történo kommunikáció lebonyolítására szolgál. Analóg jelekkel a pozícionáló rendszerek szabályozóköreit, illetve a szenzorok analóg kimeno jeleit csatoljuk a robotvezérlohöz. Ezen analóg jelek kezelése nem tartozik közvetlenül a programozó, illetve a robot operátor hatáskörébe, ezért részletes tárgyalásától itt eltekintünk.

Kezeloi kommunikációt megvalósító interfész
A robot és a robotvezérlés optimális esetben automatikusan, emberi beavatkozás nélkül végzi munkáját. Azokban az esetekben azonban, amikor az operátor közremuködése is szükséges (beállítás, betanítás, programozás, hibakeresés, stb.), megfelelo és jól használható kezelo szervek hatékonnyá tehetik a munkát. Hasonlóan a korszeru CNC vezérlésekhez, a megjelenítés és az adatbeadás kényelmessége, áttekinthetosége, a kezelot segíto funkciók megvalósítása az eladhatóság egyik fontos szempontja lett. A korszeru robotvezérlésekben is egyre fontosabb a grafikus megjelenítés.

Kijelzok:
alfanumerikus
display
(grafikus)

Nyomógombok:
funkciógombok
soft-key
alfanumerikus

Betanító elemek:
botkormány
egér
pozícionáló gömb
robotmodell

Robotprogram beviteli interfész
Az online vagy offline megírt programok bevitelére és a kipróbált és kijavított, "belott" robotprogramok eltárolására szolgáló interfész. Egyedi robotvezérlok esetén leggyakoribb a floppy lemez, rendszerbe integrált robotvezérléseknél a hálózati interfész az optimális megoldás.

- MDI
- Kazetta
- Floppy
- DNC (RS 232, LAN)

Robot kommunikációja a kiszolgált berendezéssel

A robot és a társberendezés közötti kommunikáció legfontosabb célja az együttmuködo berendezések szinkronizálásának megvalósítása. A következokben néhány példán keresztül a robotvezérlo és a társberendezés közötti bináris jelek segítségével történo információcsere megvalósítási lehetoségeit vizsgáljuk. A példák egy, a késobbiekben ismertetendo programnyelven készültek, de a példák egyszerusége miatt már ezen a helyen is értheto a muködésük.

A robot és a társberendezés együttmuködésének hardver és szoftver feltételei vannak. A következokben feltételezzük, hogy a megfelelo összeköttetések léteznek, a társberendezés vezérlésében a kommunikáció kezelés megoldott, és csak a robot programozási kérdéseivel foglalkozunk.

A társberendezéssel való kommunikáció leggyakrabban és legegyszerubben bináris I/O-n kereszül valósítható meg, ezért példáinkban csak ezzel foglalkozunk.

A robotnak (mint minden automatikusan muködtetett berendezésnek) "bizalmatlannak" kell lennie, minden feltételt és a végrehajtást ellenoriznie kell. Ha egy berendezést ember muködtet, akkor egy rendkívül jól szenzorált, magas intelligenciafokú, nagyon bonyolult döntéshozatali algoritmusokat használó felügyelo rendszer, az ember végzi a szinkronizálási, hibadetektálási, beavatkozási funkciókat. Automatikus, gépi muködtetés esetén egy átlagos ipari rendszerben ilyen fokú intelligens megoldásra általában sem igény, sem gazdaságos lehetoség nincsen, ezért egyszerubb, de megbízható megoldásokra van szükség. Ezért szükséges a feltételek alapos és gondos vizsgálata, a lehetoségek elozetes áttekintése, és lehetoleg az összes felmerülo lehetoségre a válasz megtervezése. A mintapéldákban ezeknek az alapjaival foglalkozunk.

A megoldás bonyolultsága mindig a társberendezés vezérlojének tudásától is függ.

1. példa: a robot vezérel mindent.

Feltételezéseink:

- "buta" társberendezés
- a robot felel mindenért
- a robot ellenoriz mindent
- az idozítés nem múlhat a végrehajtási sebességen, lassú és gyors társberendezés esetén is megfeleloen kell muködnie

A végrehajtás lépései:

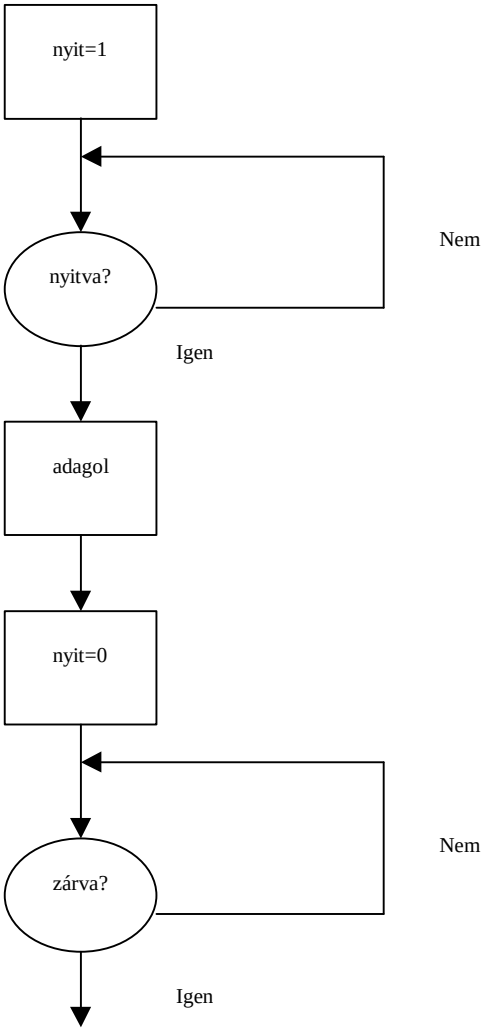
1. parancskiadás
2. a végrehajtás ellenorzése

3. a parancs megszüntetése

A parancskiadás elott a végrehajthatóság feltételeit ellenorizni lehet/kell.

- Feladat: adagolás (pl. szerszámgép)
- Megoldandó: burkolat nyitása, zárása
- Vezérelendo: nyit=NOT(zár) . Output, kimeno jel, mert a robotvezérlo állítja az érétékét.
- Ellenorizendo: nyitva, zárva. A burkolat szenzorált, érzékelt állapotai, input változók, a robotvezérlo a társberendezéstol kapja az értékeket.
- nyitva és zárva egymásnak nem negáltjai, a burkolat állapota nem bináris, ezek csak a szélso értékei.

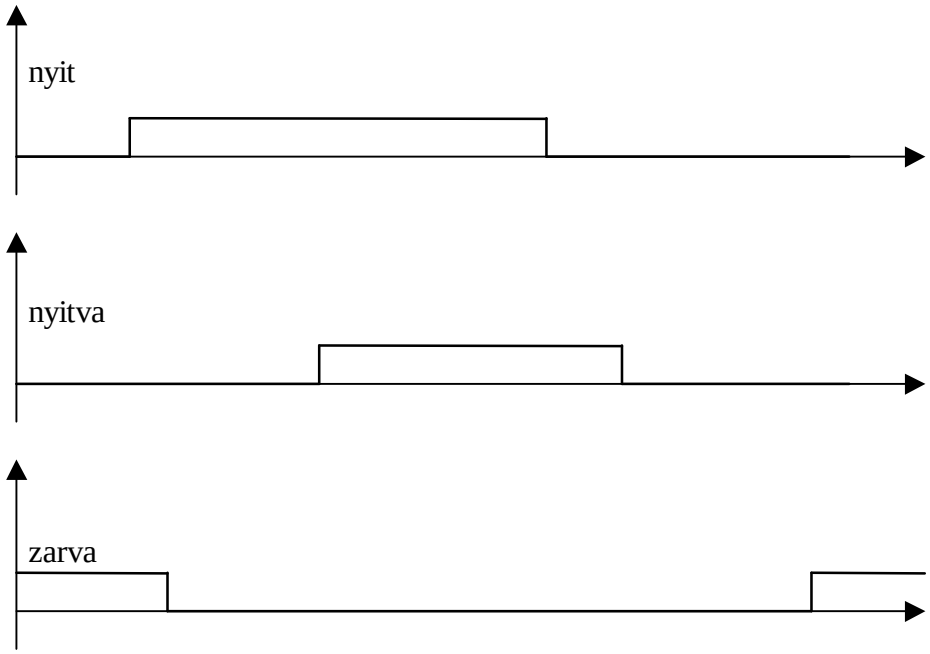
Blokkvázlat:



Programrészlet

```
pelda1:  subr();
         writeo(nyit,on);          /* nyit nevu output csatornara 1-et ad, parancs */
                                   /* a burkolat nyitasara */
nyit_var: testi(nyitva, off, nyit_var);     /* mindaddig, amig a burkolat nincs */
                                   /* nyitva, itt var          */
         adagol();                 /* a burkolat kinyilt, az adagolas elvegezheto*/
         writeo(nyit, off);        /* parancs a burkolat zarasara */
zar_var: testi(zarva, off, zar_var);        /* mindaddig, amig a burkolat nincs */
                                   /* zarva, itt var */
         end;                      /* vege a programreszletnek */
```

Idodiagram:



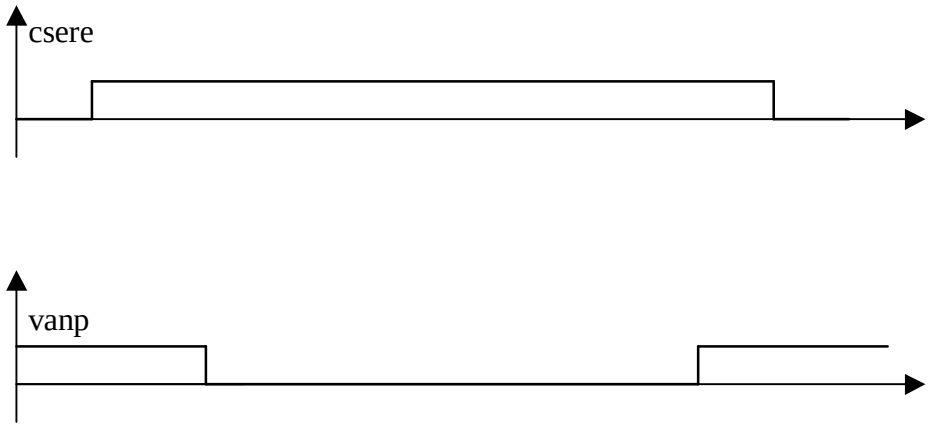
2. program: palettacsere

- Feladat: palettacsere
- Vezérlendo: csere. Output, a robot állítja.
- Ellenorizendo: vanp. vanp=1, ha érzékeli a paletta jelenlétét.
- Tájolás mechanikusan történik, nem kell vezérelni.
- Az idozítéstol nem függhet, a palettacsere és a robotprogram végrehajtási sebességétol függetlenül jól kell muködnie, ezért ellenorizni kell a paletta eltávolítását és megérkezését is!

Programrészlet:

```
cserel:     subr();
            writeo(csere, on);      /* parancs a csere inditasara   */
meg_van:    testi(vanp, on, meg_van);     /*meg ott a regi pal    */
meg_nincs:  testi(navp, off, meg_nincs);  /* meg nincs uj pal.    */
            writeo(csre, off);      /* csre megtortent, leall       */
            end;
```

Idodiagram:



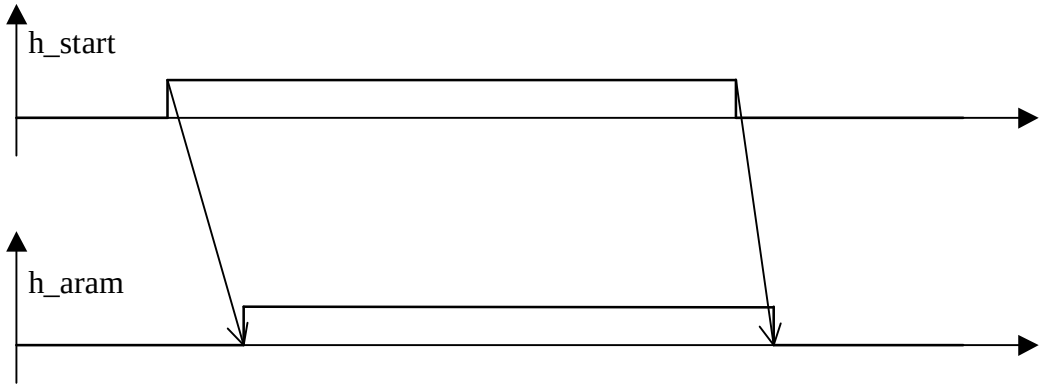
3. példa: kézfogás

- Feladat: hegeszto berendezés be- és kikapcsolása
- Vezérlendo: h_start=NOT(h_stop)
- Ellenorizendo: h_aram. h_aram=1, ha van hegeszto áram

Programrészlet:

```
hegeszt:  subr();
          writeo(h_start, on);    /* bekapcsolas */
varj:     testi(h_aram, off, varj);       /*varj, amig nincs hegeszto aram*/
          linear(50);             /* egyenes interpolacio, 50 mm/perc sebesseg*/
          pmove(<p1, pfelso, p4>);      /* egyenesek vegpontjai       */
          writeo(h_start, off);   /* kikapcsolas  */
vege:     testi(h_aram, on, vege);        /* varj, mig tenyleg kikapcsol*/
          end;
```

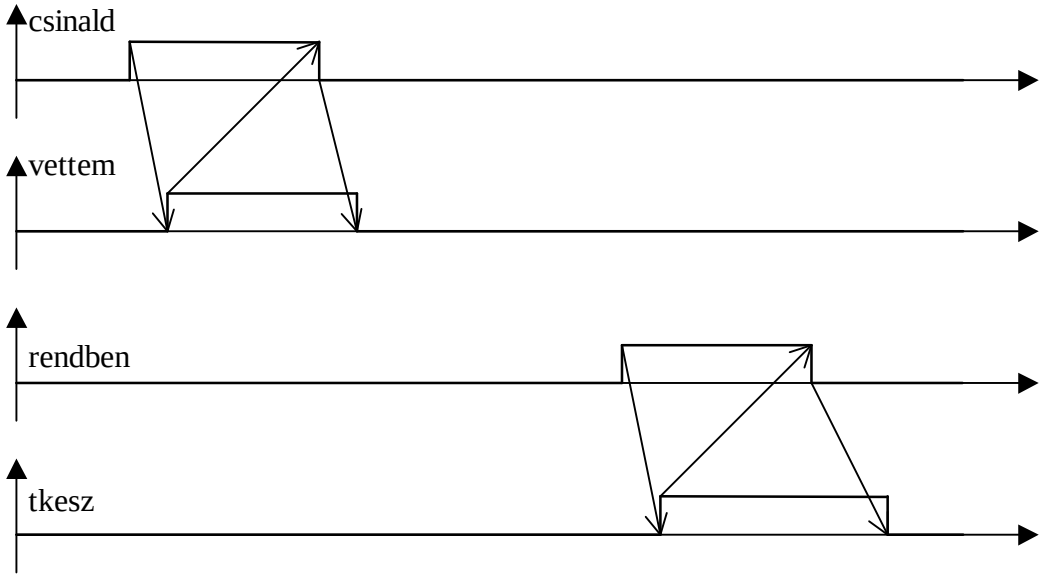
Idodiagram:



4. példa: a robot együttmuködik egy intelligens vezérlovel.

- A kommunikáció nem feltétlenül egyszerubb, de a társberendezés által ellátott feladat sokkal összetettebb lehet!
- A parancskiadást és az ellenorzést egyaránt kezfogással célszeru megvalósítani. (csináld - értettem - tudomásul vettem hogy értetted - tudomásul vettem, hogy tudomásul vetted)
- A robotvezérlo kiadja a parancsot, és megvárja, amíg a társberendezés elkezdi végrehajtani. (O=csinald, I=vettem)
- Ezután a robotvezérlo megvárja, (tétlenül vagy más feladat végrehajtásával), amíg a társberendezés jelzi, hogy kész, és visszajelez, hogy értette (I=tkesz, O=rendben).

Idodiagram:



Programrészlet:

```
okos:   subr();
        writeo(csinald, on);    /*inditas*/
indult: testi(vettem, off, indult);     /*var, mig visszajelez*/
        writeo(csinald, off);   /*jelzett*/
i0:     testi(vettem, on, i0);  /* var, mig visszaveszi*/
kesz:   testi(tkesz, off, kesz); /*var, mig a tars jelzi, hogy kesz*/
        writeo(rendben, on);    /* szol, hogy latja*/
res:    testi(tkesz, on, res);  /* var, mig szol, hogy eszrevette*/
        writeo(rendben, off);   /* visszaveszi a jelzest*/
        end;
```

Robotok programozása

On-line programozás

Robotok on-line programozásáról akkor beszélünk, ha a programozás a bekapcsolt robot mellett, a robotvezérlo szolgáltatásait használva történik. A programban szereplo pontok megadása gyakran betanítással, a robot megfogójának a kívánt pontba történo mozgatásával valósul meg. Elonye, hogy az elkészült program szintaktikailag nagy valószínuséggel helyes, nincsenek a munkatéren kívüli pontok. Hátránya, hogy a betanításos helyszíni programozás idejére a robot kiesik a termelésbol.

Off-line programozás

A program írása a robottól térben elválik, a program íráshoz nincs szükség a robot és a robotvezérlés használatára. A program írását célszeruen valamilyen számítógépes eszköz segíti, menurendszerrel, szintaktikus ellenorzéssel, szimulátorral. Elonye, hogy a programozás nem vesz el gépi idot a robottól, a program jól dokumentált, archivált lehet. Hátránya, hogy nem eredményez feltétlenül végrehajtható programot, ütközések, idozítési hiányosságok esetleg csak a program "belövésekor" derülnek ki.

Programnyelvek

Programnyelvek szintjei

Gépi kódú robotprogramozás

A programozás nehézkes, de nagyon hatékony program írható.

NC szeru (G formátumú) programnyelv

Az NC gépekéhez hasonló szintaktikájú programnyelv, annak elonyeivel és hátrányaival.

Robotfunkciókra orientált nyelvek

A robotok összes lehetoségét kihasználó, viszonylag alacsony szintu programnyelvek, nem nagyon kényelmesen használhatóak.

Mozgásleíró nyelvek

A robotok legfontosabb funkciójára, a mozgásra, mozgatásra koncentráló programnyelvek, ma a legelterjedtebb, jól használható, hatékony nyelvek. Egyik legfontosabb képviselojük az AML nyelv.

AML

Az AML az A Manufacturing Language (kb. Egy gyártásprogramozási nyelv) kezdobetuibol képzett szó. Az IBM fejlesztette ki saját céljaira.

Robotprogramozásra az AML-nek egy szukített változatát használjuk, amely IBM 7576 típusú SCARA robot programozására alkalmas.

A rendszer moduljai:

- AML programozási nyelv
- editor, szerkeszto program
- betanítás szimulátor (a robot betanításának képernyon történo szimulálására)
- programszimulátor (az AML program végrehajtásának szimulálására, ellenorzésére)
- help (segíto rendszer)
- debugger (hibakereso)
- kommunikátor a robotvezérlovel

Az AML programozási nyelv fobb részei:

Deklarációk

Formája:

cimke: utasítás;

A címkével nevet adunk a definiálandó változónak vagy szubrutinnak, az utasításban pedig meghatározzuk a típusát és általában a kezdoértékét is.

Az AML változó típusai:

- geometriai pont. A pontot 3 koordinátájával és a megfogó elfordulási szögével adhatjuk meg.

Példa:

p1: **new pt**(300, -300, -100, 180);

Itt **x**=300 mm, **y**= -100 mm, **z**= -100 mm, a megfogó elfordulása R=180 fok pozitív irányban.

A **pt()** függvény a megadott adatokból geometriai struktúrát képez.

- string. A stringet értékével adhatjuk meg.

Példa:

st: new 'Ez egy string valtozo erteke';

- tömb. Tömb megadása a változók értékeinek felsorolásával történhet. Egy tömbnek bármilyen típusú elemi lehetnek, vegyesen is.

Például:

sok: new<1, p1, st, -100>;

A **sok** nevu tömb elso eleme egy szám, melynek értéke **1**, második eleme az elobb definiált **p1** pont, harmadik eleme az **st** nevu string, negyedik eleme a **-100** érték.

- valós változó

a: new real(1.1);

- logikai változó

v: new TRUE;

– szenzor típusú változó. Az AML feltételezi, hogy a szenzor jelet **n** bites A/D átalakító kimenete adja az **s.** bittol kezdve.

ero: sensor(s, n);

– szubrutin. A szubrutinoknak nevük van és üres vagy nem üres paraméterlistájuk. Egy rutin különbözo számú paraméterekkel is hívható.

felvesz: subr(mit, honnan);

– paletta. Az AML rendkívül jól használható változó típusa a paletta. A paletta logikailag együtt kezelheto pontok halmaza, melyen az AML muveleteket enged meg. A pontok elore vagy hátrafelé bejárhatók, láncoltak (az utolsó után az elso következik, az elso elott az utolsó), és tetszoleges sorrendben is elérhetok. Palettázás jellegu feladatok ezért AML-ben könnyen és elegánsan programozhatók.

talca: new pallet(ba, ja, jf, dbs, n);

A **talca** a definiált paletta neve, **ba** a bal alsó pontja, **ja** a jobb alsó, **jf** a jobb felso, **dbs** az egy sorban levo darabok (pontok) száma, **n** az összes darab száma.

Operátorok és standard függvények.

+	összeadás
-	kivonás
*	szorzás
/	osztás
++	inkrementálás
--	dekrementálás
AND	logikai és
OR	logikai vagy
NOT	logikai nem
**	hatványozás
IDIV	egész osztás
EQ	= reláció
NE	nem egyenlo
GT	nagyobb mint

GE	nagyobb egyenlo
LT	kisebb mint
LE	kisebb egyenlo
ABS	abszolút érték
ACOS	arkusz koszinusz
ASIN	arkusz szinusz
ATAN	arkusz tangens
ATAN2	arkusz tangens x/y
COS	koszinusz
EXP	exponenciális fv
LN	természetes logaritmus
MAX	maximum
MIN	minimum
MOD	modulo
SIN	szinusz
SQRT	négyzetgyök
TAN	tangens

Mozgásutasítások

Növekményes mozgások.

djmove(<i,j,k,l>,<a,b,c,d>);

Az i., j. k., l. robotcsuklót növekményesen a, b, c, d értékkel elmozdítja.

Az 1. csukó a váll, a 2. a könyök, a 3. a kéz lineáris z irányú mozgatása, a 4. a megfogó elfordítása.

Pl.:

djmove(2,45);

a 2. csuklót, a könyököt elfordítja 45 fokkal pozitív irányban.

djmove(<1,2>, <30,-15);

az 1. csuklót pozitív irányban 30 fokkal, a 2. csuklót negatív irányban 15 fokkal elfordítja a jelenlegi helyzetükhöz képest.

dmove(<x,y,z,r>,<a,b,c,d>);

A robotkoordinátákat a megadott értékkel megváltoztatja növekményesen.
Pl.:

dmove(z, -50);

A kezet lefelé mozgatja 50 mm-rel.

dmove(<x,y>,<100,100>);

A programozott pont x és y koordinátáját 100-100 mm-rel megnöveli.

dpmove(pont);

A programozott pont koordinátáit a pont értékével megváltoztatja.

PL.
pont: new pt(100,100,0,0);
.
.
dpmove(pont);
az elozo példabeli elmozdulást okozza.

Abszolút mozgások

jmove(<i,j,k,l>, <a,b,c,d>);

Az i., j., k., l. csuklókoordinátát az a, b, c, d értékre mozgatja.

Pl.

jmove(<1,3>, <100,-100>);

Az 1. koordinátát, a vállat +100 fokos helyzetbe állítja, a 3. koordinátát, a kezet -100 mm-re mozgatja.

move(<x,y,z,r>, <a,b,c,d>);

A robotkoordinátákat a megadott értékre mozgatja.

Pl.:

move(z,-100);

A z koordinátát a -100 mm-es helyzetbe mozgatja.

move(<x,y,r>,<300,300,360>);

Az x és az y koordinátákat a 300, 300 mm-es pontba mozgatja, miközben a megfogót egy teljes fordulattal pozitív irányban elfordítja.

pmove(pont);

A programozott pontot a pont-ba mozgatja.

Pl:

pmove(p1);

pmove(pt(300,300,-150,720));

A második példában a **pt()** függvény a megadott érékekbol egy pont típusú változót generál, majd a **pmove()** függvény ebbe a pontba mozgatja a robot programozott pontját.

pmove(<p1,p2,p3>);

A programozott pontot a **p1, p2, p3** pontokon át mozgatja. Természetesen mindegyik pontnak a robot munkaterében kell lennie!

zmove(a);

A robot kezét **z** irányban az **a** méretre mozgatja. Az **a** értéknek negatívnak kell lennie!

Palettával kapcsolatos utasítások

nextpart(pal);

prevpart(pal);

A **pal** nevu palettán a következo (nextpart) illetve az elozo (prevpart) paletta pozíciót teszi aktuálissá. Emlékeztetoül: a paletta pozíciók láncoltak!

getpart(pal);

Mozgás a paletta aktuális munkadarab pozíciójára.

setpart(pal, n);

A paletta **n**. munkadarab pozícióját jelöli ki aktuálisnak.

Egyéb, mozgással kapcsolatos utasítások

linear(v);

Bekapcsolja a lineáris interpolációt (a programozott pont a következokben térbeli egyenes mentén fog mozogni), és beállítja a pályasebességet **v** mm/perc értékure. **v=0** esetén a lineáris interpolációt kikapcsolja, a programozott pont az arányos csuklóinterpolációnak megfelelo módon mozog.

circular(v);

Bekapcsolja a körinterpolációt és beállítja a pályasebességet **v** mm/perc értékure. **v=0** esetén kikapcsolja a körinterpolációt, a továbbiakban a programozott pont arányos csuklóinterpolációval mozog. Bekapcsolt körinterpoláció esetén a mozgásutasításokban körív pontjainak kell szerepelniük célpontokként.

home(i,j);

Az i. és j. robotcsuklót referencia pontra küldi.

Pl.:

home(1,2);

A váll és a könyök referencia helyzetbe megy, a z és a megfogó elfordulása nem változik.

Megfogó utasítások

grasp();

megfogó zár

relase();

megfogó nyit.

Kommunikációs és várakozó utasítások

break();

Feltétel nélküli programozott stop.

resume();

Programvégrehajtás folytatása.
delay(t);

Programvégrehajtás késleltetése **t** másodpercig.

waiti(d, e, t, sr);

t másodpercig vár arra, hogy a **d** input csatorna értéke **e** legyen. Ha a megadott idon belül ez nem következik be, hívja a **sr** szubrutint.

writeo(d,e);

A **d** output csatornára **e** értéket ad ki. Az e 0 vagy 1 lehet, d-nek output változónak kell lennie.

blink(d);

Villogtatja a megadott d output csatornát.

endblink(d);

Villogtatás kikapcsolása.

guard(d,e);

Figyeli a megadott **d** input csatornát. Ha értéke **e**-vé válik, megszakítja a mozgást.

endguard(d);

Kikapcsolja a figyelést.

monitor(d,e,sr);

Figyeli a megadott **d** inputot. Ha az aktuális érték a megadott **e** értékuvé válik, hívja a **sr** szubrutint.

endmonitor(d);

Kikapcsolja a figyelést az adott csatornára.

testi(d);

A d címu input csatornán levo értéket adja vissza.

testi(d,e,c);

Feltételes ugrást hajt végre a **c** címkére, ha a **d** input csatornán **e** értéku jel van.

Programtechnikai utasítások

s: subr(p1,p2,…,pn)
.
.
.
end;

s nevu szubrutint definiál, **p1, p2, … pn** paraméterekkel.

branch(c);

Feltétel nélküli ugrás a megadott **c** címkére.

```
while (felt) do
begin
.
.
.
end;
```

A **begin** és **end** közötti utasítások végrehajtódnak, amíg a **felt** feltétel teljesül.

```
if felt then ut1
else ut2;
```

A vezérlés a **felt** feltétel teljesülésétol függoen **ut1** vagy **ut2** utasítást hajtja végre. Ha az else hiányzik, a következo utasítás érvényes.

Az AML editor

Az aktuális programfile-t a fomenuben választjuk ki. Ha nem létezo nevet adunk meg, a rendszer új file-t hoz létre.

Az AML programoknak .aml kiterjesztésueknek kell lenniük.

Az editorba belépve kiadhatunk:

- elsodleges parancsokat (a képernyo tetején levo mezoben adhatók meg. A mezobe a HOME billentyu lenyomásával juthatunk el.)
- sor parancsokat (a képernyo bal oldalán, a sorok elott adhatók)
- karakterszintu parancsokat (a szerkesztett sorokban adhatók)

A funkcióbillentyuk használata:

F1	help
F2	törli a nem kívánt változtatást
F3	reset, törli a hibát vagy kiadott parancsot
F4	kiírja az utolsó keresési (find) parancsot
F5	kiírja az utolsó cserélési (change) parancsot
F6	belépés a betanító üzemmódba
F7+ALT	az aktuális, betanított robotpozíciót betölti a szerkesztett programba.
F8	kilépés az editorból
F9	ugrás a file elejére

F10 ugrás a file végére

Néhány fontos elsodleges parancs:

C change, csere: a megadott stringet minden elofordulásában a megadott másikra cseréli

DI dir: listát ad az AML könyvtárról

F find, keresés: megkeresi a megadott stringet

PR print, nyomtatás: kinyomtatja a filet

SA save, mentés: elmenti a szerkesztett filet más néven

SY syntax check, szintaktikus ellenorzés: ellenorzi a programot szintaktikai szabályok alapján

Néhány fontos sorparancs

A after: blokk kezdetének kijelölése

B before: blokk végének kijelölése

C copy: másolja a megadott sort a fenti kijelölés után (A) vagy elé (B)

CC-CC másolja a megadott sorokat a kijelölés szerint

D delete: törli a sort

DD-DD törli a megadott sorokat

Ixxx insert: beszúr xxx db üres sort

M move: mozgatja a megjelölt sort a kijelölés szerint

Az AML szimulált betanítás

Kiadható parancsok:

F1 help

TAB robotkar konfiguráció váltás (jobbkezes-balkezes)

SHIFT+TAB konfiguráció váltás

cursor finom robotpozícionálás

SHIFT+cursor durva pozícionálás

PgU, PgD R tengely mozgatás

F7, F8 Z tengely mozgatás

F9 grasp

F10 release

ESC eltárolja a betanított robotkoordinátákat

Az AML szimulátor

F9	szimuláció start
F5	visszatérés az editorhoz
F8	visszatérés a fomenuhöz
F3,F4	digitális output kijelzés léptetése
F6	opciók beállítása
F7	digitális input értékének változtatása

BAPS

A Bosch cég saját robotjaihoz kifejlesztett programnyelv. Hatékony, sok jól használható szolgáltatással.

VAL

Eredetileg a PUMA robotokhoz kifejlesztett programnyelv, egyszerusége és áttekintehtosége miatt talán a legelterjedtebb. Ma már nagyon sok robotvezérlésben használják.

Feladatleíró nyelvek

AUTOPASS

Magas szintu, a mesterséges intelligencia elemeit is használó programnyelv, ma még csak kísérleti fázisban van.

Robotok alkalmazástechnikája

Tipikus robotizált munkahelyek

A robotizált munkahely eszközei

A robot információs kapcsolatai

Esettanulmányok

Cella kiszolgálás

Szerelés

Nemzetközi trendek, irányzatok

Robot és mechatronika

Irodalomjegyzék

1. Dr. Kulcsár Béla: Robottechnika
2. Szabóné dr. Makó Ildikó: Robottechnika.

http://www.szgt.uni-miskolc.hu/~mako/robel1.pdf, http://www.szgt.uni-miskolc.hu/~mako/robel2.pdf, http://www.szgt.uni-miskolc.hu/~mako/robel3.pdf http://www.szgt.uni-miskolc.hu/~mako/robel4.pdf