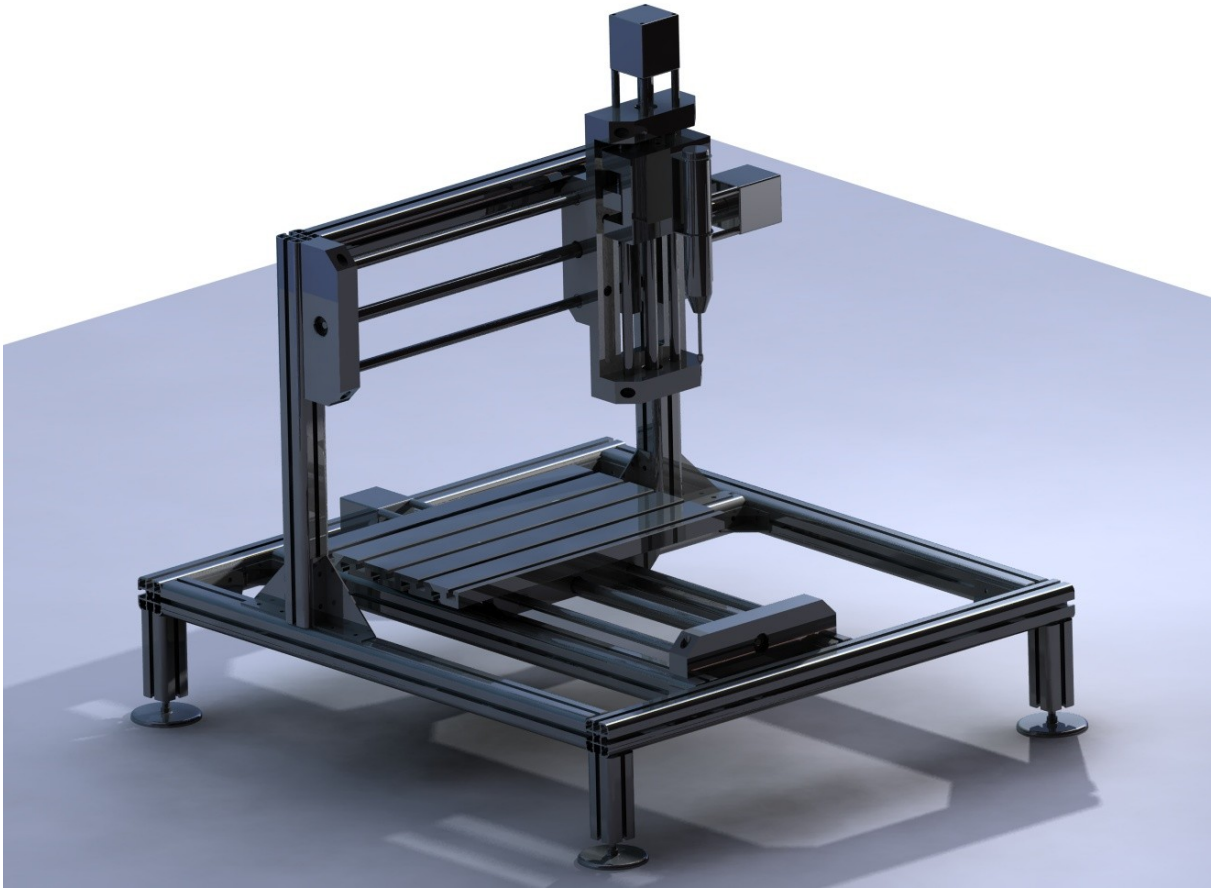
Gravírozógép tervezése és kivitelezése



Készítette:

Bozi István
Tomozi György

Tartalom

1. Bevezetés

Feladatunk nyákmaró CNC gép tervezése és kivitelezése volt mely alapvetően az Isel Kft által kiírt pályázatra készült. A gép tervezésében, kivitelezésében az alábbi pontok megvalósítását céloztuk meg:

- Egyszerű konstrukció
- Továbbfejleszthetőség
- Költséghatékonyság
- Egyediség

A felsorolt pontok alapján elképzeléseink kivitelezése sikeresnek mondható, mivel a legtöbb szempontot sikeresen teljesítettük. Ezen tulajdonságok közül talán a gép egyedisége a legfontosabbak, mivel olyan dolgokat építettünk be, mely a mai korszerű gépek közül is csak néhány gyártó által készített megmunkáló egység tud. Említésképpen a NURBS algoritmus alkalmazása, szabályozott pályakövetés és TCP/IP-én keresztüli kommunikáció alkalmazása.

Irodalom kutatásaink során kiderült, hogy egyedül egyetlen gyártó (GE Fanuc) CNC gép vezérlője támogatja a NURBS interpolációt. Ezzel „tömörítési eljárással" a géppel való kommunikáció során jelentős időt lehet megtakarítani.

A megépített szerkezet alkalmas lehet arra, hogy az amúgy is már „bekábelezett" világban nagy távolságot hidaljon át. Lehetőséget biztosít például otthoni munkavégzésre, vagy egy gyárcsarnokban elhelyezett gépegyüttes egyidejű, gazdaságos működtetésére.

2. Mechanika tervezése

2.1. Konstrukció

A mechanika tervezését piacon beszerezhető gépek áttanulmányozásával kezdtük [1, 2, 3]. A gép kialakításának fő irányadó elvei a következők voltak:

- A4-es lap megmunkálására alkalmas legyen (pályázat előírta)
- Költséghatékonyság
- Gyorsaság
- Egyszerűség

A mechanika kivitelezésére két elképzelés merült fel:

- Álló munkapad, és az x irány hordozza a többi tengelyt is (fekvőagyas)

1. ábra

- Munkapad az x tengelyen, és az y hordozza a z-t (konzolos)

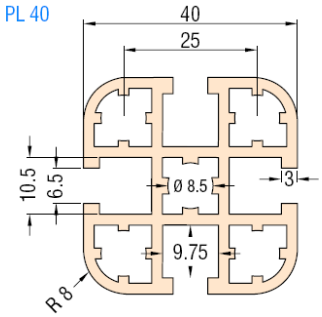
2. ábra

Az utóbbi megoldást választottuk, mert kialakításilag az első bonyolultabb, több alkatrészt (orsót, motort) igényelt volna.

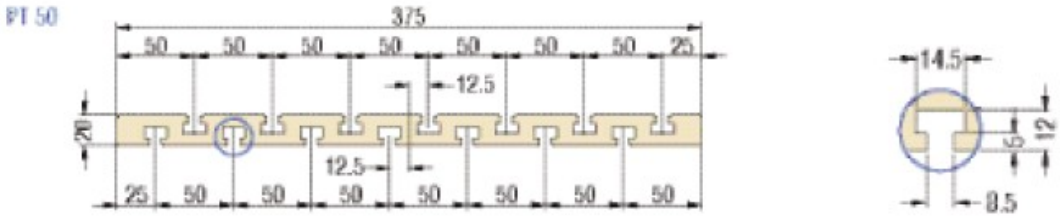
A konstrukcióhoz az ISEL cégtől rendeltünk PL40-es (3. ábra) alumínium zártszelvény, sarokelem és PT 50-es (4. ábra) rögzítő, megmunkáló felület. A rendelt zártszelvényből terveztük és készítettük el a gép állványzatát. Az állványzat tervezésénél törekedtünk arra, hogy úgy alakítsuk ki a mozgatási tengelyeket, mozgásteret, hogy a tengelyek mozgási hosszai közel hasonlóak legyenek

ezzel is optimalizálva a gyorsaságot.

Profilok:

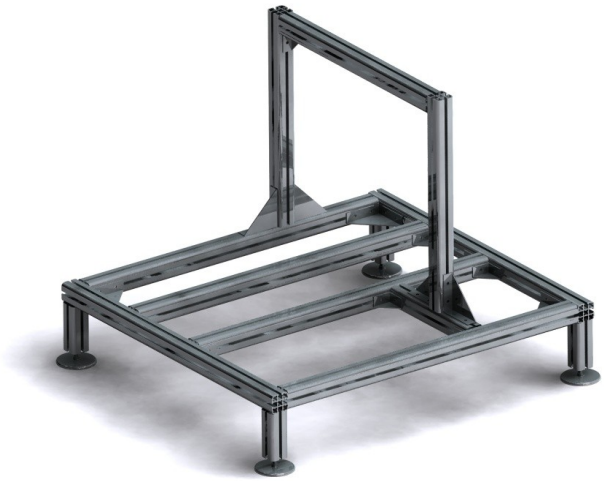


3. ábra



4. ábra

Állványzat:



5. ábra

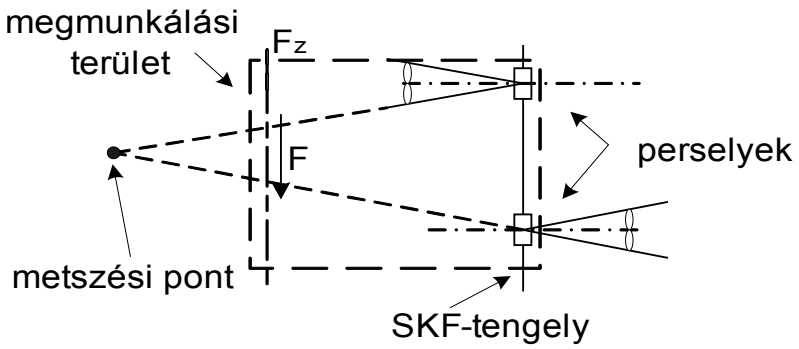
A mozgórészek tervezésénél fő szempont volt, hogy olyan kialakítású legyen a szerkezet mely nem feszül be könnyen és gyorsan mozgatható, és amennyire lehet szabványos alkatrészeket is tartalmazzon.

Tervezését megelőzően, körbenéztünk hajtásláncokat gyártó cégeknél, ötletmerítés illetve akár annak megvásárlása céljából. Kutatásunk során felmerült elképzelések:

- Bordás szíjas lineáris egység
- Golyósorsós lineáris egység
- Lineáris vezeték + hajtás, ill. hajtástervezés
- Lineáris vezeték tervezése + hajtás, ill. hajtástervezés

Mi a lineáris vezeték tervezése + hajtás tervezését választottuk mivel kész egységek beszerzése nagyon költséges lett volna, és a kiírás szerint nagy részben saját tervezésű elemekből kellett a gépet elkészíteni. Így vezetőtengelynek 12mm átmérőjű SKF tengelyt és a hozzá illő PAP12-20 ill. PAP12-25 zsugorbronz perselyeket választottuk, mivel kis súrlódású, nagy kopásállóságú és cserélhető. A mozgató orsóink gyárilag hengerelt trapézmenetes szárak, melyek tengelyvégeit kellett megmunkáltatnunk terveink alapján, és hozzá bronzból anyákat készíttetni. Ezt a konstrukciót azért választottuk, mert árban harmadrésze volt a golyósorsós hajtásnak, igaz több alkatrész kell hozzá, de rugós előfeszítéssel ez is holtjátékmentesen hajt. Az orsó befogásánál ügyeltünk arra, hogy axiálisan előfeszített legyen, mely elképzelést, kivitelezést hajtásokat gyártó cégek is alkalmaznak. A csapágyazáshoz mélyhornyú golyóscsapágyakat alkalmaztunk (61901-2Z, 61900-2Z). A hordozók tervezésénél ügyeltünk arra, hogy a terhelésnek megfelelően legyenek az alátámasztások és befeszülés végett pedig, hogy elég távol legyenek egymástól a hordozóbakok.

A 6. ábrán látható kép mutatja, hogy L1, L2 hosszakat kellet akkorára méretezni, hogy a tengelyeken kívül ébredt erő által képzett súrlódó kúpok metszéspontja a tengelyeken kívülre kerüljön.



6. ábra

A precíz hajtáshoz léptetőmotorokat alkalmaztunk, melyeket ugyancsak az ISEL cégtől vásároltunk. Mivel a különböző tengelyek igénybevétele hasonló a tervezésnek köszönhetően, így azonos motorokat alkalmaztunk mindhárom tengely hajtására. A motorok alapban 200 lépésesek, melyet a

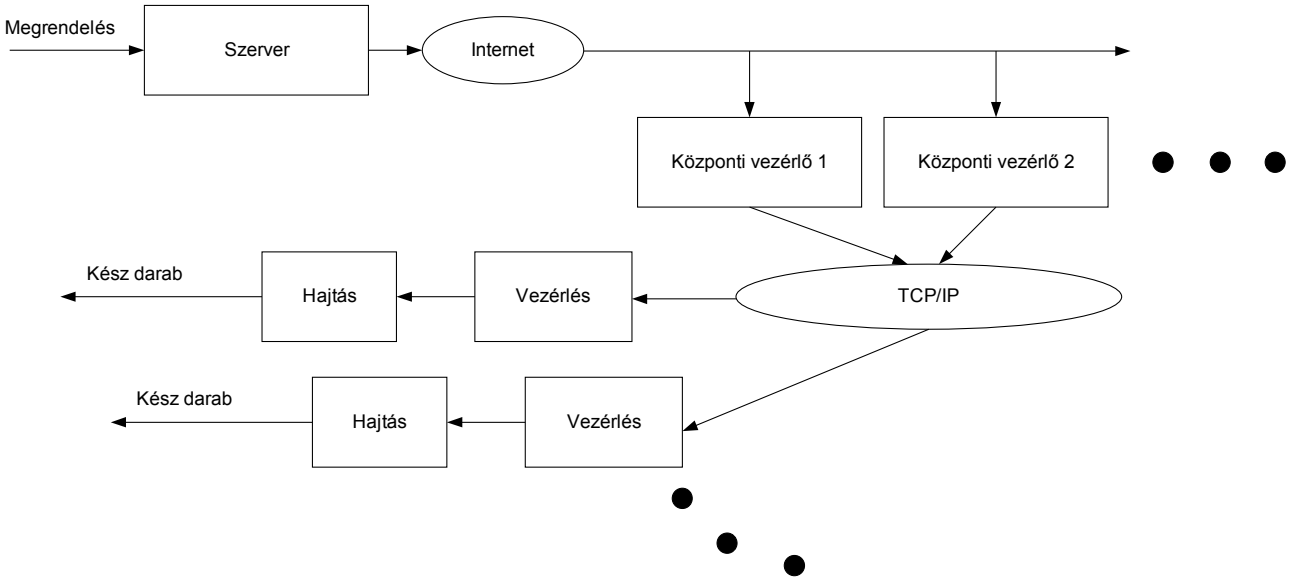
vezérléssel akár tizenhatszor nagyobbra lehet növelni, így számítás alapján a hajtás minimum pontossága 0,02 mm.

A tervezett alkatrészek műszaki rajzait és a szabványos alkatrészek listáját a mellékletekben találják.

3. A villamos vezérlés

Célunk egy olyan rugalmas, alkalmazkodó, könnyen használható berendezés elkészítése volt, amely akár ipari környezetben is megállja a helyét. Szem előtt tartottuk azt is, hogy egyszerre több berendezés is tudjon együttműködni valamint, hogy a berendezések ne kötődjenek helyhez, akár otthon is működtethetőek legyenek.

A fenti követelmények megvalósítására a manapság már mindenhol elérhető LAN hálózatot alkalmaztuk, mely a rugalmasságot biztosítja. A következő ábra mutatja a gép átfogó blokkvázlatát a megrendeléstől egészen a kész darabig.

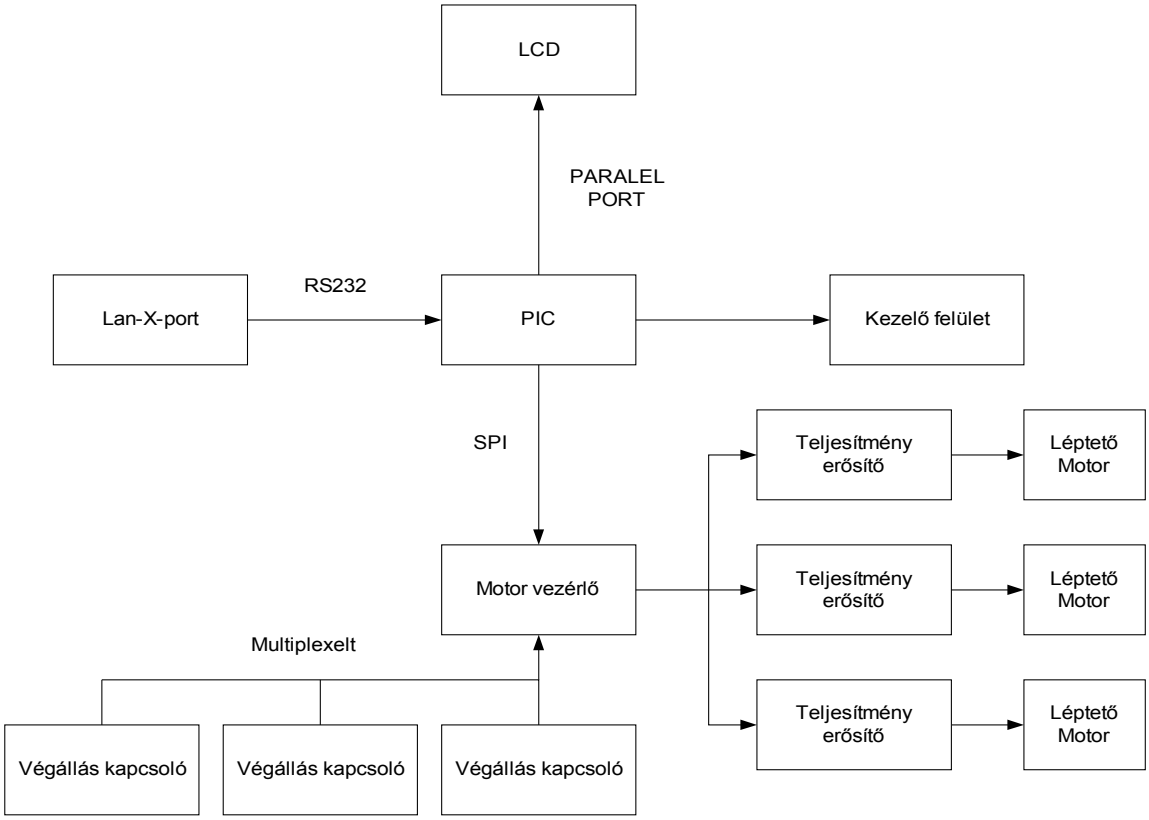


7. ábra

A megrendelést a felhasználó – azonosítóval és belépési jelszóval rendelkező személy – indítja azzal, hogy egy DXF formátumú fájlt feltölt egy távoli FTP szerverre. Amikor a feltöltés befejeződött, akkor Valamelyik szabad központi vezérlő számítógép – amelyik éppen szabad – megkezdi a fájl letöltését. Amikor a letöltés befejeződött, a fájl átkerül egy másik mappába, ahol automatikusan archiválódik. A letöltött fájlt a központi vezérlő feldolgozza – méretarány, technikai keret, mértékegység, stb. -, majd LAN hálózaton keresztül átküldi egy szabad CNC gépnek. Ott az adatok fogadása után megkezdődhet a gravírozás. A kész darab pedig akár azonnal postára adható.

A fentiek alapján, megfelelő kapacitás esetén a munkadarab elkészítése a megrendeléstől számított kb.1-2 órán belül elkészíthető. Munkánk során csak egy CNC gépre készítettük el a teljes berendezés, azonban ez könnyen bővíthető.

A villamos vezérlés blokkvázlata a következő ábrán látható. A gép vezérlését egy dsPIC mikrokontroller végzi 80 MHz-es órajellel. A kontroller SPI, RS232, párhuzamos kommunikáció segítségével kommunikál a hozzá kapcsolódó eszközökkel.



8. ábra

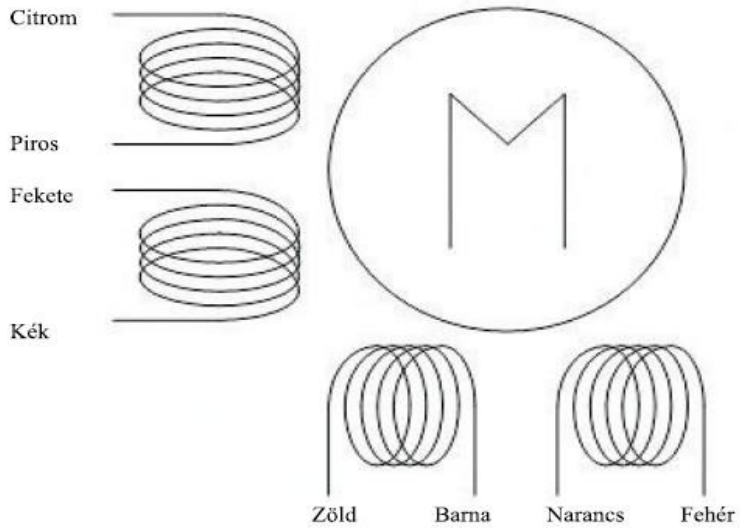
A LAN hálózat protokoll konverzióját egy Lantronix gyártmányú X-port modul végzi. Sorosan kapcsolódik a mikrokontrollerhez egy Trinamic gyártmányú léptetőmotor vezérlő IC. A motorvezérlő szintén soros kommunikációval kapcsolódik a léptetőmotor meghajtó modulokhoz, melyek adják a léptetőmotorok bemenő jeleit. A PIC-hez kapcsolódik párhuzamosan a kezelő felület, illetve egy LCD kijelző.

3.1. A léptetőmotorok

A léptetőmotorokat az ISEL cég forgalmazza, az Ő termékkínálatukból választottuk őket a várható nyomatékszükséglet és áramfelvétel alapján.

A léptetőmotorok néhány tulajdonsága:

- Lépésszög: 1,8 fok (200 lépés/fordulat)
- Pozicionálás pontossága: 5%
- Fázisok száma: 2
- Hőmérséklet max.: 80 Celsius
- Dielektromos szilárdság: 500V DC
- Névleges áramfelvétel: 1,25 Amper
- Kihajtótengely átmérő: 10mm
- Tömeg: 0,84kg
- Tekercsfeszültség: 1,5V
- Alkalmazott bekötés: bipoláris soros
- Nyomaték: 1,5Nm



9. ábra

A fenti ábra a motor bekötési vázlatát mutatja. A motorokat sorba kötött tekercsekkel üzemeltetjük, mert ebben az üzemmódban fellépő áram a meghajtó áramát nem haladja meg.

3.2. A motormeghajtó – TMC246

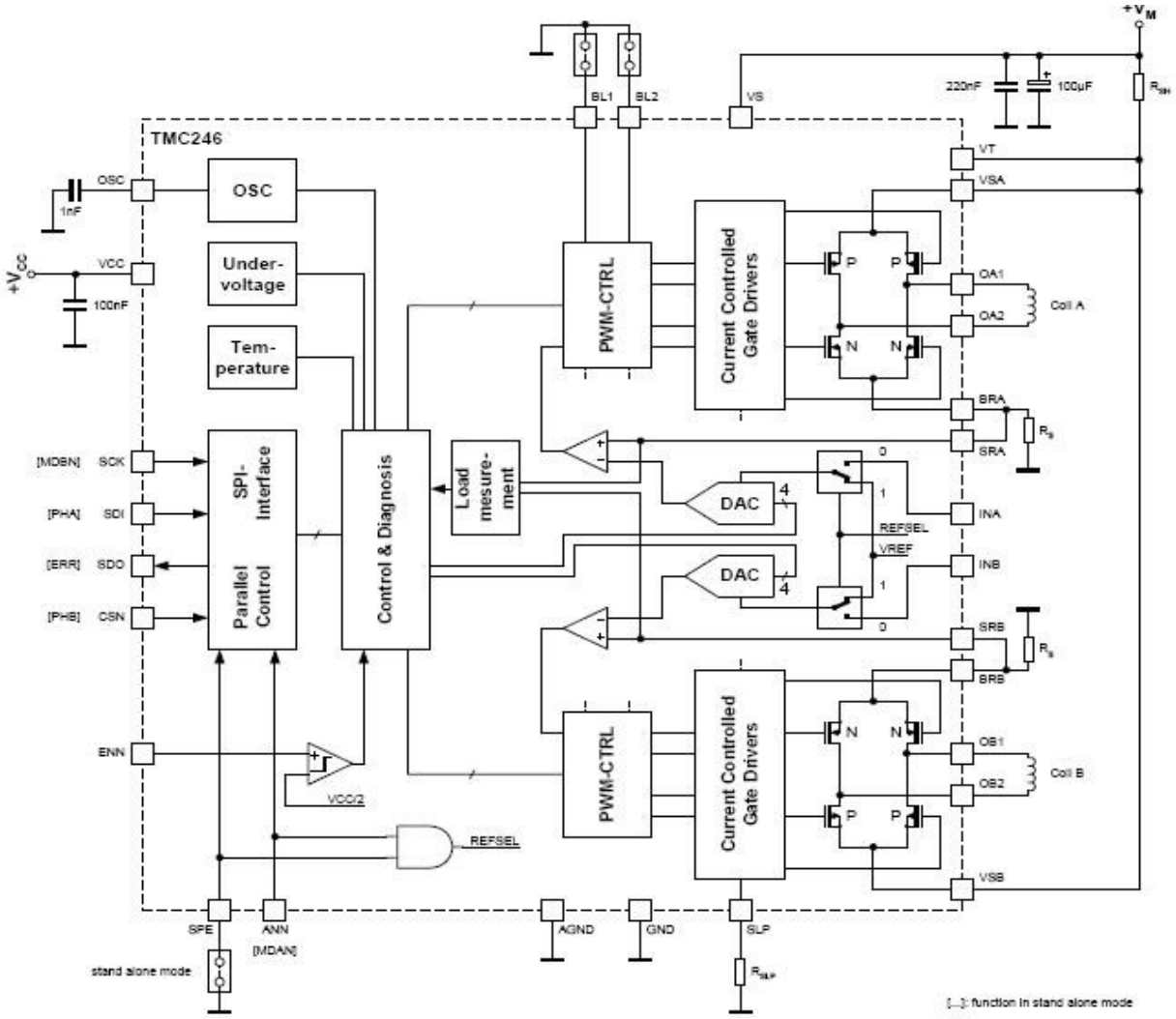
A Trinamic által kifejlesztett TMC246-os céláramkör bipoláris, kis teljesítményű motorok meghajtására alkalmas akár mikrosztep üzemmódban is. Rendelkezik egy szabványos SPI

interfésszel is az analóg vezérlés mellet. Soros kommunikációt használva egy driver-lánc is kialakítható több áramkör felhasználásával. Több diagnosztikai funkcióval is rendelkezik, és az alacsony bekapcsolási ellenállású úgynevezett TrenchFET® -ek alkalmasak akár 1,5A-ig motormeghajtásra hűtés nélkül. Alacsony energiafelvétele sleep állapotban, kicsi mérete miatt akár elemes alkalmazásokban is felhasználható.

Fontosabb paraméterek:

- Kimenet: 1500mA/tekercs
- Tápfeszültség: 7-34V DC
- Logikai tápfeszültség: 3,3 és 5V DC
- Tokozás: PQFP44
- Interfész: analóg és SPI ®
- Mikrolépés: 64
- Csendes üzem változó meredekségű táplálással
- Védelem: túláram, melegedés, álló motor
- RoHS kompatibilitás

Az eszköz belső blokkvázlatát mutatja az ábra.



10. ábra

A motormeghajtó az OSC lábra kapcsolódó kondenzátor segítségével állítja elő a belső órajelét. A vezérlése történhet sorosan vagy analóg módon, mi a soros SPI ® kommunikációt választottuk. Az SPE lábon keresztül állítható be, ha a meghajtókat láncba kapcsoljuk. Láncba kapcsolás esetén a soros bemenő jel transzparens módon folyik a kimenet felé. Az SLP lábra kapcsolt ellenállással állítható be a tekercsáramok maximális meredeksége. Az aktuális tekercsáramot az SRA/SRB lábra kapcsolt belső műveleti erősítő figyeli és PWM jel segítségével ehhez állítja a kimeneti H-híd tranzisztorait. A motor az OA és OB kimenetekre csatlakozik. A VT lábon keresztül egy általános rövidzárlat figyelés történik, ami az IC védelmét szolgálja. Érdekes még a BL1 és BL2 láb, melyek az egyes tekercsek bekapcsolása közötti milliszekundumos nagyságrendű holtidőt állítják be. Ezzel termelt zavart lehet csökkenteni, a motorokhoz igazítani.

Az áramkör kicsi mérete és a nagy áram miatt a gyártó erős követelményeket szab a nyák kivitelével kapcsolatban. A legfontosabb ezek közül a masszív földpotenciál. Szintén fontosak a

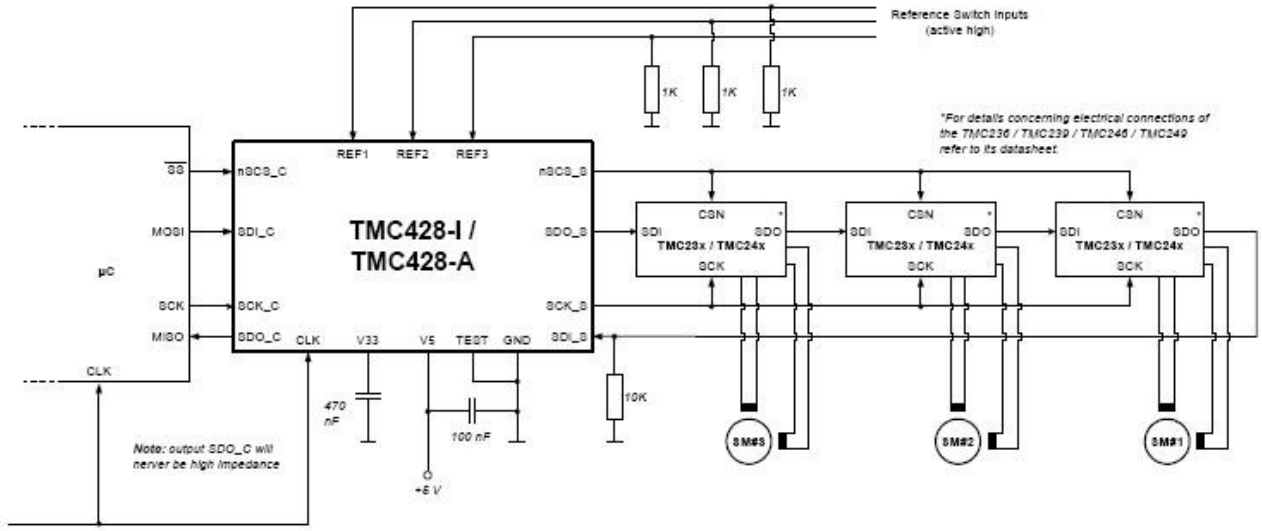
szűrőkondenzátorok helye megválasztása és elhelyezés. A berendezés beüzemelése során ez okozta a legtöbb gondot és fejtörést, hogy a termelt zavart a bemeneti soros kommunikációtól el tudjuk választani. A motor kimeneti lábak egyúttal hűtésként is szolgálnak, ezért azokat vastag vezetékkel kell bekötni. Szintén ezért fontos a nyák szimmetrikus kivitele.

A driver részletes adatlapját mellékeltük.

3.3. A motor vezérlő – TMC428

Szintén a Trinamic által fejlesztett áramkör, ami nagy segítséget jelentett számunkra, hiszen levette a mikrovezérlő válláról kritikus realtime motorvezérlési feladatot, ezzel növelve az interpoláció sebességét. Árát és műszaki tartalmát tekintve mondhatjuk, hogy olcsóbb, mint egy mikrokontrolleres alkalmazást fejleszteni.

Az áramkör akár 3 fent említett meghajtót tud kezelni egyidejűleg soros SPI ® kommunikáció segítségével, ha azok láncba vannak fűzve. Az összes kritikus taszkot a vezérlőbe integrálták úgy mint: motor áram figyelés, hiba figyelés, lépés számolás, sebesség vezérlés, mikrostep, referencia kapcsolók kezelés rámpagenerátor stb. Mivel két SPI ® kimenettel rendelkezik ezért a mikrokontrollerrel is képes sorosan kommunikálni. Képes kezelni és tárolni a motor pozíciót, sebességet ás gyorsulást. A motor üzeme során is változtathatók az előre beállított paraméterek. Egészen 64 mikrolépésig finomítható a motor forgása, csökkentve így annak zaját. És növelve a pozicionálási pontosságot. A mikrolépés tábla is módosítható, a motor paramétereihez igazítható.



11. ábra

A fenti ábra mutatja a vezérlő egy tipikus elrendezését a driver-lánccal.

▪ Lépés frekvencia

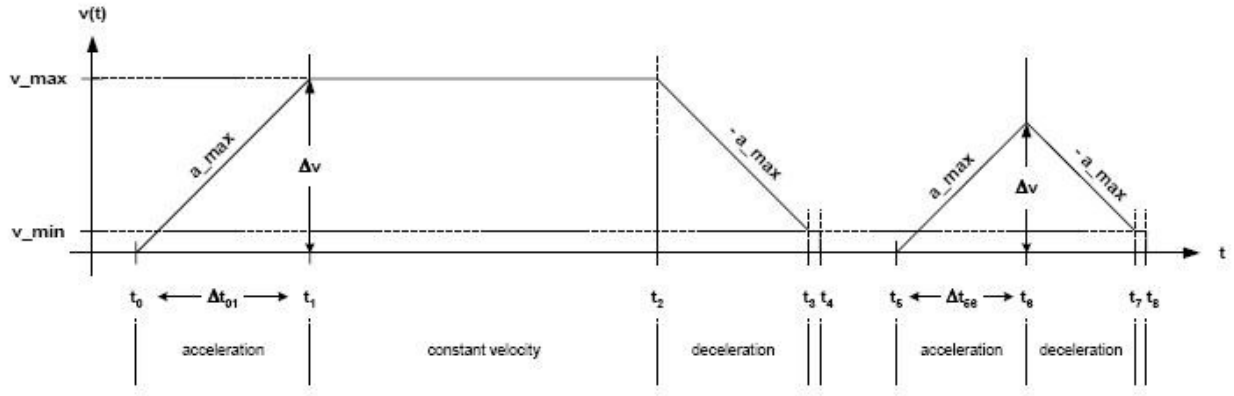
A maximális kommunikációs sebesség az órajel 16-od része. Az ebből adódó értéket osztva a kiküldött bitek számával kapjuk a maximális lépési frekvenciát. Ez esetünkben – 10MHz-es órajel és 3x16bites driver kommunikáció – 10/16/(3*16) = 19 kHz. Ez bőven elegendő hiszen így a motor kb. 96 fordulat/másodperces sebességgel forogna, ha tudna, de ezt a sebességet a motor nem tudja elérni. A maximális lépésfrekvencia ilyen motoroknál kb. 5-10kHz.

▪ Vezérlési lehetőségek

A kontroller négyfajta vezérlési lehetőséget kínál:

- **Pozícióvezérlés (RAMP MODE):** ez a vezérlés a pozicionálási feladatokra a legalkalmasabb. Csak a kívánt pozíciót kell beállítani és a vezérlő automatikusan generálja a trapéz sebesség profilt, ami a motort a kívánt pozícióba juttatja.
- **Pozícióvezérlés (SOFT MODE):** Hasonló az előbbihez, azonban a lefutás egy exponenciális görbe.
- **Sebességvezérlés (VELOCITY MODE):** A felhasználó által megadott maximális gyorsulással gyorsítja a motorokat és állandó sebességet tart.
- **Sebességvezérlés (HOLD MODE):** Hasonló az előbbihez, azonban a maximális gyorsulási paraméter nincs figyelembe véve.

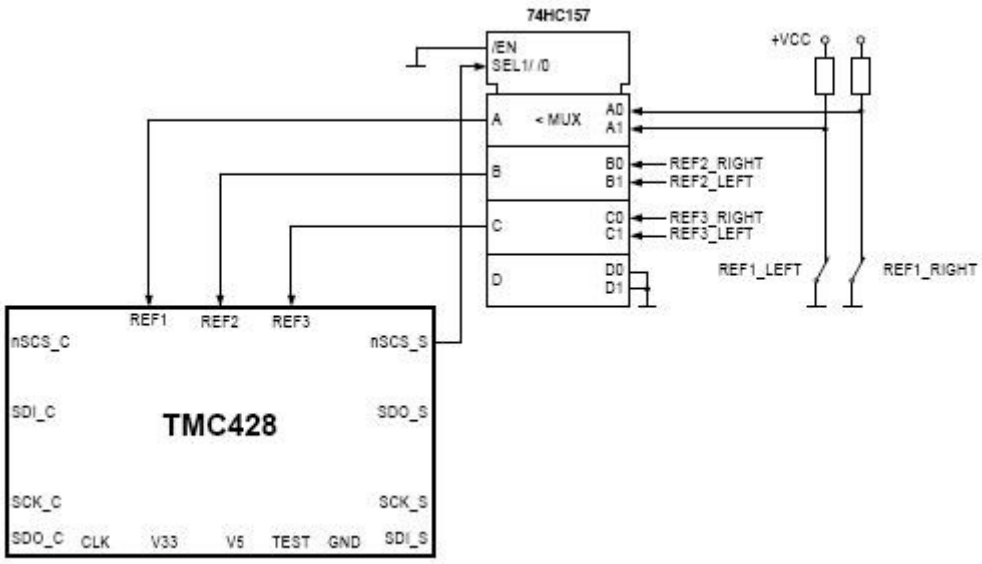
A fentiek közül mi a VELOCITY MODE-ot választottuk mert ez biztosítja a legpontosabb és legösszehangoltabb mozgást.



12. ábra

- **Végállás kapcsolók**

A blokkvázlatból látszik, hogy ehhez a vezérlőhöz csatlakoznak a végálláskapcsolók, melyek a megmunkálási teret korlátozzák. Mindegyik tengely mindkét végén helyeztünk el végálláskapcsolókat, így összesen 6db kapcsolót kellett alkalmaznunk. Mivel a vezérlőnek csak három ilyen bemenete van ezért az órajel segítségével multiplexeltük a bemenetre, ezt mutatja a következő ábra.



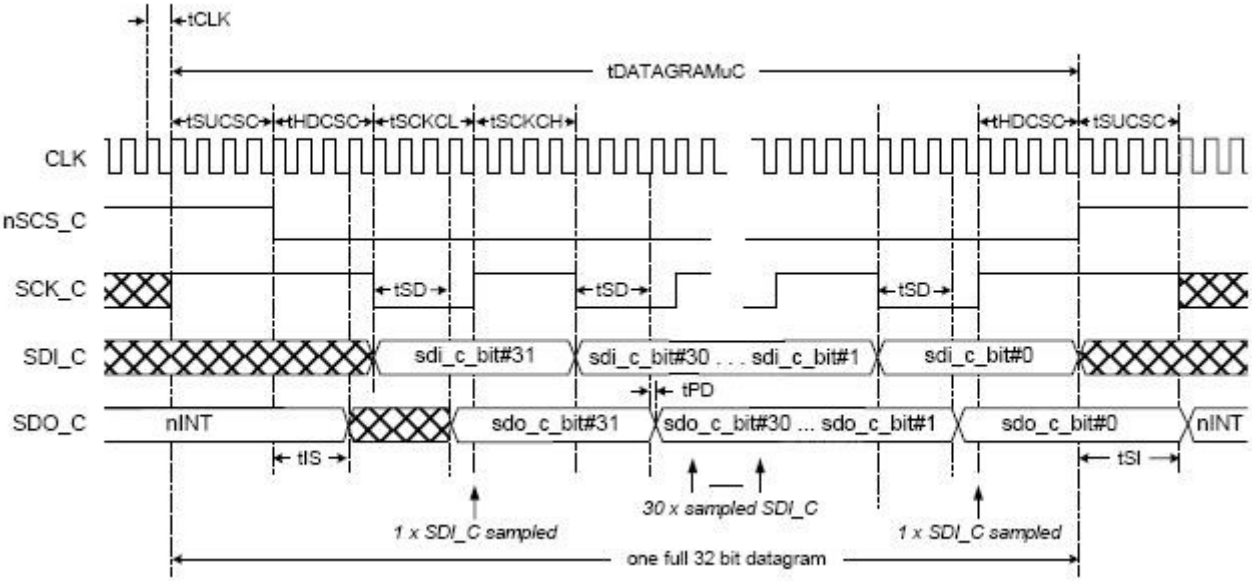
13. ábra

A végálláskapcsolók OMRON típusú hagyományos mikrokapcsolók. Ugyan a vezérlő a véghelyzet elérésekor nullázza a pozíció számlálóját, de ez számunka érdektelen, mivel nincs beleszólása a sebességvezérlés üzemmódba.

Fontos még az is, hogy a végállás elérésekor a kontroller önállóan megállítja a motort és egy később ismertetett flag-el jelzi ezt a mikrokontrollernek.

- **SPI kommunikáció**

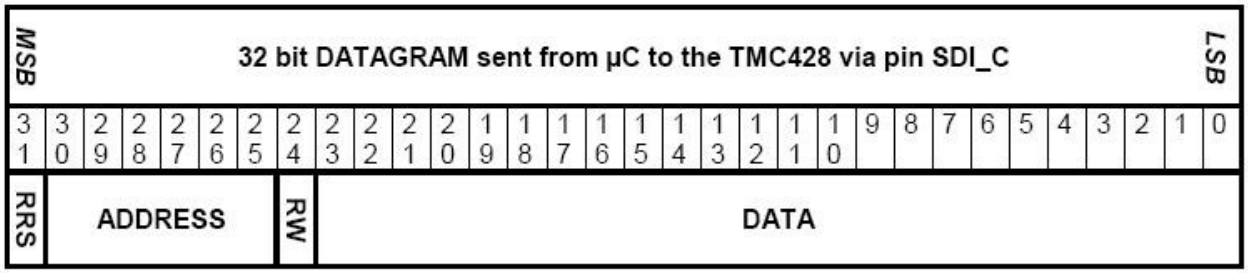
A soros SPI kommunikációt mutatja a következő ábra:

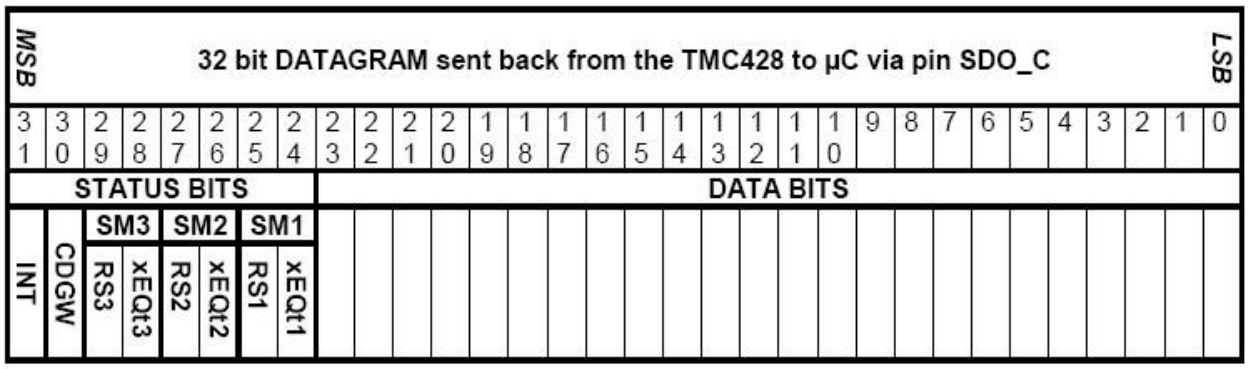


14. ábra

Az adatok a ChipSelect láb lefutó élére sorban egymás után az órajellel szinkronban kerülnek kiküldésre. Ezzel párhuzamosan érkeznek a mikrokontroller felől az adatok.

A datagrammok struktúráját mutatja a következő két ábra:





15. ábra

A mikrokontroller az RRS biten (alsó és felső RAM tábla)és az ADRESS címen keresztül címzi a TMC egyes regisztereit. Az RW bittel választható ki, hogy az adott regisztert írni vagy olvasni szeretnénk. Majd következnek a regiszterfüggő adatok.

A TMC az INT bit segítségével közli ha valami olyan esemény történt, ami esetleg megszakítást igényel (motor végálláson, túl magas áramfelvétel stb.), ezt egyébként a mikrokontroller megszakítás lábára is lehet kötni. A vezérlőn keresztül közvetlenül küldhetünk adatokat a meghajtó áramköröknek. Amikor ilyet küldünk a CDGW bit jelzi hogy a datagramm feldolgozás nélkül áthalad a vezérlőn. Az RS1-3 bitek a referencia kapcsolók állapotait reprezentálják, míg az xEQt1-3 bitek akkor jeleznek, ha pozícióvezérlés estén a motor eléri a kívánt pozíciót. Ebből bizonyos beállítás esetén akár az INT biten megszakítás is generálható.

A teljes RAM-tábla a mellékelt adatlapon megtalálható (15.oldal). A tábla alsó felében találjuk a regisztereket, melyekkel a motorok paraméterei változtathatók. A felső félben az összes motorra hatással bíró regiszterek találhatók, illetve a mikrolépés tábla állítható be.

32 bit DATAGRAM sent from a µC to the TMC428 via pin SDI_C																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RRS	ADDRESS						RW	DATA																							
																		data @ odd RAM addresses								data @ even RAM addresses			
1	0	32 x (2x6 bit) driver chain datagram configuration range																NxM_1	signal_codes							NxM_0	signal_codes			
1	1	32 x (2x6 bit) quarter period sine wave LUT range																quarter sine wave values (amplitude)								quarter sine wave values (amplitude)			

16. ábra

Ezt a táblát minden indításkor fel kell tölteni adatokkal. Az első a hajtáslánc konfiguráció, melyben megadható a motorok ébredési állapota, az hogy melyik tekercs mekkora áramot kapjon,

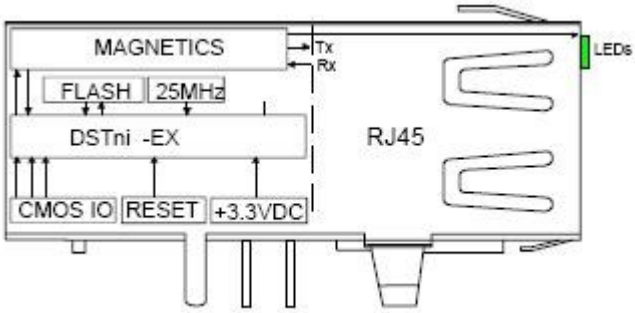
forgásirányt, illetve, hogy milyen meredekséggel változhat az áram. Ez a rész gyakorlatilag azonnal átkerül a driver-lácra és ott aktualizálódik.

A második része a mikrolépés tábla, melyet tetszőlegesen kialakíthatunk. A 4-64 bites mikrolépés használata esetén ez kerül alkalmazásra.

A kontroller részletes adatlapját mellékeltük.

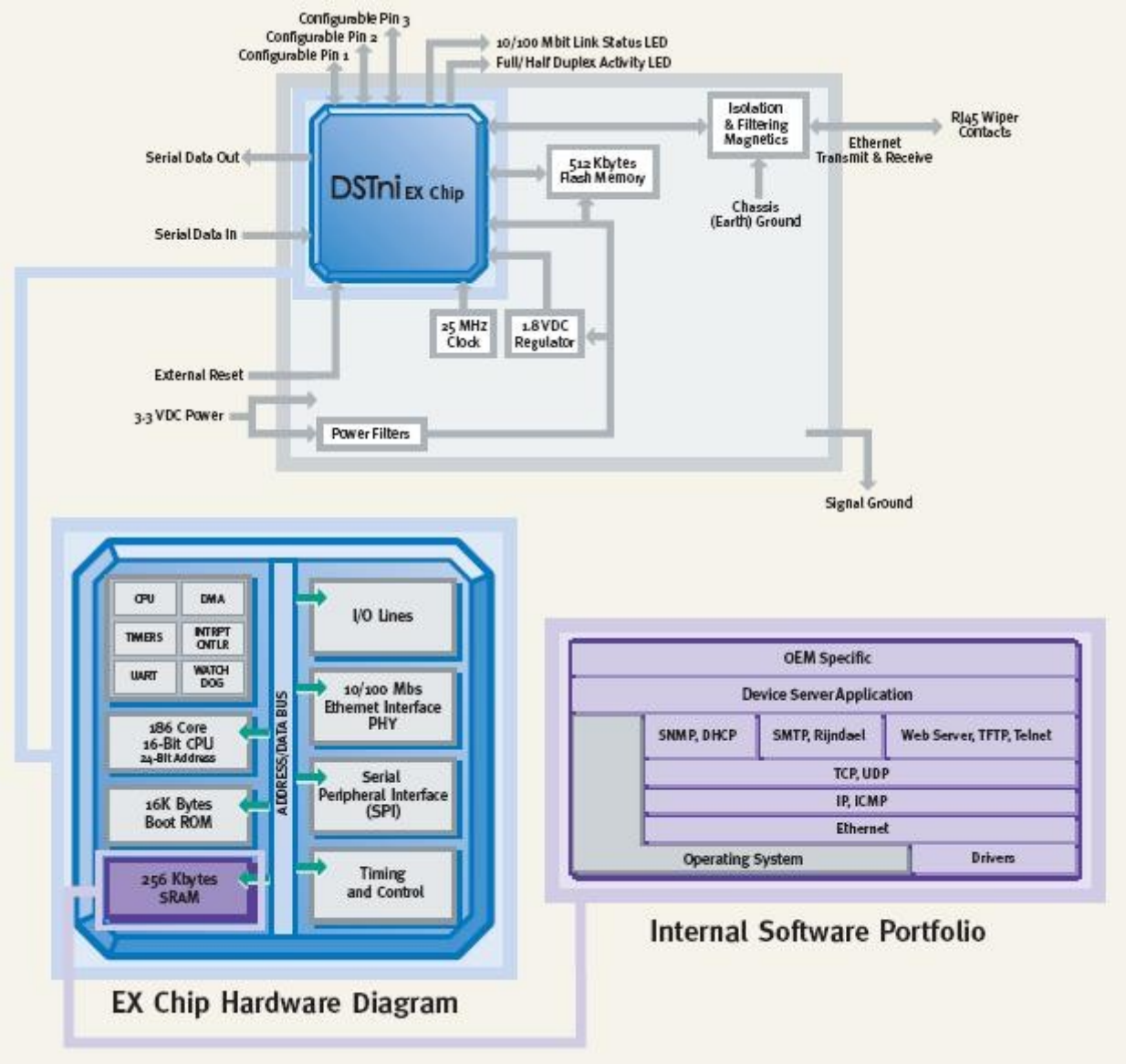
3.4. X-Port modul

Tekintettel arra, hogy az egységünket a LAN hálózatra akartuk csatlakoztatni és a TCP/IP stack-el nem akartuk terhelni a mikrokontrollert, ezért egy olyan külső modult kerestünk hozzá, mely ezt elvégzi a PIC helyett. A LANTRONIX cég sokféle megoldással rendelkezik ilyen esetekre.



17. ábra

Az általunk választott modulba mindent beleintegráltak, ami ahhoz szükséges, hogy a TCP/IP és akár az UDP kommunikációt biztosítani tudja. A modul a beállított IP címre érkező adatokat a soros porton keresztül továbbítja.



18. ábra

Egy kicsi operációs rendszer fut a modulban egy 16-bites CPU segítségével, mely a feldolgozás után a bejövő adatokat egy szintén a modulba épített SRAM memóriában puffereli. A modul beállítása akár böngészőből akár Telnetes kapcsolaton keresztül is levégezhető. Sőt akár a mikrokontroller is elvégezhetné a soros porton keresztül. Mi tekintettel arra, hogy nem szükséges a beállításokon később változtatni, böngészőn keresztül állítottuk be a modul paramétereit.

19. ábra

A beállítások között számos paraméter megtalálható úgy, mint:

- IP cím beállítása
- Fix IP vagy DHCP beállítás
- Hozzáférési jelszó
- CPU teljesítmény 48/88MHz
- Soros port beállítása (Baudrate, paritás, handshake, csomagok mérete, buffer törlés stb.)
- TCP beállítások (port, kapcsolat paraméterei)
- UDP kapcsolat beállításai
- Esemény vezérel E-mail beállítások
- GPIO lábak beállításai
- stb.

A modul részletes adatlapját mellékeltük.

3.5. A mikrokontroller

A mikrokontrollert két fontos paraméter alapján választottuk ki, az egyik és legfontosabb, hogy

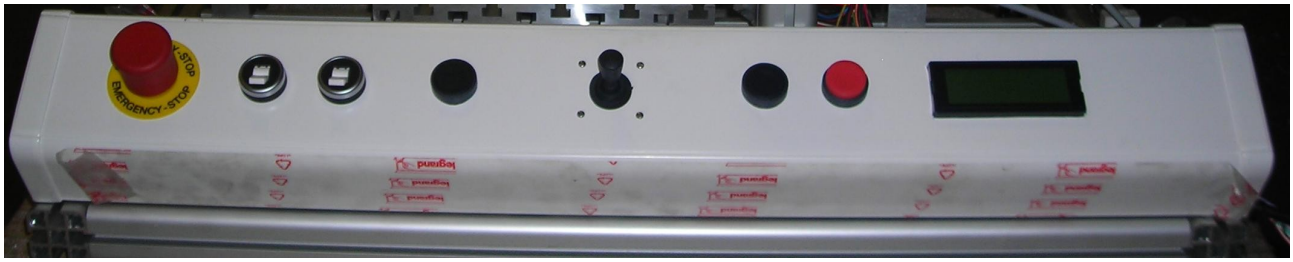
rendelkezzen annyi és olyan kommunikációs perifériával, amennyi nekünk szükséges. A másik szempont pedig a sebesség volt. Választásunk a dsPIC30F6014-re esett. Melynek nagy előnye a 16 bites regiszterszintű matematikai műveletek (összeadás, kivonás, osztás, szorzás), valamint a PLL hurokkal növelhető oszcillátor frekvencia.

A mikrokontroller tápfeszültsége 3,3V, úgy mint az X-port modulnak és a léptető motor vezérlésnek és meghajtásnak, az oszcillátorfrekvencia pedig 10MHz. Ez is igazodik a léptetőmotor vezérlőhöz. Az órajel a PIC-en belül PLL hurokkal biztosítja magának a 80MHz-es órajelet (20MIPS). A kontroller egyik SPI buszára csatlakozik a motor vezérlő, a másikra pedig egy Flash memória, amit ebben a konstrukcióban nem használunk ki. Az UART perifférián keresztül tartja a kapcsolatot az X-port modullal. Párhuzamos 8 bites kommunikáció folyik az LCD egységgel.

A PIC részletes adatlapját mellékeltük.

3.6. Kezelő felület

A kezelőfelületet igyekeztünk úgy kialakítani, hogy könnyen használható legyen.



20. ábra

Mivel a gép kezelése általában jobb kézzel történik, ezért a szabadon maradó kézzel könnyen elérhető a bal oldalra telepített vészkapcsoló, mely a gépet azonnal megállítja a táplálás megszüntetésével. A második kapcsoló a gép főkapcsolója, a harmadik most még nem üzemel, de később a géplámpa, vagy a hűtőfolyadék bekapcsolását végezheti. A következő fekete gomb megnyomásával érhető el, hogy botkormány a Z-tengely mozgassa. Felengedett állapotban ugyanis csak az X és Y-tengely kezelésére alkalmas. Az LCD kijelző mellett lévő két gomb a nyugtázó valamint a törlő gomb. Ezeknek a nullpontbemérésnél, a program indításnál van szerepük.

Az LCD kijelző egy kétsoros, háttérvilágítással rendelkező típus, mely a fontos információkat (darab mérete, gép állapota, hibaüzenetek) jeleníti meg a felhasználónak.

A kezelőegységeket egy műanyag dobozban helyeztük el, melybe hátulról tömszelencén keresztül érkezik a két kábel (kommunikáció, és vezérlés).

3.7. A tápellátás

A gép tápellátását egy szabványos ATX-es PC táp biztosítja, melynek teljesítménye 400W. Az elektronikának többféle feszültségszintre is szüksége van:

- Léptető motorok: 12V DC
- LCD kijelző: 5V DC
- PIC, motor kontroller, motor driver logika: 3,3V DC

4. Fájlfeldolgozás és matematikai alapok a vezérléshez

4.1. Az AutoCAD DXF formátum

A DXF formátumot (Drawing Exchange Format) az AutoDesk fejlesztette ki abból a célból, hogy az AutoCAD programot más hasonló jellegű szoftverekkel összekapcsoljon. Először ez a formátum 1982. decemberében jelent meg a az 1.0-s szoftververzióval. A 10-es verzió óta (1988) már nem csak az ASCII formátumú DXF létezik, hanem bináris fájlban is tárolhatók az információk. A verziószámok növekedésével egyre komplexebb objektumok lettel leírhatók a formátumban.

A legtöbb mai CAD szoftver képes DXF formátumot generálni. A teljesség igénye nélkül néhány:

- EPLAN Electrical (erősáramú villamos tervező program)
- EAGLE Layout editor (nyáktervező program)
- ProEngineer (gépész-villamos tervező program)
- Maple (matematikai program)
- Altium (nyáktervező)
- Paint Shop Pro (grafikus program)

4.2. A DXF struktúra

Egy ASCII formátumú fájlt egy egyszerű szövegszerkesztővel is megnyithatunk és az alábbi struktúrát találjuk benne:

- **HEADER szekció** – Általános információk a rajzról (mértékegységek, szögformátum,

technikai keret stb.). Mindegyik paraméternek van egy változó neve és a hozzá tartozó értéke

- **CLASSES szekció** – Információkat tartalmaz az egyes osztályokról: **BLOCK, ENTITI,** és **OBJECT**
- **TABLES szekció** – Definíciókat tartalmaz néhány nevesített egységről

  Alkalmazás ID (APPID) tábla

  Block Recod (BLOCK_RECORD) tábla

  Méretezési stílusok (DIMSTYPE) tábla

  Rétegek (LAYER) tábla

  Vonal típusok (LTYPE) tábla

  Szöveg stílus (STYLE) tábla

  Felhasználói koordináta rendszer (UCS) tábla

  Nézet (VIEW) tábla

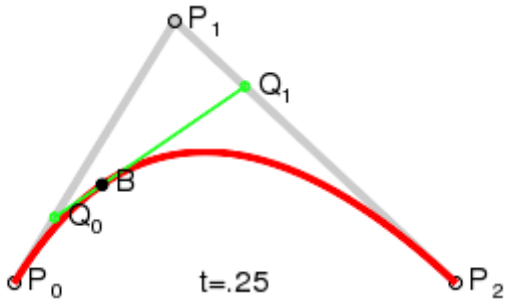
  Nézet konfiguráció (VPORT) tábla

- **BLOCKS section** – Blokk definíciókat tartalmaz a rajzban szereplő entitásokról pl.: sraffozás
- **ENTITIES section** – Ez tartalmazza a rajban szereplő entitásokat (B-spline, kör, egyenes stb.)
- **OBJECTS section** – A nem grafikus objektumok adatait tartalmazza pl.: csoportosított entitások
- **THUMBNAILIMAGE section** – A DXF fájl előnézeti képét tartalmazza
- END OF FILE

4.3. A B-spline-ok és tulajdonságaik

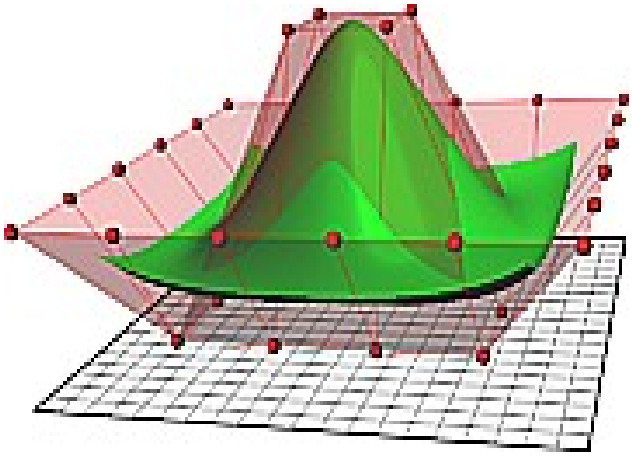
A számítógéppel segített tervezésben (CAD) és a számítógépes grafikában spline-on szakaszosan parametrikus polinomokkal leírt görbét értünk. A spline-okat azért használják előszeretettel ezen a területen, mert egyszerű és interaktív szerkesztést tesznek lehetővé, pontosságuk, stabilitásuk és

könnyű illeszthetőségük révén igen komplex formákat lehet velük jól közelíteni, vagyis egy nagyon jó tömörítési lehetőség. A spline angol szó, és nevét arról a rugalmasan hajlítható vonalzóról kapta, melyet hajóépítők és rajzolók használtak korábban.



21. ábra

A B-spline (Bezier-spline) görbék a spline-ok speciális esetei. Míg a spline-ok interpolálják a kontrollpontokat, addig a B-spline-ok csak approximálják azokat. Ezt a görbetípust először Paul de Castiljau fedezte fel 1959-ben és később továbbfejlesztve főként az autóiparban alkalmazták, leginkább karosszéria elemek felületeinek leírására. A görbe egy nagyon fontos tulajdonsága, hogy C^2 folytonos, azaz a görbe szakaszainak második deriváltja egy pontban találkozik.



22. ábra

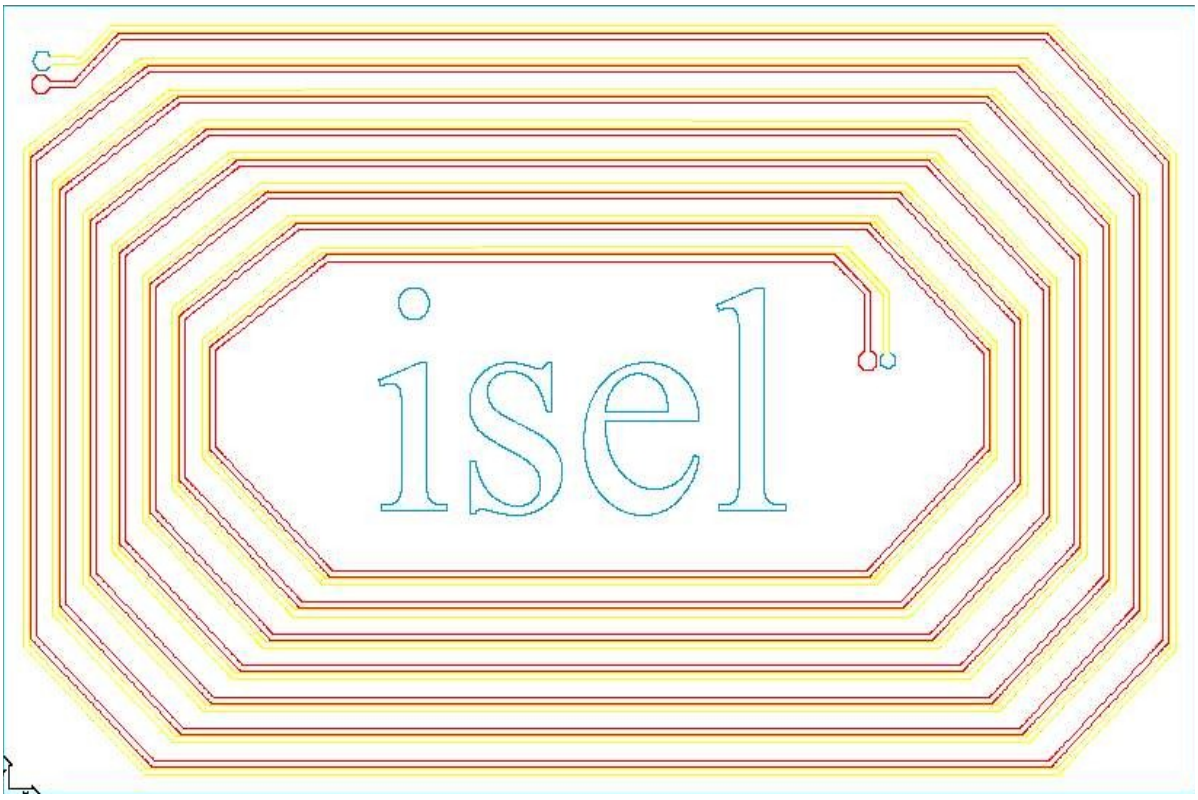
Egy másik speciális esete a Bezier-görbéknek a NURBS (Non-Uniform Rational B-Spline) görbék és felületek. A bonyolult felületek és görbék leírásának ez a ma legelfogadottabb és leghasználtabb alakja. A legtöbb mai program ezt az approximációt alkalmazza.

A görbék legelőnyösebb tulajdonsága, hogy létezik olyan numerikus módszer, amellyel a görbe egyes pontjai egymást követően kiszámíthatóak, viszonylag gyorsan és kevés műveletet igényelve. Mi számításaink során a De Boor algoritmust használtuk, mert ez tűnt a legstabilabb és leggyorsabb

iterációnak.

4.4. Az iteráció

A főként nyáktervező programok és a grafikus programok, melyek a használat során nem megfelelő paraméterekkel megadott egyszerű vektorgrafikus objektumokat készítenek (kör, egyenes, négyszög, stb.) a konverzió során az egyszerűsítés miatt numerikusan könnyen számítható B-splineokkal írják le az egyes objektumokat (betűk, egyenesek, körök stb.). A feladatkiírás során mi egy olyan DXF fájlt kaptunk, ami a fentiek miatt csak B-spline-okat tartalmazott.



23. ábra

A fenti kép S-betűjét is és az egyeneseket is spline-ok írják le a fájlban:

SPLINE	8	-2.325429999999999
5	73	20
2D	4	0.47539
330	74	30
1F	0	0.0
100	42	10
AcDbEntity	0.0000000001	-2.325429999999999
8	43	20
layer 1	0.0000000001	0.47539
62	40	30
142	0.0	0.0
370	40	10
5	0.0	-2.325429999999999
100	40	20
AcDbSpline	0.0	-0.9913999999999999
210	40	30
0.0	0.0	0.0
220	40	10
0.0	1.0	-2.325429999999999
230	40	20
1.0	1.0	-0.9913999999999999
70	40	30
24	1.0	0.0
71	40	0
3	1.0	
72	10	

A fenti paraméterekkel először a görbe fontos tulajdonságait határozza meg (fokszám, kontrollpontok száma, a görbe síkjának normálisa stb.), majd következnek a görbét leíró adatok.

- **Knot vektor – 40-es kód** → a knotok egy olyan szekvenciát alkotnak, melyek meghatározzák, hogy melyik kontrollpontot mennyire közelíti meg a görbe. Számuk mindig

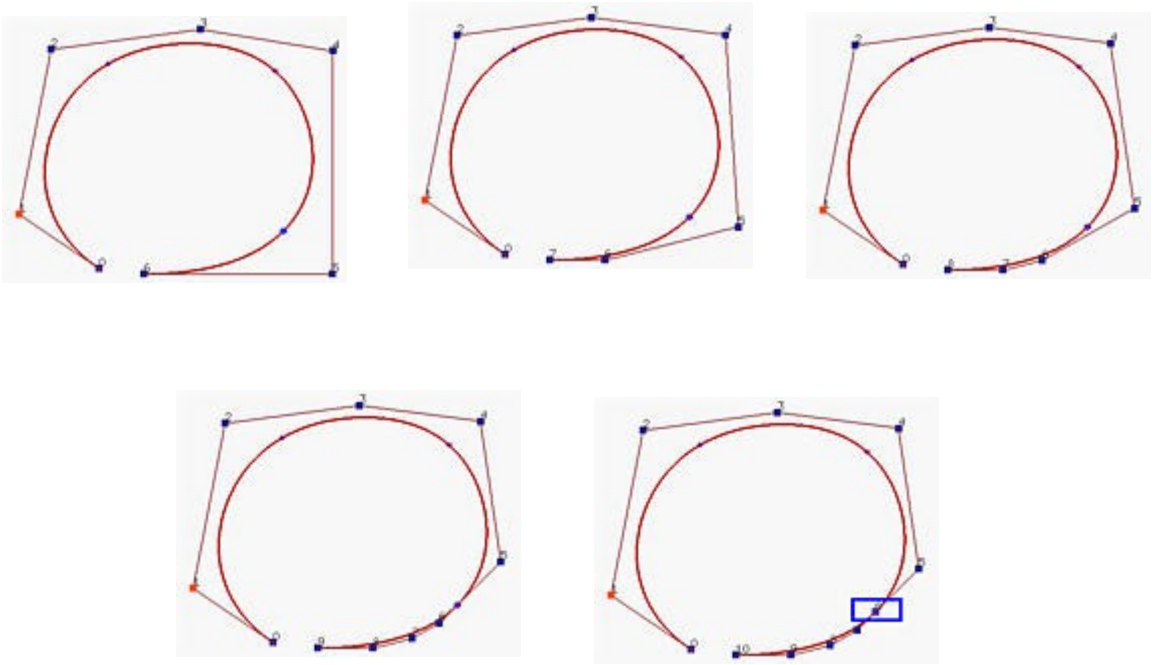
$$Knotszám = Kontrollpontok\ száma + görbe\ fokszáma + 1$$

Ha a görbe átmegy a kontrollponton, akkor a hozzátartozó knot fokszámszor szerepel a szekvenciában. A knotok távolsága a nagyobb görbületű pontokban kisebb, itt több görbe pontot számít az algoritmus.

- **Kontroll pontok – 10,20,30-as kód** → ezek azok a pontok, melyeket a görbe approximál.

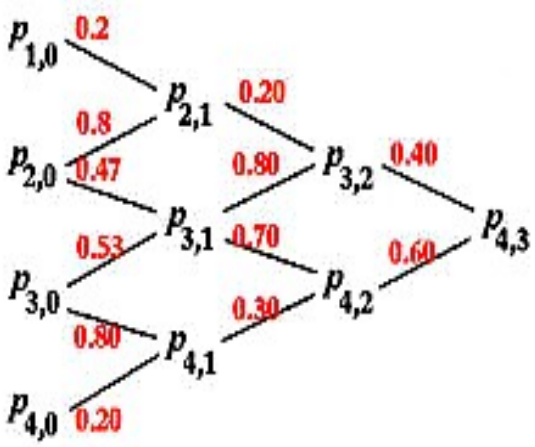
Hogy a kontrollpontokból és a knot vektorból ki tudjuk számítani a görbe egyes pontjait, a De Boor algoritmust használtuk, mely egy háromszög séma alapján számítja az egymást követő pontokat.

Az algoritmus lényege: ha egy tetszőleges knot-ot, fokszámszor illesztünk be, akkor az utoljára keletkező pont a knothoz tartózó görbére illeszkedő pont lesz. Ez látható a következő ábrán.

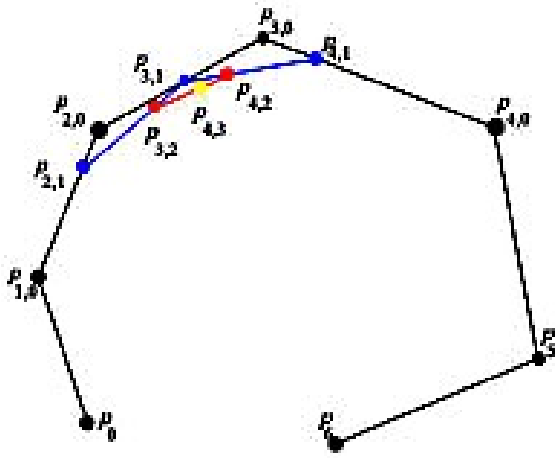


24. ábra

A háromszög séma segítségével egy igen gyors „néhány” egyszerű műveletből álló rekurzív algoritmust kapunk:



25. ábra



26. ábra

Egy harmadfokú polinomiális B-spline egy pontjának kiszámításához összesen négy kontrollpontra és hat knotra van szükség. A séma alapján a kontrollpontokból a knotokból kiszámított konstansok segítségével kapunk új kontrollpontokat. Míg végül el nem jutunk a keresett ponthoz.

A kiszámításhoz az alábbi rutint ajánlja a szakirodalom:

Input: a value u
Output: the point on the curve, p(u)

If u lies in [u[k],u[k+1]) **and** u != uk, **let** h = p (i.e., inserting u p times) **and** s = 0; **(I.)**
If u = u[k] **and** u[k] **is** a knot of multiplicity s, **let** h = p - s (i.e., inserting u p - s time); **(II.)**

Copy the affected control points pk-s, pk-s-1, pk-s-2, ..., pk-p+1 and pk-p to a new array and rename them as pk-s,0, pk-s-1,0, pk-s-2,0, ..., pk-p+1,0;

for r := 1 **to** h **do** **I. ciklus**

for i := k-p+r **to** k-s **do** **II. ciklus**

begin
Let a[i,r] = (u – u[i]) / (u[i+p-r+1] – u[i])
Let p[i,r] = (1 – a[i,r])*p[i-1,r-1] + a[i,r]*p[i,r-1]
end

pk-s,p-s is the point p(u).

A kiszámítások meggyorsítása érdekében a fenti kódon némi változtatást eszközöltünk. Mivel számunkra nem szükséges olyan helyeken pontosan kiszámítani a pontokat, ahol knot érték megegyezik a kiszámítandóval ezért az a II. feltétel a rutin elejéről töröltük, megspórolva így egy döntési és számítási ciklust. Helyette azt a feltételt szabtuk, hogy ha egy ilyen pontban kellene kiszámítanunk a görbe pontját, akkor számítsuk ki inkább egy hozzá nagyon közel álló értéket.

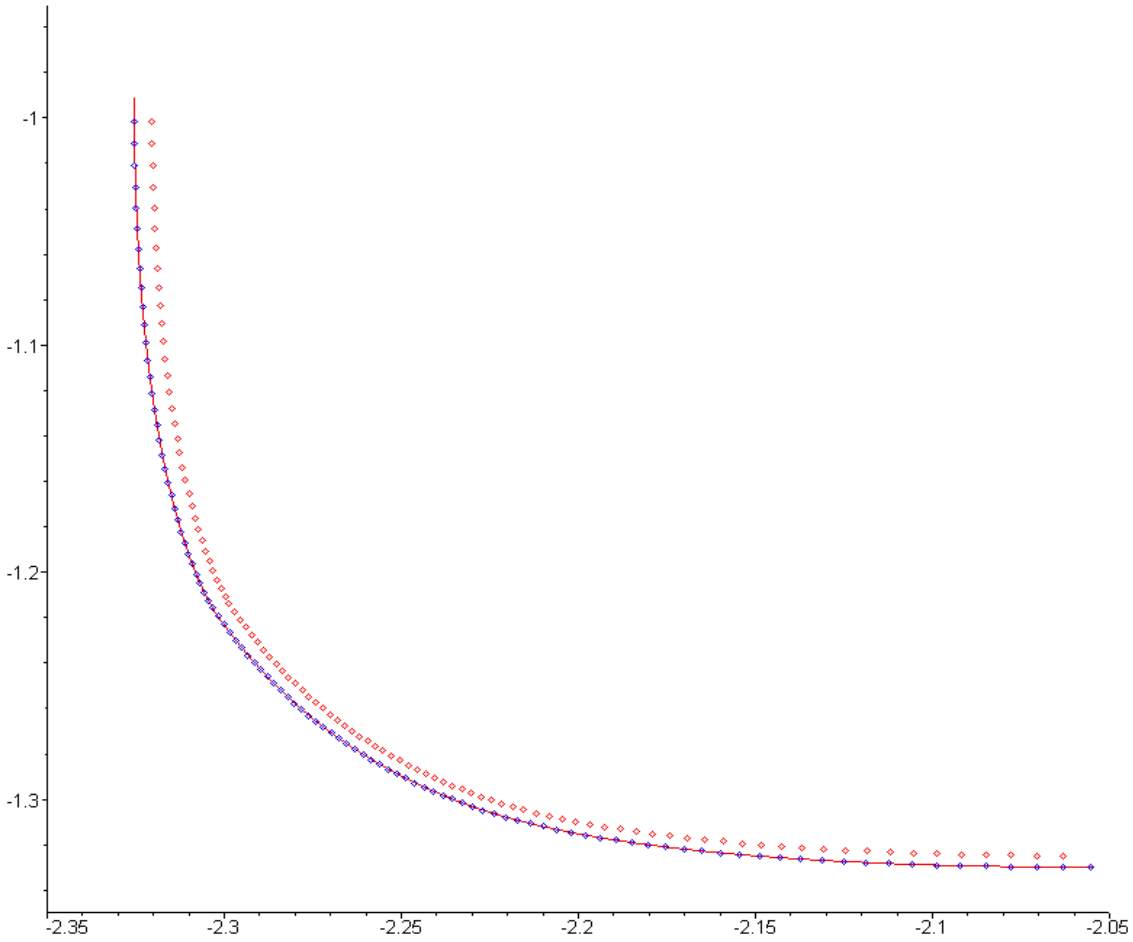
A mi esetünkben csak harmadfokú B-spline-okat kellet feldolgozni. Ekkor az első ciklus háromszor fut le a második pedig először háromszor, majd kétszer és végül egyszer. Így az értékadási, számítási műveletek összesen hatszor hajtódnak végre. Ez a számítások alapján a járulékos számítási és ciklusutasítások miatt, ha lebegőpontos számokat használunk, akkor kb. 500 mikroszekundumot vesz igénybe. Ha viszont fixpontos számokkal számolunk akkor ez az idő 100 mikroszekundumra rövidül ezt foglalja össze a táblázat.

Művelet	**Darabszám**	**Instrukció/db – float**	**Instrukció/db – integer**	**Össz – float**	**Össz. - integer**
Összeadás	6*2	122cyc	1cyc	1464	12
Kivonás	6*3	124cyc	1cyc	2232	18
Szorzás	6*2	109cyc	1cyc	1308	12
Osztás	6*1	361cyc	1cyc	2166	6
ÖSSZESEN				**7170cyc**	**48cyc**
80MHz-en				**3,59E-04sec**	**2,40E-06sec**

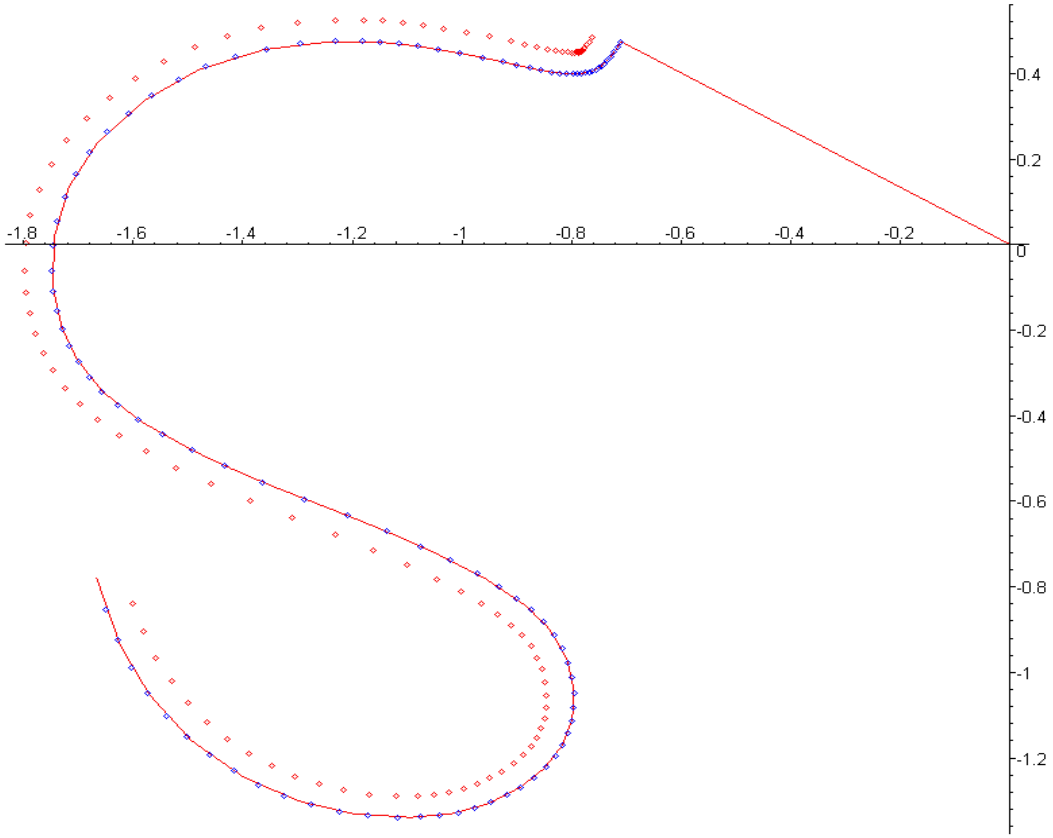
Megjegyzés: a 80MHz-es órajel mellet a PIC 20MIPS-es (Mega Instruction Per Secundum) végrehajtási teljesítményre

képes.

Mielőtt a fentieket beprogramoztuk a kontrollerbe csináltunk egy számítási szimulációt a MAPLE program segítségével ez látható a következő két ábrán. A szimulációt pedig a mellékletben csatoltuk. Az első ábra az „I” betű egyik alsó szárának részlete, míg a másik az „S” betű részlete az ISEL feliratból. A folytonos vonal a MAPLE által illesztett görbe, míg a kék a számított pontok a görbén. A piros pontoknál pedig a szerszámkorrekciót is figyelembe vettük. A pontokat a gép lineárisan interpolált szakaszokkal köti össze a mikrokontroller szoftverének ismertetésénél taglalt módon.



27. ábra

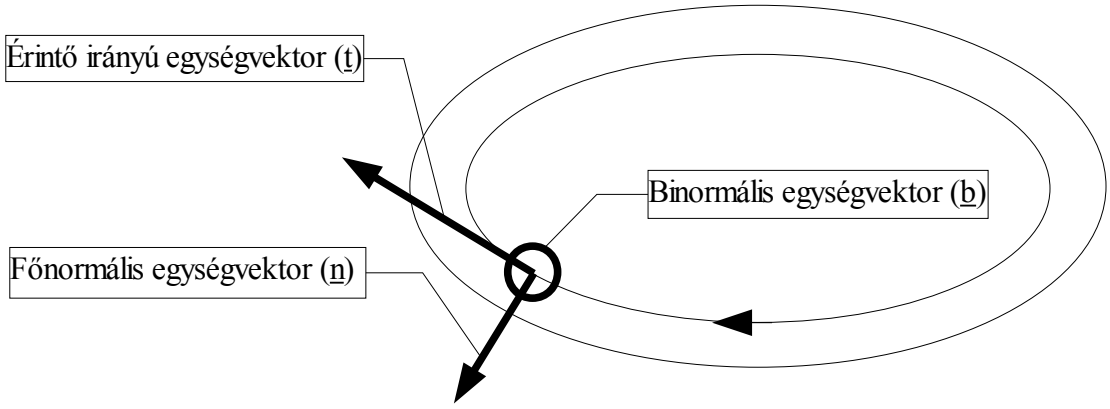


28. ábra

Az ábrán látszik még az interpoláció egyik fontos tulajdonsága: minél nagyobb a görbület, annál nagyobb a számított pontok sűrűsége. Ez a pályavezérlést gyorsítja az egyenes szakaszokon, illetve pontosítja a görbe szakaszokon.

4.5. Szerszám korrekció

Ha a szerszámmal a fentiek alapján számított pályán haladnánk végig, akkor nem ugyanazt a méretű objektumot kapnánk, mint ami a rajzon van. Ennek kiküszöbölésére egy korrekciós pályán kell végigvezetnünk a szerszámot, melyet az főnormális egységvektor felhasználásával állítunk elő.



29. ábra

Először a görbe számított egymást követő pontjaiból differencia számítással előállítjuk az érintő irányú vektort, mely nagyon jól közelíti a valóságos érintő irányt, mivel két pont közötti távolság általában nem haladja meg az 0,01 mm-t és ez a görbület növekedésével egyre kisebb lesz. A vektor normálása a Pitagorasz-tétel szerint történik. Ezután vektoriális szorzatot képezünk a kísérő triéder harmadik tagjával a binormális vektorral, ami egy egyszerű előjelcsere, tekintve, hogy mindkét vektor merőleges egymásra és a binormális vektor csak „z" irányú összetevőt tartalmaz. A binormális vektor iránya hordozza azt az információt, hogy melyik irányból kell körüljárni az objektumot („+" vagy „-" érték), nagysága pedig a szerszám rádiuszát határozza meg.

$$\vec{t} = \frac{\overrightarrow{dxdy}}{\left|\overrightarrow{dxdy}\right|}, \quad ahol \left|\overrightarrow{dxdy}\right| = \sqrt{dx^2 + dy^2}$$

$$\vec{n} = \vec{t} \times \vec{b} = \begin{vmatrix} i & -j & k \\ tx & ty & 0 \\ 0 & 0 & bz \end{vmatrix}$$

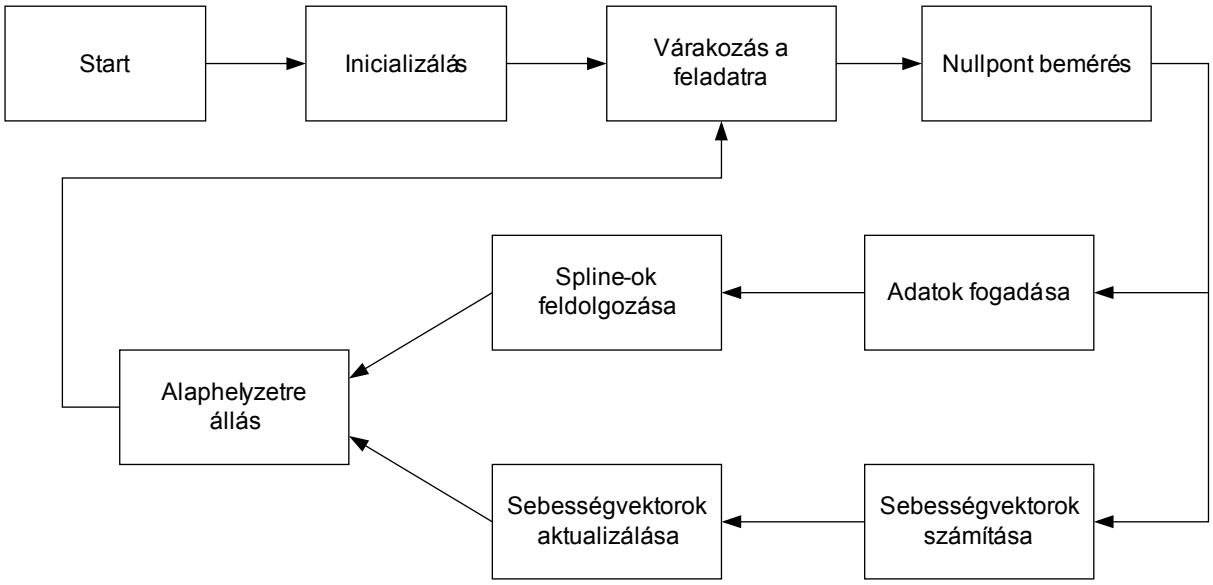
elvégezve

$$\vec{n} = (ty \cdot bz)\,\vec{i} - (tx \cdot bz) \cdot \vec{j} = bz \cdot (ty \cdot \vec{i} - tx \cdot \vec{j})$$

Tehát nem kell teljesen elvégeznünk a vektoriális szorzást, hanem elegendő a két koordinátát megcserélni és az egyiket megszorozni -1-el, valamint az egész vektort felszorozni a szerszám sugárral.

5. A mikrokontroller programja

A mikrokontroller programját az ingyenesen használható MPLAP C30-as c-fordítóval készítettük. Az alábbi ábra kontrollerben futó szoftver blokkdiagrammját mutatja.



30. ábra

5.1. Inicializálás

A gép bekapcsolása után, amikor a PLL hurok is „behúzott” a PICinit rutin inicializálja az egyes perifériákat úgy, mint:

- Analóg bemeneti modul kikapcsolása, mivel ez ébredés után aktív
- IO lábak beállítása, felhúzó ellenállások bekapcsolása
- SPI kommunikáció beállítása a léptetőmotor kontrollernek
- Driver-tábla feltöltése (motorok inicializálási paraméterei, mikrolépés-tábla)
- Globális paraméter regiszter feltöltése:
  - multiplexelt referencia kapcsolók
  - folyamatos kommunikáció a meghajtókkal
  - 3db motor
  - meghajtók soros kommunikációs órajele
- Motorok egyéni inicializálása:
  - áramértékek beállítása (állóhelyzetben, gyorsításkor, állandó sebességnél)
  - mikrolépés

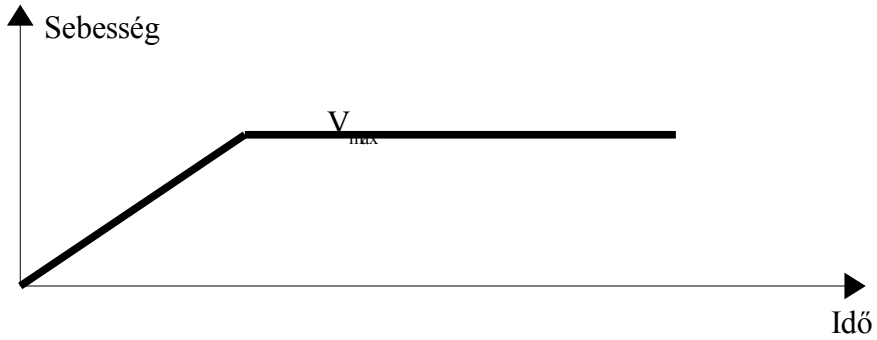
- VELOCITY MODE beállítások:
  - minimális, maximális sebesség
  - maximális gyorsulás
  - megállítás referencia kapcsolóra
- Koordináta buffer inicializálása
- UART kommunikáció konfigurálása (115 200 Baud, 8 bit, nincs paritás, 1 stop bit, megszakítás engedélyezése)
- Pályavezérlés megszakítás beállítása

Az inicializálás után a gép áll és várakozik a következő feladatra. Innentől a gép csak a hálózaton keresztül beérkező parancsokra reagál.

5.2. Nullpontbemérés

Amikor a hálózaton keresztül a gép parancsot kap egy új munka megkezdésére (#M), akkor elindítja a kézi nullpontbemérést. Alapesetben a botkormány csak az X és Y tengelyeket mozgatja. Ha megnyomjuk a fent említett váltó gombot, akkor van lehetőség a Z tengely mozgatására.

A tengelyek mozgatása gyorsjáratban történik maximális sebességgel. Így azonban a finom mozgatásra nem lenne lehetőség. Hogy mindkét igény kielégíthessük a botkormány megmozdításakor a tengelyek sebesség egy lineáris rámpán fut fel. A joystick rövid idejű megnyomásával nagyon finom pozicionálások végezhetők, míg hosszan nyomva tartva a gép felgyorsul a maximális sebességre, hogy a célt mielőbb elérje.



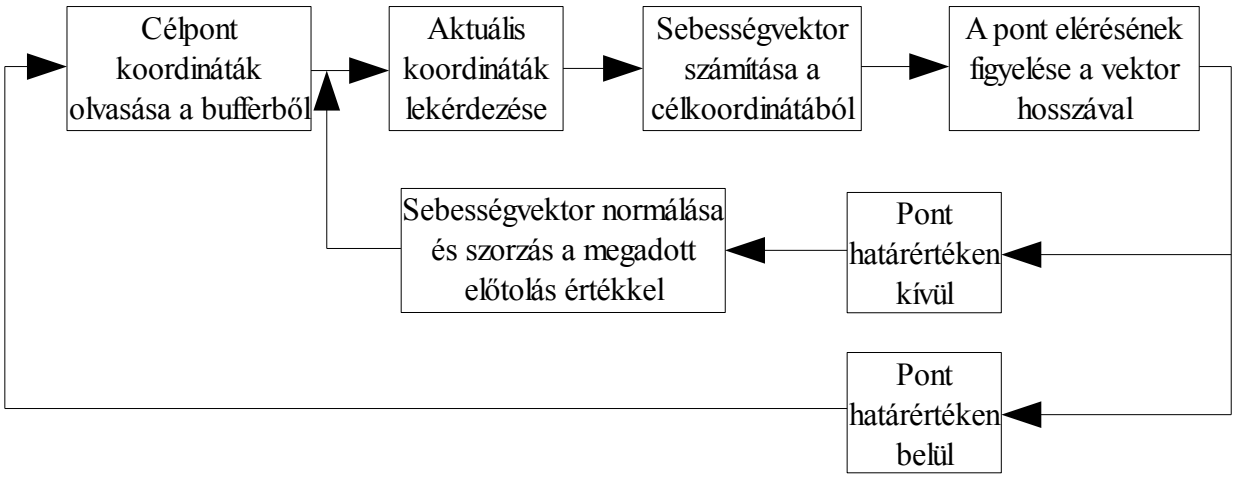
31. ábra

A nullpontbemérés a jóváhagyás gombbal fejeződik be. Ekkor a kontrollerben is kinullázzuk az aktuális koordinátákat.

A nullpontbemérés után a szoftver két párhuzamos ágon folytatódik, ezek közül az egyik az adatok fogadása és feldolgozása a fent ismertetett algoritmussal, míg a másik a pályavezérlés, melynek feladata a kiszámított koordinátákon való végighaladás lineáris interpoláció segítségével.

5.3. Pályavezérlés

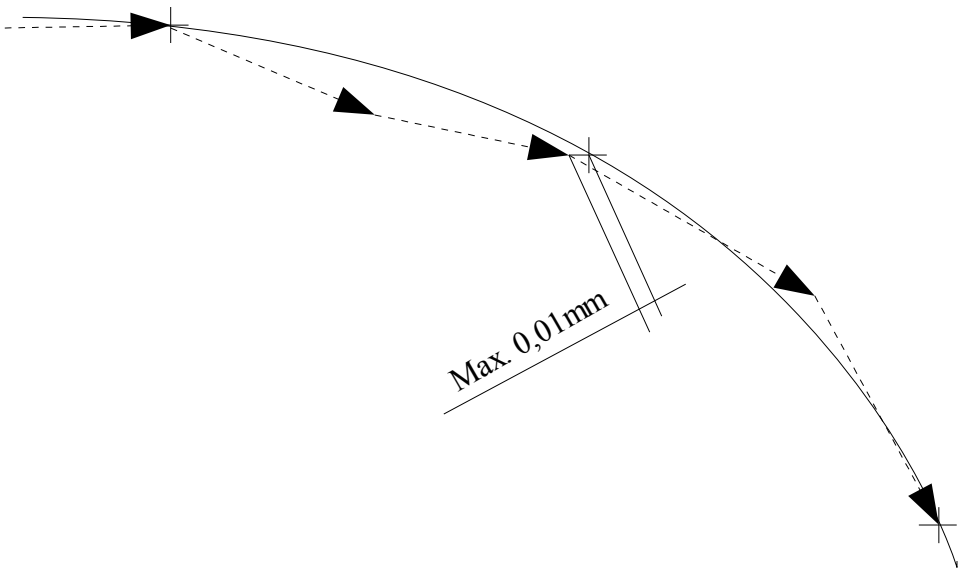
A pályavezérlés blokkvázlata látható a következő ábrán.



32. ábra

A pályavezérlés rutin megszakításos üzemben fut, mégpedig 1 milliszekundumonként egyszer fut le. Ekkor a bufferből kivesz egy célkoordinátát, majd összehasonlítja az aktuális koordinátával, amit a motorvezérlőből olvas ki. Ennek eredményeként képződik egy vektor, amelyet normálunk, majd megszorozzuk az előtolás értékével és ezt az új sebességvektort állítjuk be a motorokon.

Fontos része a rutinnak célpont koordináta és az aktuális koordináta összehasonlítása, melyet a vektor hosszának figyelésével oldottunk meg, amit a normáláshoz is felhasználunk. Amikor a számított vektorhossz kisebb, mint egy előre definiált érték (0,01mm) akkor a program lekéri a következő célpontkoordinátát és annak irányába kormányozza a gépet. Ez ugyan bevisz némi pontatlanságot a rendszerbe, de ez a feldolgozási sebesség növelésével tovább kicsinyíthető. Jelen alkalmazásban a fenti 0,01mm bőségesen kielégíti a követelményeket.



33. ábra

Ennek az úgynevezett sebesség-szabályzásnak nagyon nagy előnye, hogy a sebesség vektorok mindig a a célkoordináták irányába állnak. Ha a gép az egyes tengelyek tehetetlensége vagy a kvantált sebesség értékek miatt letér a megadott pályáról, akkor a szabályzás képes korrigálni azt. Szintén hasznos azért, mert az fent említett minimális megközelítésből keletkező hibák nem adódnak össze. Ezt a fajta szabályzást a szinkronmotorok szabályzásánál is elterjedten alkalmazzák.

A számítás fixpontos számokkal történik, hogy minél kevesebb időt vegyen igénybe. Ahhoz hogy a vektorok hosszát fixpontos számokkal is ki tudjuk számítani az úgynevezett CRENSHAW numerikus algoritmust vettük igénybe. Egy számítás körülbelül 300 mikroszekundum alatt fut le a járulékos utasításokkal együtt.

▪ CRENSHAW gyökvonás

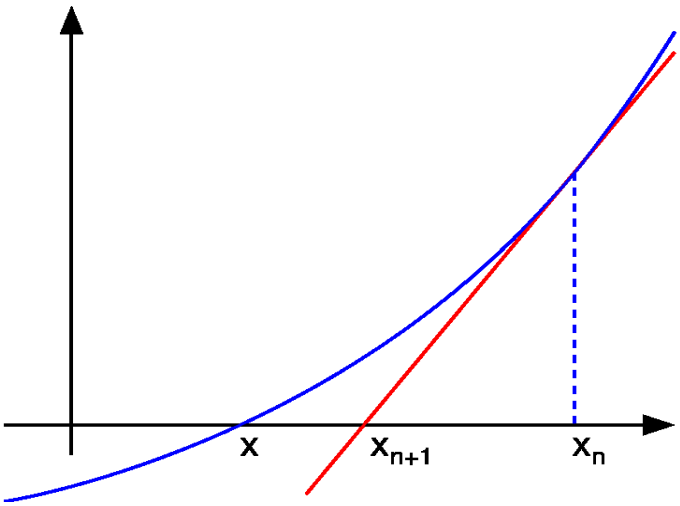
Általában numerikus gyökvonásra a Newton-iterációt alkalmazzák a programozók, még a zsebszámológépekben is. A képlete a következő:

$$x = \frac{1}{2}\left(x_0 + \frac{a}{x_0} \right)$$

A Newton-iteráció lényege, hogy egy ismeretlen folytonos függvény gyökét úgy keressük meg, hogy egy tetszőleges pontjába húzott érintőjének (első fokú polinom) gyökét meghatározzuk, majd a meghatározott pontban ismét érintőt húzunk a függvényhez, és így tovább. Az iteráció nagyon

gyorsan (exponenciálisan) konvergál a végső megoldáshoz.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



34. ábra

Ennek az iterációnak két hibája is van:

- viszonylag sok műveletet igényel
- integer aritmetikánál nem konvergál (pl.: a=15 esetén 3 és 4 között oszcillál)

A fentiek miatt Jack W. Crenshaw egy teljesen új algoritmust dolgozott ki, mely az integer aritmetikában is remekül alkalmazható.

```
unsigned short sqrt(unsigned long a){
  unsigned long rem = 0;
  unsigned long root = 0;
  for(int i=0; i<16; i++){
    root <<= 1;
    rem = ((rem << 2) + (a >> 30));
    a <<= 2;
    root ++;
    if(root <= rem){
      rem -= root;
      root++;
    }
    else
      root--;
  }
  return (unsigned short)(root >> 1);
}
```

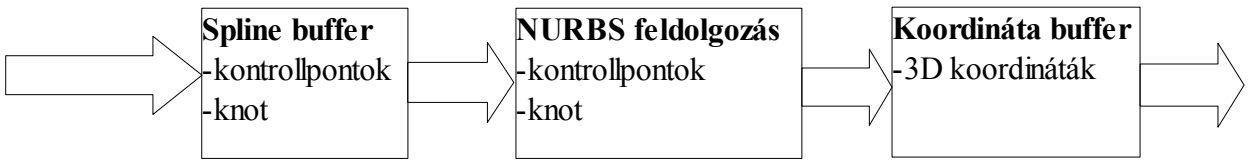
Látható hogy az algoritmus nem tartalmaz semmilyen különleges műveletet és a ciklus is csak

16-szor fut le. Az alapja egy nagyon régi mechanikus gyökvonási technika, melyet először a NASA-nál alkalmaztak.

5.4. Adatok fogadása és spline feldolgozás

A fenti megszakításos rutinnal párhuzamosan normál programban fut az adatfeldolgozás, tehát a PIC szabadidejében kiszámít néhány pontot – amit egy bufferben tárol – majd az a megszakításban beolvassa és a fentiek alapján feldolgozza. Az adatok fogadása és a parancsok feldolgozása a soros porti vétel kivételével nem megszakításos programban fut.

A beérkező adatok a soros portról egy bufferbe kerülnek, ahonnét a NURBS rutin meghívásával 4 kontrollpont és 7 knot érték kerül feldolgozásra az előző fejezetben leírtak alapján. Mivel a spline egy nagyon jól tömörített görbe, ezért a fogadott néhány bájtból akár több száz pontot is ki tudunk számolni. A számítás ideje alatt pedig bőven van idő újabbak fogadására. Az kiszámított célkoordináták szintén egy tömbbe kerülnek eltárolásra, ahonnét a pályavezérlés feldolgozza majd őket.



35. ábra

A fenti ábra mutatja az adatok áramlását a modulban. Amikor a soros port vételi regiszterbe legalább két bájt kerül, akkor kiolvassuk azt és megnézzük, hogy milyen parancs érkezett:

- **#S** → Szöveg kiíratása az LCD-re
- **#B** → B-spline kezdő adatok (fokszám, knotok száma, kontrollpontok száma)
- **#D** → B-spline adatok (kontrollpontok, knot)
- **#U** → Knot-értékek osztásának beállítása (felbontás beállítása)
- **#G** → Marás lineáris interpolációval
- **#F** → Gyorsjárat
- **#Z** → Csak Z irányú mozgás
- **#M** → Nullpont bemérés indítása

Mindig a parancsnak megfelelő rutin fut le, mely elvégzi a soron következő adatok beolvasását. Például egy b-spline érkezése esetén a kommunikáció egy **#B** paranccsal indul, melyben beolvassuk a b-spline alapparamétereit, majd ezt tárolva tudjuk, ellenőrizni a fogadott adatok darabszámából a kommunikáció helyességét. A **#B** parancsot minden esetben a **#D** parancs követi, melyben egy kontrollpont pár és egy knot érték kerül fogadásra. A kommunikációs hibák elkerülése miatt a PC oldalon egy ***O** paranccsal jelzi a gép minden adattömb megérkezését. Az itt beérkezett adatok egy bufferbe kerülnek.

- **Vételi buffer kezelés**

A vételi buffer egy egyszerű FIFO (First In First Out) egység, amely az adatokat a beérkezésük sorrendjében küldi tovább a NURBS algoritmusnak.

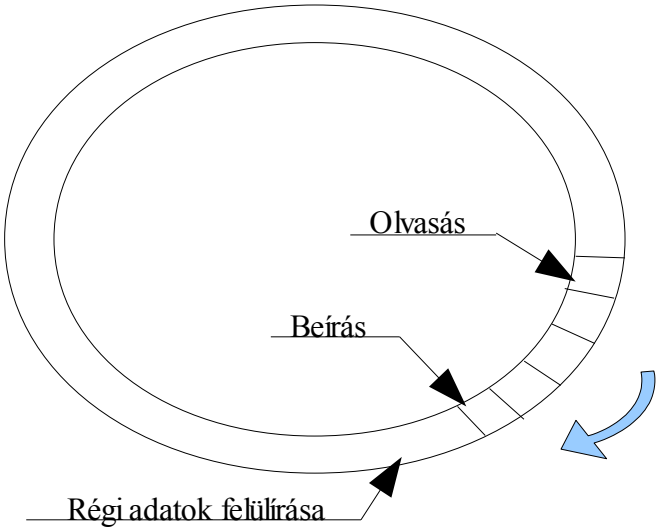
- **NURBS algoritmus**

A NURBS algoritmus a fent részletezett módon fut le és a 4 kontrollpontból és a 6 knot értékből előállítja a pontokat. A knot értékeket az intervallumban egy előre definiált értékkel növelve, egymás után következő két dimenziós pontokat kapunk amit egy bufferben tárolunk el. Arra azonban vigyázni kell hogy a buffer ne csorduljon túl, ekkor az iterációt meg kell állítani.

- **Koordináta buffer kezelés**

Egy buffert hoztunk létre a koordinátáknak a zökkenőmentes tárolásához, melybe a NURBS rutin végzi a beírást és a pályavezérlés rutin végzi az olvasást. Ennek kezelését úgynevezett gyűrű buffer üzemmódban végezzük.

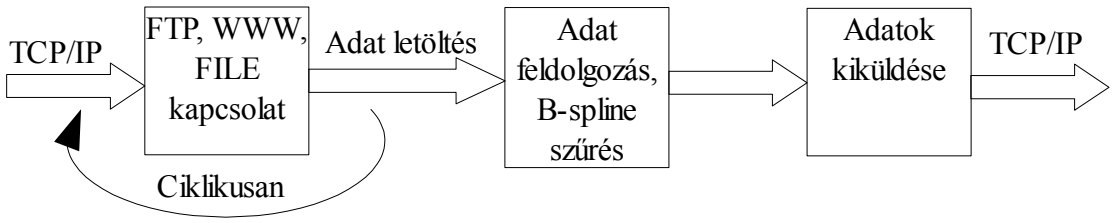
36. ábra



A gyűrű bufferben a beírás és a kiolvasás folyamatosan halad előre. Amikor a buffer végére ér a beíró rutin, akkor automatikusan az elejére ugrik és ott folytatja tovább. Ugyanez a helyzet az olvasással is. Arra azonban ügyelni kell, hogy mindig az írás haladjon elöl, hiszen ellenkező esetben az adatok összekeverednek. Az írás során a már kiolvasott adatok felülíródnak.

6. A számítógépes szoftver

A számítógépes szoftver objektumorientáltan Visual C++ nyelven íródott. A program 3 fő lépésből áll. Első a kijelölt FTP, vagy WEB szerver folyamatos figyelése és az ott megjelenő DXF fájl letöltése. A letöltést követi a fájlfeldolgozás, majd az adatok áttöltése a gravírozó gépre. Ezt a folyamatot mutatja az ábra.



37. ábra

Az internethez való hozzáférést a .NET keretrendszer támogatja, az URI (uniform resource identifier) osztályon keresztül valósítottuk meg a kapcsolatot, amellyel nemcsak WEB hozzáférés, hanem akár a saját számítógépen tárolt fájlokhoz is rugalmasan hozzáférhetünk. Ez utóbbira inkább a tesztfázisban volt szükség. Amikor a Start polling gombot megnyomjuk, akkor a megjelölt kapcsolatot a gép megnyitja (legyen az helyi vagy hálózati) és beállítható időközönként

automatikusan figyeli, hogy érkezett-e használható információ, ehhez a programba egy időzítőt építettünk be. Természetesen azt is vizsgáljuk, hogy a megadott cím valós-e. Ha nem akkor azt hibaüzenet jelzi a felhasználónak.

A WebRequest osztály segítségével nyitjuk meg a megadott címet, majd az ott lévő adatot szövegfolyamként (string stream) olvassuk be a memóriába. A beolvasott adatokat a www.ribbonsoft.com oldalról díjmentesen letölthető és felhasználható DXF olvasó könyvtár segítségével elemezzük, mely kielemzi és rendezi a fájlban található objektumokat. Az B-spline objektumokat a programunkban egy tömbben tároljuk, majd az adtokat a mértékegységnek megfelelően úgy nagyítjuk, hogy a gravírozóra az adatok már [mm] -ben kerüljenek át, továbbá egy eltolást is kell alkalmaznunk, hogy a nullpontbemérésnél a nullpont mindig sarokpontra essen.

A transzormált adatokat a binWriter osztály segítségével küldjük át a gépnek, amely az adatok vételekor egy *O üzenettel válaszol minden vett parancsra. A következő parancs nem kerül kiküldésre egészen addig, míg a nyugtázó üzenet meg nem érkezett. Amikor egy B-spline-hoz tartozó adatcsoag végére érünk, akkor a gravírozó fejet fel kell emelni és a következő kezdőpont pozícióba mozgatni. Ez a parancs is itt kerül kiadásra.

Ahhoz, hogy az összetartozó splájnok között a gravírozófej ne emelkedjen fel, az egyást követő görbék kezdő és végpontjának távolságát számoljuk és amikor ez egy előre beállított tűrést meghalad, csak akkor küldünk kiemelés parancsot a gépnek.

7. Továbbfejlesztési lehetőségek és célok

A gép diplomamunka keretében továbbfejlesztésre kerül. Elsődleges cél, hogy a gép képes legyen három dimenziós görbéket is kezelni. Ez a gép oldalon nem igényel sok munkát, hiszen a De-Boor algoritmus három dimenzióban is működik, ugyanez igaz a pályavezérlésre is.

A PC-s szoftverben viszont több módosítást is be kell iktatni, ha a távlati terveknek megfelelően három dimenziós pontfelhőből szeretnénk modellt létrehozni. Ehhez egy fordított eljárást kell kidolgozni, mely a pontfelhőből készít síkmetszeteket spline darabokból. Ami könnyít a szoftver módosításán, hogy a PC oldalon nem lényeges a műveleti igényt meghatározni. Szintén itt kell megoldani, ha a DXF fájl feldolgozását ki szeretnénk terjeszteni a többi egyszerű objektumra.

A dokumentáció teljes anyag, valamint a hozzá kapcsolódó összes adatlap és rajz megtalálható a mellékletben.

8. Irodalomjegyzék

1. Isel által forgalmazott gépek leírása (2008.11.10.)
   http://isel.hu/?page=termekek&op=csoportok&catid=2

2. Hiwin cég hajtásláncainak leírása (2008.11.10.)
   http://www.hiwin.hu/index.php?scriptlet=&id=4&language=hu

3. Egyedi tervezésű gépek leírása
   http://hobbycnc.hu/Magyar.html

4. A DXF formátum –
   http://www.autodesk.com/techpubs/autocad/acad2000/dxf/dxf_format.htm

5. Bezier görbék – http://en.wikipedia.org/wiki/Bézier_curve

6. Non Uniform Rational B-Spline – http://en.wikipedia.org/wiki/NURBS

7. De Boor algoritmus – http://en.wikipedia.org/wiki/De_Boor's_algorithm

8. Crenshaw algoritmus – http://www.embedded.com/98/9802fe2.htm

9. A CNC interpolation algorithm for boundary machining Sotiris L. Omirou Department of Mechanical Engineering, Technological Educational Institute of Patras, Patras 26500, Greece

10. Fast real-time NURBS path interpolation for CNC machine tools W.T. Lei, M.P. Sung, L.Y. Lin, J.J. Huang Department of Power Mechanical Engineering, National Tsing Hua University, Hsinchu 300, Taiwan, ROC