

Óbudai Egyetem
Bánki Donát Gépész és Biztonságtechnikai Mérnöki Kar
Mechatronikai és Autótechnikai Intézet



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

IRÁNYÍTÓ EGYSÉG FEJLESZTÉSE SCARA MANIPULÁTORHOZ

Szerző: **Zólyomi Attila**
Mechatronika szak, 4. évfolyam

Konzulens: **Dr. Nagy István**
adjunktus

Budapest, 2011

Tartalomjegyzék:

1.	A SERPENT 1	
	A mechanika	4
	A vezérlés	4
2.	A választott egységek bemutatása	
	A mikrovezérlő	6
	A kommunikációs modul	8
	A motorok vezérlése.....	9
	Hőmérsékletmérés.....	10
3.	Elektromos biztonság	
	Valós áramköri veszély	11
	Funkcionális áramköri veszély	13
4.	Villamos tervezés	
	Az USB átalakító panel	14
	A hőmérő panelek	14
	A mikrovezérlő panelja	15
	A motorvezérlő panel	17
5.	Szoftveres megvalósítás	
	A mikrokontroller programja.....	21
	Analóg és digitális világ	23
	Integrált funkciók.....	25
	A PC-re írt program	26
	A kezelőfelület kialakítása.....	27
6.	Összefoglalás.....	28
7.	Irodalomjegyzék.....	29
8.	Rezümé	31
9.	Mellékletek	32

Bevezetés

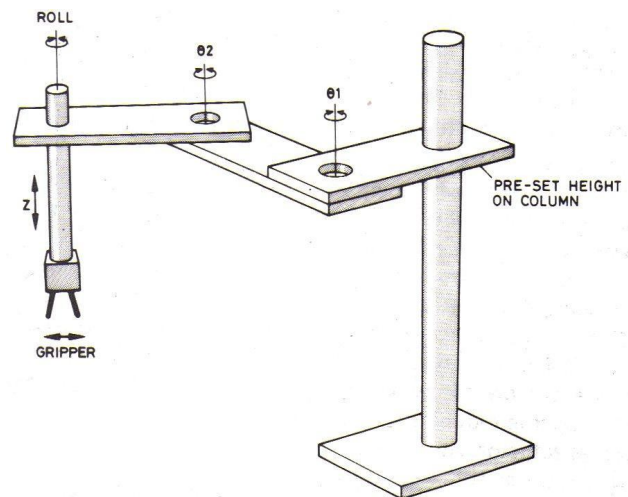
A mechatronika három, egyenként is komoly műszaki terület egymást erősítő integrációjából jött létre. Ezek a gépészet, az elektronika, és az informatika. Mindhárom terület elengedhetetlenül fontos ahhoz, hogy egy összetett mechatronikai rendszer működését megtervezzük. Jelen esetben a gépészeti rész adott. A mechanikai konstrukció kellő mélységű elemzésével lehetőség nyílik arra, hogy az elektronikát teljes mértékben a meglévő konstrukcióhoz illesszük. A végső cél egy olyan, számítógéppel kommunikáló – ám attól függetlenül is működőképes – egység létrehozása, amelynek segítségével a robot a számítógépen futó programon keresztül rugalmasan kezelhető, valamint a vezérlő egységhez további mérő / adatgyűjtő modulok illeszthetők, melyeket a megfelelő programozás után teljes funkciójukban képes kezelni. Ennek értelmében tehát a vezérlő egységnek egy olyan beágyazott rendszernek (*embedded system*) kell lennie, melyhez rugalmasan illeszthetők egyéb perifériák a hardver módosítása nélkül. Továbbá képesnek kell lennie arra, hogy egy már meglévő *API*-n keresztül a program teljesen alkalmazáshoz szabható legyen.

Terjedelmi okok miatt eltekintek a mikrovezérlő programjának ismertetésétől, csak a perifériák elvi működését mutatom be.

1. A SERPENT 1

A mechanika

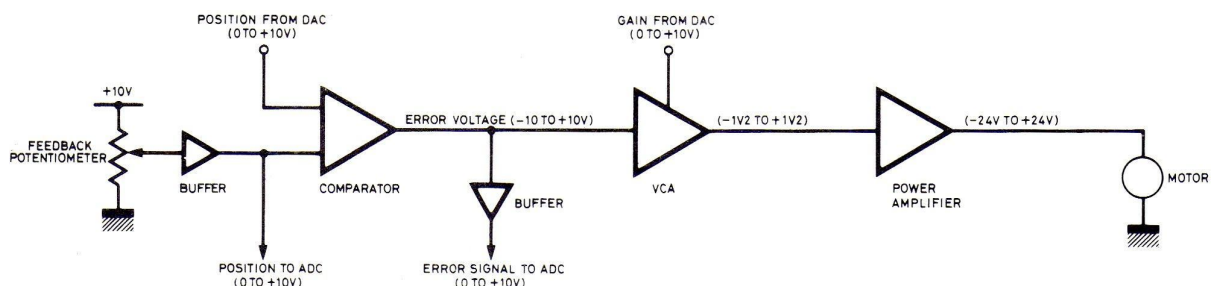
A SERPENT 1 egy SCARA típusú robot. A SCARA egy mozaikszó, a *Selective Compliant Articulated Robot Arm* kifejezésre. Ez azt jelenti, hogy minden szervó által vezérelt kar „automatikusan” engedékeny abban a síkban, amelyikben működik, tehát a csuklók nem képesek minden irányba mozogni, vertikális irányú elmozdulás esetén mereven, míg horizontális irányú elmozdulásnál engedékenyen viselkedik. Az 1.1 ábra mutatja a SERPENT robotok anatómiáját. A függőleges mozgatót egyszerű megoldani: a robot előre beállított magasságban dolgozik, csak a megfogó használható egy másik síkban is. A vízszintes mozgatót DC motorokkal valósították meg, ami a visszacsatolásokkal szervohajtást alkot. Ehhez már szükség van a vezérlő áramkörre, a rendszer „agyára”.



1.1 ábra: SCARA anatómiája

A vezérlés

„A tervezők több megoldás közül választhattak: az egyik, hogy egy processzor alapú egységnek küldik el a pozíció adatokat, ami aztán kiszámolja a szükséges meghajtási feszültséget, és azt a motorokra kapcsolja pl. egy PWM generátoron keresztül. Ennek a hátulütője, hogy egy processzor egyszerre csak egy dologgal tud foglalkozni. Ha a motorokra figyel, nem tud a géppel kommunikálni, ha a géppel dolgozik, akkor pedig a motorok vezérlés nélkül maradnak. Egy második processzor, amely csak a motorokra figyel, megoldotta volna a problémát, ám ezzel elveszítik a szoftver könnyű módosíthatóságát és hozzáférhetőségét, amely jelentős hátrány pl. az oktatásban. A másik megoldás, amelyet végül alkalmaztak, hogy analóg elektronikát használnak a szervohajtáshoz. Az analóg elektronika működik processzoros felügyelet nélkül is, és a karakterisztikája változtatható működés közben.

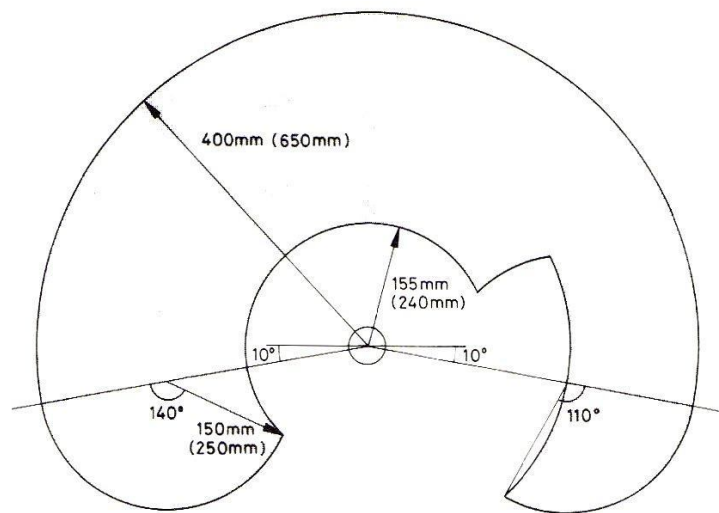


1.2 ábra: SERPENT szervó-vezérlő sematikus rajza

Az 1.2 ábra a SERPENT robotok szervo-erősítőjét mutatja. A karok érzékelői jó minőségű potenciométerek (conductive polymer). Ezek 10V-os jelszinttel dolgoznak. Az innen kijövő jelet először egy tárolóba vezetik, majd egy komparátorba. A komparátor másik bemenő feszültségét egy DAC (Digital to Analog Converter) adja, melyet mi állíthatunk be számítógépről. Ez tehát meghatározza a különbséget a jelenlegi, és a kívánt helyzet között. Ez lesz feszültségben kifejezve az Error voltage (1.2 ábra). Ezt a jelet bevezetik egy VCA-ba (Voltage Controlled Amplifier), melyet szintén egy DAC vezérel. Ezáltal lehetőség nyílik arra, hogy a rendszer erősítését, így az érzékenységet a felhasználó állítsa be. Ezt követően a jel egy teljesítményerősítőt hajt meg, a kimeneten pedig megjelenik a motorok feszültsége. Erdemes megfigyelni, hogy itt a motorok egyik kivezetése mindig föld potenciálra (0V) kapcsolódik, a forgásirányt és a sebességet pedig az ettől való pozitív vagy negatív eltérés, és annak mértéke határozza meg.

Azért, hogy ne csak az áramkör, de a szoftver is tudja a karok helyzetét, a komparátor előtti feszültséget, mely a potenciométertől jön, megméri egy ADC-vel (Analog to Digital Converter), és az így kapott digitális bitmintát a program átalakítja, majd kijelzi.

Végeredményben a potméter és a DAC feszültségének különbsége határozza meg azt, hogy meddig menjenek a motorok. Ez a két feszültség, ha egyenlő, akkor a robot elért oda, ahova mennie kellett.” [1]



1.3 ábra: SERPENT robotok munkatere

A rendszernek igen nagy hátránya, hogy csak egy speciális illesztőkártyán keresztül tudott kommunikálni, ez viszont nem használható minden számítógéphez.

Annak érdekében, hogy az általam tervezett rendszer funkciójában többet nyújtson, mint a Cybernetic Applications mérnökei által tervezett egység, orvosoltam a fentebb taglalt problémákat. A vezérlő és a gép közötti adatátvitelre egy napjainkban szinte minden számítógépen (hordozhatón és PC-n egyaránt) elérhető kommunikációs interfészt választottam. Az egész rendszer felépítését más irányból közelítettem meg, ezért a végeredmény egy olyan vezérlő egység lett, mely a robotot többféle üzemmódban képes irányítani, illetve tartalmaz hardveres és szoftveres biztonsági elemeket, ezáltal védve meg magát a nem szakszerű használattól.

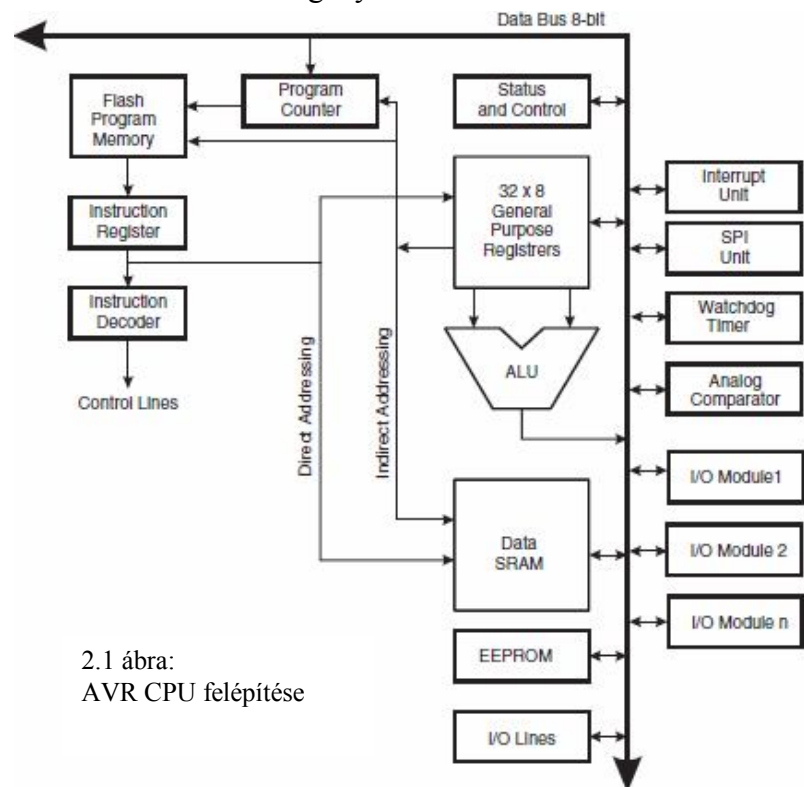
2. A választott egységek bemutatása

A mikrovezérlő:

A tervezés kezdete óta olyan eszközt kerestem, amelyet lehet magas szintű nyelven programozni, és ehhez a magas szintű nyelvhez rendelkezik megfelelő támogatással. Az egyik lehetőség a PLC lett volna, de ezt a magas árak miatt elvetettem. A másik opció egy mikroprocesszoron alapuló rendszer. Ennél a lehetőségnél maradtam, ezért esett a választásom az ATMEL cég termékeire. Már csak az volt kérdéses, hogy hány lábú eszköz legyen, illetve, hogy miket kell tudnia. A fentieket megfontolva az ATmega16A mikrokontroller mellett döntöttem.

De mi is az a mikrokontroller? Első körben úgy lehet elképzelni, mint egy mini számítógépet. Rendelkezik processzorral, ALU-val, perifériavezérlőkkel, buszvezérlőkkel, időzítőkkal, stb. Tehát szinte olyan, mint egy PC alaplap. A hasonlóság azonban ezzel véget is ért. A mikrovezérlő processzora sokkal „butább”, mint egy PC processzora. Ez egy RISC processzor (Reduced Instruction Set Computer), ami annyit tesz, hogy viszonylag kevés utasítással rendelkezik, ám mivel ezeket hardveresen támogatja, nagyon gyorsan el tudja őket végezni. Ezeket a processzorokat elsősorban nem számításgényes feladatokra tervezték. El tud

végezni bonyolult számításokat is, de ez sok időbe telik neki, ellenben a vezérlési, mérési, és ezekkel kapcsolatos számításokat hihetetlenül gyorsan végzi el. A másik szembeötlő különbség, hogy Harvard-architektúra elvén működnek, ellentétben a PC-k többségével, amik a Neumann-architektúrát követik. Ez annyit jelent, hogy míg a PC a memóriájában programot, és adatot is tárolhat, addig a mikrovezérlőknek erre külön memóriájuk van. Ez elsőre nem szembeötlő, de ha belegondolunk, akkor a PC processzora, amikor végre kell hajtania valamit, akkor



2.1 ábra:
AVR CPU felépítése

először programot, majd adatot olvas a memóriából ugyanazon a buszon. Ellenben a mikrovezérlő processzora, mivel a két külön memóriához két külön buszon fér hozzá, ezért egyszerre be tudja olvasni a programot, és az adatot is. Van még egy érdekes ötlet, amivel az ATmega16 sebességét növelték. Ez a *pipelining*. Ez alatt azt értjük, hogy amíg a processzor egy utasításon dolgozik, addig a belső hardver már elkezd betölteni a következőt a programmemóriából, ezzel is időt takarítva meg. A fenti elveknek köszönhetően másodpercenként akár 16 millió utasítást is elvégezhet (16 MIPS).

Az eddigiekből látszik, hogy memóriával rendesen el van látva. Nézzük bővebben:

Az programmemória esetünkben egy FLASH memória, melyet az adatlap szerint minimum 10000-szer lehet újraprogramozni. Amikor az eszközt felprogramozzuk, ide töltjük fel a programot, melyet aztán a processzor lépésről lépésre hajt végre. Az adatmemória SRAM. Ez

egy felejtő memória. Itt tárolja a processzor a saját működéséhez szükséges adatokat, a változók értékeit, stb. Ráadásként tartalmaz EEPROM-ot is. Mivel a programmemóriába nem írhatunk bele, az SRAM pedig nem őrzi meg az értékét, ezért az EEPROM-nak nagy hasznát vesszük, ha akkor is szeretnénk adatot tárolni, amikor nem kap feszültséget a rendszer, pl. beállítás mentése.

Ahhoz, hogy a külvilággal kapcsolatot tartsunk, szükség van ki és bemeneti lábakra. Ezeket a digitális ki és bemeneteket (továbbiakban I/O) portokba rendezve találjuk, ezeken keresztül történik a tényleges vezérlés. A legtöbb I/O lábnak van alternatív funkciója, melyek segítségével képes pl. soros portot kezelni, többféle buszon kommunikálni, analóg jeleket mérni, stb.

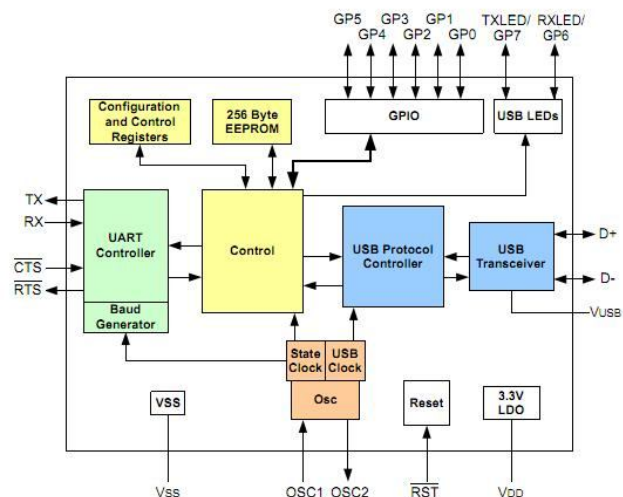
A kommunikációs modul:

A felhasználóval történő kapcsolattartás jelen esetben kiemelten fontos, hiszen a tényleges irányítás számítógépről történik. Kommunikációs interfésznek elsőként soros portot akartam használni, mert a mikrokontroller is támogatja, de a jelszintjük nem egyezik, valamit manapság egyre ritkább, ezért újragondolva a kommunikációs sémát, az USB-re esett a választásom. Egyrészt mert szinte minden számítógépen megtalálható, másrészt a felhasználó szempontjából egyszerű kezelni, viszonylag kis méretű csatlakozókkal kell dolgozni, ellentétben a DB-9-es RS-232 csatlakozóval (2.2 ábra).



2.2 ábra: USB-A és DB-9 típusú csatlakozók

Ahhoz, hogy a mikrokontroller és a PC USB-n keresztül kommunikáljon, szükség van egy cél IC-re, amivel a mikrokontroller soros porton keresztül, a PC pedig USB-n keresztül kommunikál. Ilyen eszköz elég sok van a piacon, én a Microchip által gyártott MCP2200-at választottam, elsősorban azért, mert könnyű hozzájutni, és igen kedvező az ára. Ez támogatja a teljes sebességű USB adatátvitelt (12Mb/sec), CDC (*Communication Device Class*) és HID (*Human Interface Device*) módban is működik, rendelkezik hardveres adatirányítással, 8 általános I/O lábbal, stb. Mivel az USB sokkal gyorsabb, mint az USART, ezért a tervezők 128 bájt belső memóriával látták el, mely az USART modulhoz tartozik, fele-fele arányban megosztva a ki és a bemenő adatoknak. Amíg az USB-ről érkező adatot az USART vonalon nem tudja továbbítani, addig ebben a FIFO (*First In First Out*) elven működő memóriában tartja.



2.3 ábra: MCP2200 belső felépítése

A motorok vezérlése:

Úgy döntöttem, hogy a motorok vezérléséhez szükséges teljesítményelektronikának egy ilyen célra kifejlesztett integrált áramkört használok. Ezt azért tartom előnyösebbnek, mert a gyártók olyan plusz funkciókat is integráltak ezekbe az eszközökbe, amiket csak bonyolult kombinációs logikával lehetne megvalósítani. Ezáltal tovább növekedne a panel mérete, másrészt olcsóbb, mint ha én építenék össze különálló elemekből egy hasonló teljesítményű áramkört. A megfelelő IC kiválasztásához szükséges tudni néhány adatot:

- ♦ a motorok 24VDC-ről üzemelnek.
- ♦ méréseim szerint a karok motorjainak áramfelvétele nem haladja meg az 1A-t terheletlenül (váll motorja 0,6A, kar motorja 0,4A, csukló motorja 0,3A). Ez terhelés alatt növekedni fog, ezért ennél az értéknél nagyobbbat választok, azonban a robot funkciójából fakadóan, ugyanis nem nagy terhek mozgatására találták ki, (adatlapja szerint maximális kapacitása 2kg) ezért kb. 1,5-2,5-szeres biztonsági tényezővel számolok.

Az alábbiakban összegyűjtöttem néhány ilyen célra tervezett áramkört, tulajdonságaikkal, és határértékeikkel.

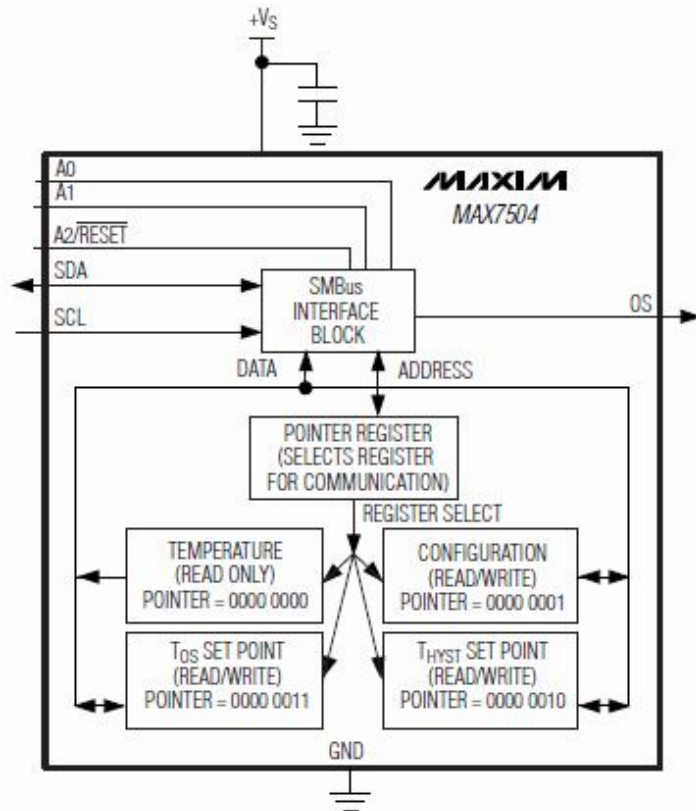
Áramkör neve	Tulajdonságai	Paraméterei
L293B/E	Túlmelegedés elleni védelem, 4 db félhíd, BJT kimenet, TTL logika, az E jelű típushoz külső ellenállások köthetők a kapcsolóüzemű vezérléshez	Max. 1A csatornánként, Max. 36V
L293D	Túlmelegedés elleni védelem, 4 db félhíd, BJT kimenet, TTL logika,	Max. 600mA csatornánként, Max. 36V
L298N/P	Túlmelegedés elleni védelem, 4 félhíd (2 híd) BJT kimenet, TTL logika, a félhidak közös pontja felhasználható árammérésre	Max. 2A csatornánként, Max. 46V
TDA1085CG	Beépített frekvencia-feszültség átalakító, lágyindítás, terhelő áram korlátozás, triak kimeneti fokozat, közvetlenül táplálható AC vonalról	Max. 200mA, Max 15V
UC3176	2 db nagy teljesítményű műveleti erősítő, integrált precíziós erősítő kimenet áramméréshez, feszültség-kimaradás érzékelése	Max. 2A, Max. 35V
TLE 5205-2	Túlmelegedés és kimeneti rövidzár védelem, TTL szintű logika, FET-es kimenet, integrált védődiódák, feszültség-kimaradás érzékelése, terhelés megszűnésének érzékelése	Max. 5A, Max 40V
L6203	Túlmelegedés elleni védelem, rövidzár védelem, belső feszültség-stabilizátor a logikának, TTL szintű logika, félhidak közös pontjának kivezetése árammérésre	Max 5A, Max 48V

A fenti paramétereket kiértékelve az STMicroelectronics L298N típusú áramkörére esett a választásom. Mindegyik IC 2 hidat tartalmaz, ezért a 3 motor meghajtásához a 2 IC szükséges, a maximálisan megengedhető árama a korábban meghatározott intervallumba esik, valamint a fentiek közül az egyik legjobb ár / érték aránnyal rendelkezik.

Hőmérsékletmérés:

Hőmérőkre a jelen alkalmazásban azért van szükség, hogy a motorvezérlő áramkörök hűtőbordáinak hőmérsékletét mérjük. Ha a hűtőbordák nagyon melegek, akkor túl nagy a terhelés a motorokon, ebből következhetni lehet esetleges rendellenességekre. Többféle hőmérő áramkör létezik, és ennek megfelelően többféle elven mérhetnek. Én egy digitális hőmérőt választottam a Maxim Integrated Products katalógusából. A digitális hőmérő azt jelenti, hogy egy tokba integrálták a mérő elektronikát, valamilyen kommunikációs interfészt, kezelhet hőmérsékletkorlátokat, riasztási kimenetet, és egyéb funkciókkal bírhat, melynek csak a tervezők képzelete szabhat határt.

A választott áramkör a MAX7503. Ez egy „*local temperature sensor*”, ami annyit jelent, hogy a saját tokozásának a hőmérsékletét méri, maximum 9 bites pontossággal, ami $0,5^{\circ}\text{C}$ -os felbontást eredményez -55°C -tól $+125^{\circ}\text{C}$ -ig. Rendelkezik programozható alsó és felső riasztási hőmérsékletekkel, melyekhez tartozik egy riasztási kimenet is. I²C (*Inter Integrated Circuit*) buszon kommunikál, ami az integrált áramkörök közti kommunikációra lett kifejlesztve, elsősorban periféria-áramkörök kezelésére. A kommunikációhoz szükség van egy master-re, ami a mikrokontroller lesz, valamint egy vagy több slave-re, ami jelen esetben a digitális hőmérő.



2.4 ábra: MAX7503 digitális hőmérő blokkdiagramja

3. Elektromos biztonság

Kétféle biztonságról beszélhetünk: a valós áramköri veszélyt jelentő túlfeszültségről, és esetlegesen túlámról, melyek a mikrokontroller tönkremeneteléhez vezethetnek, és a funkcionális veszélyt jelentő áramköri zavarokról. Fontosnak tartom, hogy tervezésnél ezekre kiemelt figyelmet kell fordítani, hiszen ezek már döntően befolyásolhatják a rendszer működőképességét.

Valós áramköri veszély:

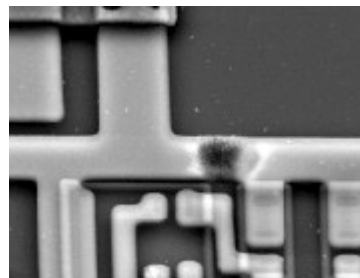
Ide tartozik az ESD (*Electrostatic Discharge*) védelem, az egyéb forrásból származó túlfeszültség és fordított polaritású feszültség elleni védelem, valamint az I/O lábak túlterhelése.

Mivel a rendszeren belül rendelkezésre áll a stabil 5V és 0V, továbbá nem feltételezem, hogy a jövőben olyan szinten bővítik a rendszert, ami plusz tápforrás alkalmazását igényelné, ezért elegendőnek tartom, hogy a rendszerbe csak egy fordított polaritás elleni védelmet építettem, a stabilizátor rész elé. Ha a tápfeszültség polaritása megfelelő, akkor a panelon kigyullad egy zöld LED. Ellenkező esetben semmi nem történik.

Mivel a főpanel közvetlen kapcsolatban áll a motorvezérlő panellal, amely jóval magasabb feszültségen dolgozik, ezért a motorvezérlők meghibásodása esetén ez a feszültség megjelenhet a mikrovezérlő lábain. Ha ez megtörténik, nem lehet pontosan megmondani, hogy az egymás után következő egységek (USB modul, PC) hogyan viselkednek, illetve milyen valószínűséggel milyen károsodást szenvedhetnek, de az nagyon valószínű, hogy a mikrokontroller tönkremegy. Ezt elkerülendő, optocsatolókat használok a mikrovezérlő motorokat irányító kimenetein. Az optocsatoló a galvanikus leválasztás egy fajtája. A jelet egy nem elektromos mennyiséggé alakítjuk, jelen esetben fénné, amit a másik oldalon visszaalakítunk elektromos jellé. Ennek köszönhetően a kontroller és a 24V között egy „fény” teremt kapcsolatot. Az optocsatoló hátránya, hogy csak egy irányba működik, ha egy lábat kimenetnek terveztünk, és optocsatolóval láttuk el, az csak kimenet lehet, és ugyanígy fordítva.

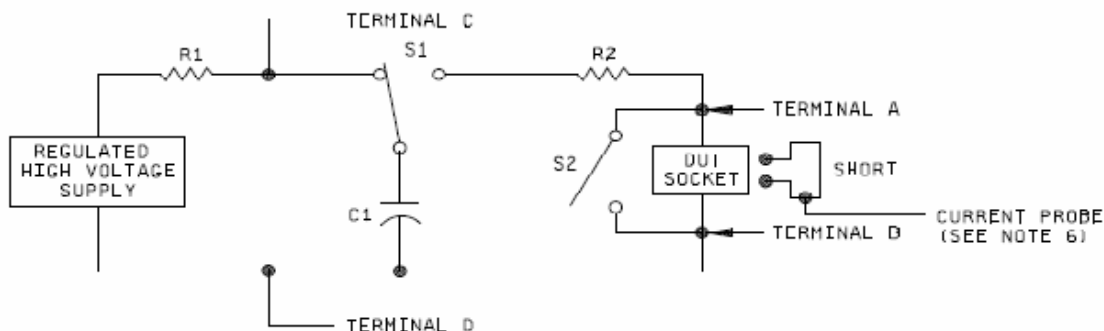
A fentebb említett veszélyforrások az esetek többségében egyértelmű nyomokat hagynak maguk után, ha bekövetkeznek (pl. az IC szokatlanul melegszik, esetleg füstöt látunk / érzünk, vagy nem működik).

Létezik azonban egy olyan alattomos veszélyforrás, amely csak ritkán hagy egyértelmű nyomot maga után, ellenben az elektronikai iparban készült selejtek nagy része ennek a számlájára írható, ezért megéri bővebben foglalkozni vele. Ezt bizonyára mindenki tapasztalta már, amikor pl. bevásárlókocsi fém felületéhez érve csípést érzett, vagy a feltöltődött ruha levételkor szikrázott. Ez az elektrostatikus kisülés, vagy ESD. Sokan hallottuk már, hogy nem szabad a számítógép alaplapját fogdosni, mert tönkremehet. Hát ez az oka. Nem nehéz elképzelni, hogy tesz tönkre egy IC-t, akár tokozáson keresztül is. Ma a félvezető technológiában a vezetősávok szélessége a mikrométer és a nanométer határán van. A hirtelen fellépő nagy áramerősség gond nélkül tönkretesz egy ilyen vékony vezetősávot (3.1 ábra).



3.1 ábra: ESD kisülés eredménye

A gyártók az eszközök adatlapjában feltüntetik, hogy az ESD veszélyes-e az IC-re, illetve megadják azt a határértéket, amelyre az eszközt tesztelték. Ezt az úgynevezett Human Body Model-el (Emberi Test Model) vizsgálják. A tesztelésre vonatkozó előírásokat, részleteket a JEDEC JS001 és a United States Military Standard MIL-STD 883H szabvány 3015.8-as metódusa is tartalmazza.



3.2 ábra: ESD teszt berendezés funkcionális ábrája

Pl. a MAX7503 adatlapja a max. ESD feszültséget $\pm 2000\text{V}$ -ban határozza meg, továbbá megadja, hogy 100pF kondenzátorral, és $1,5\text{k}\Omega$ ellenállással tesztelték (szabványban rögzített). Ezt a következőképpen kell értelmezni: az elektrosztatikusan feltöltött emberi testet modellezi a 100pF -os kondenzátor, az 1500Ω -os ellenállás pedig a test ellenállását jelképezi (innen a Human Body Model elnevezés). A teszt úgy zajlik, hogy feltöltik a kondenzátort, jelen esetben 2000V -ra, majd az ellenálláson keresztül kisütik úgy, hogy az közben csatlakoztatva van az IC-hez, ezzel szimulálva azt, hogy egy hasonló mértékben feltöltődött ember „sül ki” rajta keresztül. A mikrokontrollernek a gyártó szerint min. $\pm 4\text{kV}$ -os kisülést kell elviselnie, ami már egy viszonylag jó érték a fentebb bemutatott $\pm 2\text{kV}$ -hoz képest, ám a kisülés érzékelési küszöbe 3kV körüli (ettől felfele érezzük), és sajnos ez az alsó határ; nem ideális esetben akár néhány tíz kV is lehet. További szomorú tény, hogy beépítéssel az alkatrész érzékenysége megmarad, beszerelt állapotban is tönkre lehet tenni. Az ESD ellen történő védekezés egyik legegyszerűbb módja, ha megakadályozzuk, hogy a felhasználó hozzáérjen az érzékeny alkatrészekhez. Ezt a MAX7503 esetében úgy oldottam meg, hogy az IC-khez nem lehet hozzáférni, csak szerelés után, így a rajtuk keresztül történő véletlen kisülés esélye nagyon minimális. A mikrokontroller helyzete kicsit más, hiszen elég frekvenciált helyen van, és nem lehet csak úgy „eldugni”. Ez a későbbiekben egy műszerdobozba szereléssel lesz megoldva, azonban a dobozolás jelenleg csak akadályozná a fejlesztést.

Harmadik veszélyforrás az, ha a mikrokontroller kimeneteit túlterheljük. A kimeneti lábak feszültsége maximum a tápfeszültség lehet, aminek felső értéke $5,5\text{V DC}$, valamint minden általános célú kimenet maximum 40mA áramot képes leadni. A legegyszerűbb megoldás az, ha nem engedem, hogy a felhasználó módosíthassa a hardvert. Ennek érdekében minden felhasznált kimenet fix helyre csatlakozik (nem lehet máshova bekötni a csatlakozók miatt), és ahol szükséges volt, áramkorlátozó ellenállásokat használtam. Ezeknek az ellenállásoknak az értéke az adott láb feladatától függ, de sehol nem használtam ki a teljes terhelhetőséget

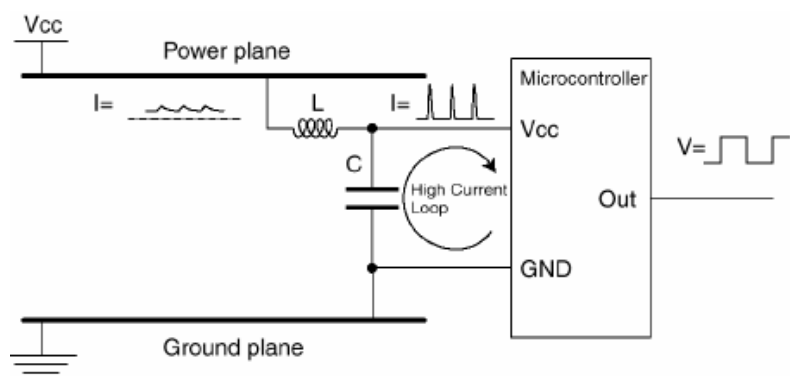
Funkcionális áramköri veszély

Ide nem a megszokott módon értelmezett áramköri veszélyek tartoznak. Ezek nem veszélyeztetik közvetlenül az áramkör fizikai épségét, ellenben mérési, kommunikációs és egyéb problémák forrásai lehetnek. Két csoportba sorolom őket attól függően, hogy a rendszeren belül keletkeznek, vagy kívülről jutnak be.

A külső források közül az egyik leglényegesebb a tápfeszültség. Az nem jó, ha csak egyenirányított és puffertelt feszültséget használunk tápfeszültségnek, hiszen az még nem elég stabil, kis feszültségtűskék szuperponálódhatnak az alap feszültségre. A legjobb megoldás egy feszültség-stabilizátor IC használata. Jelen alkalmazáshoz az L7805 IC-t választottam. Ennek eredményeként stabil 5V DC lesz a kimenetén, ami alkalmas az áramkörök táplálására.

A rendszeren belül keletkező zajok egy részét maga a mikrokontroller állítja elő. Ha a kontroller adatlapját megnézzük, akkor látható, hogy az áramfelvétele 15mA körül van. Ez azonban nyugalmi érték. Amikor a kimeneteket kapcsolgatjuk, akkor ez az érték egy pillanatra megnő. Ha az egész portot kapcsoljuk egyszerre, elég nagy „tűskék” keletkeznek. Nem túl szerencsés dolog ezt figyelmen kívül hagyni, hiszen zavarhatja más áramkörök működését.

Köztudott, hogy áram csak zárt hurokban folyik, ahogy a villamos jelek is csak zárt áramkörben terjednek, azonban a villamos jelekhez hasonlóan a zavarok is terjedhetnek. Minél hosszabb egy ilyen áramhurok, annál több zavart szed össze a környezetéből. További sajnálatos tény, hogy a fentebb említett nagyfrekvenciás jelek és zavarok ezeket az áramhurkokat „antennaként használják”, ezáltal nem csak a saját áramkörüket, hanem a közelükben lévő egyéb berendezéseket is megzavarhatják. Ezt a



3.3 ábra: Áramhurok helyes kialakítása és leválasztása

problémát úgy orvosoltam, hogy a lehető legrövidebb áramhurkokat hoztam létre,

amiket a tápvezetésekről ún. hidegítő kondenzátorok segítségével választottam le, így csak az a kis hurok sugároz.

Az ATmega16 analóg áramkörökkel is rendelkezik, szükség van egy analóg tápfeszültségre is. Az ilyen rendszereknél – ahol analóg és digitális táplálás egyaránt szükséges – bevett gyakorlat, hogy analóg és digitális tápfeszültségnek is ugyanazt a feszültségforrást használjuk, de az analóg tápfeszültséget közvetlenül a feszültségforrás bevezetési pontjától vezetjük el. Ez némileg ellent mond a józan logikának, hiszen fentebb felvázoltam, mi a következménye a hosszú tápvezetéseknél és áramhurkoknak, azonban ezzel a módszerrel megelőzhető az, hogy a digitális rendszer kapcsolási és egyéb zavarai befolyásolják az analóg mérés eredményét.

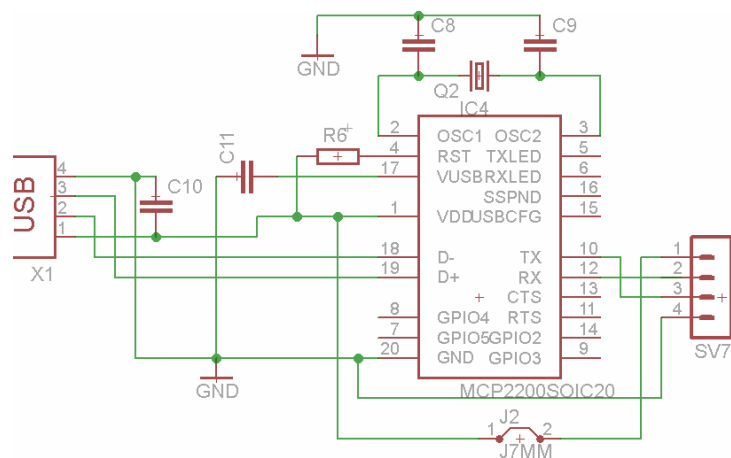
4. Villamos tervezés

Az USB átalakító panel:

Ennek az egységnek a feladata, hogy megvalósítsa az adatátvitelt a számítógép USB portja, és a mikrovezérlő soros portja között. Erre a feladatra a már fentebb említett MCP2200 céláramkört használtam. A helykihasználás miatt panelen furatszerelt és SMD alkatrészek is helyet kaptak. Az MCP2200 működéséhez egy külső 12MHz-es kvarc oszcillátorra van szükség. Ezt az órajel bemeneti lábához a lehető legközelebb kell elhelyezni, és 1-1 kerámia kondenzátorral hidegíteni kell a föld felé.

Az USB erőssége, hogy differenciális átvitelt használ. Az így megvalósuló közös módusú zajelnyomás miatt egy nagyon stabil átvitelt kapunk. Ennek egyik követelménye, hogy a két párhuzamosan futó adatvezeték ugyanolyan hosszú legyen.

Az áramkör úgy lett kialakítva, hogy csak az USB panel veszi a működéséhez szükséges 5V-os feszültséget az USB buszról. A bekötést ennek alapján az IC adatlapja szerint végeztem el. Az IC-n található egy *active-low reset*, amely a külső, manuális újraindításhoz használható. Ennek használatát nem tartottam fontosnak, hiszen ha kihúzzuk, és újra bedugjuk a csatlakozót, ugyanazt a hatást érzük el, ezért logikai magas szintre kötöttem. A számítógéphez egy USB-B kábellel csatlakozik, míg a mikrokontroller paneljába egy négy tűskés precíziós tűskesorral illeszthető, valamint a megfelelő mechanikai stabilitás érdekében csavarral is rögzítettem.



4.1 ábra: Az USB kommunikációs panel kapcsolási rajza

Hőmérő panelek:

A MAX7503 csak SMD tokozással készül, ami nagyban megkönnyítette a dolgomat, amikor a hűtőbordákra kellett szerelni. Ezek az áramkörök gyárilag úgy lettek kialakítva, hogy az egész felületük hő-fellevő felület legyen. Amikor a mért hőmérséklet eléri az előre programozott hőmérsékletet, akkor a riasztási láb az általam beállítottak szerint viselkedik. Ezt a változást figyelem a mikrovezérlővel, és ebből el tudom dönteni, hogy éppen túllépte a felső hőmérsékletet, avagy az alsó küszöb hőmérséklet alá csökkent.

Kialakításukat tekintve egyszerű eszközök, a működésükhöz csak egy felhúzó ellenállás szükséges a riasztási kimenetre, és egy hidegítő kondenzátor a tápfeszültségre. Mivel itt különösen törekedtem a helytakarékosságra, csak SMD alkatrészeket használtam. Az eszközök I²C buszon kommunikálnak. Mivel itt nem építettem be a busz felhúzó ellenállásait, ezért azt majd a mikrovezérlő paneljén kell megtennem. Ezek nélkül nem működne a kommunikáció.

A mikrovezérlő panelja:

Ez a panel a rendszer lelke. A mikrokontroller tudja azt, hogy milyen érkező parancsra mit kell elvégezni, és egész idő alatt felügyeli a robottól, és az egyéb perifériáktól visszatérő érzékelők jeleit, a működéshez azonban szükség van kiegészítő áramkörü elemekre.

Az egyik legalapvetőbb ilyen, a stabil tápfeszültséget előállító alkatrész. Erre egy L7805 típusú lineáris feszültségszabályzó IC-t használok, amely stabil 5V-ot ad a kimenetén. Előnye, hogy olcsó, szinte mindenhol beszerezhető, és nagyon minimális a sugárzott zajszintje. Hátránya, hogy viszonylag nagy feszültségkülönbség és átfolyó áram esetén melegszik. Pl. ha a bemenő feszültség 15V, a kimeneti pedig 5V, akkor a $15-5=10\text{V}$ -ot „elfűti” magán.

Mivel a félvezető alapú elektromos áramkörök nem viselik jól a fordított tápfeszültséget, az L7805 elé beépítettem egyenirányító diódát. Ha a tápegység csatlakozó pontjára fordított polaritású feszültség kerül, a dióda lezár, a modul gyakorlatilag nem kap tápfeszültséget. Ha a polaritás megfelelő, a dióda kinyit, a feszültség stabilizátor IC bemenetén pedig a DC tápfeszültség – a dióda nyitófeszültsége mérhető (kb. 0,6V).

Normál működés közben a panel bal felső sarkán kigyullad egy zöld LED. Ez jelzi a tápfeszültség jelenlétét a rendszeren. Másodlagos funkciója, a fejlesztés és bővítés során, a rövidzárlat „kijelzésekor” hasznos. Ha pl. véletlenül a tápvezetékeket (Vcc és GND) rövidre záránk, a LED elaludna, hiszen az áram mindig a legkisebb ellenállású úton igyekszik eljutni GND-hez, és a LED előtét ellenállása sokkal nagyobb értékű, mint a zárlat.

A LED előtét ellenállását Ohm törvénye alapján határoztam meg. Egy átlagos LED maximális árama 20mA, viszont az áram növelése nem arányos a fényerő növekedésével. Ennek érdekében 500Ω-os ellenállást választottam, amely 10mA-re korlátozza a LED áramát.

A gyakorlatban az igazi munkát a mikrokontroller végzi. Ahhoz, hogy egyáltalán elinduljon, be kell kötni a tápfeszültséget, és megfelelően kell meghajtani a RESET áramkört. A RESET egy active-low-reset, ami akkor indítja újra a kontrollert, ha alacsony logikai szintet kap a bemenetére. Alapállapotban magas szintre van húzva, nem szól bele a működésbe, a RESET gomb megnyomása viszont alacsony logikai szintet eredményez a bemeneten.

Ezekén felül szükséges lehet a külső oszcillátorra. Az összes AVR mikrokontroller rendelkezik belső oszcillátorral, de ezek $\pm 10\%$ -os tűréssel rendelkeznek, ezért a kritikus időzítésekhez, valamint nagy sebességű soros kommunikációhoz a külső kvarc oszcillátor elengedhetetlen. Ugyan a panelra felkerült egy 4Mhz-es kvarc kristály, két darab hidegítő kondenzátorral együtt, ám jelenleg a belső oszcillátorról dolgozik, mert egyelőre nem szükséges a külső használata.

Az egész rendszerben a jelszintek viszonyítási alapja a GND. Az USB panel csatlakozásánál a tápvezetékek közül csak ez van közösítve (fentebb említettem, hogy az USB modult az USB busz táplálja). A mikrokontroller soros interfésze, valamint az USB modul jelszintje teljes mértékben kompatibilisek egymással, így nincs szükség további csatoló elemekre.

A következő sarkalatos kérdés az ATmega16 analóg áramköre. Ezt használják az analóg komparátorok, és az ADC-k is. Három csatlakozási pontja van analóg föld (AGND), analóg táp (AVCC), és analóg referencia feszültség (AREF).

Az AGND a chip-en össze van kötve a digitális földdel, így ezzel nem sok teendőm volt, kívül is összekötöttem GND-vel, hiszen így a terhelő áram két lábon keresztül jut el a földig.

Az AVCC-t a fentebb tárgyaltaknak megfelelően alakítottam ki: minimális áramhurok, hidegítő kondenzátor, és egy 10uH induktivitású tekercs, és a feszültséget a bevezetés pontjához legközelebb eső pontból kapja.

Az analóg referenciát, a fentiekkel ellentétben, kicsit szokatlanul oldottam meg. Mivel ez lesz a mérés alapja, – tehát AREF és GND között fogja a rendszer 10 bites pontossággal mérni az érzékelők feszültségét – meg kellett oldani, hogy egyáltalán ne legyen ingadozás a referenciában. Ez nem egyszerű feladat, hiszen a félvezető alapú áramkörök erősen hőfokfüggők (megközelítőleg $-2\text{mV}/^\circ\text{C}$). Ennek az lenne a következménye, hogy a hőmérséklet változására más értékeket olvasna be. Ez még szembetűnőbb lenne, ha az eszköz dobozba kerül, ugyanis ott a szellőzés korlátozott. Hogy ez ne legyen probléma, egy referencia áramkört építettem be (Texas Instruments, REF02). Ugyan ez is hőfokfüggő, ám a precíziós kimenetét képes ez bizonyos határon belül stabilan tartani.

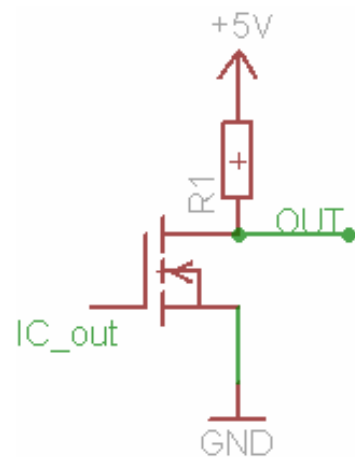
Mivel a REF02 kimenete terhelésszabályozott, megtehettem azt, hogy az érzékelő potméterek is erről kapják a feszültségüket. Erre azért volt szükség, hogy ha potenciométerek és a referencia bemenet együtt változzanak. A terhelésszabályozás azt jelenti, hogy saját maga állítja be a kimenő áramát, így dinamikusan képes követni a terhelés megváltozását. Mivel a potenciométerek párhuzamosan vannak kapcsolva, az ellenállásuk kicsi is lehet, ezért ez a terhelésszabályozás egy elég fontos tényező jelen alkalmazásban. Ezeken felül a REF02 kimeneti feszültsége bizonyos tartományban beállítható, így egy elég precíz referenciát kapunk. A trimmeléshez egy 4,7k Ω -os trimmer potenciométert használtam.

Az I²C busz kapcsolatot is a mikrokontroller valósítja meg, hardveresen. Az ilyen hardveres támogatásoknak az a nagy előnye, – a következő fejezetben kitérek rá – hogy a processzor csak az utasítás kiadásához kell, ezután a processzor is, és az éppen használt hardverem is végzi a saját dolgát tovább.

Az I²C egy kétvezetékes buszrendszer, SDA az adat, SCL az órajel. Maximum 127 slave eszközt lehet a buszra kapcsolni. Hosszát korlátozza a terhelő kapacitás, és a sebesség. Az I²C portok open-drain-es portok (4.2 ábra). Ez azt jelenti, hogy a perifériák csak alacsony logikai szintet tudnak adni. Amikor magas szintre van szükség, csak engedik, hogy a busz felhúzó ellenállásai „felhúzzák” a buszt. Minél gyorsabb a kommunikáció, annál gyorsabban kell az ellenállásoknak felhúzniuk a buszt, illetve minél nagyobb a busz terhelő kapacitása, a felhúzás annál nehezebb. Ez bizonyos határokon belül megoldható erősebb felhúzással, de figyelni kell arra, hogy az alkatrészek meddig képesek ezt kezelni, nehogy tönkretegyük őket. Ha nem sikerül „tartani a ritmust” (nem tudjuk elég gyorsan felhúzni), akkor a küldendő adatok összemosódhatnak, gyakorlatilag összeomlik a busz. Ha szerencsénk van, akkor a periféria észleli, hogy valami nem jó, és újraindítja az I²C áramkört, megóvva minket attól, hogy a rossz kommunikáció miatt esetleg elrontsunk valamit.

A felhúzó ellenállások tehát kulcsszerepet töltenek be a busz kommunikációjában. Értéküket az alapján határoztam meg, hogy milyen erős felhúzásra van szükség.

Most, hogy nagyjából ismerjük az I²C kulcsfontosságú elemeit, érthető, hogy a mikrovezérlő hogyan kommunikál a digitális hőmérőkkel. A hőmérők rendelkeznek egy riasztási kimenettel is. Ez akkor hasznos, ha termosztát módba akarjuk használni. Ez is egy open-drain-es kimenet, tehát itt is a fenti metódust kell követni a felhúzó ellenállás megválasztásához. Ezt a riasztási kimenetet a mikrovezérlő egyik külső megszakítást érzékelő lábára kötöttem.



4.2 ábra: Open-Drain-es kimenet

Szerintem ez a lehető legjobb megoldás, mert ahelyett, hogy állandóan szoftveresen figyeltetném a bemenet állapotát, csak felfprogramozom a hardvert, hogy tegye ezt meg helyettem. Amikor a lábon változás történik, akkor egy meghatározott programrészletet a processzor soron kívül végrehajt, ezután visszatér az eredeti feladatához.

A mikrokontrolleres panel jobb oldalán egy szalagkábel foglalat kapott helyet. Az ehhez kapcsolódó szalagkábel viszi a jeleket a motorvezérlő paneljére. Azért ezt a fajta kapcsolatot választottam, mert a szalagkábel csatlakoztatásakor két fül rácsukódik a kábel végén található csatlakozóra, így egy nagyon stabil, véletlen oldódástól mentes kontaktust hoz létre.

Található még a modul bal alsó sarkán egy hatpólusú csatlakozó. Ez az eredeti programozó csatlakozó, a gyári AVRISPMkII típusú programozóhoz alakítottam ki. Ezen keresztül módosíthatjuk a programot, átállíthatjuk a mikrokontroller speciális regisztereit, és lekérdezhetünk róla információkat. Annak érdekében, hogy a programozáshoz ne kelljen ilyen speciális programozó eszköz, a mikrokontroller felfprogramozható a kommunikációs USB vezetékén keresztül is. A másik oka annak, hogy így oldottam meg az, hogy a hozzá nem értők nem tudják elállítani a speciális regisztereket, hiszen ilyen módon csak a program feltöltésére van lehetőség.

A motorvezérlő panel:

Ha funkciójában nézzük a rendszert, a mikrovezérlő a robot „agya”, de a motorvezérlők „adnak erőt” a mikrokontrollernek.

A motorokhoz sokkal nagyobb áramok kellenek, mint pl. egy LED meghajtásához, ezért a mikrokontroller nem ideális választás a közvetlen motorvezérlésre. Korábban bemutattam, hogy miért az L298N típusú motorvezérlőt választottam. Most kitérek arra, hogy hogyan kell ezeket megfelelően használni, és hogy milyen elemeket építettem be annak érdekében, hogy a lehető legnagyobb biztonsággal a lehető leghosszabb ideig működjenek.

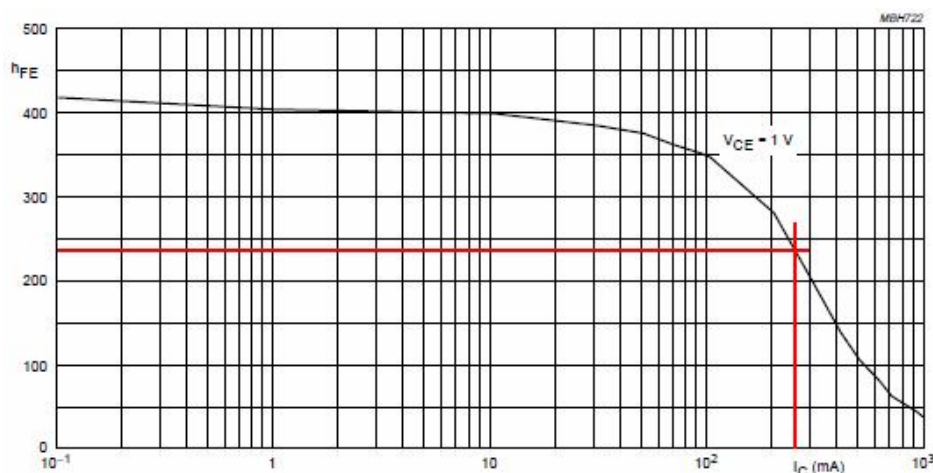
Hogy teljes legyen a kép: az I²C buszos hőmérőket arra használom, hogy a motorvezérlők hőmérsékletét mérjék. Ha egy beállított hőmérsékletet meghaladnak, a mikrokontroller lekapcsolja a motorvezérlő oldal feszültségét. Hogy ezt megtehesse, kell egy kapcsoló elem. Ehhez egy „mezei” monostabil relét használtam. Ennek jellegzetessége, hogy csak egy irányba szükséges vezérelni, és ha a behúzó tekercs gerjesztése megszűnik, akkor egy rugó bontja a kapcsolt kört, és így ugyan az egész működés alatt gerjeszteni kell a tekercset, ám ha a mikrovezérlő paneljén kimarad a tápfeszültség, akkor a relé bontja a motorvezérlő panelon is. A relé kapcsoló köre 5V-ról működik, a kapcsolt körön pedig akár 10A-t és 30V-ot is tud kapcsolni. Sajnos még a relé meghajtása is túl nagy falat a mikrokontrollernek, ezért szükség van egy kapcsoló elemre, amivel a vezérlő kimenete is megbirkózik. Egy BC337-40 típusú bipoláris tranzisztort választottam. Ez 500mA áramot képes vezetni a kollektorán keresztül. A relé behúzó tekercsének névleges ellenállása 47Ω, ±10% tűréssel. A legrosszabb esetben az ellenállás kb. 43Ω, így ezzel számolok a továbbiakban. Ohm törvényéből adódik, hogy 5V-on az árfolyó áram kb. 116mA, ami bőven alatta marad a tranzisztor terhelhetőségének.

Ezzel sajnos nem ért még véget a számítás. A bipoláris tranzisztorokat teljesítménnyel vezéreljük, ezért meg kell határozni a bázis ellenállását is.

Első lépésben megkerestem a BC337-40 DC áramerősítési tényezőjét. Ez határozza meg, hogy a bázison bevezetett áram hányszorosát képes kapcsolni a kollektoron. Jelen esetben ez minimum 250.

$$h_{FEmin} > \frac{I_{load}}{I_{druvmax}} \times 5$$

Ez az egyenlet azt határozza meg, hogy van-e a tranzisztorban elég biztonsági tartalék. Ebben az esetben $I_{load} = 116\text{mA}$, $I_{drv\ max} = 3\text{mA}$ (az az áram, amivel a kontroller vezérli a tranzisztort). $I_{drv\ max}$ -ot én választottam 3mA -re. A kapott érték megfelelő. Ezután megkerestem a BC337 adatlapjában a $h_{FE} - I_C$ diagramot, és leolvastam róla a 116mA -hez tartozó áramerősítést.



4.3 ábra:
A BC337-40
 $h_{FE} - I_C$ diagramja

A diagramról leolvasható az erősítési tényező: kb. 240. A következő egyenletből számítható a bázis ellenállásának értéke:

$$R_b = \frac{V_{cc} \times h_{FE}}{I_C \times 5}$$

$V_{cc} = 5\text{V}$, $I_C = 116\text{mA}$. Az eredmény 2064Ω . Ehhez a legközelebb eső az E96-os ellenállás sorból a $2\text{k}\Omega$ -os ellenállás.

Induktív terhelések kapcsolásánál, amikor megszűnik a tekercs gerjesztése, akkor egy ún. ellen elektromotoros erő (*back emf*) jelentkezik. Ez egy olyan feszültség, amely az ellen az áram ellen dolgozik, amelyik azt gerjesztette. Ez ellen sokan elfelejteneek védekezni, pedig fontos, mert adott esetben tönkreteheti a félvezető kapcsoló elemet. Hogy ezt elkerüljük, egy nagy sebességű diódát (free wheeling diode) kapcsolunk párhuzamosan a relé tekercsével. Hogy hogyan, és miért, arról később lesz szó.

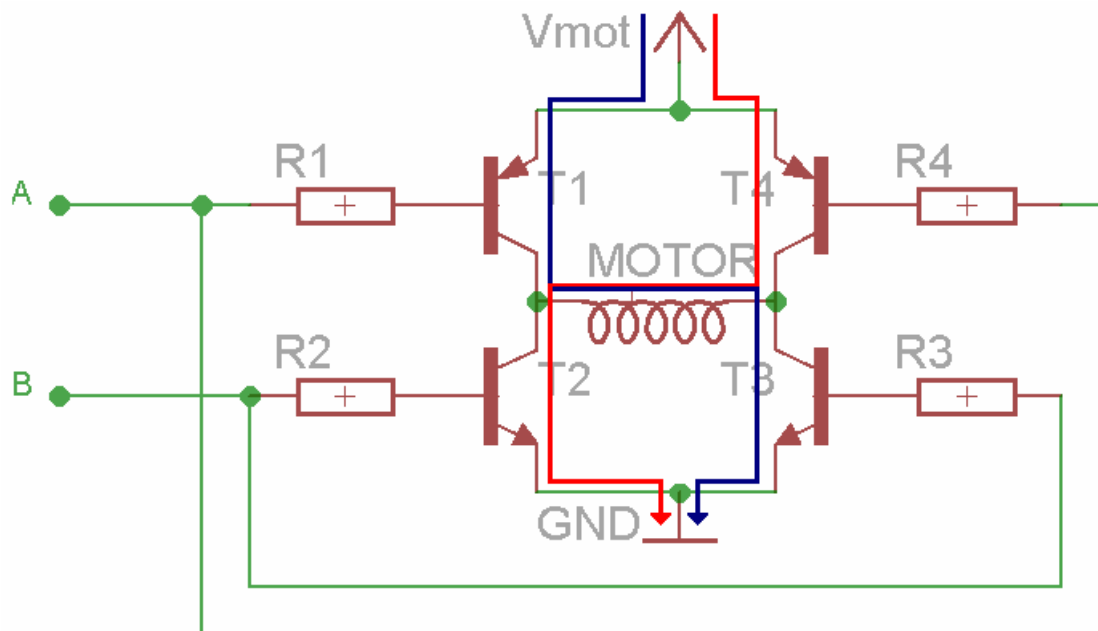
A következő három elem az áramköri biztonságnál már említett optocsatolók. Úgy lehet elképzelni őket, mint egy eszköz, amelynek egyik felén a jelet átalakítjuk fényre, majd a másik végén visszaalakítjuk. Ez azért hasznos, mert a fény – mivel nem vezető közeg – ugyan elektromosan elszigeteli egymástól az áramköröket, ám a jelet átviszi. Jelen esetben tehát a mikrokontroller jeleit átviszi, ám ha a motorvezérlő oldalon rendellenesség folytán nem megengedett nagyságú feszültség kerülne a vezérlő lábakra, a kontrollert és a többi áramkört megvédi. A mikrokontroller oldaláról úgy kell kezelni, mint ha egyszerű LED-eket hajtánánk meg active-low üzemmódban. A motorvezérlő oldaláról pedig úgy kell használni, mint bármilyen open-kollektor-os kimenetet: szükség van felhúzó ellenállásra, azonban itt nem kritikus a felhúzás erőssége, mivel csak a jelszintre van szükség, ezért ide $2,2\text{k}\Omega$ -os ellenállásokat választottam, a LED-ek elé pedig 330Ω -os áramkorlátozó ellenállásokat.

Mivel a motorvezérlők logikai áramköre is 5VDC-ről üzemel, viszont a mikrokontroller oldali 5V-ot leválasztottam, ezért ide is kellett egy ugyanolyan 5V-os stabilizált táplálás. Továbbá itt is megtalálható egy ugyanolyan LED, ami az aktív állapotot jelzi.

A motorvezérlő áramkörökről: 5V-os TTL jelszintekkel kommunikálnak (0V vagy 5V), motoronként 2A a maximális folyamatos áram, a motorok névleges feszültsége 24V. Egy IC két motort tud meghajtani. Kétféle bemeneti lábbal rendelkeznek:

- ♦ INPUTn: ezekkel lehet a forgásirányt, illetve a fékezést beállítani
- ♦ ENABlen: ezzel lehet engedélyezni, hogy az adott motorhoz tartozó meghajtó áramkör hajtja-e a hozzá tartozó motort.

Ezek az IC-k 2-2 teljes H-hídat tartalmaznak. A H híd lényege, hogy a híd két fele ellenütemben működik, ezáltal a motor az egyik, vagy a másik irányba forog (4.4 ábra). Ha az „A” jelű bemenetet vezéreljük, akkor a kék nyíllal jelzett úton fog folyni az áram, a motor elindul az egyik irányba. Ha „A” helyett „B”-t vezéreljük, akkor a motoron folyó áram útját a piros nyíl jelzi. Ekkor a motor az ellenkező irányba indul el. Láthatjuk, hogy azzal, hogy melyik tranzisztorpárt vezéreljük, gyakorlatilag a motoron folyó áram folyásának az irányát határozzuk meg.



4.4 ábra: H-híd elvi áramköri rajza

Az elméleti lehetőség meglenne arra, hogy félhidak közös pontján nagyjából megmérjük a motorok áramát, ám ezzel a rendszer elveszítené azt az előnyt, hogy a motorvezérlés galvanikusan le van választva a mikrokontrollertől.

Korábban, a relénél említettem, hogy induktív terhelés esetén szükség van ún. free wheeling, vagy flyback diódákra. Ezek igen gyors kapcsolású diódák. A motorvezérlőhöz $t_{rr} < 200\text{ns}$ ellenirányú feléledési idővel rendelkező dióda szükséges. Ez határozza meg, hogy a dióda milyen gyorsan képes váltani a ki- és bekapcsolt állapotok között.

De miért is van rájuk szükség? A tekercs egy elég lassú elem, nem azonnal reagál az őt ért hatásokra, gerjesztésekre. Amikor a motorra tápfeszültséget kapcsolunk, akkor a tekercsein keresztül áram kezd el folyni, ami mágneses teret gerjeszt a tekercs körül. Többek között ezért működik a motor. Amikor a félvezetők kikapcsolják a tápfeszültséget a motorokon, nem folyik tovább áram, megszűnik a tekercsek gerjesztése, és a megszűnő mágneses mező

feszültséget indukál a motor tekercsén. Ennek a feszültségnek a neve ellen-elektromotoros erő (*back / counter electromotive force*). Ez a feszültség ellentétes irányú, mint az eredeti feszültség. Sajnos, a félvezetők többségének megvan az a rossz tulajdonsága, hogy nem szeretik az ellenirányú feszültséget, ha az egy bizonyos értéket meghalad. Értékét így számolhatjuk ki:

$$U = -L \frac{di}{dt}$$

Mivel di sokkal nagyobb, mint dt , ezért U is nagy lesz.

Ez a feszültség adott esetben elég nagy lehet ahhoz, hogy tönkretegy a félvezető alapú kapcsoló elemeket. Ezért fontos az, hogy egy gyors feléledési idejű (t_{rr}) dióda segítségével elejét vegyük a félvezetők károsodásának. A dióda bekötése kétféleképpen történhet:

- ♦ Egyik eset az, amikor (pl. a relé esetén) párhuzamosan kötjük a diódát az induktív elemmel úgy, hogy az a tekercs normál működése közben záróirányban legyen előfeszítve. Amikor kikapcsoljuk, és a mágneses mező feszültséget indukál a tekercsben, akkor a dióda nyitóirányban lesz előfeszítve, és rövidre zárja a tekercset. Ennek következtében a félvezető kapcsoló elemen nem az indukált feszültség jelenik meg, hanem csak a dióda nyitófeszültségével megegyező feszültség.
- ♦ Másik eset, amit a motorvezérlő hidaknál alkalmaztam, hogy a diódák nem a motorral vannak párhuzamosan kötve, hanem közvetlenül a motorok kivezetése és a tápfeszültség (V_{MOT} és GND) közé úgy, hogy normál üzemben záróirányban legyenek előfeszítve. Ez azt eredményezi, hogy a motor megállításakor a diódák ugyanúgy kinyitnak, mint az első esetben, de itt a forgásiránytól függően V_{MOT} vagy V_{GND} -re vezetik el a motor indukált feszültségét.

5. A szoftveres megvalósítás

A mikrokontroller programja

A dolgozat elején leírtam a SERPENT robotok tervezőinek problémáit. Az egyik az volt, hogy ha egy processzorral akarták volna megoldani az irányítást, akkor vagy a robotra, vagy a kommunikációra nem tudnak figyelni. Én mégis egy processzorral felügyelem a rendszer összes funkcióját.

Jogos lehet a kérdés: „nálam hogyhogy nem jelentkezik a fenti probléma?”.

A válasz: Jelentkezik. Csak az egész olyan rövid ideig tart, hogy nem érdemes vele foglalkozni.

Ennek az oka a processzorok és mikrovezérlők fejlődésében keresendő, ugyanis a különböző belső perifériák beállíthatók úgy, hogy bizonyos funkciókat ellássanak a CPU felügyelete nélkül is, és amikor azzal végeztek, akkor szólnak a processzornak, hogy feldolgozhatja az adatot. Ezt hívják megszakításnak, szakmai nyelven *interrupt*-nak. Az ATmega16 számos megszakítás-forrással rendelkezik. Jelen esetben az USART (soros kommunikációhoz használt interfész) bejérkező adatokhoz tartozó megszakítást használom. Ez úgy dolgozik, hogy automatikusan érzékeli, ha a soros vonalon adat érkezik, és elkezd összegyűjteni a beérkező bájtot, bitenként. Amikor az egész beérkezett, egy megszakítást küld a processzornak. Minden megszakításnak külön vektora van a vektortáblában, így ha megszakítják a CPU-t, akkor ő mindig tudja, hogy melyik periféria tette ezt.

Képzeljük el a folyamatot:

- Alap eset: processzor éppen egy „világmegváltó” folyamaton dolgozik.
- Miközben a CPU a saját dolgával van elfoglalva, tegyük fel, hogy meg szeretnénk tudni az aktuális hőmérsékletet. Kiküldjük neki soros vonalon a parancsot, hogy mérje le, és küldje vissza.
- Az USART szépen elkezd begyűjteni a biteket, majd amikor a teljes bájt összeállt, megszakítást küld a processzornak.
- A processzor még befejezi az utolsó műveletet, amin dolgozik, majd nekilát, hogy soron kívül kiszolgálja a hőmérsékletmérést. Először elmenti azt az állapotot, ahol abbahagyta a főprogramot, majd elugrik a megszakítás-kiszolgáló rutinhoz. Ennek a neve ISR (*Interrupt Service Routine*). Ez mondja meg a processzornak, hogy mit kell éppen tennie, jelen esetben hőmérsékletet mérni.
- A mikrokontroller leméri az aktuális hőmérsékletet (attól most tekintsünk el, hogy hogyan), és visszaküldi soros vonalon.
- A CPU visszatölti a megszakítás előtti állapotát, és onnan folytatja, ahol abbahagyta, az USART pedig tovább figyeli a kommunikációt.

A fentiekből látszik, hogy az a megszakítás értelme, hogy nem nekünk kell figyelni, hogy a perifériáknak szükségük van-e a processzorra, hanem ezt maguktól jelzik, és mivel a megszakítás-kiszolgáló rutinok magasabb prioritásúak, mint a főprogram, ezért a processzor egy ilyen kérés esetén amint tud, nekilát a megszakítás kiszolgálásának.

Az USART megszakítással nem kell azt figyelmem, hogy érkezett-e új adat, hanem azt a hardver fogja jelezni, nekem csak el kell döntenem, hogy mire használom. Ezzel le is küzdöttem azt az akadályt, ami miatt a SERPENT tervezői elvetették a processzoros megoldást.

Az I²C kommunikáció hardveres követelményeiről korábban írtam. Most az áramkör e részét vesézem kicsit. Ahhoz, hogy érdemi kommunikációról lehessen beszélni, kell egy master, aki

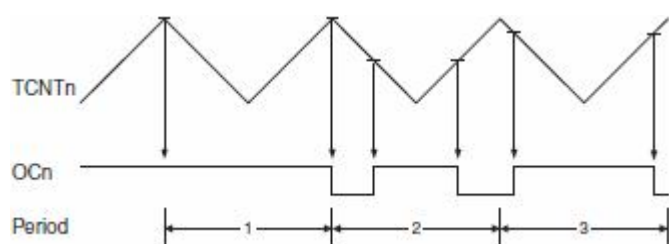
a kommunikációt irányítja. Ez lesz a mikrovezérlő. Továbbá kell egy vagy több slave, akik a master utasításait, feldolgozzák, és végrehajtják. A kommunikációnak több fázisa van:

- ◆ A master először küld egy START jelet a buszon. Ennek hatására az összes slave eszköz „fülel a buszra”.
- ◆ Ezt követően a master kiküldi annak az eszköznek az egyedi címét, amelyikkel kommunikálni akar. A cím legfelső bitje határozza meg, hogy master olvasni akar az eszköztől, vagy éppen írni akar rá. Ezt mindegyik slave megkapja, és amelyik eszköz felismerte a saját címét, az egy ún. ACK-t (Acknowledge) küld vissza a masternek, tudatva vele, hogy kész a további műveletek elvégzésére.
- ◆ A master ezután elküldi az adatot, vagy éppen arra vár, hogy a slave kitegye azt a buszra.
- ◆ Ha továbbra is dolgozni szeretne az eszközzel, akkor egy ismételt start jelet követően a 2. lépéstől ismétli a folyamatot.
- ◆ Ha végzett a kommunikációval, akkor egy STOP jelet küld a buszon.

A kommunikáció a hőmérőkkel a fentiek alapján zajlik. Elsőként kiküldöm a START jelet, majd megcímzem írásra az eszközt. Ezután a konfigurációs regiszterben beállítom a riasztási lábat „Interrupt mode”-ra, majd a felső és alsó riasztási hőmérsékleteket is beírom a regiszterekbe. Ezzel a beállítással, ha a hőmérséklet a felső határértéket meghaladja, akkor a riasztási láb logikai alacsony szintre változik, és egészen addig az is marad, amíg ki nem olvassuk a hőmérsékletet tartalmazó regisztert. Ha kiolvastuk, akkor a láb állapota visszatér magas szintre. Ugyanez megtörténik, ha a hőmérséklet lecsökken az alsó határérték alá, de csak akkor, ha előzőleg túllépte a magas szintet. Szerencsére a riasztási kimenet változását is tudom figyelni a kontrolleren megszakítással, és csak annyi a dolgom, hogy az ISR-ben megmondom, hogy mit tegyen, ha logikai szint változását érzékeli. Én arra programoztam be, hogy ilyen esetben kapcsolja ki a motorvezérlő panel tápfeszültségét addig, amíg nem esik a hőmérséklet az alsó határérték alá.

A motorvezérlés szoftveres oldala: korábban írtam, hogy a motorvezérlők 5V-os TTL logikával rendelkeznek. Digitális rendszerben körülményes analóg jelet előállítani, tehát a TTL logika a legjobb választás. Itt jön képbe az impulzus-szélesség moduláció (PWM). Ezt úgy lehet a legegyszerűbben elképzelni, hogy van egy tetszőleges hosszúságú periódusunk, és ebben a periódusban szabadon állíthatjuk a magas és alacsony szintek arányát. Mivel itt csak magas és alacsony szintek vannak jelen, ezért a motorvezérlő logikája is ugyanúgy fogja értelmezni, és ugyanúgy „pulzálni” fog. Ezzel szemben a motor tekerce – mivel egy viszonylag lassú elem – számára egy analóg jelnek fog tűnni, és egyenletesen, simán fog járni. Minél hosszabb a magas jel aránya a perióduson belül, annál gyorsabban megy a motor. Ezt a PWM jelet a mikrovezérlő egyik legsokoldalúbb perifériái, a TIMER-ek állítják elő. A TIMER nem más, mint egy számláló, aminek az értéke mindig növekszik az általunk megadott ütemben. A növekedés alapvetően a rendszer órajeltől függ, azonban ezen tudunk lassítani ún. előosztók segítségével. Ha pl. az előosztás 1:1, akkor a TIMER értéke a rendszer órajelének ütemére növekszik. Többféle PWM jelet hozhatunk létre, attól függően, hogy a TIMER milyen módban üzemel. Én a fázishelyes PWM jelet használom, mert ez a legideálisabb DC motorok meghajtására.

„Ez a TIMER kettős meredekségű (dual slope) üzemmódján alapul. A kettős meredekségű működés lényege, hogy a TIMER 8 bites módban 0-tól 255-ig számol, majd vissza 0-ig, és az aktuális értékét mindig a TCNTn regiszterben tárolja. Egy OCRn nevű regiszterbe kell a komparálási értéket betölteni. Ezt az OCRn fogja a hardver minden ciklusban összehasonlítani a TCNTn aktuális értékével.” [2]



4.5 ábra: Timer értéke és a kimeneti jel

Nem invertált működés esetén a kimenet akkor magas szintű, ha a számláló értéke kisebb, mint a komparálási érték ($TCNTn < OCRn$).

Ezzel a módszerrel gyakorlatilag fokozatmentes sebességszabályozást lehet megvalósítani. Az így megszagatott jelet a motorvezérlők ENABLEn bemenetére kell kapcsolni.

Ez két okból is előnyös:

- ◆ Egyrészt induláskor a vezérlés nem rántja meg a motorokat, hanem fokozatosan felfuttatja a sebességüket
- ◆ A másik pedig, hogy így korlátozni lehet az áramot a motorok indulásakor. Ez a DC motorok sajátosságából adódik: a rajtuk átfolyó áramot a kapocsfeszültség és az indukált feszültség különbségéből, valamint a motor ellenállásából határozhatjuk meg. Induláskor azonban – mivel a motor nem forog – még nincsen indukált feszültség, tehát marad a kapocsfeszültség, és a tekercs ellenállása, ami igen nagy áramot eredményez. Ennél a módszernél tehát a teljes motor feszültséget rákapcsolom a motorokra, de ezt kezdetben csak nagyon rövid időre, majd ahogy a motor gyorsul, egyre tovább. A tekercs – párszor már utaltam rá, hogy lassan reagál az őt ért hatásokra – ezért úgy viselkedik, mint ha egyre nagyobb analóg feszültséget kapna. Továbbá megtartottam azt az előnyt, hogy a motor mindig leadja a maximális nyomatékát, hiszen ha csak egy pillanatra is, de a teljes motor feszültség mérhető rajta. Ezzel szemben, ha analóg feszültséggel vezérelném – ahogy azt az eredeti vezérlés tette – akkor figyelnem kellett volna arra, hogy a motornak kis feszültségen kicsi a nyomatéka. Ebben az esetben elképzelhető, hogy el sem indul.

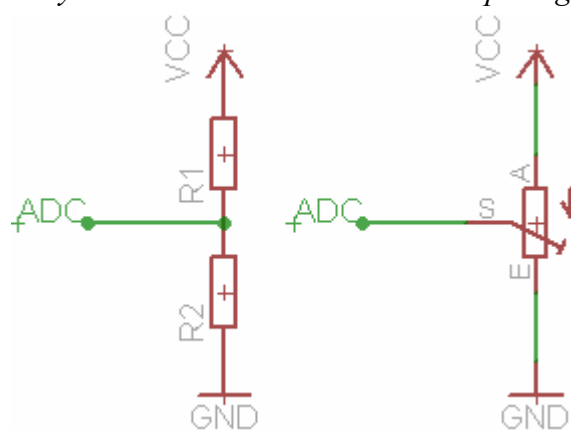
Analóg és digitális világ:

Eddig már van egy olyan robot, amelyik képes mozogni, de valahogy tudnia kell, hogy éppen hol van, és milyen messze a céltól. Ezt *conductive polymer* potenciométerekkel mérem.

„A *conductive polymer* olyan szerves polymer, amely elektromosan vezető. Vezetőképesség szempontjából viselkedhetnek úgy, mint egy fém, vagy úgy, mint egy félvezető.” [3]

Ilyen anyagból csak nagyon precíz potenciométereket gyártanak. Előnyük a kivételesen hosszú élettartam, és a stabil értéktartás az élettartamuk alatt.

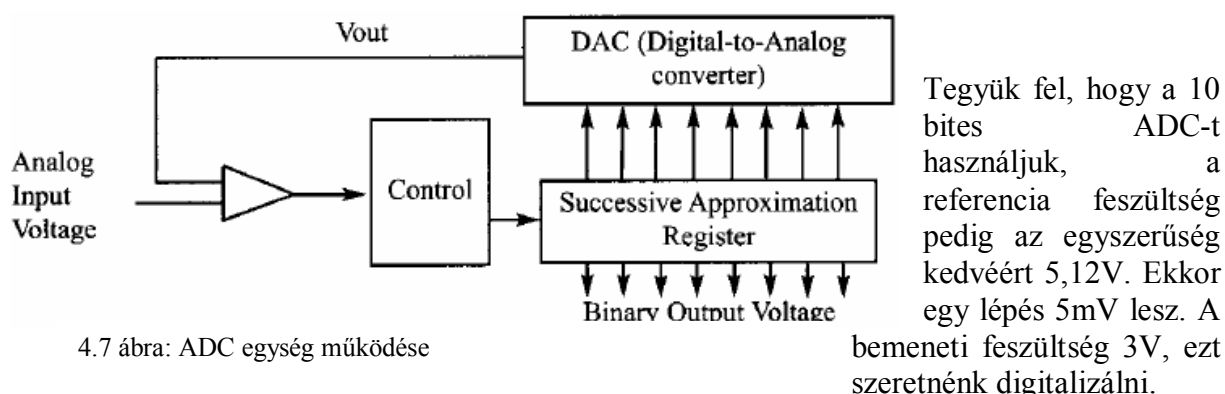
A helyzet mérése úgy történik, hogy a potenciométerek által leosztott feszültséget mérem. A potméterek kivezetéseit úgy határozhatjuk meg, hogy nagyjából középpállásba tekerjük a tengelyt, és megmérjük mindhárom kivezetés közti ellenállást. Ahol a legkisebbet mértük, azon a kivezetésen mérjük a



4.6 ábra: Feszültségosztó és potenciométer

feszültséget, a másik kettőt pedig tápfeszültségre és földre kötjük. Ekkor a potméterünk úgy viselkedik, mint egy feszültségosztó. Ha tekergetjük a tengelyét, akkor megváltoztatjuk a feszültségosztóban szereplő ellenállások értékét, és ezzel együtt a leosztott feszültséget is. Ezzel gyakorlatilag egy változtatható feszültségosztót kapunk. Ha ezt a robot csuklóira szereljük, akkor a leosztott feszültség a robot mozgásával arányosan fog megváltozni.

A potenciométerek alkotta feszültségosztók által leosztott feszültséget ADC-el (*Analog to Digital Converter*) mértem. Ehhez nem kell külön egység, az ATmega16 ADC egysége megfelelő. Ez 10 bites felbontással tud mérni, ami azt jelenti, hogy 1024 különböző feszültségértéket tud megkülönböztetni. Mivel ez már elég jó felbontás, vigyázni kell arra, hogy az eredményt semmi ne hamisítsa meg (ezért kellett különleges megoldások a referencia feszültség, és az analóg áramkörök táplálásánál). Ugyan mérés nyolc egymástól független csatornán történhet, de valójában csak egy AD átalakító egység van, ezért a csatornákat multiplexelve kötik a konverter bemenetére. Az ADC a fokozatos megközelítés elvén (Successive Approximation) működik. Ezt a következő ábra szemlélteti:



4.7 ábra: ADC egység működése

1. „A SAR (Successive Approximation Register – ez alakítja ki az analóg jel digitális bitmintáját) úgy kezd el dolgozni, hogy a legfelső helyiértékű bitet egyesbe állítja (1000000000).”
2. „Ezt a feszültséget egy DAC előállítja, majd egy komparátor összehasonlítja a DAC feszültségét a mérendő feszültséggel. Ha a mérendő feszültség magasabb, akkor a 10. bitet meghagyja 1-nek, ha alacsonyabb, akkor törli. Jelen esetben, mivel ez a referencia feszültség fele, ami kisebb mint 3V, ezért 1 lesz.”
3. „Ebben a lépésben a második bitet állítja egyesbe, az első lemásolja, és kiküldi a DAC-nak, tehát most a referencia feszültség háromnegyed részével hasonlítja össze a bejövő jelet. Mivel ez nagyobb, mint 3V, ezért ezt a bitet törli (0-ba állítja).”
4. „Tovább ugrik a 3. bitre, és ugyanúgy beállítja egyesbe, így a DAC a 1010000000 bitmintát alakítja analóg feszültséggé. Ez 3,2V, ami még mindig nagyobb, mint az eredeti jel, ezért ez is 0 értékű lesz.”
5. „A SAR a fenti szisztéma szerint végiglépked a biteken addig, amíg meg nem határozza a legalsó helyiértékű bitet is. Amikor végzett, akkor jelzi a processzornak, hogy kiolvashatja a bitmintát, ami jelen esetben 1001011000 lesz.” [4]

Ez a hardver oldaláról mutatta be, hogyan is zajlik egy ilyen konverzió. Természetesen nekünk nem kell a biteken ugyanúgy végiglépkedni, ahogy ezt a mikrokontroller teszi. Használhatjuk a következő egyszerű képletet:

$$D_{\text{bitminta}} = \frac{V_{\text{in}}}{V_{\text{step}}}$$

D_{bitminta} = az analóg feszültség digitális megfelelője

V_{in} = mérendő feszültség

V_{step} = egy lépés feszültsége

Integrált funkciók:

Most már a mikrokontroller képes kommunikálni a számítógéppel, van információja a teljesítményelektronika hőmérsékletéről, tudja, hogy milyen helyzetben vannak a csuklói, és mozgatni is tudja őket. Az funkciók tehát összeálltak, már csak alkalmazni kell őket. Az alábbiakban bemutatok néhány olyan függvényt, és támogatást, melyeket azért írtam, hogy könnyebb legyen a különböző igényeket megvalósítani.

Ahhoz, hogy a robot a karokat tudja forgatni, meg kell adni, hogy a motorvezérlő melyik lábára milyen jelszintet kell kapcsolni. A könnyű felhasználhatóság érdekében ehhez saját függvényeket írtam, így elég csak annyit megadni, hogy „Jobbra(motor száma)”, és mikrovezérlő a függvény paraméterében lévő motor irány-vezérlő lábait úgy állítja be, hogy amikor meghívom a „Start(motor száma, max sebesség)” függvényt, akkor jobbra indul el, természetesen a motor sebességét fokozatosan növelve. Ugyanilyen függvényeket írtam még a balra történő mozgásra, a megállásra, és a léptetésre. Ezek közül a léptetésről érdemes említést tenni: a program akkor hívja meg ezt a függvényt, ha a cél annyira közel van, hogy egyszerűbb ezzel megközelíteni, hiszen így a milliméter tört részének megfelelő távolságokat is képes mozdulni. Amikor megközelíti a célt, akkor ha egy bizonyos távolságon belülre ér, visszaveszi a motor fordulatszámát – a korábban részletezett PWM módszerrel – és így közelít tovább. Ezzel elérhető, hogy pontosan a megadott helyen álljon meg a kar. Sajnos a gyakorlatban ez nem mindig működik, mert a megközelítő algoritmus még nem tökéletes. Ritkán előfordul ugyanis, hogy egy kicsit „túlszalad” a célon. Ilyenkor is a már említett léptetés függvénnyel tereli vissza a kart a megfelelő pozícióba, tehát észreveszi, és korrigálja is a tévedést.

A karok pozícióba irányítása inntől kezdve nem túl bonyolult feladat. Csak meg kell adni azokat a pontokat, ahova szeretnénk, hogy a karok menjenek, és a mikrovezérlő addig hajtja a motorokat – természetesen sebesség felfuttatással és a célpontok megközelítésével – amíg a karok helyzete, és a kívánt helyzet meg nem egyezik.

A robothoz tartozik egy betanító kar is. Ez a robot egyszerűsített, és kicsinyített másolata. A feladata az, hogy ha a kis kart mozgatom, a robot lekövesse a mozgását. Ez ugyanúgy potenciométerekkel van felszerelve, tehát a mérési elv nem változik a robot helyzetének méréséhez képest. A feladat annyiban ki is merült, hogy a kis karok helyzetét állítottam be a robotkaroknak, mint követendő helyzet.

Ezen felül implementálásra került egy alaphelyzet funkció is, amelynek segítségével a robotot egy előre definiált helyzetbe lehet állítani. Ennek gyakorlati jelentősége akkor van, ha kikapcsoláskor azt szeretnénk, hogy a robot „összecsukja” magát. Továbbá a rendszer minden indulásakor, amikor a robot nem ebben az állapotban van, automatikusan felveszi ezt a helyzetet.

A robot irányítható ún. kézi vezérléssel is, egy PC szoftveren keresztül. Itt teljes egészében én irányíthatom a robot viselkedését, egy bizonyos határon belül. Ennek a határnak a definiálására azért volt szükség, hogy elkerüljem mechanika és a vezérlés károsodását abban az esetben, ha nem sikerül időben megállítani a mozgást (ilyenkor ugyanis végállásba érkezne, a motor nem forog, és a fellépő nagy áramerősség tönkretethetné a teljesítményelektronikát). Ezáltal kézi üzemmódban a robot karjai csak egy, a munkaterénél kisebb tartományban mozgathatók. Ha ebből kilépünk, akkor azt a kart, amely kiért a

tartományból átállítja automata üzemmódra, és visszaviszi a biztonságos tartományon belülre. Ettől függetlenül a többi kar ugyanúgy mozgatható kézi vezérléssel addig, amíg el nem hagyjuk velük is a definiált területet. Ha egy kar kiért a megengedett mozgástartományból, akkor újra kérni kell a mikrokontrollert, hogy a kart engedje át kézi vezérlésre. Ilyenkor az összes kart kézi üzemmódra állítja. Az alapból kézi üzemmódban működő karokra ez a parancs nincsen hatással.

Mivel a potenciométerek nagyobb szögben képesek elfordulni, mint a robot csukló, ezért fennáll a lehetősége annak, hogy olyan pontot adjunk meg a robotnak, ahova már fizikailag nem képes menni. A mikrovezérlő szoftverének pontosan definiáltam, hogy mi az az intervallum, amiben a robot még képes mozogni. Ha ezen kívül eső pontot adunk meg, akkor a vezérlő nem fogja végrehajtani, annak ellenére, hogy értelmezte a parancsot. Ellentétben a kézi vezérléssel, itt már meg lehet adni bármilyen helyzetet a robot fizikai munkatérén belül, mint célhelyzet.

Sajnos a betanító kar kialakítása nem egyezik a robot valós kialakításával (a betanítónak nagyobb a munkatere, mint a robotnak), ezért itt is a fent említett algoritmust használok annak eldöntésére, hogy a robot képes-e felvenni a helyzetet. Itt is a teljes munkatér kihasználható.

Az USB adatkommunikációnak köszönhetően a rendszer szinte minden számítógépről irányítható. A telepítés ugyanúgy történik, mint ha egy szokványos USB eszközt telepítenénk:

- ◆ csatlakoztatjuk a vezérlőt a számítógéphez,
- ◆ az operációs rendszer, jelzi, hogy új hardvert talált,
- ◆ megmondjuk neki, hogy hol találja hozzá a driver-t, és feltelepíti.

A mikrokontrollerbe feltöltöttem egy *bootloader* programot. Ez kényelmes fejlesztést tesz lehetővé, mivel nincs szükség külső programozó eszközre. Hátránya is van ennek a módszernek: ez csak a program letöltésére használható. A mikrokontroller működését beállító (pl. órajel, memória tiltások, stb.) regiszterekhez ezzel nem férhetünk hozzá, ami nem feltétlenül nagy hátrány, tekintve, hogy az AVR-ek elég kényesek a helyes órajel beállításra (értsd: ha rossz a beállítás, leáll a processzor, és csak trükközve lehet helyre tenni).

A PC-re írt program:

A vezérlővel való kommunikáció kétféleképpen történhet. Az egyik lehetőség az, ha a parancsokat egy terminál programból küldjük a mikrokontrollernek. Ez a fejlesztés során kétségkívül nagyon hasznos, ám egyáltalán nem kényelmes. Nem mellékesen pedig igencsak jó agytorna kb. 19 parancsot fejben tartani, és azokat rutinszerűen használni.

A másik opció az, hogy egy direkt ehhez a vezérlőhöz, s ehhez a robothoz készített szoftvert használjuk. Ezt a grafikus felületű programot azért írtam, hogy megkönnyítsem a robot használatát, ugyanis itt a grafikus felület elfedi a parancsok összetett rendszerét. Ami marad, az egy áttekinthető felület, amelyben elég csak a gombokra kattintgatni a különböző funkciók és üzemmódok eléréséhez. Úgy gondolom, ez egy elég vonzó alternatíva a terminálos parancs-küldözgetés kiváltására.

Jelenleg a szoftver direkt kinematikával dolgozik. Ez azt jelenti, hogy a robot csuklóinak helyzetét adhatjuk meg a programban, és azt kiküldve a mikrovezérlő felveteti a robottal. Ennek ellentéte, az inverz kinematika, amikor a robotot egy Descartes koordinátarendszerbe helyezzük, és a manipulátor kar pozícióját X-Y-Z koordinátákkal, orientációját pedig szöggel adjuk meg. Ez, a tervek szerint egy későbbi fejlesztés témája lesz.

A kezelőfelület kialakítása:



4.8 ábra: Felhasználói program

A **Menüt** lenyitva három funkciót érhetünk el:

- ◆ a motorvezérlő panel tápfeszültségét manuálisan kapcsolhatjuk
- ◆ a robot karjait alaphelyzetbe küldhetjük
- ◆ kiléphetünk a programból.

Az **Irányítási módok** menüben a következő irányítási módok közül választhatunk:

- ◆ **Auto:** ekkor a robot karjait a mikrovezérlő irányítja a PC-ről beérkező helyzetek alapján, melyeket a három fehér színű „Editbox”-ba írhatunk be, majd a **Pozícióba** gombra kattintással küldhetünk el.
- ◆ **Manuális:** Ekkor a főablakban megjelenő gombokkal irányíthatjuk a robot karjait egymástól függetlenül.
- ◆ **Kar követés:** Ebben a módban a betanító kar mozgását követi.
- ◆ **Kar követés vége:** a **Kar követés** módból **Auto** üzemmódba lép.
- ◆ **Végpozíció olvasás:** erre kattintva a mikrovezérlő elküldi a PC-nek azt a helyzetet, amibe a karokat vezérli

Ahogy a képen látszik, minden menü kinyitható menüablakká, ezáltal megkönnyítve a gyakran használt funkciók elérését.

A háttérben látható diagram egy viszonylag új fejlesztés eredménye. Ez a motorvezérlő áramkörök hőmérsékletéről szolgáltat információt, másodpercenkénti frissítéssel. A vízszintes, szürke vonalak 10°C-ként emelkednek. A diagram egyelőre csak tájékoztató funkciót tölt be. (A fenti képen a környezeti hőmérséklet kb. 16°C. Jól megfigyelhető, hogy a kék színű görbéhez tartozó IC hajt meg két motort, míg a másik csak egyet.)

6. Összefoglalás:

Összességében tekintve, a végeredmény egy olyan rendszer lett, amely számos tulajdonságában és paraméterében túltesz az eredeti vezérlésen.

A rendszer előnye, hogy használatához nem szükséges semmiféle programozói tudás, csak a megírt grafikus felületet kell használni, ellenben, ha valaki affinitást érez, a nyílt forráskód miatt a rendszer számos funkcióval bővíthető, – mind a mikrovezérlő, mind a PC tekintetében – így olyan funkciók is megvalósíthatók, melyek e dolgozat írásakor fel sem merültek. Emel a rendszer értékén, hogy nem feltétlenül szükséges külső programozó eszköz a programozásához, hiszen a mikrovezérlő képes a saját memóriáját felforprogramozni.

A hardver tekintetében is van lehetőség egyéni ötletek megvalósítására, hiszen a kivezetett kommunikációs buszra felfűzhetők egyéb perifériák, vagy akár egy másik mikrovezérlő, amely a jelenlegivel együttműködhet.

Hasznos lehet a felsőoktatásban tanuló hallgatók számára, hiszen szemléletesen sajátíthatják el a robotika alapjait. Mivel a kommunikációs protokoll kidolgozott, és a szoftver kötöttségek nélkül módosítható vagy fejleszthető, ezért lehetőség van pl. a MATLAB, vagy egyéb műszaki szoftverek által számított eredmények kipróbálására a gyakorlatban is.

Munkám során vizsgáltam a robot ismétlési pontosságát az új vezérléssel. Méréseim szerint ez 1mm, vagy az alatti.

A közeljövőben megvalósítandó célként tűztem ki a mikrokontroller algoritmusainak fejlesztését, az ismétlési pontosság javítását, új funkciók implementálását, valamint a PC szoftver fejlesztését, elsősorban az inverz kinematika irányába.

Ez a projekt egy jó lehetőség volt arra, hogy megtanuljak egy projektre összességében tekinteni, elvonatkoztatva a részletektől. Sokat fejlődtem az elektronikai tervezés és a nyomtatott áramkörök készítése terén is. Ahhoz, hogy egy számítógépen futó szoftvert írjak, meg kellett tanulnom egy új programnyelvet. Emellett fejlesztettem a C programozási tudásomat is, hiszen a mikrovezérlő programját abban írtam.

7. Irodalomjegyzék:

- [1] Cím: Serpent Robot Review (Serpent Robotok Áttekintése)
Forrás: Practical Electronics
Megjelenés: 1985. Szeptember
Szerzők: Dick Becker, Robert Gay
- [2] Cím: Phase correct PWM mode (Fázishelyes PWM mód)
Forrás: ATmega16 adatlap, 79. oldal
Szerző: Atmel Corporation
- [3] Cím: Conductive Polymer (Vezető Polimer)
Forrás: Wikipedia (http://en.wikipedia.org/wiki/Conductive_polymer)
- [4] Cím: Successive Approximation ADC (Fokozatos megközelítésű ADC)
Forrás: The AVR microcontroller and embedded system, page 478. (Az AVR mikrokontroller és beágyazott rendszer, 478. oldal)
Szerzők: M. Ali Mazidi, Sarmad. Naimi, Sepher Naimi
Megjelenés: 2010. január. 30.
Egyéb: ISBN-10: 0138003319

- 1.1 ábra Cím: SCARA anatómiája
Forrás: Practical Electronics, 39. oldal
Megjelenés: 1985. szeptember
Szerzők: Dick Becker, Robert Gay
- 1.2 ábra Cím: SERPENT szervó-vezérlő sematikus rajza
Forrás: Practical Electronics, 40. oldal
Megjelenés: 1985. szeptember
Szerzők: Dick Becker, Robert Gay
- 1.3 ábra Cím: SERPENT robotok munkatere
Forrás: Practical Electronics, 41. oldal
Megjelenés: 1985. szeptember
Szerzők: Dick Becker, Robert Gay
- 2.1 ábra Cím: AVR CPU felépítése
Forrás: ATmega16 adatlap, 8. oldal
Szerző: Atmel Corporation
- 2.2 ábra Cím: USB-A és DB-9 típusú csatlakozók
Forrás: Árgép.hu (<http://argep.hu>)
- 2.3 ábra Cím: 2.3 ábra: MCP2200 belső felépítése
Forrás: MCP2200 USB to UART Protocol Converter adatlap, 2. oldal
Szerző: Microchip Inc.

- 2.4 ábra Cím: MAX7503 digitális hőmérő blokkdiagramja
 Forrás: MAX7503 adatlap, 10. oldal
 Szerző: Maxim Integrated Products
- 3.1 ábra Cím: ESD kisülés eredménye
 Forrás: http://www.static-sol.com/library/articles/ESD_damage.htm
- 3.2 ábra: Cím: ESD teszt berendezés funkcionális ábrája
 Forrás: United States Military Standard MIL-STD 883H method 3015.8
 (Egyesült Államok 883H katonai szabványának 3015.8 metódusa)
 Megjelenés: 2010. február 26.
- 3.3 ábra: Cím: Áramhurok helyes kialakítása és leválasztása
 Forrás: AVR042 application note, 3. oldal
 Szerző: Atmel Corporation
- 4.1 ábra Cím: Az USB kommunikációs panel kapcsolási rajza
 Szerző: Zólyomi Attila
- 4.2 ábra Cím: Open-Drain-es kimenet
 Szerző: Zólyomi Attila
- 4.3 ábra Cím: A BC337-40 $h_{FE} - I_C$ diagramja
 Forrás: BC337 datasheet (BC337 adatlap), 5. oldal
 Szerző: Discrete Semiconductors
- 4.4 ábra Cím: H-híd elvi áramköri rajza
 Szerző: Zólyomi Attila
- 4.5 ábra Cím: Timer értéke és a kimeneti jel
 Forrás: ATmega16 adatlap, 79. oldal
 Szerző: Atmel Corporation
- 4.6 ábra Cím: Feszültségosztó és potenciométer
 Szerző: Zólyomi Attila
- 4.7 ábra Cím: ADC egység működése
 Forrás: The AVR microcontroller and embedded system, page 478. (Az AVR mikrokontroller és beágyazott rendszer, 478. oldal)
 Szerzők: M. Ali Mazidi, Sarmad. Naimi, Sepher Naimi
 Megjelenés: 2010. január. 30.
 Egyéb: ISBN-10: 0138003319
- 4.8 ábra Cím: Felhasználói program
 Szerző: Zólyomi Attila

IRÁNYÍTÓ EGYSÉG FEJLESZTÉSE SCARA MANIPULÁTORHOZ

Zólyomi Attila

Óbudai Egyetem

Bánki Donát Gépész és Biztonságtechnikai Mérnöki Kar, 4. évfolyam

Dr. Nagy István, adjunktus

Kulcsszavak: mikrovezérlő, teljesítményelektronika, direkt kinematika.

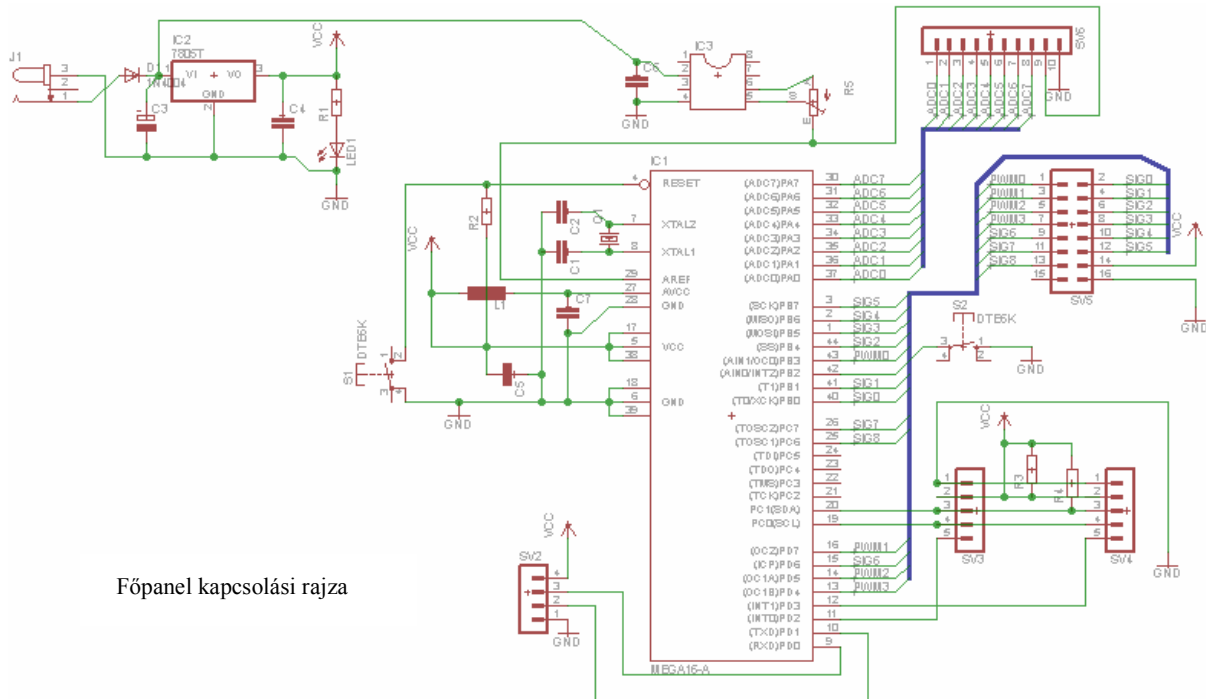
A TDK dolgozatom fő témája egy olyan projekt bemutatása, amelynek fő célja, hogy egy már meglévő robot mechanikáját életre keltsen, és azt alkalmazhatóvá tegye mind a robotika alapjainak elsajátításában, mind pedig a beágyazott rendszerek fejlesztésében.

A vezérlő eredetileg oktatási célra készült, ezért a hozzá készült szoftver kezelőfelülete egyszerű, könnyen áttekinthető. Többféle integrált biztonsági funkcióval rendelkezik, melyeknek köszönhetően elsajátíthatók a robotika alapjai anélkül, hogy a rendszerben kárt tennénk. Forráskódja és dokumentációja nyílt, így bárki szabadon fejleszthet hozzá szoftvert, mind PC-re, mind pedig a vezérlőre.

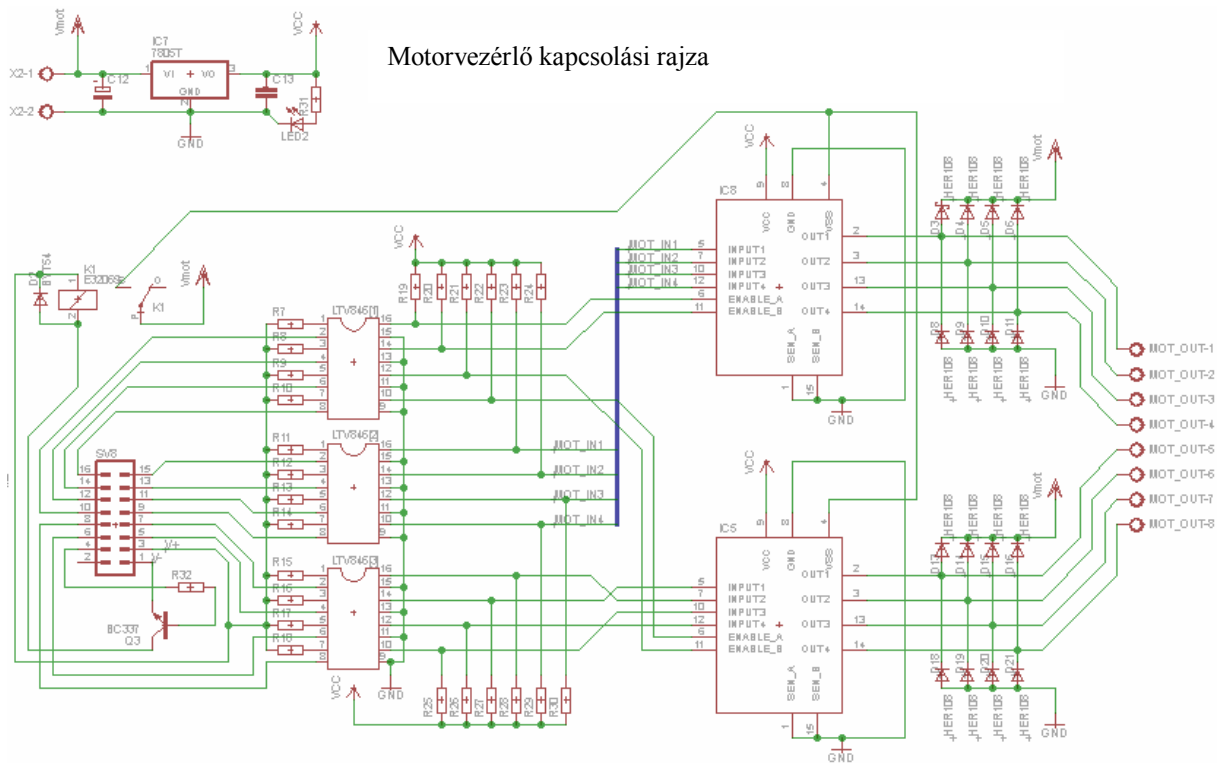
Dolgozatomban először bemutatom az eredeti irányító rendszer sajátosságait, kitérve arra, hogy a tervezők milyen eredeti ötletekkel álltak elő a felmerülő problémák megoldására. Ezután ismertetem azokat a problémákat, amelyeket nekem is figyelembe kellett venni a tervezés során, és bemutatom, hogy a napjaink technológiájával ezeket a problémákat hogyan lehet áthidalni úgy, hogy a végeredmény egy olyan rendszer, mely messze túltesz az eredeti koncepción.

Végezetül elmondok néhány továbbfejlesztési lehetőséget, melyek megvalósítása további fejlődési lehetőséget jelent mind elektronikai, mind informatikai területen.

9. Mellékletek:



Főpanel kapcsolási rajza



Motorvezérlő kapcsolási rajza