Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Szenzorszimulátor tervezése elektronikus motorvezérlő rendszerekhez

Diplomaterv

Gusztáv Tamás
2008

Nyilatkozat

Alulírott, *Gusztáv Tamás*, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, és a diplomatervben csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

.....

Gusztáv Tamás

Kivonat

A modern járművek a környezeti terhelés csökkentése érdekében bonyolult motormenedzsmenttel rendelkeznek, melyek számos szenzor jele alapján, speciális beavatkozók segítségével érik el a jobb hatásfokot, kevesebb káros emisszió mellett.
A meghibásodó nagy bonyolultságú elektronikus alkatrészek a szervizelést, majd a hibás alkatrész analízisét is megnehezítik. A hiba okának felderítése viszont a magas minőségi követelmények teljesítéséhez elengedhetetlen.

Ennek megoldására egy ígéretes lehetőséget jelent, a motorvezérlő elektronikában már amúgy is jelenlévő diagnosztikai funkciók felhasználása, illetve a motorvezérlő köré épített autonóm rendszerben, hosszas szerelés nélkül szimulált járatási tesztek végrehajthatóak. Ehhez arra van szükség, hogy a motorvezérlőhöz csatlakoztassuk a motorban egyébként szükséges beavatkozókat, valós fizikai körülmények hiányában pedig a szenzorok jelét szimulált jelek helyettesítik.

A dolgozat egy olyan rendszer megtervezéséről és megvalósításáról szól, amely képes az AUDI HUNGÁRIA MOTOR Kft. által gyártott R4 PDTDI motormenedzsment szenzorainak szimulációjára, és biztosítja a rendszer kiterjeszthetőségét más motortípusokra is. A dolgozat keretében a fordulatszámjeladók hullámformájának szimulációja, a töltőlevegőnyomás-mérő, légtömegmérő, a hőmérsékletjeladók és az olajszintmérő szimulációja készült el.

A megtervezett rendszer egy alaplapból és a szenzorok szimulációját végző szenzorkártyákból épül fel. A szenzorok kimenete egy PC felhasználói programjából állítható, mely szabványos soros vonalon kommunikál az alaplappal. A szenzorkártyák az alaplappal I^2C buszon keresztül vannak kapcsolatban. Ez a felépítés lehetővé teszi, hogy az új motortípusok különböző típusú szenzorait is további kártyák hozzáadásával szimuláljuk. A megvalósított rendszer a következő kimenettípusokat tartalmazza: analóg feszültségkimenet 0-5 V@0,01 V felbontás, ellenálláskimenet 60 Ω-50 kΩ@~5% pontosság, digitális kimenet a fordulatszám jeladók szimulációjához.

A megépített rendszer segítségével valóban szimulált motorműködés érhető el, mely segítségével valódi beavatkozók vizsgálhatóak. Az elektronika képes a szenzorok jelével közel megegyező tulajdonságú kimenetek előállítására, a szimuláció során a motorvezérlőben hibatári bejegyzés nem történik. A szimuláció lehetővé tesz továbbá olyan beállításokat, melyek valóságban csak nagyon ritkán, vagy csak speciális körülmények között fordulnak elő, így másképpen nehezen előállíthatóak.

Abstract

To reduce environmental pollution modern vehicles are use sophisticated engine control units (ECU) combined with numerous electronic sensors and actuators. The increasing complexity of these parts makes the repairing of the faulty vehicles complicated, and the analysis of the replaced failure parts a real challange. However is it impossible to achieve high quality without finding the root cause of the malfunction.

For a better analysis it is a promising possibility to use the integrated on-board-diagnostic of the ECU in addition to the common test equipments to reproduce and recognize the fault, without the disadvantages and high cost of a real engine test. To simulate the engine behavior the simulatio of the sensor outputs for the engine control unit is necessary, like in a „Hardware-in-the-loop" tester used for ECU development, but the desired performace is available by an essential simplified system, because the ECU is only needed to operate the real actuators according to the sensors.

This diploma work is about the design and realization of a system, which is able to simulate the sensor outputs for an electronic engine control system of the motortype R4 PDTDI. This simulator is meant to be a prototype that serves as a universal basis of future applications with other motortypes of AUDI HUNGARY LTD. For that very reason the main requirements are flexibility and expandability. The simulation of the following sensors are involved: mainfold pressure, mass air flow, cam and crankshaft sensors, temperature sensors, oil level sensor.

The designed system consists of a motherboard and a simulator card for each sensor. The desired output value data of the simulated sensors determined by the user is sent over RS232 port of a PC to the mainboard. The connection between the cards and the mainboard is provided by an I^2C bus. This system architecture enables the use of a different sensor set for each motortype. The mainboard and the cards are controlled by Atmel AVR microcontrollers. The realized output types: analog voltage output with feedback 0-5 V@0,01 V resolution, resistance output 60 Ω-50 kΩ@~5% accuracy, digital output for cam and crankshaft sensors.

The designed system is able to simulate motor starts to test real actuators, without getting storage of failure codes for the simulated sensors.

1. Bevezetés[1],[2],[3]

Napjainkban egyre növekvő szerepet kap a környezetszennyezéssel szembeni védekezés, melynek oka elsősorban következményeinek fokozódó megjelenése. A folyamatosan növekedő népességgel a közlekedés környezetre gyakorolt növekvő negatív hatása jelenti az egyik legsúlyosabb problémát, így ezen a területen is sürgős beavatkozásokra van szükség. Az első rendelet, mely a gépjárművek károsanyag-kibocsátását korlátozta, a 60-as évek végén az USA-ban született. Azóta világszerte számos, egyre szigorodó intézkedés látott napvilágot. Ezek a normák azonban nem teljesíthetőek a hagyományos karburátoros gépjárművekkel, szükség van a motorok belső folyamatainak pontosabb szabályzására, hogy jobb hatásfokkal, és kisebb szennyezőanyag-emisszióval történjen meg az égés. Ehhez fejlett szenzorokra van szükség, melyek menet közben is tájékoztatást adnak a motor üzemi állapotáról, illetve megfelelő beavatkozókra, melyek az egyes folyamatokba az üzemállapotnak megfelelően beavatkozhatnak. Az egyes mennyiségek összefüggései miatt azonban a működés összetett, a szabályzást egy központi egységnek kell végezni a lehető legtöbb információ figyelembevételével. Ez vezetett az elektronikus motormenedzsment kialakulásához. A motorvezérlő egység (ECU) gondoskodik az optimális működésről, melyet a befecskendezés, gyújtás, kipufogógáz-visszavezetés, és különböző szelepek összehangolt vezérlésével ér el. A szenzorok és beavatkozók a szigorodó normák következtében egyre nagyobb számban vannak jelen a motorban, és egyre összetettebbek. A beavatkozók egyre precízebben, nagyobb felbontással működtethetőek, a kezdeti vákuumszelepes megoldásokat felváltják a motoros, beépített szabályzóval rendelkező intelligens szelepek. A fejlődés ma már ott tart, hogy a szervizek a meghibásodásokat csak speciális műszerek segítségével tudják felderíteni, illetve elektronikus hibáknál ezt a feladatot egyre inkább a beépített öndiagnosztika látja el. A meghibásodó elektronikus alkatrészek javítása nem lehetséges, a szerviz ezeket kicseréli, a hiba okát általában csak részletesebb analízis tárja fel.

Az AUDI Hungária Motor Kft. győri gyárában készített motorokra felépített, majd később a működés során meghibásodott szenzorok és beavatkozók vizsgálata a m.r. feladata. A vizsgálatok célja a hiba okának pontos meghatározása, mely a beszállítóval közösen történik, és melynek eredményeképpen az alkatrészek

megbízhatóságának növelését hozó intézkedések születnek. Az alkatrészek összetett felépítése miatt a hiba viszont csak triviális esetekben határozható meg rövid idő alatt. Gyakran sporadikus jelenségről van szó, amely csak bizonyos körülmények között, meghatározott idő után lép föl. Ilyenkor az analízis hosszú időt vesz igénybe, és igen költséges is lehet. A működési rendellenesség reprodukálásához, mely nélkül a hibaok feltárása esélytelen lehet, gyakran hosszú járatási tesztek vezetnek.

Ehhez hasonló esetekben nagy segítséget nyújtana egy olyan rendszer, amely a járatási próbák szimulált elvégzéséhez biztosítana környezetet. Itt tetszőleges üzemi körülmények beállíthatóak lennének, ám a vizsgálat sokkal kisebb költséggel lenne megoldható. Egy ilyen teszt célja nemcsak az adott alkatrész teljes működési tartományban történő vezérlése, illetve szenzorok esetén kimenetének ellenőrzése, hanem az, hogy az adott alkatrészt a motorvezérlő elektronika a beépített öndiagnosztikai rendszerével saját maga is ellenőrizze. Ez azért fontos, mert gyakran az adott alkatrész és a motorvezérlő együttes nem megfelelő működése hatására jelenik meg a hibajelzés a műszerfalon, ami aztán a szervizben a diagnosztikai műszer segítségével a hibakód kiolvasása után a megfelelő alkatrész cseréjéhez vezet.

Egy erre alkalmas tesztkörnyezet megvalósításához a motorvezérlő egységhez csatlakoztatni kell az általa vezérelt beavatkozókat, illetve a tényleges működés szimulálásához, valós fizikai körülmények hiányában, szükség van a szenzorok működés közbeni jeleinek külső előállítására. Egy, a motorvezérlő fejlesztéséhez szükséges „Hardware-in-the-Loop” teszterhez hasonló komplexitású szimuláció elkészítése azonban nem szükséges, egyszerűbb, lassabb eszközökkel is elérhető a kívánt funkció: a motor szimulált elindítása, illetve olyan üzemállapotok elérése, melyben a vezérlő a csatlakozó beavatkozókat valós körülményekhez hasonlóan működteti. Közben a diagnosztikai interfészen megfigyelhetők az esetlegesen fellépő hibaüzenetek, illetve egyéb műszerekkel megfelelő csatlakoztatása is lehetővé válik további paraméterek megfigyelésére. A szenzorok vizsgálatára is lehetőség nyílik ily módon, amennyiben az adott szenzor megfelelő gerjesztését valamilyen eszköz biztosítja, a többi szenzort pedig továbbra is helyettesítheti a szimulátorelektronika.

A diplomaterv keretében megvalósítandó feladat az AUDI Hungária Motor Kft. m.r.-en készülő, ilyen elven működő vizsgálóberendezésének szenzorszimulációt végző elektronikájának megtervezése és elkészítése. Jelenleg a vizsgálóberendezés egy

konkrét motortípushoz készül, egy adagolóporlasztós befecskendező rendszerrel rendelkező PDTDI diesel motorhoz. Ez egy elterjedt típus, hosszú ideig nagy darabszámban készült, mára már kezdi kiszorítani a közös nyomócsőves változata, de az egyszerűbb motormenedzsmentje jó kiindulási alapot szolgáltat a további motortípusokra történő kiterjesztéshez.

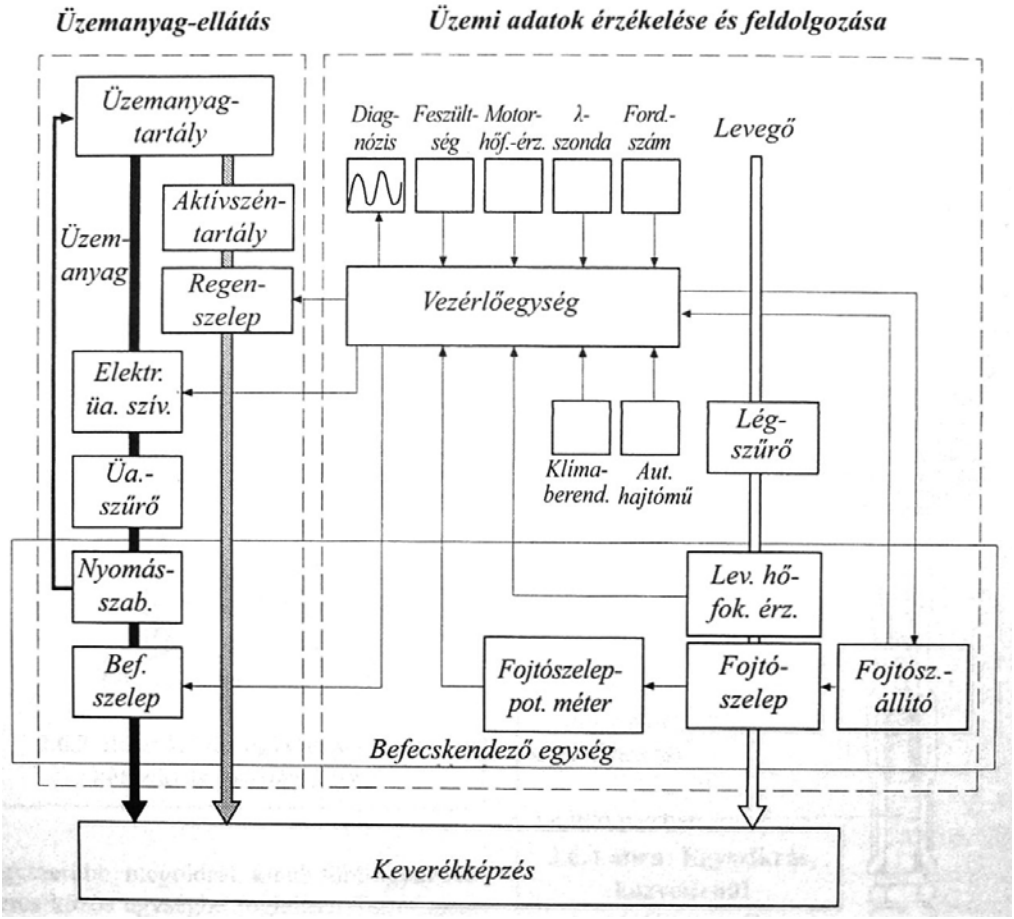
1.1. Motormenedzsment rövid áttekintése[1],[2]

A központi vezérlőegység kialakulásának alapvető oka tehát az alacsonyabb fogyasztás mellett nagyobb teljesítmény, azaz jobb hatásfok elérése, illetve a környezetvédelmi normák betartásához szükséges szennyezőanyag-emisszió csökkentése. Ennek eszköze az első motorvezérlő rendszereknél érthető módon a keverékképzés optimalizálása volt. Ez egy jelleggörbe alapján meghatározott mennyiségű üzemanyag befecskendezését jelenti, melyhez a bemeneti paraméterek elsősorban a fordulatszám, illetve a terhelés. A vezérlő feladata itt a fordulatszámjeladó, illetve a légmennyiségmérő jelei alapján meghatározott terhelési állapot számítása, és az ehhez tartozó üzemanyag-mennyiség befecskendezése, ami az üzemanyagnyomás, illetve az akkumulátorfeszültség ismeretében a befecskendezőszelep meghatározott idejű nyitva tartásával oldható meg. Emellett hamar megjelent az elektronikus gyújtásvezérlés is, ami a precíz fordulatszámfüggő gyújtási-szög szabályzással még hatékonyabbá teszi az égési folyamatot.

Az ilyen optimalizálással elért fajlagos terhelésnövekedés, illetve az üzemanyag-fogyasztás jelentős csökkenése által okozott égési csúcshőmérséklet-növekedés azonban lehetővé tette a nitrogénoxidok képződését az égéstermékek között, ami további szabályzásokat tett szükségessé, ami újabb szenzorokkal és beavatkozókkal valósítható meg. Így ahogy a motorvezérlők számítási teljesítménye megnőtt, a katalizátorral együtt megjelent a lambda-szonda, amely az optimális keverési arány pontos beállításához nyújt visszacsatolást, illetve a kipufugógáz-visszavezető szelep, mely megfelelő vezérlés esetén akár 60%-kal csökkenti a nitrogénoxidok kibocsátását az égési hőmérséklet csökkentésével. Elterjedőben van továbbá a beáramló levegő perdületét szabályzó szelep, amely a homogén keverékképzést segíti elő.

Mára ez a tendencia addig fejlődött, hogy egy korszerűbb autó motorjának megfelelő vezérléséről akár harmincnál több szenzor jele alapján hasonló számú beavatkozó gondoskodik, melyek összehangolása a motorvezérlő feladata, ami az

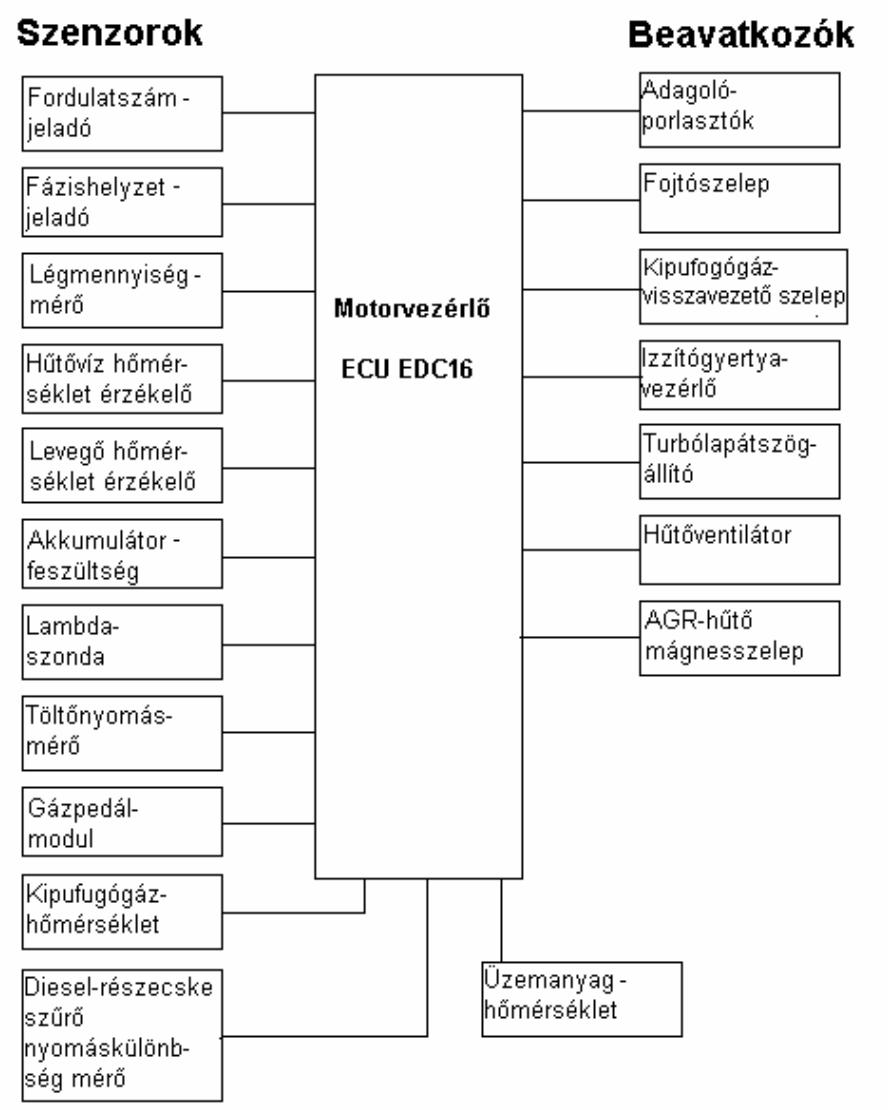
autóban jelenlévő többi vezérlőegységgel (ABS-fék vezérlő, váltóvezérlő, légzsák-vezérlő) is folyamatos kapcsolatban áll. A motormenedzsment általában ezt a komplex rendszert jelenti. A belső kapcsolati rendszer általános felépítését az alábbi ábra szemlélteti:



1.1. ábra. Befecskendezés-vezérlés kapcsolati rendszere [1]

1.2. Az R4 PDTDI motormenedzsment felépítése[4]

Kezdetben a vizsgálóberendezés a R4 PDTDI diesel motort működtető Bosch Motronic EDC16 motorvezérlő felhasználásával készül, így az ezeken a motortípusokon megtalálható szenzorok szimulációjának elkészítésére van szükség, melyek az alábbi ábrán láthatóak:



1.2. ábra. Az R4 PDTDI motormenedzsment felépítése

A szimuláció szempontjából a szenzorok rendelkeznek kitüntetett jelentőséggel, így azokat célszerű röviden bemutatni:

Fordulatszám-jeladó: a motor főtengelyén található mágnesezett tárcsa közelében elhelyezkedő Hall-jeladó, amely a fordulatszám pontos számításához szükséges jelet szolgáltatja.
Fázishelyzet-jeladó: a vezérműtengelyen található kódtárcsa közelében elhelyezett Hall-jeladó, amely a tengely pontos szöghelyzetéről ad információt.
Légmennyiség-mérő: A szívócsőben elhelyezkedő szenzor, mely a szívott levegő mennyiségét méri, mely arányos a terheléssel.
Hőmérséklet-érzékelők: A motor különböző közegeinek hőmérséklete az üzemállapotról ad tájékoztatást (hűtővíz, kipufugógázhőmérséklet-szenzor), vagy egy másik számított vagy mért mennyiségnél ad korrekciós tényezőt (levegő, üzemanyag-hőmérséklet-szenzor)
Lambda-szonda: kipufugógáz oxigénarány-mérő. A keverékképzésről ad visszacsatolást.
Töltőlevegőnyomás-mérő: a szívócsőnyomást méri, mely a terhelés meghatározásához és a turbólapátszög szabályzáshoz szükséges.
Gázpedál-modul: A gázpedál tulajdonképpen egy potenciométer, mely a motorvezérlővel van elektromos kapcsolatban.
Diesel részecskeszűrő (DPF) nyomáskülönbség-mérő: Ha a DPF a kipufogórendszerben kezd telítődni, a bemenete és kimenete közötti kipufogógáz-nyomáskülönbség nőni kezd. Egy bizonyos érték fölött megkezdődik a regenerálás.

2. A motormenedzsment egyes összetevőinek tesztelése

Ahogy a bevezetésben már elhangzott, az egyes meghibásodott alkatrészek tesztelése, a hiba okának mihamarabbi feltárása különös jelentőséggel bír. Az egyszerűbb alkatrészek, pl. hőmérséklet-jeladók működésének vizsgálata természetesen nem jelent problémát, de a bonyolultabb, akár saját hibatárral rendelkező beavatkozók vizsgálata már nagyobb körültekintést igényel. Ehhez számos vizsgálati eszköz áll rendelkezésre,

melyeket pl. az alkatrész beszállítója bocsát rendelkezésre, vagy belső fejlesztés eredményei, ezek a berendezések azonban pl. egy beavatkozó esetén elsősorban annak alapvető funkcióit, működtethetőségét ellenőrzik, sporadikus jellegű, rövid ideig fennálló hibát nem tudnak jelezni, melyre viszont adott esetben a motorvezérlő hibajelzést ad.

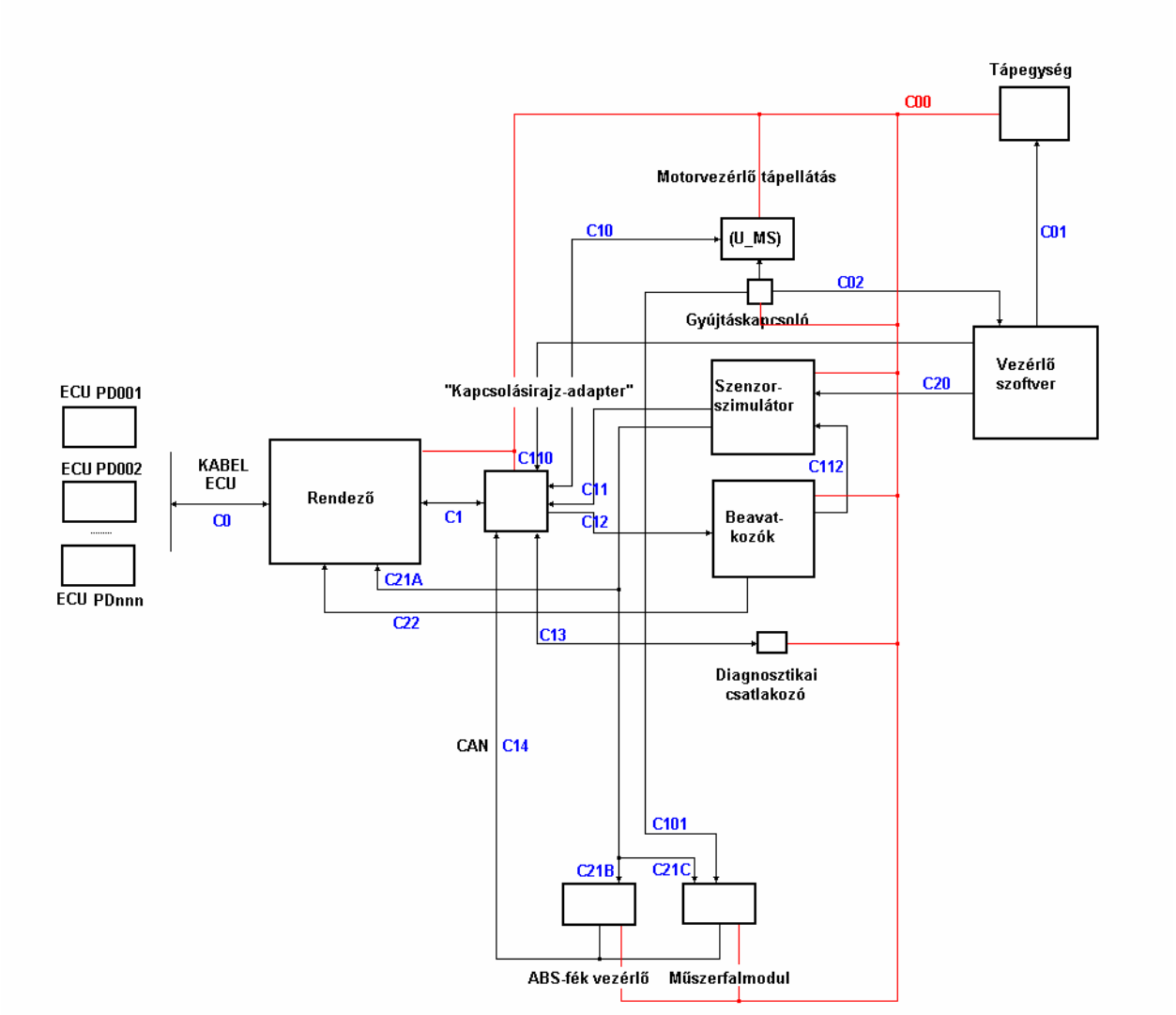
2.1. Szenzorok és aktuátorok tesztelése a motorvezérlő fedélzeti diagnosztikai rendszerének kihasználásával[3]

A gépjárművek emissziós normáinak lefektetése, és az első ezeket teljesítő motorvezérlők megjelenése után hamar felismerték azt a tényt, hogy ha a motormenedzsmentben valamilyen meghibásodás történik, az nem jelent feltétlenül fogyasztásnövekedést, vagy szolgál valamilyen, az üzemeltető által észrevehető tünettel, viszont megnövekedhet az adott gépjármű károsanyag-kibocsátása. Egy esetleges ellenőrzés során ez nem róható fel az üzembentartónak. Ennek megoldására kötelezővé tették a motorvezérlőben egy olyan beépített, folyamatosan működő diagnosztikai rendszer integrálását, amely képes a motorvezérlővel kapcsolatban lévő összes szenzor és beavatkozó rendellenes működésének felismerésére. (Kezdetben különféle fedélzeti diagnosztikai rendszerek, majd az egységes OBD II. szabvány: SAE J 1850, ISO 9141-2, ISO 15 031-3). A motorvezérlő a műszerfalon jelzi a hiba megjelenését, a pontos körülmények és a hiba típusa a diagnosztikai interfészen keresztül olvasható ki.

Természetesen a vezérlő a hiba okáról nem szolgálhat részletes információval, de a cél nem is ez, hanem sokkal inkább az, hogy a szimulált járatás során, melyet akár klímakamrás és rázópados vizsgálat is kiegészíthet, jelezze a hiba megjelenését, illetve a tünet típusát. Mindezt akár olyan üzemi tartományban, amelyet a motor egyébként nem viselne el tartósan egy valós járatás során, de az adott alkatrész itt kap jelentősebb szerepet.

3. A megvalósítandó tesztkörnyezet rendszerterve[5]

A megvalósítandó tesztkörnyezet felépítése a motorvezérlő rendszer járműbeli felépítését követi, a más motortípusokkal szemben megkívánt kompatibilitás miatt azonban néhány egyéb modul is helyet kapott:



3.1. ábra. A tesztkörnyezet rendszerterve

A modulok funkciója az ábra alapján balról jobbra haladva:

Az R4 PDTDI motortípuson belül több altípus is megtalálható, melyek részben különböző motorvezérléssel rendelkeznek. A cél, hogy a rendszerhez bármelyik típusú vezérlőegység csatlakoztatható legyen (ECU PDmn). A motorvezérlők egy kábelkorbácson keresztül csatlakoznak egy rendezőhöz, amelyen az egyes jelek megszakíthatóan ki vannak vezetve, ezáltal lehetővé téve bármely csatlakozó alkatrész áramának illetve feszültségének mérését. A „kapcsolási rajz" adapter feladata, hogy a rendszerben jelenlévő, a PDTDI motortípusokon előforduló összes beavatkozó és szenzor közül az adott motorvezérlőnek megfelelőket kapcsolja a vezérlő egységhez. Ez az adapter van kapcsolatban tehát a beavatkozókkal, a műszerfalmodullal, az ABS vezérlővel, és a szenzorokat szimuláló elektronika kimeneteivel. A szimulátor-elektronika megfelelő vezérlése egy PC-n futó felhasználói program segítségével történik, mely a központi tápegység vezérléséről is gondoskodik.

4. A szimulálandó szenzorok bemutatása, a hardver és szoftver terv elkészítése

A következőkben a szenzorok szimulációjához szükséges rendszerfelépítés bemutatása következik, amit az ezt alkotó alegységek leírása követ. A szimulációt megvalósító modulok leírása a következőképpen épül fel:

- kimenet típusa
- érintett szenzortípusok részletesebb bemutatása
- szimulációs áramkör terve

4.1. A rendszerfelépítés kidolgozása

A szenzorszimulátor elektronika megtervezésének első lépése a rendszer alapvető felépítésének kidolgozása, a feladat részfeladatokra, modulokra bontása, illetve az egyes modulok közti kapcsolatok meghatározása. A hibalehetőségek elkerülése érdekében két fontos szempont szem előtt tartása célszerű, melyek a tervezés további szakaszaira is vonatkoznak:

- A szimulátorral szemben megkövetelt jelenlegi és jövőbeni ismert igények minél pontosabb felmérése;

- Tartalékok biztosítása, persze ésszerű keretek között, melyek a menet közben felvetődő újabb funkciók megvalósítását teszik lehetővé, vagy egy adott részfeladat megoldásánál jelentkező nehézségek leküzdésére adnak alternatív lehetőséget.

Elsőként lássuk az ismert rendszertervet befolyásoló igényeket:

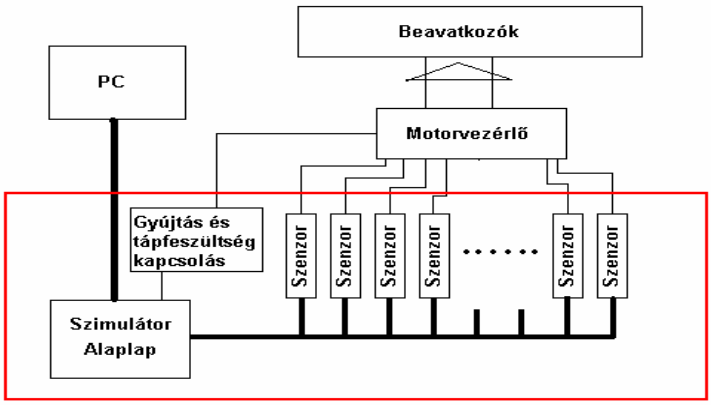
-A feladat egy rendszer elkészítése, mely PC-től kapott parancsok alapján, a motoron található szenzorok kimenetén üzemi körülmények között megjelenő elektromos jeleket szimulálja. A rendszer legyen bővíthető új szenzortípusokkal a további motortípusokkal való kompatibilitás biztosítása végett.

A felépítés szempontjából a bővíthetőség a legfontosabb meghatározó tényező. Ennek biztosítására többféle lehetőség is kínálkozik.

Egy korlátozott, de egyszerű megoldást jelent, ha a rendszert felépítő modulok számával típusonként valamekkora redundanciát hozunk létre. Ilyenkor maga a rendszer nem bővíthető, de egy kívánt új funkció, amennyiben az illeszkedik valamelyik redundáns modul funkciójához, az adott modullal megvalósítható. Ennek előnye, hogy egyetlen központi egység szükséges, mely az összes funkciót ellátja. Hátránya viszont, hogy egy előre nem ismert típusú funkció nem lesz implementálható, illetve egy adott funkció változtatásánál akár az egész rendszerbe bele kell nyúlni. (Egy szoftver felelős az összes funkcióért.)

Ennél sokkal rugalmasabb, épp emiatt azonban munkaigényesebb megoldást jelent a busz architektúra. Itt az egyes modulok egy buszon keresztül kapcsolódnak a központi egységhez, mely a PC-től kapott parancsokat a megfelelő egységhez továbbítja. A bővítés itt egyszerűen egy újabb, tetszőleges felépítésű modul hozzáadását jelenti a buszhoz. Előnye, hogy szinte korlátlan (a busz paraméterei által korlátozott) mértékben bővíthető, és a bővítés nem teszi szükségessé más egység programjának módosítását. Hátránya, hogy a PC és a rendszer közötti kapcsolaton túl a rendszeren belüli kommunikációt is meg kell oldani.

Mivel a most készülő szimulátor egy számos további, már meglévő, és ezután gyártásba kerülő motortípusra kibővítendő rendszer prototípusa, így mindenképpen számolni kell új típusú szenzorokkal. Emiatt a megvalósításhoz a busz architektúrát választottam. Ez a következő ábrán vázolt felépítést jelenti:



4.1. ábra Szimulátorelektronika rendszerterve

A pirossal keretezett rész jelenti a megvalósítani kívánt egységet. A PC-n futó szoftver (nem része a diplomamunkának) csak a szimulátorral van közvetlen kapcsolatban. A motorvezérlő felé a tápellátás illetve a gyújtáskapcsolás szintén a szimulátor feladata. A szimulátor két fő részre bontható, egy alaplapra és a szenzormodulokra. Az alaplap feladata a PC és a modulok közötti kapcsolat megteremtése, illetve a modulok tápellátásának biztosítása. A PC felől érkező parancsok a buszra kerülnek, ahol a parancs címrészében szereplő szenzor-modul végrehajtja azokat. Minden modul azonos felületen csatlakozik a buszra, mely a tápellátását is biztosítja. Így egy új típusú szenzor csak egy újabb modul csatlakoztatását jelenti a buszhoz.

4.2.1. Az alaplap megvalósítása

4.2.1.1. Alaplap rendszerterve

Az alaplap az előbbiek alapján a következő feladatokat kell, hogy ellássa:

A, a hardverrel szemben támasztott követelmények:

- kommunikációs felület biztosítása PC és a szenzorkártyák között;
- a szenzorkártyák tápellátásának biztosítása;
- tápellátás illetve a gyújtáskapcsolás megoldása a motorvezérlő felé;
- bővíthetőség biztosítása további alaplapok csatlakoztatási; lehetőségének biztosításával.

B, a szoftverben megvalósított funkciók:

- kétirányú kommunikáció a PC és a kártyák között;
- kártyák közötti kommunikáció lehetőségének biztosítása;
- hibajelzés, buszra kapcsolódó modulok lekérdezésének lehetősége.

Az alaplap funkcióinak megvalósításához célszerű valamilyen mikrokontroller alkalmazása, mellyel egy jóval rugalmasabb konfiguráció alakítható ki, mint egy diszkrét célintegrált-áramkörökkel megvalósított buszillesztővel. A korábbi

tapasztalatok alapján az Atmel márkájú AtMega16 típusú mikrokontroller alkalmas a feladatra. A kontroller a következő főbb jellemzőkkel rendelkezik:

- 8 bit RISC architektúra
- 1 MIPS / MHz (maximum 20MHz)
- 1 kbyte SRAM
- 16 kbyte Flash programmemória
- 512 byte beépített EEPROM
- HW-szorzó
- számos beépített periféria:
  - 10 bit ADC
  - soros port
  - I^2C interfész
  - 3 timer modul PWM kimenettel
  - Analóg komparátor.

A választást a feladathoz jól illeszkedő paramétereken és perifériakészleten kívül a kedvező ár és a nyílt forráskódú fejlesztői környezet is indokolja, mely C fordítót is tartalmaz. A felprogramozáshoz szükséges áramkör kapcsolási rajza és meghajtó program szintén ingyenesen hozzáférhető.

Ezen kontroller választásával további tartalékok is biztosíthatóak a rendszernek, hiszen ugyanilyen tokozásban teljes láb- és szoftverkompatibilitással további két chip is rendelkezésre áll nagyobb, 32 kbájt illetve 64 kbájt programmemóriával.

A, kommunikációs felület biztosítása PC és a szenzorkártyák között:

A PC és az alaplap közötti kommunikáció megoldására többféle lehetőség is kínálkozik. A PC felől megközelítve a problémát a *párhuzamos port, soros port,* illetve az *USB port* tűnik a legegyszerűbb megoldásnak. Mivel a mikrokontroller USB-n keresztüli kommunikációra közvetlenül nincs felkészítve, és az USB kezelés a PC oldali szoftverfejlesztést is bonyolultabbá tenné, így a hagyományos soros és párhuzamos portok alkalmazása mellett döntöttünk. Ez azt jelenti, hogy az elsődleges kommunikációs port a soros port, de mint tartalék erőforrás, az alaplap hardverszinten alkalmas párhuzamos portról érkező üzenetek fogadására is.

Az alaplap és a szenzorkártyák között a kommunikáció a bővíthetőség miatt tehát valamilyen buszon keresztül kell, hogy történjen. Erre a célra az Inter-IC busz, azaz az I^2C busz választása tűnt legmegfelelőbbnek, a következő megfontolások alapján:

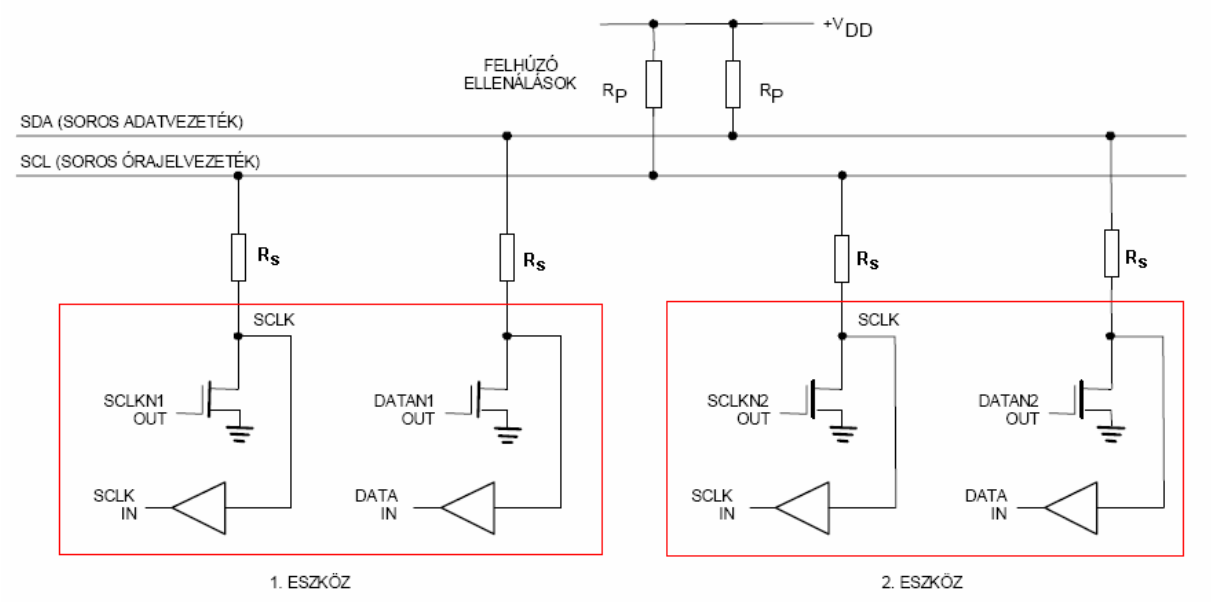
- A PC és a szenzorkártyák közötti kommunikáció időkritikus része a szenzorok kimeneteinek megváltoztatására küldött parancsot jelenti. Egy motortípusnál maximum kb. 15-20 szenzorral kell számolni. Egy parancs maximum kb. 7 byte lehet. Ha feltételezzük, hogy minden szenzornak másodpercenként maximum 40 parancsot küldünk, ami akkor is elégnek tűnik, hogy ha nem vesszük figyelembe, hogy bizonyos kártyáknál elvárás, hogy ugrás alapjel hatására a kimenetüket fokozatosan állítsák át, akkor ez $f = 40 \cdot 20 \cdot 7 \cdot 8 = 44800$ bps átviteli sebességet jelent. Az Atmel mikrokontrollerek által támogatott I^2C protokoll maximális sebessége 400 kbps.
- A párhuzamos adatbusz sok jelvezetéket jelentene, ami a szenzorkártyákon több I/O lábú mikrokontroller alkalmazását tenné szükségessé, így a soros busz jóval gazdaságosabb megoldást jelenthet.
- Az Atmel mikrokontrollerek perifériái között megtalálható az I^2C vezérlő is, nincs szükség külön illesztőáramkörre.

Az I^2C busz bemutatása[6],[7]:

Az I^2C busz egy tipikusan mikrokontrolleres alkalmazásokra kifejlesztett soros busz, mely a következő főbb jellemzőkkel rendelkezik:

- három buszvezeték: soros adat vonal, órajel, földvezeték
- 7 bites címtartomány
- több masteres üzemmód ütközésdetektálással
- kétirányú adatforgalom, 100/400 kbps sebesség
- nincs szükség külön buszinterfészre, az IC tartalmazza azt

Az eszközök a buszra az alábbi ábrán látható módon csatlakoznak:



4.2. ábra Eszközök csatlakoztatása az I2C buszra

A egyes eszközök a busz felé open-drain kimenettel rendelkeznek, amely egy huzalozott ÉS kapcsolatot valósit meg, mely egyszerű ütközésdetektálást tesz lehetővé. Az R_s soros ellenállások az eszközök opcionális túlfeszültség-védelemét szolgálják a buszon esetlegesen kialakuló tüskékkel szemben, az Atmega mikrokontrollerek I^2C interfésze azonban tartalmaz beépített védelmet, így ezekre az ellenállásokra nincs szükség.

Az open-drain kimenet miatt viszont szükség van a buszvonalak „tápfeszültségre húzására”, amiért az R_p ellenállások felelősek.

Az I^2C protokoll a kommunikáció résztvevőinek megnevezésére az alábbi terminológiát használja:

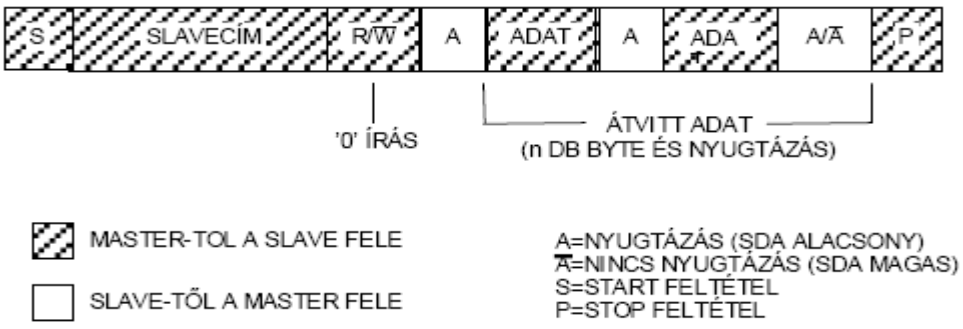
Küldő (Transmitter)	Az az eszköz, amelyik adatokat küld a buszra
Vevő (Receiver)	Az az eszköz, amelyik adatokat fogad a buszról
Master	Az az eszköz, amelyik kezdeményezi az átvitelt, generálja az órajelet és befejezi az átvitelt
Slave	A master által megcímzett eszköz

A protokoll rövid bemutatása:

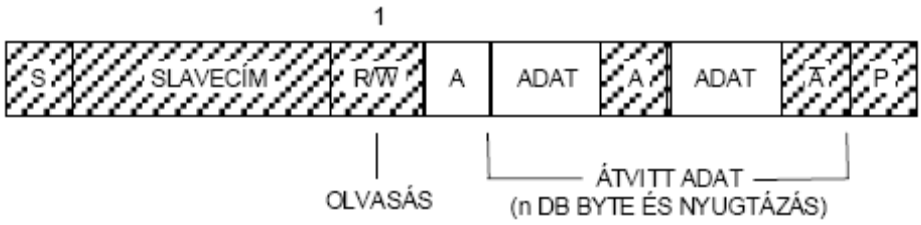
Az átvitelt mindig a master kezdeményezi. Egy buszra több master is kapcsolódhat, melyek egymástól függetlenül bármikor kezdeményezhetnek átvitelt, amikor a busz nem foglalt, de az átvitelvezérlés mindig csak egy résztvevő, a kezdeményező dolga. A többi esetleg jelenlévő master ilyenkor slave-ként viselkedik, és szintén címezhetővé válik. Ha két master egyszerre kezdeményez átvitelt, akkor arbitrációra van szükség. Ez a következőképpen történik:

Mindkét master megkezdi az átvitelt, de ezzel egyidejűleg az adatvonal szintjét is figyeli. Hogyha nem az szint van a buszon, amit küldött, ami a felépítésből adódóan azt jelenti, hogy egy „B" master lehúzta az adatvonalat, miközben az „A" magas szintet küldött, akkor a „B", azaz az alacsony szintet küldő master nyeri meg az átvitel jogát, és az „A" pedig slave-ként működik tovább, és már ebben a ciklusban megcímezhető „B" által.

A kétirányú forgalom megvalósításához azonban nem követelmény több master jelenléte:
A master kezdeményezhet írást és olvasást is, azaz a slave is lehet transmitter. Ilyenkor a master által megcímzett slave teszi az adatot az adatvonalra a master által szolgáltatott órajellel szinkronizálva.
A kommunikáció pontos menetét a következő ábra szemlélteti:



4.3. ábra Küldés protokollja az I^2C buszon



4.4. ábra Olvasás protokollja az I^2C buszon

Az átvitel tehát egy start feltétellel kezdődik, amit a megszólítani kívánt slave 7 bites címe követ, majd a 8. bit dönti el, hogy írás vagy olvasás fog-e történni. Minden bájtot egy nyugtázás követ, mely mindig az adat fogadójának feladata. A nyugtázás elmaradása nem jelent feltétlenül hibát, de mindenképpen a folyamatban lévő átvitel végét, a STOP feltétel elküldését hozza létre. Nyugtázás elmaradhat, ha

- nincs ilyen az adott címmel rendelkező slave a buszon;
- a megcímzett slave nem kész a kommunikációra.

Ha egy slave-nek valamely két átvitt byte között több időre van szüksége, de valamiért nem akarja befejezni az átvitelt (például, mert mint slave nem fog tudni maga kezdeményezni egy újabbat) akkor lehetősége van a master várakoztatására, úgy, hogy a nyugtázás után lehúzza az órajelvonalat. A master a következő byte átvitelét csak azután fogja kezdeményezni, miután az órajel meghajtásának jogát visszakapta.

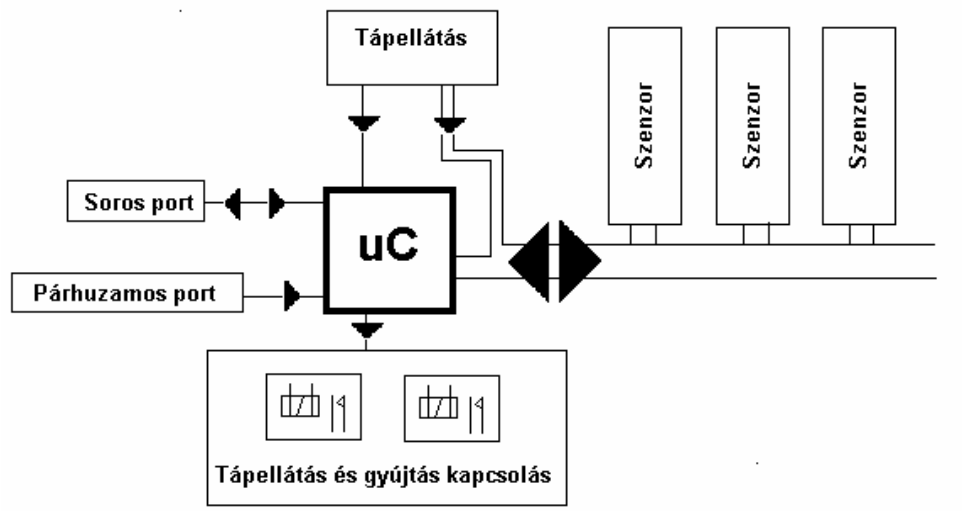
B, A szenzorkártyák tápellátásának biztosítása

Az alaplap, a motorvezérlő és a beavatkozók tápellátása egyetlen, közös tápegységről történik. Ennek kimenetén az autó üzemi feszültségtartományán belül, kb. 8-16 V, bármilyen érték beállítható, és ezen tartományon belül a szenzorszimulátornak is üzemelnie kell. Az alaplap biztosítja a szenzorkártyák tápellátását is

C, Tápellátás illetve a gyújtáskapcsolás megoldása a motorvezérlő felé

Mivel a szenzorszimulátor a tápellátást közvetlenül a központi tápegységtől kapja, annak bekapcsolása után a szimulátor kész fogadni a parancsokat a PC-től. A motorvezérlő felé a tápfeszültség és a gyújtás kapcsolását szintén a felhasználói programból kell megoldani. Ez a feladat logikailag elkülönül a szenzorok szimulációjától, így nem egy, a buszra csatlakozó modul, hanem az alaplap központi egysége közvetlenül látja el ezt a feladatot a hozzá illesztett jelfogók segítségével.

Az alaplap rendszerterve így a következőképpen alakult:



4.5. ábra Alaplap rendszerterve

4.2.1.2. Alaplap kapcsolási rajza

Az alaplap kapcsolási rajza a rendszertervet alkotó modulokra bontható fel, és a kapcsolás ismertetése is modulonként történik:

A, tápellátás:

A tápellátásnak egyszerűen a mikrokontroller tápfeszültségét kell biztosítania, ami a központi tápegységtől érkező 8-16 V közötti feszültségből stabil 5 V tápfeszültség előállítását jelenti. Ennek legegyszerűbb módja készen kapható feszültségstabilizátor IC alkalmazása. Az IC típusa az áramfelvétel várható értéke alapján határozható meg, melynek nagysága az egyes alkalmazott alkatrészek adatlapjain található átlagos fogyasztás figyelembevételével becsülhető. Az áramfelvételtől függetlenül azonban célszerű az alaplapon lévő központi egységet, és a szenzorkártyákat külön feszültségstabilizátorral ellátni, hogy a kártyáknál esetlegesen fellépő problémák ne

okozhassák a teljes rendszer leállását. Ezt az elvet követve egy alaplapon a tápfeszültség- ellátást három stabilizátor fokozat végzi, egy a központi egység, kettő pedig az egy alaplapra csatlakoztatható 10 kártya egyik illetve másik felének 5 V-os feszültségét adja.

A, központi egység az Atmega16 mikrokontrollerrel:

A mikrokontroller tápellátása a VCC lábon keresztül történik, a helyes működés feltétele, hogy 4,5 V< V_{cc} <5,5 V! A kontroller a tápfeszültség stabilizálódása után Reset állapotba kerül. A „Brown-Out-Detector" engedélyezésével a reset folyamat csak a tápfeszültség már biztonságos értékénél következik be, így nincs szükség külön reset áramkör alkalmazására. Mivel a központi tápegység szintén a felhasználói programból vezérelhető, hiba esetén annak segítségével is megoldható egy újraindítás, így a kontroller reset lába egy ellenállással tápfeszültségre húzható.

A processzor órajelét egy kvarcoszcillátor szolgáltatja, mely a C_{12}, C_{13} kondenzátorokból, a kvarckristályból és a kontrollerbe integrált meghajtó fokozatból áll. A kondenzátorok a felharmonikusok elnyomására szolgálnak, értékükre az Atmega16 adatlapja ad ajánlást a kvarc frekvenciájának függvényében. Mivel nem telepről üzemel a rendszer, így a kis fogyasztással szemben a sebesség sokkal fontosabb szempont, így az órajel frekvenciáját érdemes a lehető legnagyobbra választani. A 18.432MHz-es kristállyal a processzor áramfelvétele normál működés közben kb. 30mA.

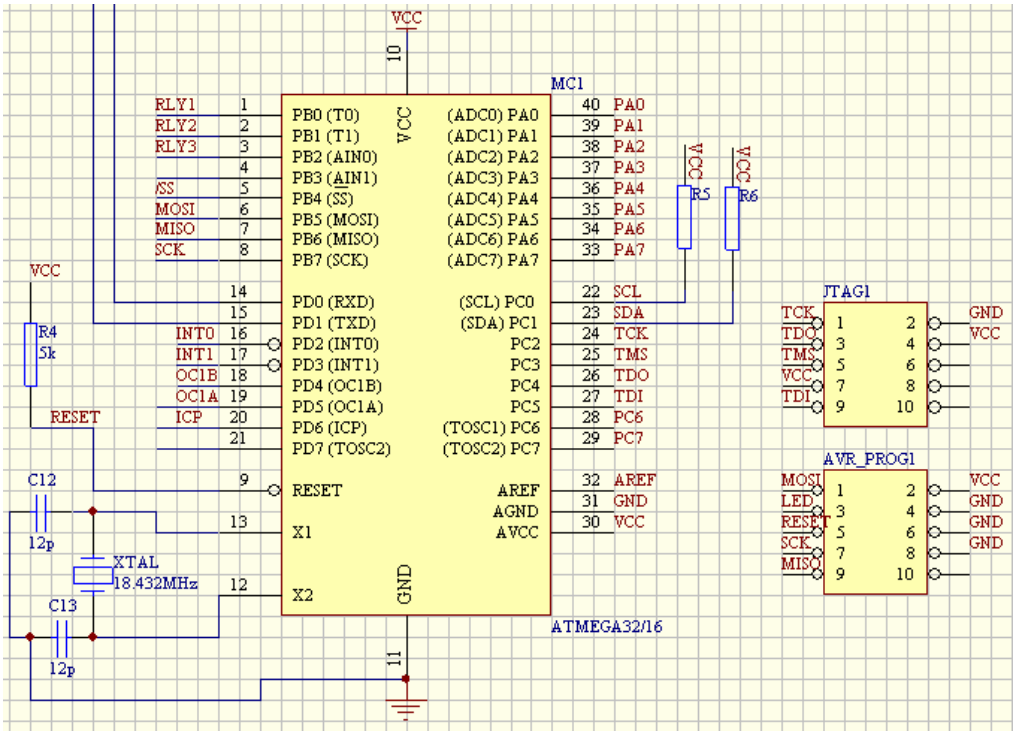
R_5 és R_6 ellenállások az I^2C busz vonalak tápfeszültségre húzásáért felelősek. Értéküket a busz sebessége és kapacitása határozza meg: Minél nagyobb a sebesség és a kapacitás, egységnyi idő alatt annál több töltésnek kell eljutni a tápfeszültségtől a buszvonalig, hogy az interfész kimenetén lévő tranzisztor zárása után a magas szint a sebességnek megfelelő idő alatt helyreálljon, ami annál kisebb ellenállásértéket követel meg. Az ellenállás minimális értéke abból adódik, hogy egy adott interfészen maximum 3mA áram folyhat keresztül.

Azaz:

$$\frac{Vcc-0.4}{3mA} < R_{pull-up} < \frac{1000ns}{C_b} \quad \text{ha } f_{SCL} < 100kHz \text{ és } \frac{Vcc-0.4}{3mA} < R_{pull-up} < \frac{300ns}{C_b}$$

ha $f_{SCL} > 100kHz$, ahol C_b a buszkapacitás pF-ban. ($f_{SCL}\big|_{max} = 400kHz$)

A kontroller felprogramozása, azaz a működtető program betöltése a Flash memóriába vagy SPI vonalon keresztül történik, vagy a JTAG interfészen át. Az alaplap mindkét lehetőségre fel van készítve.



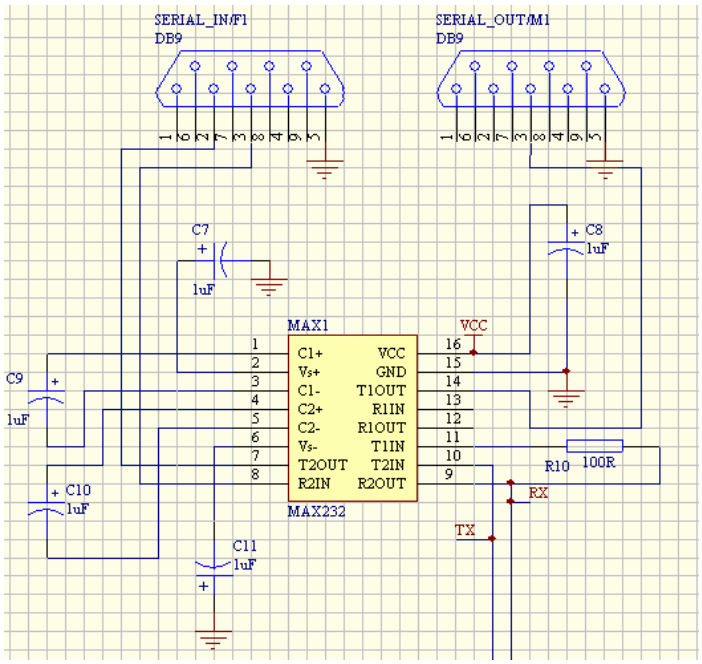
4.6. ábra Az alaplap mikrokontroller és kiegészítő elemek kapcsolási rajza

B, Soros port illesztése

A PC a parancsokat a szabványos soros porton keresztül küldi a szimulátornak, mely a bitek átvitelére -3..-25 V illetve +3..+25 V-os feszültségszinteket használ a zavarérzékenység csökkentése érdekében. Mivel a mikrokontroller beépített soros vezérlője csak TTL szintek küldésére és fogadására képes, a két interfészt illeszteni kell egymáshoz. Erre a feladatra a Maxim MAX232 típusú célintegrált-áramkör alkalmas, mely 5 V-os tápfeszültségről működve, beépített töltéspumpa segítségével mindkét

irányú illesztést megoldja a TTL és a soros vonali szintek között. Egy tokban mindkét irányú illesztésre két meghajtó fokozat is található, melyek eredeti célja a modemvezérlő jelek illesztése lenne, azonban ezekre most nincs szükség, így segítségükkel további tartalékok adhatók a rendszernek:

Mivel az I^2C busz kapacitása korlátozott, maximum 400 pF, így a buszra, bár a címtartomány lehetővé tenné, esetleg mégsem csatlakoztatható megfelelő sebesség mellett elegendő számú eszköz. Ha viszont a soros portról érkező parancsokat egy következő, ugyanilyen alaplapra szintén soros porton továbbküldjük, akkor ugyanez a címtartomány kiterjeszthető egy további, fizikailag leválasztott I^2C buszra. Ez egyszerűen megoldható, ha a kapcsolási rajzon látható módon a MAX232 IC PC felőli TTL szintű adó kimenetét nemcsak a kontroller vevő bemenetével kötjük össze, hanem a még szabad másik TTL -> RS232 meghajtón át egy másik csatlakozón keresztül küldjük tovább:



4.7. ábra A soros porti illesztő és ismétlő áramkör kapcsolási rajza

Ekkor viszont elveszítjük a teljes kétirányú kommunikáció lehetőségét a PC és a szimulátor között, hiszen az ilyen módon csatlakoztatott további alaplap(ok) nincsenek közvetlen kapcsolatban a PC-vel, mivel a független mikrokontrollerek adó vonalai nem közösíthetőek forgalomirányítás nélkül. Bár a PC felé a kommunikáció inkább csak diagnosztikai jellegű és a használhatóságot segíti, a specifikáció nem követeli meg, mégis feladása ebben a tartalék üzemmódban is hiba lenne. Megoldása legegyszerűbben

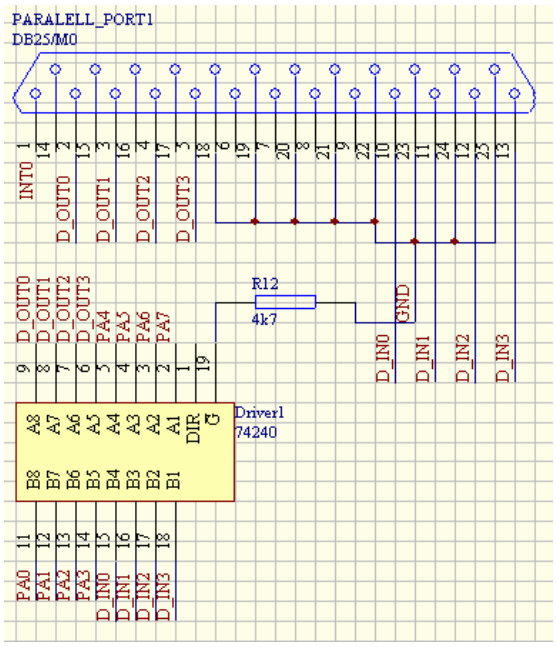
az alaplapok összekapcsolásánál, a központi egységek közötti további adatvonalak kialakításával eszközölhető.

C, Párhuzamos port illesztése

A soros port mellett egy párhuzamos nyomtató port is helyet kapott az alaplapon. Ennek oka a már említett tartalékokra való törekvés. Segítségével a terhelt busz mellett is lehetőség van gyors beavatkozásra, ha a jövőben egy speciális kommunikációt igénylő modul kiszolgálását is meg kell oldani. A kommunikáció a párhuzamos porton is kétirányú lehet, mindkét irányban 4-4bites adatszélességgel. A kapcsolatok az alábbi táblázatban részletezett módon valósulnak meg:

D-sub Pin #	Név	Információ iránya
1	STROBE	PC->Szimulátor
2	D0	PC->Szimulátor
3	D1	PC->Szimulátor
4	D2	PC->Szimulátor
5	D3	PC->Szimulátor
10	nACK	Szimulátor->PC
11	Busy	Szimulátor->PC
12	PaperEnd	Szimulátor->PC
13	Select	Szimulátor->PC

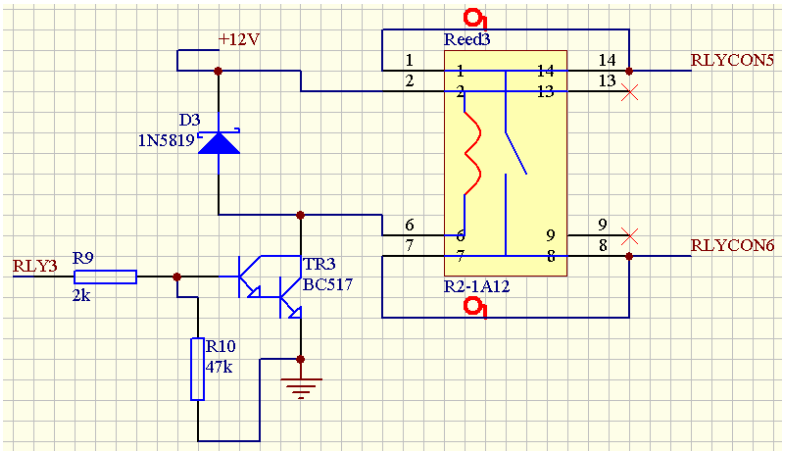
A PC az adatot a D0-D3 biteken keresztül küldheti. A Strobe jel a központi egység interrupt bemenetére csatlakozik. Új adat érkezését a PC a strobe jel felfutó élének segítségével jelezheti. A központi egység a többi négy bemeneten keresztül küldhet információt. Az átvitelvezérlés a szoftver feladata.



4.8. ábra A párhuzamos port bekötése az alaplapnál

D, Tápellátás illetve a gyújtáskapcsolás megoldása a motorvezérlő felé

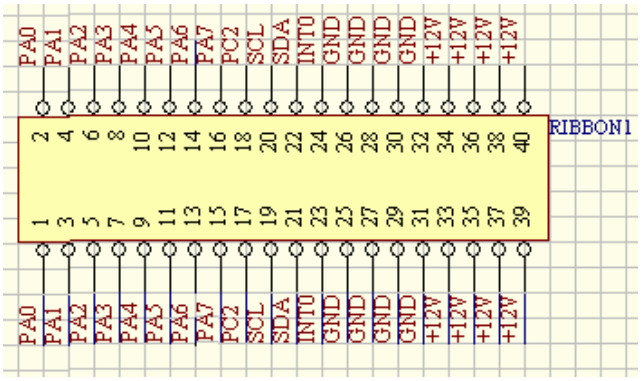
A tápellátás és gyújtás kapcsolása az autóban megszokott „relétáblán” elhelyezkedő nagyáramú relékkel történik, az alaplapnak ezek kapcsolását kell elvégeznie. Ez a teljes leválasztás végett szintén relékkel történik, ez azonban kisméretű, kis terhelhetőségű kapcsolóelemekkel is megoldható. Ilyen például az R2-1A12 típusú Reed-relé, mely DIP tokozásban maximum 1A-t tud kapcsolni. A relék a behúzótekercsét a mikrokontroller egyik I/O lába egy darlington tranzisztor segítségével kapcsolja (TR_3). A D_3 dióda egy gyors schottky dióda, mely a kikapcsolásnál a behúzótekercsen, mint induktív fogyasztón létrejövő nagyfeszültségű impulzus levezetésére szolgál.



4.9. ábra Relével megvalósított kapcsolóelem huzalozási rajza

E, bővíthetőség biztosítása további alaplapok csatlakoztatási lehetőségének biztosításával.

Több alaplap egymáshoz való csatlakoztatásával a busz fizikai mérete növelhető, ami további kártyahelyeket biztosít. Egy alaplapon két bővítő csatlakozóra van szükség, hogy kettőnél több alaplap is láncbafűzhető legyen. A csatlakozók az alábbi jeleken keresztül létesítenek kapcsolatot az alaplapok között:



4.10. ábra A bővítőcsatlakozó bekötési rajza

- Tápellátás
- I^2C busz
- Párhuzamos port adatbusz (PA0-PA7)
- Egy interrupt vonal (INT0)
- Egy általános célú I/O vonal (PC2)

A bővítésnek két alapvető módja lehetséges:

- Ha a cél egyszerűen újabb kártyák illesztése a buszhoz, és még nem értük el a maximális buszkapacitást (400 pF), akkor a csatlakoztatni kívánt alaplapon csak a tápellátást biztosító kapcsolás megléte szükséges, mely a kártyák tápfeszültségét biztosítja, a többi alkatrész, beleértve a központi egységet is, beépítése elmaradhat. A csatlakozón keresztül az új alaplap fizikailag a busz részévé válik.
- Ha a kapacitáskorlát az utóbbi megoldást nem engedi, akkor a busz bővítése a soros porton keresztül történhet, úgy, hogy az előző alaplap kimeneti soros portját az új alaplap bemenetével kell összekötni. Ilyenkor az új alaplapnak szintén rendelkeznie kell a központi egységgel, mely a parancsokat, bár az előző alaplapon keresztül, de közvetlenül a PC-től kapja. A két I^2C busz ilyenkor fizikailag elkülönül egymástól. A bővítő csatlakozó szerepe ilyenkor másodrendű,

ha a párhuzamos adatvezetékek használatára nincs szükség, akkor el is hagyható, ekkor azonban a tápellátást az új kártyának a központi tápegységről külön biztosítani kell. A két I^2C busz viszont nem köthető össze, a csatlakozásnál az SDA és SCL vonalakat meg kell szakítani.

4.2.1.3. Alaplap szoftver terve

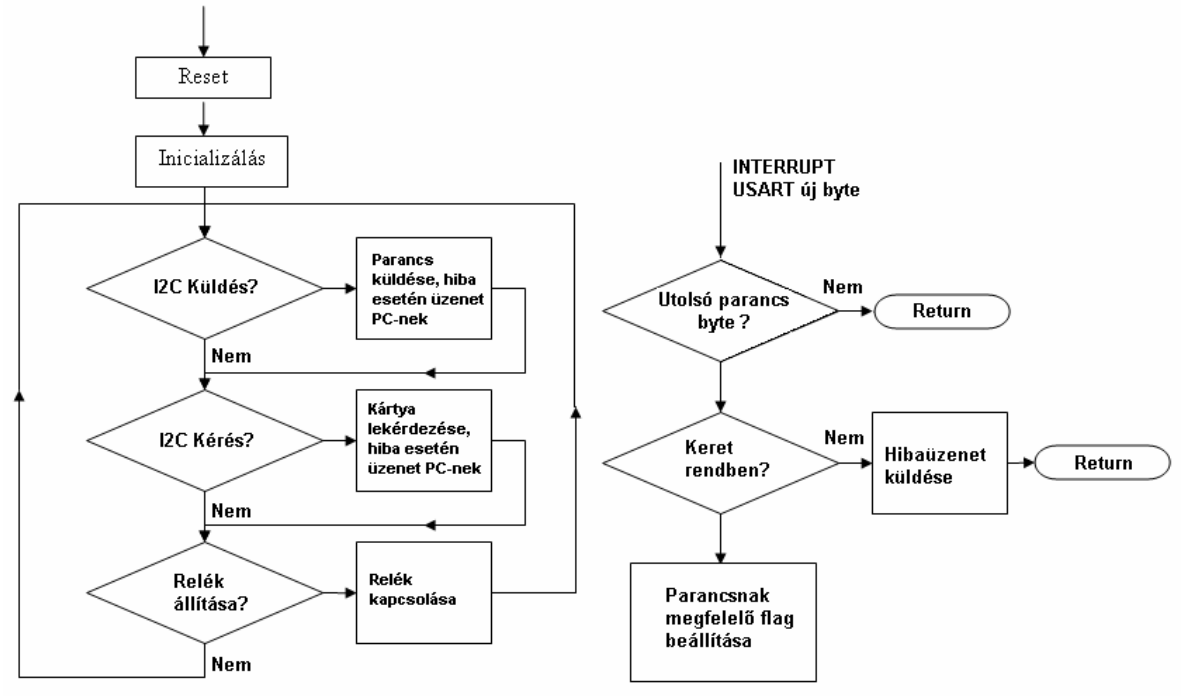
Az alaplap szoftver feladata a PC és a szimulátorkártyák közti kommunikációs protokoll megvalósítása, a motorvezérlő tápfeszültség- és gyújtásjel-kapcsolása.
A protokollal szemben támasztott követelmény, hogy a feldolgozás sebessége elég gyors legyen ahhoz, hogy folyamatos küldést is lehetővé tegyen, anélkül, hogy az adatvesztés okozna. Követelmény emellett, hogy jelezze, hogyha az átvitel közben a buszon hiba történt, például egy megcímzett kártya nem elérhető.
A kommunikáció két részre osztható:

- soros port kommunikáció
- I^2C busz vezérlés

Mivel a cél a PC és a kártyák közti kapcsolat létrehozása, a két közegben az adatok átvitelére használt protokollt célszerű azonosra választani, hiszen akkor csak a fizikai rétegek közötti konverziót kell megvalósítani, így gyorsabb működés érhető el.
Az átvitelt minden esetben a PC kezdeményezi, mely lehet a kártyák kimenetét módosító parancs, vagy a buszhoz csatlakoztatott kártyák adatainak lekérdezése, paramétereinek módosítása.

A feladathoz jól illeszkedik az interrupttal kiegészített ciklikus programszervezés, így a szoftver felépítése ezt az elvet követi. Ez azt jelenti, hogy a feladatvégző rész, ami jelen esetben a küldés az I^2C buszra, olvasás a buszról, illetve a jelfogók működtetése, egy végtelen ciklus magját alkotja, de csak akkor, és az a függvény hajtódik végre, amelyiktől az interrupt rutinban érkező parancs kiszolgálást kér.
Ez egy jól kezelhető, éppen ezért elterjedt architektúra: az új esemény egy újabb interrupt rutin, egy új feladat pedig a ciklusban egy újabb feltétellel, vagy anélkül végrehajtandó függvény hozzáadását jelenti.

A szoftver folyamatábrája a következőképpen alakult:



4.11. ábra Az alaplap mikrokontroller szoftver működésének egyszerűsített folyamatábrája

Mivel az RS-232 kapcsolat elegendőnek bizonyult a kívánt adatforgalom lebonyolítására, a párhuzamos portot kezelő rutin nem került implementálára. Szükség esetén későbbi továbbfejlesztéskor ez bármikor megoldható a program kiegészítésével, ami a STROBE jelre kötött külső interrupt engedélyezését és a kezelő függvény beszúrását jelenti.

A szoftverfejlesztés a mikrokontroller egyik ingyenesen hozzáférhető fejlesztői környezetében, a WinAVR-ben történt. A program C fordítót is tartalmaz, így jóval gyorsabb fejlesztést tesz lehetővé, mint az assembly nyelv.
A program a következő fájlokból épül fel:

RS232.c: Ez a fájl tartalmazza a soros port inicializálását végző rutint, a küldést és fogadást végző függvényeket. A fogadást végző függvény maga az interrupt rutin.

TWI_master.c: Ebben fájlban kerültek implementálásra az I^2C buszt master módban működtető függvények. (interfész inicializálás, Start, Stop feltétel fogadása, írás a buszra, ACK és NACK fogadás)

main.c: Ebben a fájlban található a főprogram. Az egyes alkalmazott perifériák inicializálása után a main ciklus részeként egy végtelen ciklusban történik az I^2C busz kezelése, a küldés és fogadás, melyet az interrupt rutin engedélyez a beérkező parancsok szerint.

A soros porti kommunikáció protokollja:

A kommunikáció a soros porton a következőképpen történik:

A PC a parancsokat mindig egy 8 bájtos üzenetben küldi az alaplapnak, melynek felépítése:

1. szinkron karakter
2. célkártya címe
3-8. 6 adatbájt

A szinkron karakter a kommunikáció során esetlegesen fellépő hibák valószínűségének csökkentésére szolgál. Értéke mindig ugyanaz, és az alaplap szoftver csak akkor kezdi meg a parancs fogadását és értelmezését, ha az üzenet ezzel a karakterrel kezdődik. Ily módon, ha valamilyen hiba folytán az üzenet csonkolva érkezik az alaplaphoz, újraindítás nélkül, esetleg néhány adatbájt elvesztése árán ismét szinkronizálódik a kommunikáció.

Ezt követi annak a kártyának a címe, melynek az üzenetet továbbítani szeretnénk. A megengedett címtartomány 0x00-0x7F-ig terjed. Van azonban két kivétel: Ha az üzenet a 0xF9 címet tartalmazza, akkor ez a parancs az alaplapon lévő gyújtás- és tápfeszültség-kapcsoló relékre vonatkozik, ezt az üzenetet a mikrokontroller nem továbbítja az I^2C buszra. Ha a cím 0xFA, akkor az alaplap egy olvasást kezdeményez az 1 adatbájtban megadott című kártyától. Ha a megcímzett kártya a buszon van, akkor egy 6 bájt hosszú üzenetben elküldi eepromban tárolt paramétereit, melyek a következők: cím, szoftververzió, típus, és 3 egyelőre kihasználatlan bájt. Amennyiben a buszon nem található a megcímzett kártya, az alaplap a „NO_ADDR” hibaüzenetet küldi vissza.

Az adatbájtok tartalmazzák a címben szereplő kártya kimenetét módosító paramétereket. Az adatok jelentését a 6. bájt értéke módosíthatja. (Lásd az egyes kártyák esetén.)

Az I2C busz kommunikáció protokollja:

Miután az alaplap megkapta a PC-től az üzenetet, és az valóban egy kártyának továbbítandó parancs, továbbküldi azt az I^2C buszra. Itt azonban már nem jelenik meg a

szinkronbájt. A master, az alaplap mikrokontrollere kiküldi a buszra a címezni kívánt slave címét, és ha az elfogadja a címzést, akkor megkezdi a PC-től kapott 6 adatbájt küldését. Jelenlegi állapot szerint a szimulátorkártyák nem kezdeményeznek átvitelt.

4.2.2. Szimulátorkártyák tervezése

Az előző pontokban leírtak alapján a szenzorok jelét az I^2C buszra csatlakoztatott független modulok, továbbiakban szimulátorkártyák, kártyák, fogják szimulálni, melyek a működéshez szükséges paramétereket a buszról kapják. A kártyáknak a motormenedzsment bemutatásánál megismert típusú szenzorokat kell helyettesíteniük, melyek a következők:

- fordulatszámjeladók:

- *vezérműtengely*
- *főtengely*

- hőmérséklet-jeladók:

- *motorhőmérséklet*
- *kipufogógáz-hőmérséklet*
- *üzemanyag-hőmérséklet*

- egyéb szenzorok:

- *légtömegmérő*
- *töltőlevegőnyomás- és hőmérséklet-szenzor*
- *diesel-részecskeszűrő nyomáskülönbség-mérő*
- *olajszint és -hőmérséklet-jeladó*
- *gázpedál-modul*
- *olajnyomás-kapcsoló*

A szenzorok által mért különböző fizikai mennyiségeket négyféle (a konkrét megvalósításnál részletezett) kimeneti jeltípus reprezentálja:

- Analóg feszültségváltozás
- Ellenállásváltozás
- digitális hullámforma
- kapcsolóállapot

A kártyákkal szemben támasztott általános követelmények:

- szimuláció a szenzor teljes működési tartományában;
- robusztus felépítés és működés;
- univerzális felhasználhatóság.

Egy-egy szenzorkártya egy adott motorvezérlővel a működés során mindig egy kitüntetett szenzor jelét szimulálja, azért azonban, hogy a rendszer egyszerűen bővíthető legyen, ugyanaz a kártya egy másik motorvezérlővel felépítve egy másik, típusának megfelelő kimenettel rendelkező, ám más karakterisztikájú szenzort fog helyettesíteni. Mivel a különböző motortípusokon még az azonos mennyiséget mérő szenzorok karakterisztikája is gyakran eltérő, így célszerű egy kártyát nem egy adott szenzorhoz elkészíteni, hanem egy adott kimenettípushoz: azaz egy kártya PC-től kapott paramétere nem a szenzor által mért fizikai mennyiség értéke, hanem a felhasználói programban beállított fizikai mennyiséghez az adott szenzor-karakterisztika szerint tartozó kimeneti jel adott paraméterének fizikai értéke. Így nem kell a kártyának tárolnia a szenzorkarakterisztikákat, és új szenzor esetén gondoskodni az új karakterisztika beprogramozásáról, hanem ez a felhasználói programban jóval egyszerűbben megoldható egy táblázat segítségével. Ezen táblázatok a felhasználói programnak megfelelő formátumú elkészítése szintén része volt a feladatnak, és a következőképpen történt:

A szenzorok karakterisztikája azok dokumentációjában megtalálható, általában függvény vagy táblázat formájában. A LabView-ban írt felhasználói program számára a bemenet egy mátrix, melynek első oszlopa az adott szenzor gerjesztőjelének lépésenkénti értékeit tartalmazza, a második oszlop pedig gerjesztés hatására a kimeneten megjelenő jelet. A mátrix méretét a következő két szempont korlátozza:

- legyen a mátrix minél kisebb, mivel a sok szenzor, és az egyéb táblázatok miatt a program memóriaigénye nagyra nőhet, ami lassú működést eredményezhet;
- mivel két lépés között a LabView lineárisan interpolál, a túl rövid mátrix miatti ritka lépéssűrűség nemlineáris függvény esetén viszonylag nagy hibát okozhat, ezért legyen a mátrix minél hosszabb, hogy nagyobb legyen a pontosság.

Ezek alapján a lépésköz meghatározása a következőképpen történt, Matlab segítségével:

Mivel a szenzorkártyák által megkövetelt pontosság általánosan az adott szenzor hibával terhelt karakterisztikájához képest kb. 1%, körülbelül ekkora pontosságot kell biztosítani a karakterisztikában is, azaz a megadott szenzorkarakterisztika, és az ennek mintavételezésével adódó, mátrixszal megadott töréspontos karakterisztika eltérése ne legyen nagyobb 1%-nál. Ez a függvénnyel megadott karakterisztika esetén a Matlab *polyval* függvényének segítségével történt, a függvényértékek megfelelő elem-lépésközű bemeneti vektorral történő kiszámításával. A táblázattal megadott karakterisztikánál ezt a lépést megelőzte a *polyfit* függvénnyel történő polinomillesztés a karakterisztika pontjaira. Így optimális méretű mátrixot kaphattunk, mely például lineáris szenzorkarakterisztikánál mindössze a kezdő és végértéket kellett, hogy tartalmazza.

A kártyák funkcióinak megvalósításához szintén érdemes valamilyen mikrovezérlőt alkalmazni. Meg kell oldani az adott kártya illesztését a buszhoz, a beérkező parancsok értelmezését, és ennek megfelelően a kimenet módosítását. Ezek a feladatok mikrokontroller használatával sokkal rugalmasabban, és hardverszinten is univerzálisabban oldhatóak meg. A feladathoz célszerű az Atmel AVR család egy másik, kisebb tagjának választása, mely rendelkezik a szükséges perifériakészlettel, és mivel a processzormag ugyanaz, mint a többi AVR mikrokontrollernél, teljes szoftverkompatibilitás is jellemzi, így például az I^2C busz vezérlése majdnem teljesen átvehető. A választás a dokumentációk áttanulmányozása alapján az AtMega8 típusú kontrollerre esett. Ez az alaplapnál már ismertetett Atmega16-hoz hasonló felépítésű, de kevesebb, 8 kbájt programmemóriával, és kevesebb általános célú I/O perifériával rendelkezik, de ugyanúgy megtalálható benne az I^2C illesztő, az ADC, vagy például az USART, mely a JTAG interfész híján a programfejlesztésben jelentős segítséget nyújthat.

Mindegyik szimulátorkártya az alaplaphoz tehát azonos interfészen keresztül kapcsolódik, és a fizikai méreteiknek is célszerű egyezniük, hogy az elektronika védelmére készülő burkolat megfelelő megvezetést nyújthasson a kártyáknak. Ez célszerűvé teszi a szimulátorkártyákhoz egy univerzális PCB tervezését, melyen az összes kimenet meg van valósítva, azonban mindig csak az adott típusnak megfelelő

alkatrészek vannak beültetve. Ez tulajdonképpen az egyes kimenetek kapcsolási rajzában nem jelent különbséget, de a kis lábszámú mikrokontroller miatt néhány jumper beültetését teszi szükségessé, melyek funkciója a 6. pontban kerül részletezésre.

4.2.2.1. Analóg feszültségkimenetű szenzorok

Általában azok a szenzorok, melyeknél a mért fizikai mennyiséggel arányos elektromos jel az alacsony jelszintek miatt helyi jelkondicionálás (és akkor már előfeldolgozás) nélkül nem továbbítható a motorvezérlő felé, kimenetükön az információt valamilyen digitális interfészen keresztül adják át, vagy egy analóg feszültségkimenet szintje hordozza a mérési eredményt. Ez a feszültség, a rendelkezésre álló dokumentációkat megvizsgálva, minden esetben a szenzor tápfeszültség-tartományán belül változik, melynek szabványos értéke 5 V minden esetben. A vizsgált motortípusnál ezek a következő szenzorokat jelentik:
Légtömegmérő, *töltőlevegőnyomás- és -hőmérsékletmérő* levegőnyomás-mérő része,
diesel-részecskeszűrő nyomáskülönbség-mérő, *gázpedál-modul*

4.2.2.1.1. A kimeneti paraméterek meghatározása

A szenzorkarakterisztikák tanulmányozásával megállapítható, hogy a legtágabb feszültségtartomány, melyben a kimeneti feszültséget elő kell tudni állítani, az a 0,5 – 4,5 V-os tartomány. Követelmény továbbá, hogy a hardver alkalmas legyen hibás szenzorműködés szimulációjára is, így ezt a határt valamennyire bővíteni kell. Logikus megoldás a 0 – 5 V nyitott intervallum alkalmazása, hiszen ez a szenzor tápfeszültség-tartománya, de így a határokat elég csak megközelíteni, ami a célt teljesíti, de nem igényli negatív tápfeszültség alkalmazását a kimeneti puffer erősítőhöz.

A kimenet pontosságát jelentős mértékben befolyásolja, hogy a szenzor és a motorvezérlő között a csatolás hogyan jön létre. A szenzorok egy részénél a kimenet értéke tápfeszültségfüggő. Itt a motorvezérlő által biztosított tápfeszültség szolgál a szenzor kimeneti DA-átalakítójának referenciájául, illetve a motorvezérlő bemenetén lévő AD-átalakító is ehhez képest végzi a konverziót. Ez a felépítés biztosítja, hogy maga az átvitt információ közelítőleg tápfeszültségfüggetlen legyen. A másik

alkalmazott megoldás, hogy a DA és AD fokozatok egy-egy, egymástól független, ismert, állandó feszültségű referenciával rendelkeznek, melyek tápfeszültségfüggetlenek. Ekkor a kimenet értéke ismét tápfeszültségfüggetlen, de ebben az esetben a tápfeszültség változása nem is okozza a kimenet változását.
Azaz fontos az adott kimenet típusának ismerete, hiszen ez határozza meg az átalakítás referenciáját, melynek rossz megválasztása hibát okozhat.

A kártyán a DA-átalakítás megoldására két alapvető lehetőség kínálkozik. Az egyik egy külön DAC IC alkalmazása, a másik pedig a mikrokontrollerben rendelkezésre álló PWM modul és egy aluláteresztő szűrő használata.
Külön DAC alkalmazása esetén a problémát tulajdonképpen csak a kommunikáció megvalósítása jelenti, mely az adott IC típusától függően lehet egyszerű párhuzamos, vagy újabban inkább soros SPI, vagy I2C buszon keresztül zajlik. A választott mikrokontroller ezek mindegyikével rendelkezik, így ez nem jelenthet akadályt. További előnye, hogy a konverter névleges felbontása nem frekvenciafüggő, nagyobb sávszélesség is biztosítható a pontosság megtartása mellett. Jelentős hátránya azonban, hogy igen költséges.
A PWM-mel megvalósított DA-átalakító ezzel szemben olcsó és robusztus. A választott mikrokontroller két független PWM kimenettel is rendelkezik, és ára fele a jelenleg kapható 8 bites DAC-kének. Ennek a megoldásnak azonban hátránya, hogy egy megfelelően méretezett aluláteresztő szűrőről is gondoskodni kell, illetve nagyobb felbontás csak nagy órajelfrekvenciával, vagy kis sávszélességgel lesz megoldható.

A szenzorok szimulációjánál a megkívánt maximális frekvencia, melyet a kártyáknak még teljesítenie kell, legfeljebb kb. 200Hz. Ez az alacsony frekvencia az előnyei miatt a PWM-mel megvalósított DA-nak kedvez, így emellett döntöttünk.

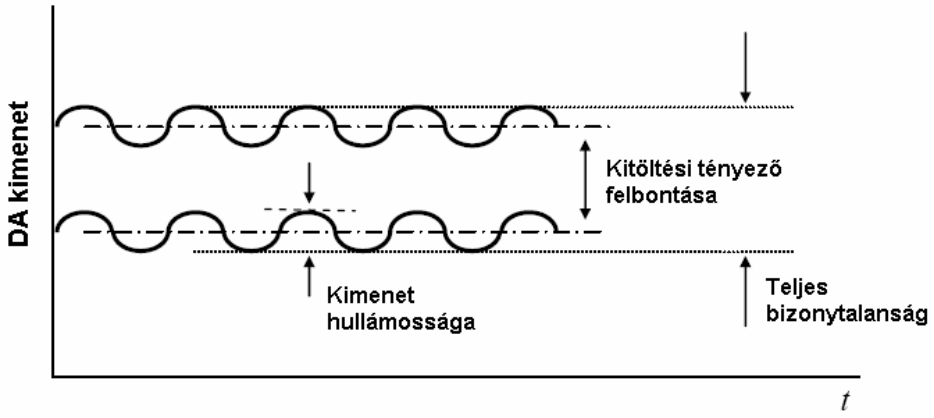
4.2.2.1.2. Az analóg kimenet kapcsolási rajza[8]

Az analóg kimenetet megvalósító kapcsolással szemben támasztott követelmények tehát a következők:

-]0-5[V közötti működés;
- <0,5% pontosság a végértékre vonatkoztatva;
- f_{max}=200 Hz;
- választható tápfeszültségfüggő és -független kimenet;
- felépítés: PWM + aluláteresztő szűrő.

A PWM-mel megvalósított DA-átalakító szerves része egy aluláteresztő szűrő, melynek feladata, hogy a mikrokontroller kimenetén megjelenő négyszögjelnek csak a középértéke kerüljön a kimenetre, a négyszögjel alapfrekvenciája, és a felharmonikusok ne jelenjenek meg ott. A kimeneten végül megjelenő jel pontosságát befolyásolja egyrészt a PWM felbontása, másrészt a szűrő véges meredeksége miatt megjelenő AC komponens nagysága is. Minél kisebb a szűrő vágási frekvenciája, és minél nagyobb a PWM alapfrekvenciája, annál nagyobb a szűrő csillapítása a váltakozó komponensekre nézve. A szűrő vágási frekvenciáját azonban alulról a kívánt sávszélesség korlátozza, illetve a PWM alapfrekvenciájával fordítottan arányos annak maximális felbontása. Tehát a legnagyobb pontosság a kívánt sávszélességhez méretezett szűrő esetén egy optimálisan megválasztott frekvenciájú PWM-mel érhető el.

A kimeneten megjelenő bizonytalanság tehát tulajdonképpen a PWM felbontásának és az áteresztett AC komponens amplitúdójának összege:



4.12. ábra A PWM-el megvalósított DA hibájának kialakulása

A szűrő vágási frekvenciája a kívánt sávszélesség (200 Hz) alapján adódik, kb. 250 Hz-re választható. A szűrő fokszámát a kívánt pontosság fogja meghatározni.
Elsőfokú szűrő alkalmazása esetén a zárótartományban a csillapítás -20dB/dekád. A mikrokontroller kimenetén megjelenő négyszögjel amplitúdója 5 V. A közelítő számításhoz vegyük csak a négyszögjel alapharmonikus összetevőjének amplitúdóját figyelembe. Mivel a magasabb harmonikusok eleve kisebb amplitúdójához a nagyobb frekvencia miatt nagyobb elnyomás is tartozik, így nem követünk el számottevő hibát.
A számítás szempontjából az 50%-os kitöltésű négyszögjelet vegyük alapul, mivel itt a legnagyobb az alapharmonikus energiatartalma[1].

$$U_k = \frac{1}{T}\int_0^T u(t)e^{-jk\omega t}dt \tag{1.1}$$

$$U_1 = \frac{2}{T}\int_{-\frac{T}{4}}^{\frac{T}{4}} U_0 e^{-j\omega t}dt = \frac{2U_0}{T}\frac{e^{j\omega\frac{T}{4}} - e^{-j\omega\frac{T}{4}}}{j\omega} = \frac{4U_0}{2\pi}\sin\frac{\pi}{2} \approx 3{,}18V \tag{1.2}$$

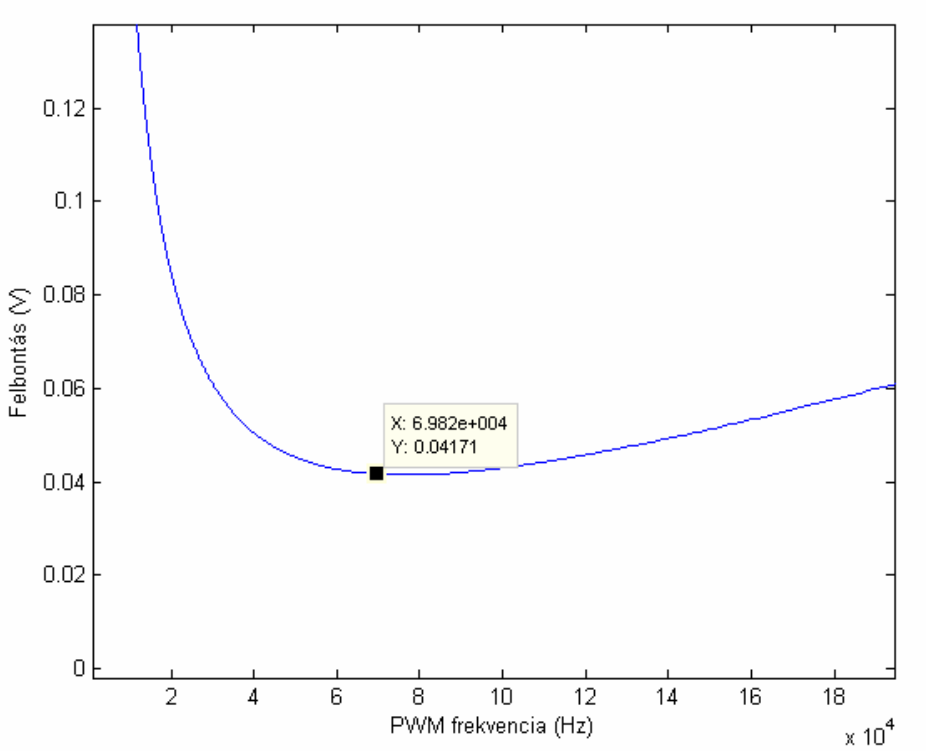
Legyen a PWM, és ezáltal az alapharmónikus frekvenciája f_{pwm}, a szűrő fokszáma n, f_c a vágási frekvencia. A szűrő kimenetén megjelenő hullámosság amplitúdója ekkor kb.:

$$u_{ripple} = U_1 \cdot 10^{\frac{-20\cdot n\cdot \lg\frac{f_{pwm}}{fc}}{20}} = U_1 \cdot \left(\frac{f_c}{f_{pwm}}\right)^n \tag{1.3}$$

A kitöltési tényező felbontása pedig:

$$u_{res} = \frac{U_0}{\frac{f_{cpu}}{f_{pwm}}} \tag{1.4}$$

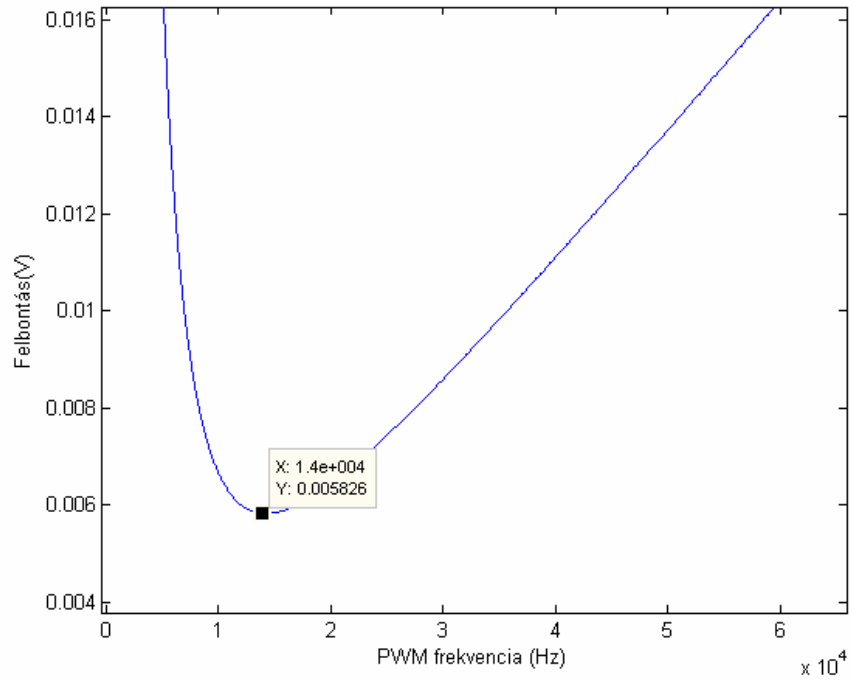
Az optimalizálás célja, egy olyan f_{pwm} keresése, melyre $u_{ripple} + u_{res}$ összeg minimális.
A mikrokontrollerben a PWM előállítására használt timer modul 16 bites, ami a lehető legnagyobb, 18,432 MHz-es órajel mellett
$281{,}25\text{Hz} = \frac{18{,}432\text{MHz}}{2^{16}} \le f_{pwm} \le 18{,}432\text{MHz}$ PWM frekvenciák választását teszi lehetővé. Az ezeken a frekvenciákon adódó $u_{ripple} + u_{res}$ összeget Matlab-ban kiszámítva és ábrázolva a következő ábrát kapjuk:



4.13. ábra PWM-mel megvalósított DA-átalakító effektív felbontása a PWM frekvencia függvényében elsőfokú szűrő esetén

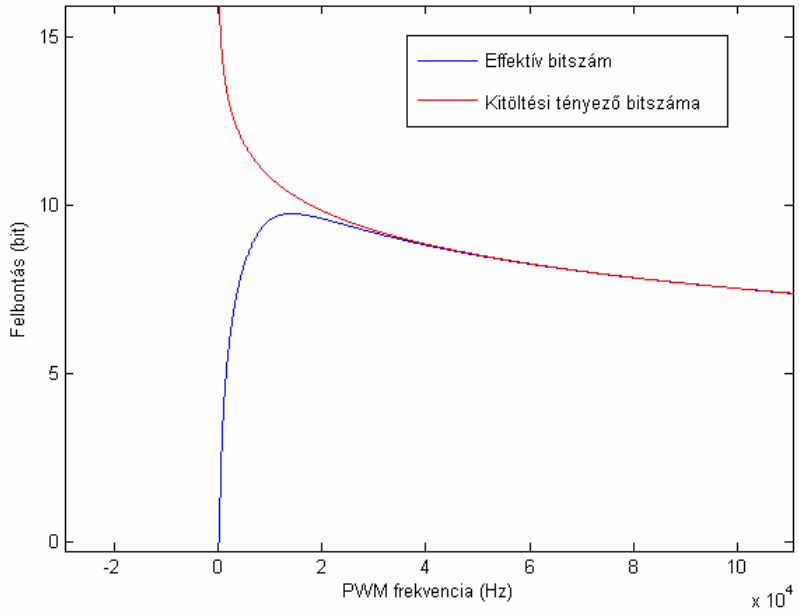
Ez azt jelenti, hogy elsőfokú szűrő esetén, 41 mV-os felbontást kapunk, ami még elmarad a kívánt 4,8 V · 0,5% = 24 mV pontosságtól.

A megoldás a másodfokú szűrő alkalmazása lehet:



4.14. ábra PWM-mel megvalósított DA átalakító effektív felbontása a PWM frekvencia függvényében másodfokú szűrő esetén

Másodfokú szűrővel a felbontás az optimális **14 kHz-es PWM frekvencián** a legnagyobb, 5,8 mV, ami már teljesíti a követelményeket.

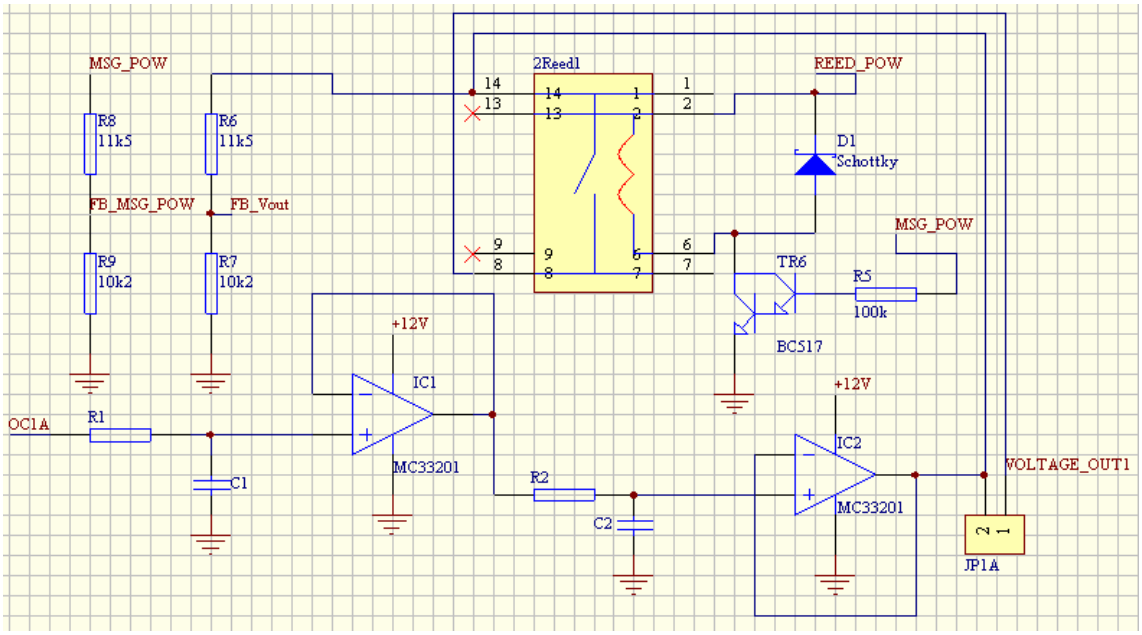


4.15. ábra A PWM-mel megvalósított DA-átalakító effektív felbontása bitben kifejezve, illetve a kitöltési tényező felbontása

Az ábrából jól látható, hogy bár alacsonyabb frekvencián a kitöltési tényező nagyobb felbontást ígér, de a szűrő kisebb elnyomása miatt az effektív felbontás alacsonyabb lesz. Nagyobb frekvenciákon viszont a felbontást már egyértelműen a kitöltési tényező felbontása határozza meg.

A kívánt másodfokú szűrő legegyszerűbben két elsőfokú RC szűrő kaszkádosításával valósítható meg.

A kimenet kapcsolási rajza (a mikrokontroller nélkül):



4.16. ábra Az analóg kimenetet megvalósító áramkör kapcsolási rajza

OC1A a mikrokontroller PWM kimeneti csatornája. A jel az R_1 és C_1 alkotta aluláteresztő szűrő után egy egyszerű műveleti erősítővel (IC1) megvalósított követőerősítőre jut. Ez azért szükséges, hogy a szűrő következő fokozata ne terhelje az előzőt, ami a meredekséget csökkentené. A kártyán nem áll rendelkezésre negatív tápfeszültség, ezért hogy a kimeneten mégis minél alacsonyabb feszültség is megjeleníthető legyen, rail-to-rail műveleti erősítő választása szükséges. Egy ilyen erősítő az MC33201. R_2 és C_2 alkotja a szűrő második fokozatát. R_1, R_2 680 Ω, illetve C_1, C_2 1 µF-ra választásával a vágási frekvencia kb. 234 Hz. Ezután ismét egy követő erősítő következik, ami a kimenet alacsony impedanciáját biztosítja. Az olcsó MC33201 műveleti erősítő szintén alkalmas erre a feladatra. Kimenete rövidzárvédett, és 80 mA-rel terhelhető, ami megfelel bármely szenzor által támasztott követelménynek.

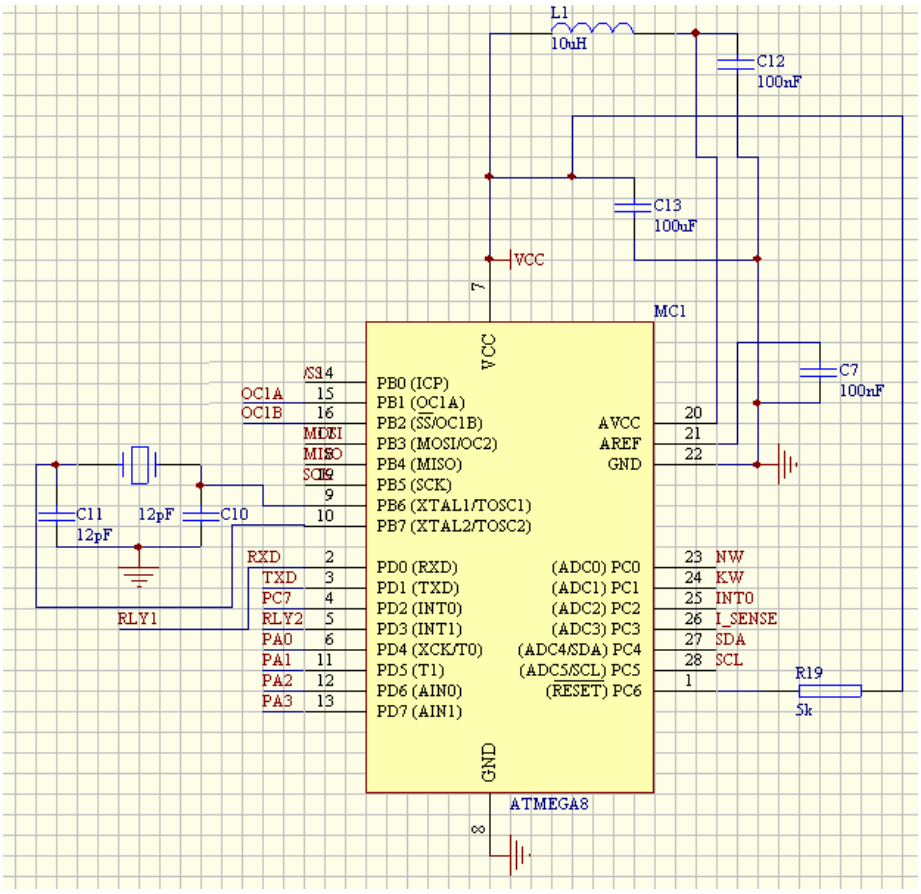
Elvárás továbbá, hogy a kimeneten csak akkor jelenjen meg feszültség, ha a motorvezérlő biztosítja az adott szenzor tápellátását. Ezt meg lehetne oldani úgy is, hogy a motorvezérlő felőli tápfeszültséget a mikrokontroller egyik I/O lábán figyeljük, és a PWM kimenetet csak megléte esetén engedélyezzük. Egy, a motorvezérlő tápfeszültsége által vezérelt relével kapcsolva a kimenetet viszont galvanikus leválasztást is biztosíthatunk, így ezt a megoldást választottuk. A relét a BC517 darlington tranzisztor kapcsolja, így motorvezérlő tápfeszültségének terhelése gyakorlatilag elhanyagolható. A JP1A jumper segítségével az előző megoldás is

realizálható, de ez inkább a teszteléshez nyújt segítséget. Meg kell még oldani a kapcsolásban továbbá, hogy a tápfeszültségfüggetlen és a tápfeszültségfüggő kimenetű szenzorok is pontosan szimulálhatóak legyenek. Ezt a feladatot részben a szoftver fogja megoldani. Két paraméter ismeretére van szükség:

- A tápfeszültségfüggetlen szenzoroknál a kimenet értéke tehát csak a mért értéktől függ a karakterisztika alapján, azaz itt nem kell figyelembe venni a motorvezérlőtől érkező tápfeszültség értékét. Viszont a PWM-mel megvalósított DA felépítése miatt a kimeneti feszültség függ a mikrokontroller tápfeszültségétől, itt ezt a függést kompenzálni kell, azaz ismerni kell a tápfeszültséget. Ehhez vagy közvetlenül a tápfeszültséget kell megmérni, vagy a kimenet értéke alapján is lehet rá következtetni. Utóbbi eset került megvalósításra, mivel ekkor a kimenet aktuális értékéről is van információnk, mellyel nem megfelelő érték esetén hibajelzést generálhatunk, illetve a megfelelő értékre szabályozhatunk.
- A tápfeszültséget mint referenciát használó szenzoroknál a mikrokontroller tápfeszültsége mellett a motorvezérlőtől kapott szenzortápfeszültséget is ismerni kell, hiszen ennek értékétől közvetlenül függ a kimenet értéke.

A paraméterek mérésére a mikrokontroller beépített AD-átalakítójával van lehetőség. Ez egy 10 bites szukszcesszív approximációs konverter, melynek felbontása pont illeszkedik a feladathoz, hiszen az analóg kimenet felbontása ettől éppen elmarad, 9,75 bit körül van (4.15. ábra). Az AD-átalakító rendelkezik egy belső referenciával is, melynek értéke 2,56 V, és ki van vezetve az egyik lábra is, így kalibrálható is a mért érték. A kísérletek alapján a 7805 típusú feszültségstabilizátorok és a motorvezérlő által kiadott szenzortápfeszültség sem nő 5,3 V fölé, így a szükséges feszültségosztó osztásarányát R6, R7 illetve R8, R9 ; 11,5 kΩ illetve 10,2 kΩ ellenállások segítségével kb. 0.47-re állítottuk be. A maximális mérhető feszültség ekkor 5,44 V. (Erre vonatkoztatva a 10 bites AD-átalakító még mindig nagyobb felbontású, mint a feszültségkimenet.) Így az AD-átalakító közel jár a végértékhez, ami nagyobb pontosságot biztosít.

A mikrokontroller és a körülötte szükséges egyéb elemek a következő ábrán láthatóak:



4.17. ábra Az Atmega8 mikrokontroller, és a szükséges egyéb kapcsolási elemek

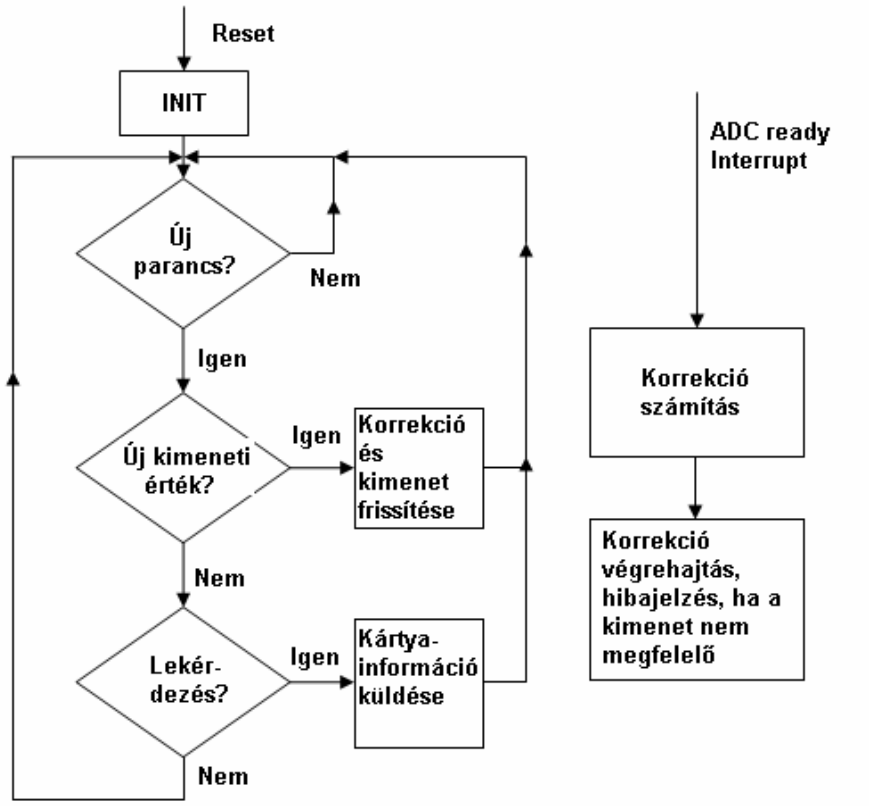
C_{10} és C_{11} szűrőkondenzátorok a 18,432 MHz-es kristállyal biztosítják a mikrokontroller órajelellátását. C_{13} a kontroller tápfeszültségét puffereli, L_1 és C_{12} pedig az AD-átalakító tápfeszültségének szűrését végzi. C_7 a beépített referencia jelét puffereli.

4.2.2.1.3. Az analóg kimenet szoftver terve

Az analóg kimenettel rendelkező kártya szoftverének feladatai:

- Parancsok fogadása I^2C buszon keresztül;
- Lekérdezés hatására információ küldése a PC felé;
- kimeneti feszültség beállítása a kapott parancsnak megfelelően;
- korrekció számítása a kimeneti feszültség mérése alapján, a szenzortípus fajtája szerint.
- hibajelzés, ha a kért kimeneti érték nem állítható be.

A szoftver folyamatábrája az alábbi ábrán látható:



4.18. ábra Az analóg kimenetet megvalósító mikrokontroller szoftverének egyszerűsített folyamatábrája

A program az inicializálás után egy végtelen ciklusban vár új parancs érkezésére. Jelenleg a program kétféle parancs fogadására képes:

- a kimeneti értéket módosító parancs a PC felhasználói programtól;
- lekérdezést illetve címmódosítást kezdeményező parancs a kártyamenedzsment programtól.

A kártya a felhasználói programtól minden esetben 6 bájtot kap, melyeket a következőképpen értelmez:

Az első 5 bájt jelentését a 6. bájt tartalma határozza meg. Ha ennek értéke 0xFA, akkor az adatok nem a kimenetre, hanem a kártya egyéb paramétereinek módosítására szolgál, ahol:

1. bájt: a kártya új címe, mely reset után érvényes

2-6. bájt: további fejlesztésekre fenntartva

Ha a 6. bájt értéke 0x00, akkor a parancs új kimeneti értéket tartalmaz:

1. bájt: Kimeneti feszültség felső bájt

2. bájt: Kimeneti feszültség alsó bájt

3. bájt: a kimenet típusa: 0x01 – tápfeszültségfüggetlen, 0x02 - tápfeszültségfüggő

4-6. bájt: további fejlesztésekre fenntartva

Az első két bájt a kimenet értékét nem feszültségben adja meg, hanem a PWM modul összehasonlító regiszterének értékét tárolja, konstans 5 V-os tápfeszültségre számítva. Ha pl. beállítani kívánt érték „u”, akkor az első két byte alkotta 16bites szó értéke a következőként számítható:

$$\text{data} = \text{round}\left(\frac{u}{5V} \cdot \frac{f_{clk}}{f_{pwm}}\right) \tag{1.5}$$

Tápfeszültségfüggetlen esetben a kimeneten is ennek a feszültségnek kell megjelennie, így ha a mikrokontroller tápfeszültsége 5 V, akkor ez az érték közvetlenül bekerül az összehasonlító regiszterbe. Ha a tápfeszültség ettől eltérő, akkor ezt az értéket még meg kell szorozni egy korrekciós taggal, ami az ideális 5 V és a tényleges tápfeszültség hányadosa, melyet azonban a kontroller a kimenet értékéből számol vissza. Ha a kimenet értéke nem megfelelő, mert valami „elhúzza” annak feszültségét, akkor a korrekció nem lesz eredményes, és a korrekciós együttható nem stabilizálódik, hanem vagy 0-hoz, vagy végtelenhez tart. A korrekciós együttható értékét ezért korlátozni kell, ha kisebb, mint 0,5 vagy nagyobb, mint 2, akkor az 1 értéket kapja, és a kártya hibát jelez, kigyújt egy piros ledet. Ez a két érték, ha a kimenet terheletlen, és megfelelően működik, akkor a 2,5 V illetve a 10 V-os tápfeszültség értékhez tartozna, ami már mindenképpen hibás működést jelent.

Tápfeszültségfüggő kimenetű szenzor szimulációja esetén a korrekciós együtthatót még meg kell szorozni a motorvezérlőtől kapott tápfeszültség és az ideális 5 V hányadosával, illetve kimenet mérésénél a mért értéket a korrekciós együttható számítása előtt ezzel el kell osztani.

4.2.2.2. Digitális hullámforma kimenet megvalósítása

A diszkrét feszültségszintekkel folytatott digitális kommunikáció előnye az analóggal szemben, hogy sokkal kevésbé zavarérzékeny, ezért azoknál a szenzoroknál, ahol valamilyen változás bekövetkezési idejének pontos ismerete nagy jelentőséggel bír, vagy a jelet a szenzortól távol kell feldolgozni, gyakran alkalmazzák. Hátránya viszont, hogy míg az analóg szintekkel kommunikáló szenzoroknál az AD-átalakító egy „univerzális dekódoló", addig a különféle digitális protokollok célhardvert igényelnek, vagy a szoftvert terheli értelmezésük, ami ugyanakkor előnyt is jelenthet, hiszen a szoftver elkészítése egyszeri költség. Az autóban viszont a gyorsan terjeszkedő elektronika számos zavarforrást is jelent, az árnyékolás megoldása viszont költséges lenne, így gyakran a digitális megoldás mellett döntenek.

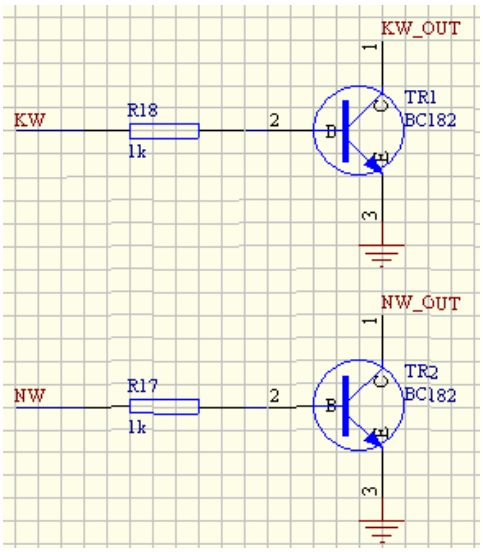
A PDTDI motormenedzsmentben a *vezérműtengely-jeladó*, *főtengely-jeladó* és az *olajszint- és –hőmérséklet-jeladó* kommunikál digitális szintekkel. Mindegyik esetben érthető a választás, hiszen a vezérműtengely és a főtengely jeladók jelei alapján pozíció,- illetve fordulatszám-értékeket számít a motorvezérlő, melyek alapján többek között a befecskendezési időpont, illetve hely kerül meghatározásra. A pozíciók pontos ismerete teljesítmény-növekedést és fogyasztáscsökkenést eredményezhet, a digitális jel határozott szintváltásaival ezek az információk pedig jól továbbíthatóak. Az olajszint- és hőmérséklet-jeladó pedig a motor alján, az olajteknőben helyezkedik el, jelét pedig az utastérbe, a műszerfalmodulhoz kell továbbítani, analóg jelszint csak nagy zajjal terhelve, vagy költséges árnyékolt kábellel lenne megfelelően vezethető.

4.2.2.2.1. A kimeneti paraméterek meghatározása

A vizsgált digitális jeladók open-collector/drain kimenettel rendelkeznek, melyeket a motorvezérlő illetve a műszerfalmodul felől felhúzóellenállás fogad. Ennek értéke 5-10 kΩ, és a 5, illetve 12 V-ra húzzák fel a kikapcsolt kimeneti tranzisztor collektor/drain feszültségét, így a tranzisztornak csak kis áramot kell elviselnie bekapcsolt állapotban is. A maximális kimeneti frekvenciát a főtengely-jeladónál kell biztosítani. Itt egy jeladótárcsán 58 mágneses jelzés található, ami a maximálisan szimulálni kívánt $8000\frac{1}{\text{min}}$ fordulatszámon kb. 8kHz-es négyszögjelet eredményez.

4.2.2.2.2. A digitális hullámforma kimenet kapcsolási rajza

A digitális kimenet kapcsolása elég egyszerűen megvalósítható, hiszen a mikrokontrollerhez csak egy tranzisztort kell illeszteni, a többi a szoftver dolga. A mikrokontroller és a működéshez szükséges egyéb külső elemek az analóg feszültségkimenet részben ismertetettel azonosak, így annak kapcsolási rajzát nem közlöm még egyszer.



4.19. ábra Az open-collectoros digitális kimenetet megvalósító kapcsolás

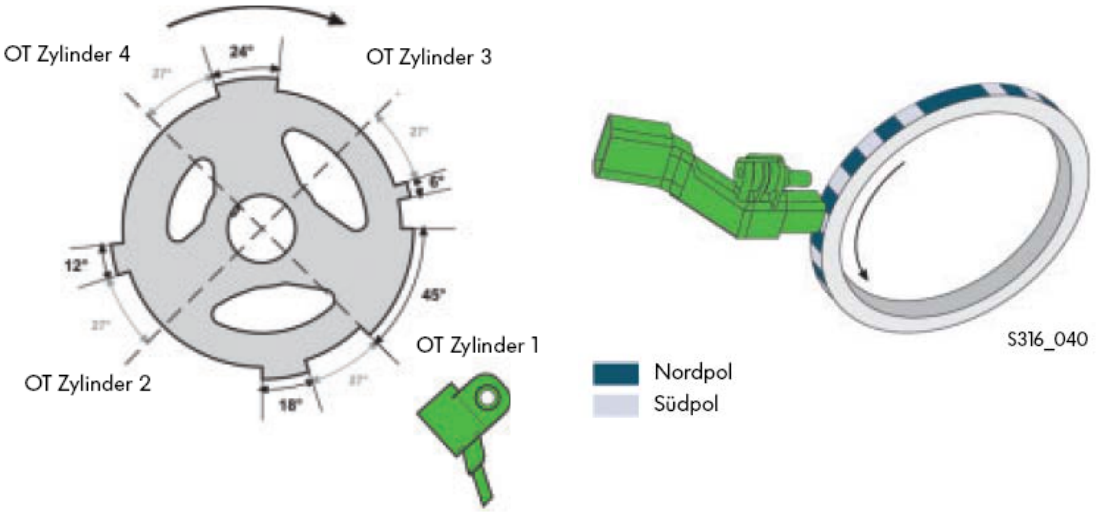
A választott tranzisztor a gyakran alkalmazott olcsó BC128, mely rendelkezik a szükséges paraméterekkel. Egy kártyán két ilyen kimenet kialakítása célszerű a következő megfontolás alapján:

Bár a főtengely és a vezérműtengely jeladó külön szenzorok, kimeneti jeleik szinkronban kell, hogy legyenek, ellenkező esetben ez azt jelentené, hogy a 2:1 fordulatszámarány felborult, ami mechanikai hibára utal, melynek hatására a motorvezérlő azonnal leállítaná a motort. Ha a két szenzor külön kártyán lenne megvalósítva, akkor a szinkron biztosítása miatt a két kártyának egymással is kommunikálnia kellene, ami persze megoldható, de a feladat felesleges bonyolítását jelentené. Sokkal egyszerűbb tehát, ha egy mikrokontroller szoftver valósítja meg mindkét jelet.

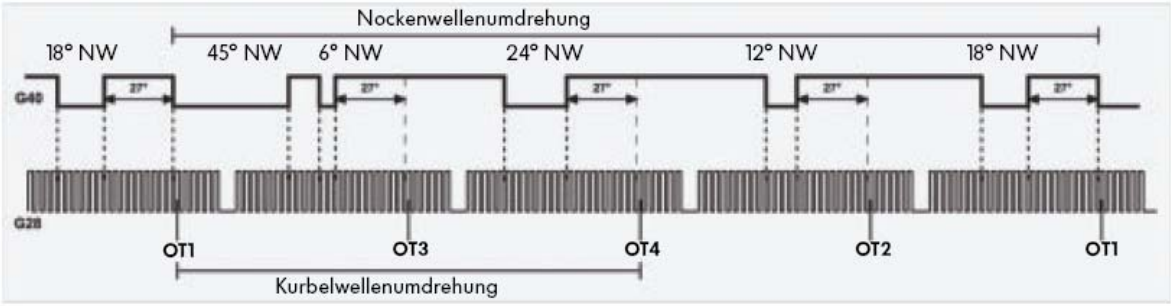
4.2.2.2.3. A digitális hullámforma kimenet szoftver terve

4.2.2.2.3.1. A főtengely- és vezérműtengely-jeladók szimulációja

A két szenzor szimulációját tehát egy kártyával, egy szoftverben kell megvalósítani. A következő ábrákon látható a két szenzor jeladótárcsája, és a szenzorok kimenetén megjelenő hullámforma a főtengely két fordulata alatt:



4.20. ábra A vezérműtengely és főtengely jeladók és a hozzájuk tartozó kódtárcsa



4.21. ábra A vezérműtengely és főtengely jeladók kimenetén megjelenő hullámforma a vezérműtengely egy fordulata alatt.

A kártyának tehát a kimeneteken ezeket a hullámformákat reprodukálnia, változtatható frekvenciával. A megoldáshoz két lehetséges programfelépítés között gondolkodtunk:

- Mivel a főtengely-jeladó kimeneti jelalakja eléggé közelíti a szabályos négyszögjelet, ami a mikrokontroller PWM moduljával könnyen megvalósítható, célszerű lehet ezt felhasználni: a négyszögjel periódusait számolva, megfelelő számú

periódusonként a kimenetet letiltva előállítható a főtengely-jeladó kimenete. A vezérműtengely-jeladó jóval ritkább beavatkozást kíván, ennek szintváltásait a főtengely-jeladó négyszögjel-periódusainak sorszámához lehetne szintén kötni, és egy I/O lábat eszerint vezérelni. A fordulatszám a PWM frekvenciájával lenne arányos.

- A másik megoldás, ha mindkét jel hullámformáját letároljuk a kontrollerben, és bizonyos időközönként kitesszük a következő mintát a kimenetekre. Az időt a timer segítségével lehetne mérni, a nagyobb fordulatszámhoz kisebb idő tartozik.

A megvalósításban a második megoldás mellett döntöttünk. Ennek oka, hogy jóval átláthatóbb programfelépítést eredményez, és sokkal egyszerűbben teszi lehetővé egy másik jeladótárcsájú szenzorra való áttérést, hiszen csak a letárolt hullámformákat kell megváltoztatni, míg az előbbi felépítésnél egy teljesen újszerű hullámforma nem illeszkedik az algoritmusba. A konkrét megvalósítás tehát a következőképpen történt:

A timer modul bizonyos, a kívánt fordulatszámból kiszámított időközönként interruptot generál. Az interrupt rutin feladata, hogy a flash-ben tárolt hullámforma-vektorból a következő értéket kiolvassa, és annak inverzét kitegye a kimenetre. A főprogram végtelen ciklusban várja a PC-n futó felhasználói programtól az új fordulatszám értékeket.

A motor fordulatszáma azonban nem változhat tetszőleges sebességgel, biztosítani kell, hogy a vezérlő szoftvertől érkező fordulatszám-ugrásjel hatására a motorvezérlő felé egy fokozatosan, adott idő alatt létrejövő változást hozzunk létre.

Az időben egyenletes fordulatszám-változtatást azonban két körülmény is nehezíti:

1, A fordulatszám hullámforma következő mintája mindig Timer Overflow Interruptnál kerül a kimenetre. A Timer maximális értéke, melynek elérésekor az Interrupt-kérés létrejön, határozza meg a fordulatszámot:

$$\mathrm{rpm} = \frac{\mathrm{F\_CPU} \cdot 120}{\mathrm{L} \cdot \mathrm{Top}} \tag{1.6}$$

ahol

Rpm: a fordulatszám [1/min]

F_CPU: mikrokontroller órajel frekvencia

L: a hullámformát tároló vektor hossza

Top: a Timer maximális értéke, melynek elérésekor interrupt-kérés jön létre

Az összefüggésből látható, hogy a Top értéke és a fordulatszám között fordított arányosság van.

2, A másik nehézséget az okozza, hogy ha a Top értékét a fordulatszám változtatásához az interrupt rutinban módosítjuk, akkor figyelembe kell venni, hogy a megszakítások között eltelt idő is függ Top értékétől:

$$\triangle t = \frac{Top}{F\_CPU} \tag{1.7}$$

Ahhoz, hogy a változás lehetőleg egyenletes legyen, szükség van egy olyan függvény előállítására, mely mentén változtatva a Top értékeket a fordulatszám lineárisan változik.

Tehát a cél:

$$\frac{\triangle rpm}{\triangle t} \cong const \tag{1.8}$$

Mivel a fordulatszámot a kontrollerben csak diszkrét lépésekben tudjuk változtatni, a feltételt is diszkrét lépésekre kell megfogalmazni:

$$\left.\frac{\triangle rpm}{\triangle t}\right|_{n-1\to n} \cong \left.\frac{\triangle rpm}{\triangle t}\right|_{n\to n+1} \tag{1.9}$$

$$\Delta rpm = \frac{F\_CPU \cdot 120}{L \cdot Top_n} - \frac{F\_CPU \cdot 120}{L \cdot Top_{n-1}} \tag{1.10}$$

$$\Delta t = \frac{Top_{n-1}}{F\_CPU} \cdot k \tag{1.11}$$

k: minden k-adik interrupt rutinban változtatjuk a Top értékét

$$\left.\frac{\Delta rpm}{\Delta t}\right|_{n-1\to n} = \frac{\frac{F\_CPU\cdot 120}{L\cdot Top_n} - \frac{F\_CPU\cdot 120}{L\cdot Top_{n-1}}}{\frac{Top_{n-1}}{F\_CPU}\cdot k} \tag{1.12}$$

$$d_n \equiv Top_{n-1} - Top_n \tag{1.13}$$

ahol d_n az n-edik lépésben szükséges változtatás top értékén.

A feltétel (1.9) és (1.12) alapján:

$$\frac{\frac{F\_CPU\cdot 120}{L\cdot Top_n} - \frac{F\_CPU\cdot 120}{L\cdot (Top_n + d_n)}}{\frac{Top_{n-1}}{F\_CPU}\cdot k} = \frac{\frac{F\_CPU\cdot 120}{L\cdot (Top_n - d_{n+1})} - \frac{F\_CPU\cdot 120}{L\cdot Top_n}}{\frac{Top_{n-1}}{F\_CPU}\cdot k} \tag{1.14}$$

Egyszerűsítve:

$$\left[\frac{1}{Top_n} - \frac{1}{Top_n + d_n}\right]\cdot Top_n = \left[\frac{1}{Top_n - d_{n+1}} - \frac{1}{Top_n}\right]\cdot (Top_n + d_n) \tag{1.15}$$

Fejezzük ki $d_{n+1}-$et, ezáltal egy rekurzív formulát kapunk a lineáris változáshoz szükséges Top vektor szomszédos elemei különbségének kiszámításához:

$$d_{n+1} = Top_n - \frac{Top_n + d_n}{1 - \frac{Top_n}{Top_n + d_n} + \frac{Top_n + d_n}{Top_n}} \tag{1.16}$$

Látszik, hogy a kapott összefüggés nem függ k értékétől, azaz a változási sebességet k értékének változtatásával befolyásolhatjuk, anélkül, hogy a lineáris felfutást elveszítenék.

Legyen az átlagos maximális fordulatszám változási sebesség :

$$\left.\frac{\Delta \mathrm{rpm}}{\Delta \mathrm{t}}\right|_{\max} \approx 5000 \frac{1}{\mathrm{min} \cdot \mathrm{sec}} \tag{1.17}$$

A maximális kívánt fordulatszám:

$$\mathrm{rpm}|_{\max} = 8000 \frac{1}{\mathrm{min}} \tag{1.18}$$

Bár a konkrét megvalósításban szereplő Diesel-motor fordulatszáma ennél csak szűkebb tartományban mozog (500-5000 $\frac{1}{\mathrm{min}}$), a más motortípusokkal való kompatibilitás biztosítása érdekében szükséges a szélesebb tartomány.

A timer modul által biztosított fordulatszám-felbontás, azaz, hogy mekkora az a legkisebb lépés, amivel a fordulatszám változtatható a következőképpen írható fel:

$$\Delta \mathrm{rpm} = \frac{\mathrm{F\_CPU} \cdot 120}{\mathrm{L} \cdot (\mathrm{Top} - 1)} - \frac{\mathrm{F\_CPU} \cdot 120}{\mathrm{L} \cdot \mathrm{Top}} \tag{1.19}$$

Az összefüggésből látszik, hogy ez is változik Top értékével. Leolvasható továbbá, hogy minél nagyobb a Top értéke, annál nagyobb lesz a felbontás is, azaz annál kisebb az egységnyi változtatásból eredő fordulatszám-változás. Tehát cél, hogy Top értékét a változtatáshoz képest magasan tartsuk a működési tartományban.
(1.6) alapján ez F_CPU nagyra választásával oldható meg.

A nagy F_CPU viszont az alsó fordulatszámhatár megnövekedéséhez vezet, amit a maximális Top érték korlátoz, melyet viszont csak szoftveresen lehet bővíteni, vagy a timer előosztó ügyes átkapcsolásával lehet megoldani.

$$F\_CPU = 18,432\text{MHz} \quad L=240 \quad \text{Top}\big|_{max} = 2^{16} - 1 \tag{1.20}$$

$$\text{rpm}\big|_{min} = \frac{18,432\text{MHz} \cdot 120}{240 \cdot 65536} = 140,625 \frac{1}{\text{min}} \tag{1.21}$$

Legyen ez a kiindulási fordulatszám.

A timer felbontása a legrosszabb esetben, azaz ha rpm=8000$\frac{1}{\text{min}}$ a fenti paraméterekkel:

$$\text{Top}\big|_{\text{rpm}=10000} = \frac{F\_CPU \cdot 120}{L \cdot 8000} \cong 1152 \tag{1.22}$$

$$\Delta\text{rpm}\big|_{\Delta\text{Top}=1} = \frac{F\_CPU \cdot 120}{L \cdot (1152-1)} - \frac{F\_CPU \cdot 120}{L \cdot 1152} = 6.95 \approx 7 \tag{1.23}$$

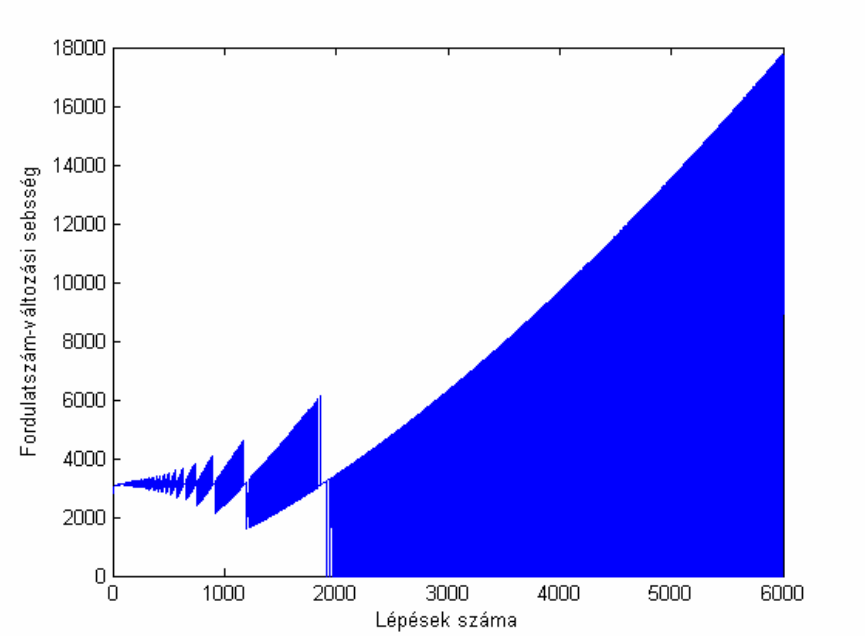
Ennél kisebb lépcsőkben talán nem is kell változtatni a fordulatszámot, a tapasztalatok alapján a motorvezérlő ennél sokkal nagyobb ugrásokat is hibajelzés nélkül kezel.
A magasabb fordulatszám-tartományban azonban, mivel egyre kisebb idő telik el két interrupt között, amikor a fordulatszámot változtatjuk, és a felbontás is csökken, maga a változási sebesség, amit egy lépés létrehoz, rohamosan nőni fog. Ahhoz, hogy magasabb fordulatszámon is tartani lehessen az átlagos változási sebességet, csak minden második, harmadik, n-edik ütemben lehet eggyel változtatni TOP értékét, ennek azonban a rekurzív formulával automatikusan adódnia kell!

A rekurzív formula kiértékelése egy MATLAB szkripttel történt, melynek bementi paraméterei az első és a második ütemben beállítani kívánt fordulatszám. E két adat a

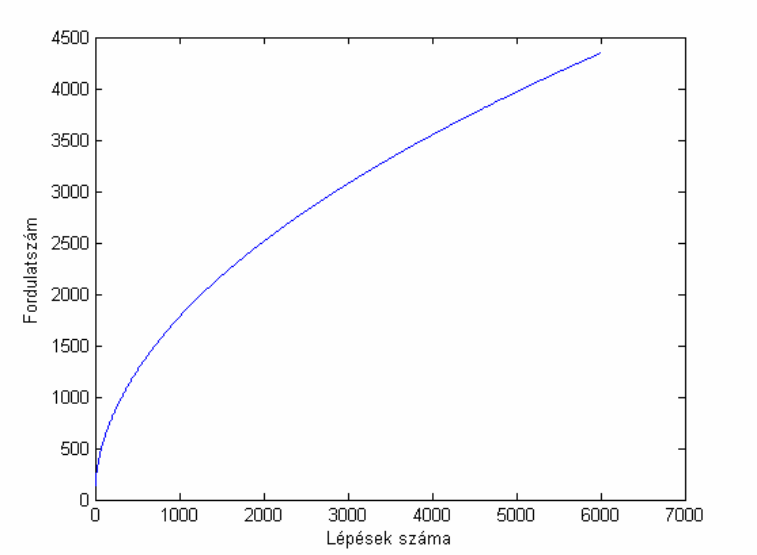
fordulatszám-változási sebességet is tartalmazza, hiszen a két lépés között a fordulatszámmal fordítottan arányos idő telik el, mivel az interrupt rutinban változtatunk.

Legyen a kezdeti fordulatszám az alsó határ, ami kb. $141\frac{1}{\min}$, a lépésköz pedig $10\frac{1}{\min}$.

Ekkor a lépések számának függvényében a következő fordulatszám-változási sebesség diagrammot kapjuk:



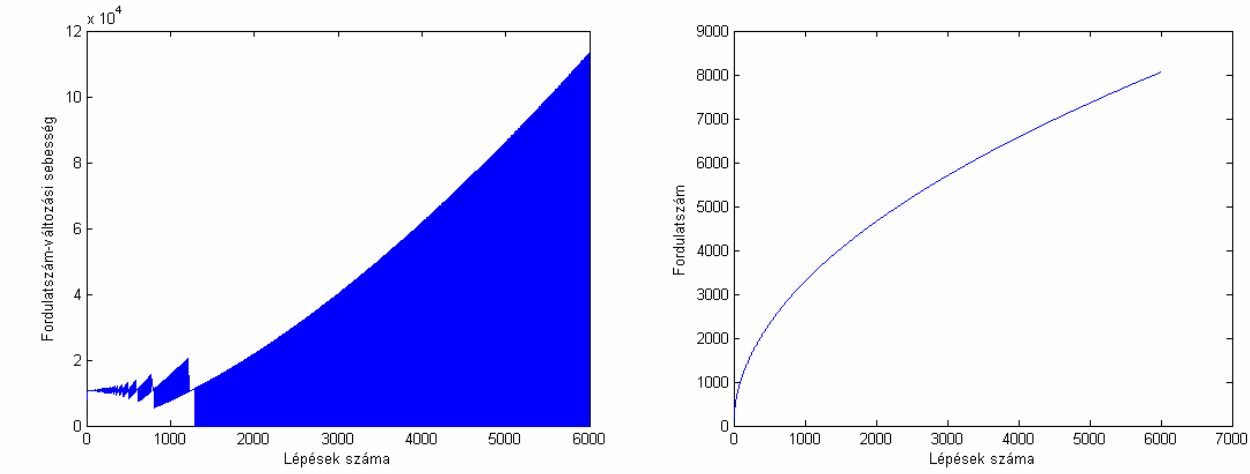
4.22. ábra A generált vektorral elérhető fordulatszám-változási sebesség a vektorindex függvényében



4.23. A fordulatszám alakulása a kapott vektor indexei függvényében

Ekkor az átlagos változási sebesség $3200\frac{1}{\min\cdot\sec}$. Az első ábrán látható, hogy az egyes lépéseknél a változási sebesség korántsem állandó, és a vége felé pedig egyre jobban oszcillál, amire számíthattunk is: mivel nagyobb fordulatszámon egyre nagyobb lépésekben lehet csak változtatni, hogy az átlagos sebesség állandó legyen, egyre több lépésnél a változási sebesség 0. A második ábrán a fordulatszám alakulása látható a lépések függvényében. Itt nem egyenest vártunk, hiszen az egyes lépések a kontrollerben egyre kisebb időközönként követik egymást. Látható, hogy csak a 6000. lépés után jutunk el egyáltalán a $4500\frac{1}{\min}$ fordulatszámig. Ez azért probléma, mert a mikrokontrollerben nincs elég idő két interrupt között, hogy a formulával a következő értéket kiszámítsuk, az eredményül kapott vektor pedig nem illeszkedik semmi olyan függvénye, ami szintén elég gyorsan kiszámítható lenne, így le kell tárolni a kontrollerben, melynek kapacitása azonban véges (8 kbájt), és a program mellett kb. 6 kbájt áll rendelkezésre erre a célra.

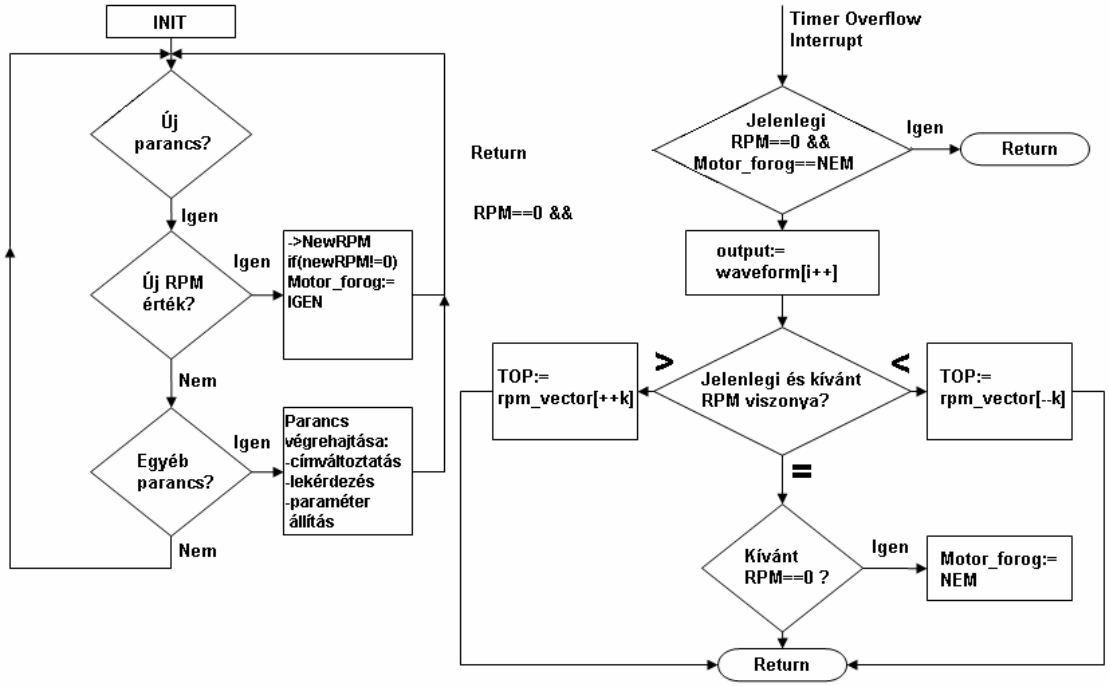
Növeljük a kiindulási lépésközt addig, hogy az átlagos változási sebesség $10000\frac{1}{\min\cdot\sec}$ legyen. Ekkor, ha minden második interrupt rutinban változtatunk (k=2), akkor visszakapjuk a kívánt $5000\frac{1}{\min\cdot\sec}$ sebességet, persze kisebb felbontással, de így nem ütközünk a kontroller memóriakorlátjába:



4.24. ábra A generált vektorral elérhető fordulatszám-változási sebesség és a fordulatszám alakulása a vektorindex függvényében

Az ábrán látható, hogy így éppen elértük a kívánt maximális fordulatszámot is. Nagyobb kezdeti lépésköz változtatása pedig azért nem célszerű, mert már így is magasabb fordulaton nagyon nagy a pillanatnyi változási sebesség.
Hogy a 6000 elemű fordulatszámvektor elférjen a kontroller memóriájában, egy lépés nem lehet több 1 bájtnál. Ez úgy érhető el, hogy nem magukat a fordulatszám értékeket tároljuk el, hanem a szomszédos értékek különbségét, ami az első 30 érték után már kisebb mint 256.

A szoftver egyszerűsített folyamatábrája:



4.25. ábra A fordulatszám jeladókat szimuláló mikrokontroller szoftver egyszerűsített folyamatábrája

Jelenleg a program kétféle parancs fogadására képes:

- a kimeneti értéket módosító parancs a PC felhasználói programtól
- lekérdezést, illetve címmódosítást kezdeményező parancs a kártyamenedzsment programtól

A kártya a felhasználói programtól minden esetben 6 bájtot kap, melyeket a következőképpen értelmez:

Az első 5 bájt jelentését a 6. bájt tartalma határozza meg. Ha ennek értéke 0xFA, akkor az adatok nem a kimenetre, hanem a kártya egyéb paramétereinek módosítására szolgál, ahol:

1. bájt: a kártya új címe, mely reset után érvényes
2-6. bájt: további fejlesztésekre fenntartva

Ha a 6. bájt értéke 0x00, akkor a parancs új kimeneti értéket tartalmaz:
1. bájt: Fordulatszám érték felső bájt
2. bájt: Fordulatszám érték alsó bájt
3-6. bájt: további fejlesztésekre fenntartva

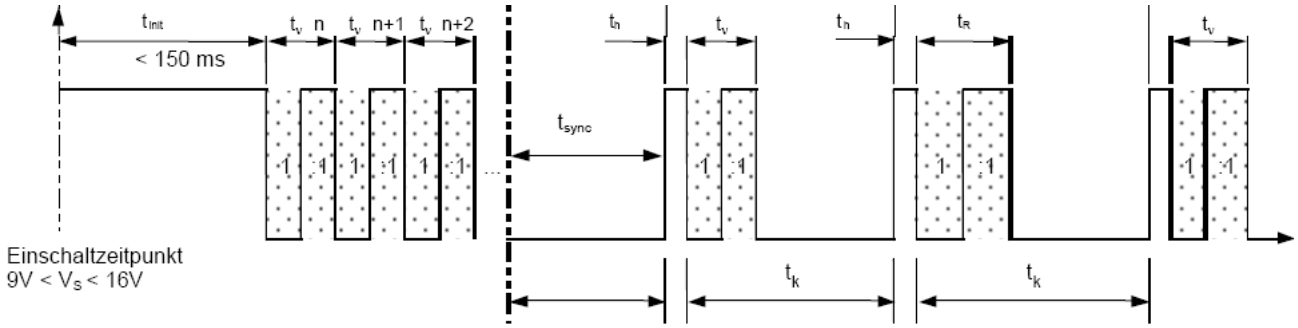
A PC felhasználói programja a mikrokontroller terhelését csökkentendő nem a beállítani kívánt fordulatszám értéket küldi el közvetlenül, hanem az azzal arányos TOP értéket, mely a következő képlet szerint számítható:

(1.24) $$\mathrm{Top} = \frac{\mathrm{F\_CPU} \cdot 120}{\mathrm{L} \cdot \mathrm{rpm}}$$

4.2.2.2.3.1. Az olajszint- és hőmérséklet-jeladó szimulációja

Az olajszint- és hőmérséklet-jeladó a motorolaj pontos mennyiségének és hőmérsékletének folyamatos monitorozására szolgál. Az általa küldött adatok alapján a műszerfalmodul a terhelés, a fordulatszám, és a menetidő, mint további paraméterek naplózásával egy integrálformula segítségével következtet a motorolaj összigénybevételére, és ez alapján annak állapotára. A szenzor a hőmérsékletet egy NTC ellenálláselem segítségével méri, az olajszintet pedig egy ultrahang adó-vevő határozza meg. A két mérés eredményét ciklikusan egy digitális interfészen keresztül küldi a PC felé. A szintmérés elve miatt a szenzor a PULS nevet kapta, így továbbiakban így hivatkozom rá.

A kommunikációs protokoll a következő időidiagram szemlélteti:



4.26. ábra A PULS szenzor kommunikációs protokollja

Amint a tápfeszültség eléri a működéshez szükséges értéket, megkezdődik az inicializálás. Ennek maximális hossza 150ms, a mérési eredmények általában 15-30ms-ot mutatnak. Ezután megkezdődik a mérési eredmények küldése. Az inicializálás után 3 hőmérsékletadat kerül átvitelre, majd a „szinkron-idő" után megkezdődik az adatok rendes, ciklikus küldése. Az egyes mért értékeket mindig valamilyen periódushossz reprezentálja. A hőmérséklet értéket a t_v-vel jelölt intervallumhossz hordozza, mely egy szimmetrikus négyszögjel egy periódusának hossza. A három hőmérsékletérték után az átvitel a következőként történik: A ciklus egy impulzussal kezdődik, melynek hossza állandó, $t_h = 80\text{ms}$. Ezután minden páratlan ciklusban a hőmérsékletet jelentő szimmetrikus négyszögjel következik, minden páratlan ciklus pedig egy $t_R = 100\text{ms}$ hosszúságú referencia négyszögjel. A fogadó oldalon a referencia intervallum valós hossza, és a 100ms elméleti érték hányadosa adja a két egység órajelének arányát, ami az egyes időadatok korrekciójának számításához szükséges. Az ábrán t_K-val jelölt intervallum hordozza az olajszint mérőszámát, mely minden ciklusban megjelenik, és a t_h impulzus végétől a következő t_h impulzus kezdetéig tart.

Az egyes időintervallumok hossza, és az általuk reprezentált mennyiségek közti összefüggés a következő:

Olajszint:

$$t_k = -7,5\left(\frac{\text{ms}}{\text{mm}}\right)\cdot h + 1050(\text{ms}) \qquad (1.25)$$

ahol h az olajszint.

Hőmérséklet:

$$T(C°) = a_5 \cdot t_v^{\,5} + a_4 \cdot t_v^{\,4} + a_3 \cdot t_v^{\,3} + a_2 \cdot t_v^{\,2} + a_1 \cdot t_v + a_0 \qquad (1.26)$$

ahol a_i: $a_{5=}$2,88969E-07 $a_{4=}$-7,73536E-05

$a_{3=}$8,34974E-03 $a_{2=}$-4,69415E-01

$a_{1=}$1,71846E+01 $a_{0=}$-2,87401E+02

A hőmérséklet-kiértékelő polinom inverzének meghatározásával, mely Matlabban a *polyfit* parancs segítségével történt, a szimulálandó t_v idő is számítható. Mivel az egyes intervallumok elég hosszúak, köztük elegendő idő áll rendelkezésre a mikrokontrollernek, hogy a polinom alapján a szükséges időket saját maga számítsa ki, így a PC-től lehetőség van a hőmérséklet és szintértékek változtatás nélküli elküldésére is, ezt a kísérletek is igazolták. A végleges verzióban azonban az adatforgalom egységesítése érdekében a többi szenzornál megszokott módon itt is a szenzor kimenetén megjelenő közvetlen fizikai paraméter értékét, azaz jelen esetben az időintervallumok hosszát kapja meg a kártya. Ezáltal egyszerűbben megvalósítható a más motortípusokon alkalmazott, azonos protokollú, de különböző kiértékelő polinommal rendelkező szenzorok szimulációja.

A szoftver felépítése a következő:

A program a PC-től a parancsokat I2C buszon keresztül fogadja, melyek a t_v és t_k időket tartalmazzák. A timer modul az inicializálást követően 0,1ms-onként interruptot generál. A kimeneti hullámforma előállítása ebben az interrupt rutinban egy állapotgép segítségével zajlik, melynek főbb állapotai az idődiagram alapján:

- inicializásás;
- 3 hőmérsékletfázis;
- „szinkron-idő”;
- páratlan ciklus;
- páros ciklus.

A kapott parancsok felépítése:
A kártya a felhasználói programtól minden esetben 6 bájtot kap, melyeket a következőképpen értelmez:
Az első 5 bájt jelentését a 6. bájt tartalma határozza meg. Ha ennek értéke 0xFA, akkor az adatok nem a kimenetre, hanem a kártya egyéb paramétereinek módosítására szolgál, ahol:

1. bájt: a kártya új címe, mely reset után érvényes

2-6. bájt: további fejlesztésekre fenntartva

Ha a 6. bájt értéke 0x00, akkor a parancs új kimeneti értéket tartalmaz:

1. bájt: t_v hőmérséklettel arányos időérték felső byte

2. bájt: t_v hőmérséklettel arányos időérték alsó byte

3. bájt: t_k olajszinttel arányos időérték felső byte

4. bájt: t_k olajszinttel arányos időérték alsó byte

5-6. bájt: további fejlesztésekre fenntartva

A egyes értékek 1 tizedesjegy pontossággal ms-ban kerülnek átvitelre, úgy hogy az adatok a tizedesre kerekített érték tízszeresét (így egész számot) tartalmazzák.

4.2.2.3. Ellenálláskimenet megvalósítása

A motor üzemállapotának felismeréséhez, a pontos üzemanyag-adagoláshoz, a kipufogógáz normák betartásához számos hőmérsékletre van szűksége a motormenedzsmentnek. Ezek a szenzorok általában valamilyen hőmérsékletfüggő ellenállást alkalmaznak az adott közeg hőmérsékletének meghatározásához. A szenzor maga gyakran csak ezt az ellenállást tartalmazza. A kiegészítő áramkörök a motorvezérlőben vannak.
A szenzorszimuláció feladata itt az adott szenzornak megfelelő ellenállásértékek megjelenítése a motorvezérlő felé. Ehhez valamilyen elektronikusan változtatható ellenállásra van szükség. Az ellenállás-tartomány, melyben a szimuláció szükséges, figyelembe véve az összes ismert hőmérséklet jeladó karakterisztikáját kb. 70 Ω-50 kΩ. Ez egy elég széles tartomány, három nagyságrend átfogással. Az elvárt pontosság elsősorban a szenzorok karakterisztikájától függ. Mivel nem a motorvezérlő vizsgálata a cél, ezért az egyszerűbb megvalósítás érdekében kisebb pontossággal is megelégedhetünk.

4.2.2.3.1. A kimeneti paraméterek meghatározása

A kipufogógáz hőmérséklet jeladók PT-200 platina érzékelőt használnak, mely -40 C°-tól 1000 C°- használható, és ebben a tartományban ellenállása 170 Ω és 850 Ω között jó közelítéssel lineárisan változik. Itt 1 C° változás kb. 0,65 Ω ellenállás-változást jelent. Ez az 1C° pontosság csak nagyon nehezen lenne teljesíthető, és értelme sem igazán lenne, mivel a kipufogórendszerben nagyon gyors hőmérsékletváltozások mennek végbe, a gáz hőmérséklete egy másodperc alatt több száz fokot is változhat. A szenzor adatlapja a pontosságot hőmérsékletben definiálja: -40-600 C° : ± 10 C°, 600-1000 C° : ± 20 C°. Így megelégedhetünk mi is kb. ezzel a pontossággal, ami ellenállásban az alsó tartományban $\pm 6,5$ Ω, a felső tartományban ± 13 Ω-ot jelent, ami már sokkal inkább biztosítható.

A hűtővízhőmérséklet-jeladó NTC-t tartalmaz, melynek ellenállása a -10-120 C° tartományban 9 kΩ és 120 kΩ között változik, közel sem lineárisan. A kívánt pontosságot ennél a szenzornál a karakterisztika legkisebb meredekségű részénél kell meghatározni. Szerencsére ez a karakterisztika vége, mely a kisebb ellenállásértékekhez közelít. Itt 1 C° hőmérsékletváltozáshoz 3 Ω ellenállás-változás tartozik. 90 C°-on ez az érték már 10 $\frac{\Omega}{\text{C°}}$, kisebb hőmérsékleten egyre nő. Az adatlap a pontosságot százalékban adja meg az ellenállás értékére, ami a legkevésbé meredek szakaszon a legszigorúbb: $\pm 5,6\%$. Ez ebben a tartományban $\pm 6,7\,\Omega$ pontosságot jelent. Az üzemanyag hőmérséklet jeladó is ebbe a kategóriába tartozik. Szintén NTC ellenállással méri a hőmérsékletet, ellenállása azonban 340 Ω és 48,5 kΩ között változik, a leglaposabb szakaszon $14,5\,\frac{\Omega}{\text{C°}}$ sebességgel.

A jeladókat a motorvezérlő mindig egy ellenállásosztó alsó tagjaként, feszültséggenerátorral hajtja meg. A szenzorok adatlapjai alapján a maximális megengedett mérőáram 1 mA.

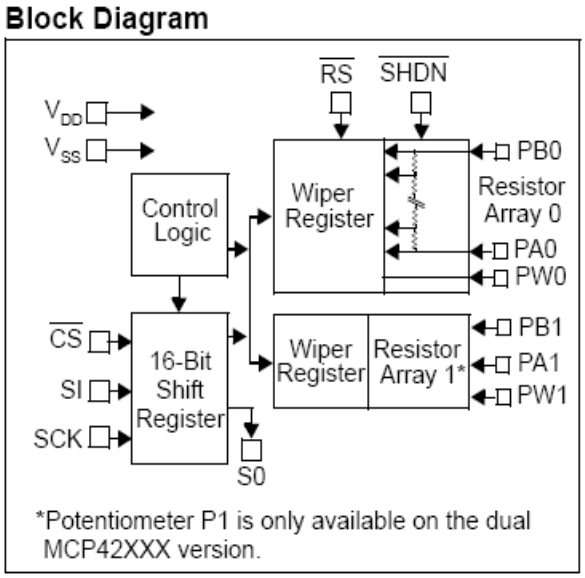
4.2.2.3.2. Az ellenálláskimenet kapcsolási rajza[9]

A kimenetet megvalósító kapcsolással szemben támasztott követelmények tehát a következők:

- Szimuláció az adott szenzor teljes ellenállástartományában 5,5-6% pontossággal;
- max. 1 mA terhelhetőség.

A probléma vagy valamilyen félvezető elem megfelelő vezérlésével oldható meg, vagy az ehhez hasonló elven működő digitális potenciométerek segítségével. A várhatóan nagyobb stabilitás és pontosabb működés miatt utóbbi mellett döntöttünk.

A digitális potenciométer tulajdonképpen egy ellenállás-vektort tartalmazó integrált áramkör, ahol a vektor két végpontja a potenciométer két kivezetése, a csúszka pedig a létra valamelyik fokához csatlakozik a „wiper regiszter" értéke alapján. A kapcsolatot a csúszka és a létra valamelyik foka között térvezérelt tranzisztorok hozzák létre, vagy szakítják meg, így ezek vezérelhetőségének biztosítása miatt a potenciométerekre csak a tápfeszültség-tartományon belüli feszültségek kerülhetnek.
A kereskedelemben számos típus megtalálható, különböző ellenállásértékekkel, terhelhetőséggel és felbontással. A terhelhetőség és a megfelelő ellenállásértékek könnyen, szinte bármelyik család tagjaival megvalósíthatóak lennének, a pontosság azonban kritikus követelménynek tűnt: 8 bitnél nagyobb felbontást találni nem sikerült, ami pedig 10 kΩ-os ellenállásnál 40 Ω-os lépéseket tesz lehetővé. A megoldást ezért egy olyan IC-től vártuk, melynél egy tokban két potenciométer is rendelkezésre áll, melyeket párhuzamosan kapcsolva, és megfelelően vezérelve nagyobb felbontást érhetünk el. A választás az adatlapok áttanulmányozása után az itthon is forgalomban lévő Microchip MCP42XXX családra esett, mely a következő tulajdonságokkal, és az ábrán látható felépítéssel rendelkezik:



4.27. ábra Az MCP42xxx digitális potenciométer család felépítése

Jellemzői:

- 2db független, megszakítható, 3 kivezetéses potenciométer;
- 8 bit felbontás;
- max. 1 mA terhelhetőség;
- 5-10-50 kΩ ellenállású típusok;
- SPI kommunikációs interfész

A választott mikrokontroller rendelkezik SPI interfésszel, így a kommunikáció ezen keresztül megvalósítható. Az SPI protokoll egy egyszerű soros adatátvitelt valósít meg, master-slave felépítésben, ahol a master szolgáltatja az órajelet. A kommunikációban résztvevő perifériák rendelkeznek egy adatkimenettel és egy –bemenettel. A bemeneten az adatokat egy shiftregiszter fogadja, mely az órajel hatására a bemeneten beléptet egy bitet a regiszter első rekeszébe, a kimeneten pedig kilépteti a regiszter utolsó bitjét. Ezáltal a busz lehetővé teszi, hogy elvileg végtelen számú perifériát ily módon láncba fűzzünk.

A választott digitális potenciométer egy 16 bites shiftregiszterrel rendelkezik, mely egy parancs- és egy adatbájtot vár a master módú mikrokontrollertől.

A parancsbájt a következő információkat tartalmazza:

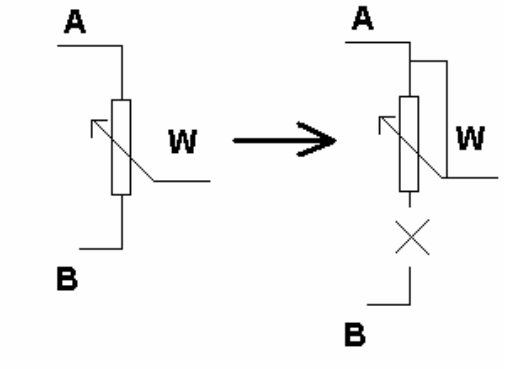
- funkcióválasztás:
  - wiper regiszter írása
  - potenciométer kikapcsolása
- parancs hatáskörének választása:
  - P0
  - P1
  - mindkettő potenciométer

Az adatbájt a parancsban meghatározott potenciométerre vonatkozó wiper regiszter érték, mely az adott potenciométer ellenállását a következő kapcsolatban szerint határozza meg:

$$R_{WB}(D) = \frac{R_{AB} \cdot D}{256} + R_W \qquad (1.27)$$

ahol R_{AB} a potenciométer névleges ellenállása, R_W a csúszka ellenállása, ami kb. 52 Ω

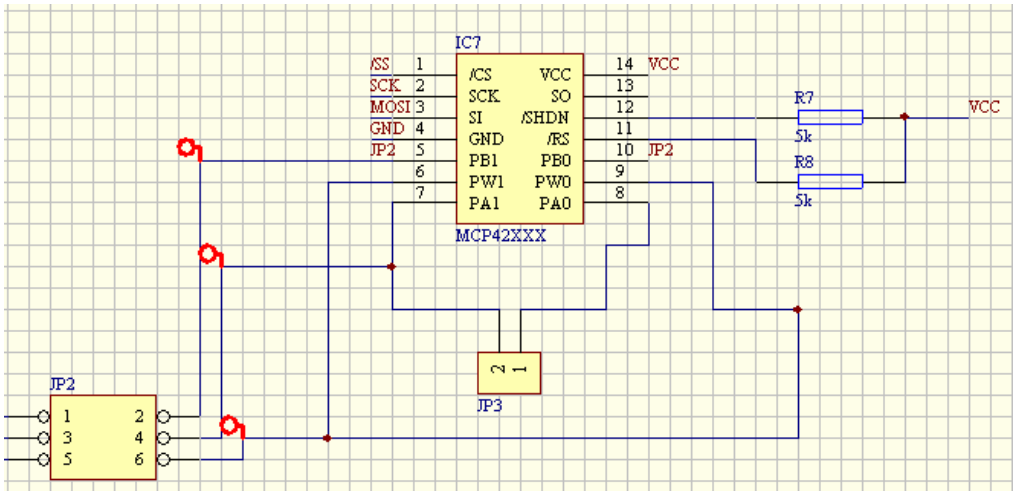
Egy potenciométer kikapcsolása a következő állapotváltozást jelenti:



4.8. ábra A digitális potenciométer viselkedése a shutdown parancs hatására

Ez a funkció leegyszerűsíti a párhuzamos kapcsolással történő felbontásnövelést, hiszen az külső kapcsoló elem nélkül csak a potenciométer néveleges ellenállásának feléig tenné lehetővé ellenállásértékek beállítását. Ha azonban a szenzort a W és B kivezetések közti ellenállás szimulálja, akkor a végérték felénél az egyik potenciométer kikapcsolása után, igaz akkor már csak az eredeti 8 bites felbontással, a teljes tartományban használható a potenciométer külső kapcsolóelem nélkül. A 10 kΩ-os típust példaként vizsgálva ez 5 kΩ felett 40 Ω-os lépéseket tesz lehetővé, ami itt 0,8%-os hibát jelent, ami még bőven teljesíti a követelményeket.

A digitális potenciométer kimenet kapcsolási rajza az analóg feszültségkimenet részben ismertetett mikrokontrolleres kapcsolás mellőzésével:



4.29. ábra A potenciométer kimenetet megvalósító áramkör kapcsolási rajza

A digitális potenciométer szintén 5V tápfeszültséget igényel. A kommunikáció az első 3 kivezetésen keresztül zajlik, /SS a chip select, melynek lefutó éle új parancs érkezésre készíti fel az IC-t, felfutó éle pedig az utasítás végrehajtását kezdeményezi az átvitel befejeztével. SCK a master által szolgáltatott órajel bemenete, MOSI az adatbemenet. (**M**aster **O**ut **S**lave **I**nput)

4.2.2.3.3. Az ellenálláskimenet szoftver terve

Az ellenálláskimenettel rendelkező kártya szoftverének feladatai:

- parancsok fogadása I^2C buszon keresztül;
- lekérdezés hatására információ küldése a PC felé;
- ellenállás beállítása a kapott parancsnak megfelelően, a párhuzamos ellenállások megfelelő beállításával

A program az inicializálás után egy végtelen ciklusban vár új parancs érkezésére. Jelenleg a program kétféle parancs fogadására képes:

- a kimeneti értéket módosító parancs a PC felhasználói programtól;
- lekérdezést illetve címmódosítást kezdeményező parancs a kártyamenedzsment programtól.

A kártya a felhasználói programtól minden esetben 6 bájtot kap, melyeket a következőképpen értelmez:

Az első 5 bájt jelentését a 6. bájt tartalma határozza meg. Ha ennek értéke 0xFA, akkor az adatok nem a kimenetre, hanem a kártya egyéb paramétereinek módosítására szolgál, ahol:

1. bájt: a kártya új címe, mely reset után érvényes

2-6. bájt: további fejlesztésekre fenntartva

Ha a 6. bájt értéke 0x00, akkor a parancs új kimeneti értéket tartalmaz:

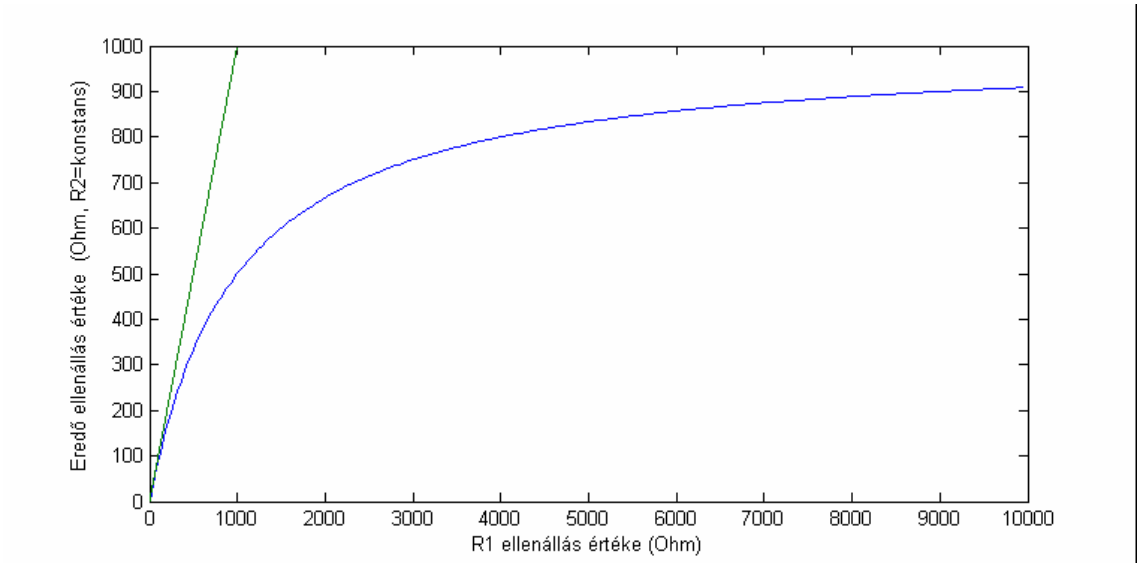
1. bájt: P0 potenciométer wiper regiszterének értéke

2. bájt: P1 potenciométer wiper regiszterének értéke

3-6. bájt: további fejlesztésekre fenntartva

Ha első bájt értéke 255, akkor a kártya P0 potenciométert kikapcsolja, ezzel megszüntetve a párhuzamos kapcsolást.

A kártya közvetlenül csak a potenciométer IC megfelelő vezérléséről gondoskodik, az adott ellenállásértékekhez tartozó regisztertartalmak számítására nem lenne kapacitása, így ezeket az értékeket a PC-től kapja. A PC szoftver a beállítani kívánt ellenállásérték alapján egy előre elkészített táblázatból olvassa ki a megfelelő regiszterértékeket.
A táblázat elkészítése szintén a feladat része volt, és a következőképpen történt:
Két ellenállás párhuzamos kapcsolása esetén az eredő ellenállás mindkét ellenállás értékénél kisebb lesz. Ha az egyik ellenállás értéke állandó, és a másik ellenállás értéke tart a végtelenhez, akkor eredő ellenállásuk aszimptotikusan tart az állandó ellenállás értékéhez:



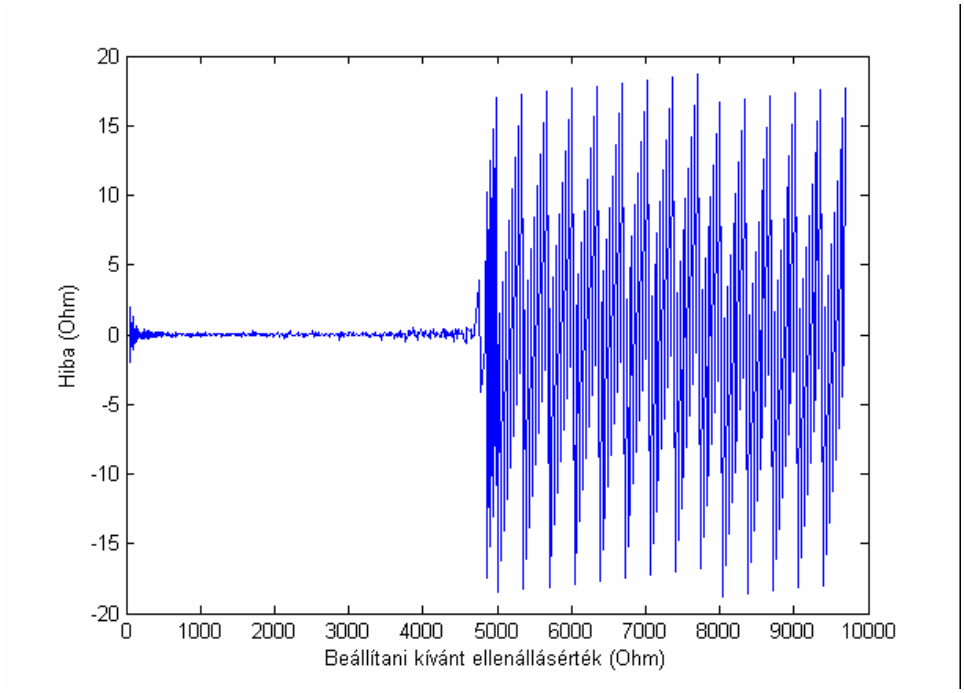
4.30. ábra Párhuzamosan kapcsolt ellenállások eredő ellenállása az egyik ellenállás függvényében. A zöld segédegyenes meredeksége 1.

Látható, hogy az eredőellenállás-görbe meredeksége a teljes tartományban kisebb mint egy. Ez azt jelenti, hogyha a digitális potenciométer egyik ellenállását egy állandó értékre állítjuk, és a vele párhuzamosan kapcsolt másik potenciométer értékét változtatjuk a felbontás által megengedett 40 Ω-os lépésekben, akkor az eredő ellenállás ennél mindenképpen kisebb lépésekben fog változni. Azaz, ha egy adott ellenállást szeretnénk beállítani, akkor a két ellenállást úgy kell megválasztani, hogy az egyik felülről lehetőleg minél közelebb legyen a kívánt ellenálláshoz, hogy a meredekség minél kisebb legyen, ekkor a másik ellenállás változtatásával elég pontosan rá lehet állni a kívánt értékre. Természetesen minél nagyobb a beállítani kívánt érték, felülről annál távolabbra kell választani a konstans ellenállás értékét, különben a másik ellenállást nem tudnánk elég nagyra választani, hogy elérjük a kívánt ellenállás értéket.

Az adott eredő értékhez tartozó két ellenállás értékének meghatározása általánosan azonban nem sikerült zárt formulával, csak viszonylag nagy hibával (kb. 10 Ω) , vagy csak szűk tartományban, így az alábbi megoldást választottam:

egy Matlab szkrip segítségével, optimalizáljuk a kapcsolás két ellenállásértékét, úgy, hogy az összes lehetőséget kipróbálva a legkisebb hibát adó értékeket tartjuk meg egy adott eredő ellenállás esetén. Így minden esetben a legjobb közelítést kapjuk, és azt a megoldást is megtaláljuk, amely éppen a karakterisztika meredekebb részén helyezkedik el, mégis jó értéket ad.

A megoldás igen jó eredményt hozott:



4.31. ábra A párhuzamosan kapcsolt potenciométerek számított hibája az ellenállás függvényében

Az ábra egy konkrét digitális potenciométerhez készített táblázat hibáját ábrázolja az ellenállás függvényében. A potenciométer névleges ellenállása 10 kΩ, melynek pontos értéke mérés alapján 9665 Ω. Párhuzamos működés tehát ezen érték feléig lehetséges, efölött marad a 8 bites felbontás, a 40 Ω lépésköz, ami jól látható az ábrán is. (± 20 Ω) A táblázatban a beállítani kívánt szomszédos értékek különbsége nem állandó, a lépésközök a Matlab-ban írt kódrészlet alapján:

```
a=50:1:499;
b=500:5:1995;
c=2000:10:5000;
d=5020:20:9680;
R_tabla=[a b c d];
```

Ennek oka, hogy nagyobb értékeknél kisebb felbontás is kielégíti a követelményeket, így rövidebb táblát kapunk, illetve a névleges érték fele fölött egyébként sem tudunk nagyobb pontosságot elérni, ez az ábrán is látható.

Viszont amíg a párhuzamosan kapcsolt ellenállások állítják elő a kimenetet, látható, hogy elvileg 1 Ω pontosság érthető el, ami a 40 Ω-os lépésekhez képest lényegesen finomabb.

A valóságban azonban csökkenti ezt a pontosságot az ellenálláslétra nemlinearitása, amelyre az adatlap $\overline{\mathrm{INL}} = 0,25$ LSB, ami a 10 kΩ-os típusnál kb. 10 Ω-ot jelent. A nagyobb problémát azonban a hőmérsékletfüggés okozza, mivel erre nem lehet alkalmazni a 7. pontban tárgyalt kalibrálást. Az adatlap alapján ennek értéke kb. 12 $\frac{\Omega}{\mathrm{C}^\circ}$ a teljes ellenállásra vonatkozóan. Ez a kimenetben a párhuzamos kapcsolás esetén

$$\frac{\Delta R_e}{R_e} = \frac{\Delta R_2}{R_2}\left(\frac{R_1}{R_1 + R_2}\right) + \frac{\Delta R_1}{R_1}\left(\frac{R_2}{R_1 + R_2}\right) \tag{1.28}$$

hibát okoz. Feltételezhetjük, hogy a két ellenállás relatív hibája megegyezik, amely azonos hőmérséklet esetén teljesül is. Ekkor látható, hogy a hiba egy az egyben megjelenik a kimeneten, ami így

$$\frac{12\frac{\Omega}{\mathrm{C}^\circ}}{10000\Omega} = 0,12\frac{\%}{\mathrm{C}^\circ} \tag{1.29}$$

relatív hibát jelent, ami még elfogadhatónak tűnik.

4.2.2.4. Digitális kimenet relével

A motormenedzsment részét képezi néhány olyan „szenzor” is, amely kétszintű kimenettel rendelkezik, és működését tekintve egy kapcsolónak tekinthető, mely a motorvezérlő vagy a műszerfalmodul felé általában egy vezetéken csatlakozik, és földet kapcsol vagy szakít meg. Ilyen kapcsoló az *olajnyomáskapcsoló*, mely a szükséges olajnyomás meglétét jelzi, illetve a gázpedálmodulban esetenként megtalálható *üresjárati-* és *kick-down* kapcsoló.

4.2.2.4.1. A kimeneti paraméterek meghatározása

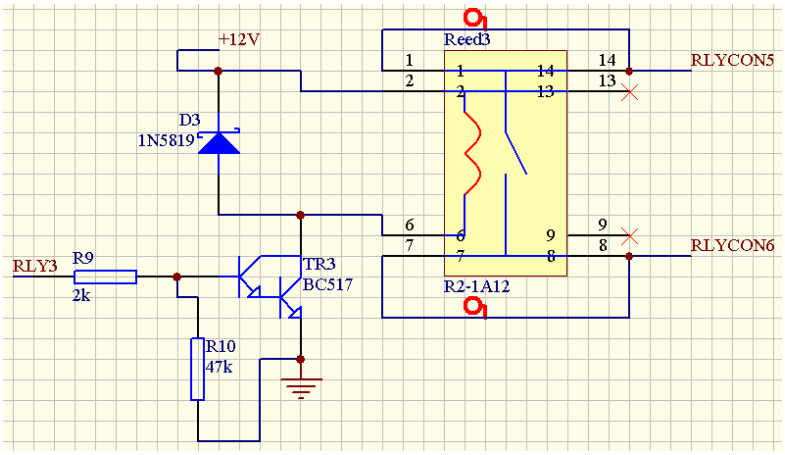
A megvalósítandó kapcsolók mindegyike csak feszültségszintet kapcsol a motorvezérlő felé, számottevő áram nem folyik rajta. A kapcsolások a működés során ritkán, vagy csak kis frekvenciával történnek.

4.2.2.4.2. A kimenet kapcsolási rajza

A kimenet egyszerű eszközökkel megvalósítható. Mivel gyors kapcsolásokra nincs szükség, érdemesebb relét alkalmazni félvezető elem helyett, így egy teljesen leválasztott kapcsoláshoz jutunk.

A konkrét megvalósítás a már ismert Reed relékkel történt, melyek maximális árama 1A, és kis méretű DIP tokozásban beszerezhetőek.

A kártya kimenetet megvalósító kapcsolási rajza az alaplapnál látott felépítéssel megegyező:



4.32. ábra A relével megvalósított digitális kimenet kapcsolási rajza

4.2.2.4.3. A kimenet SW terve

Jelenleg a program kétféle parancs fogadására képes:

- a kimeneti értéket módosító parancs a PC felhasználói programtól;
- lekérdezést illetve címmódosítást kezdeményező parancs a kártyamenedzsment programtól.

A kártya a felhasználói programtól minden esetben 6 bájtot kap, melyeket a következőképpen értelmez:

Az első 5 bájt jelentését a 6. bájt tartalma határozza meg. Ha ennek értéke 0xFA, akkor az adatok nem a kimenetre, hanem a kártya egyéb paramétereinek módosítására szolgál, ahol:

1. bájt: a kártya új címe, mely reset után érvényes;

2-6. bájt: további fejlesztésekre fenntartva;

Ha a 6. bájt értéke 0x00, akkor a parancs új kimeneti értéket tartalmaz:

1. bájt: ha 1: Relé1 ZÁR, 0: Relé1 NYIT;

2. bájt: ha 1: Relé2 ZÁR, 0: Relé2 NYIT;

3-6. bájt: további fejlesztésekre fenntartva.

4.2.3. A szimulátorkártyák általános szoftverrendszere

A szimulátorkártyák szoftverének átláthatóságát elősegítendő, a forrásfájlok írásánál a cél az volt, hogy a lehető legtöbb modul minden kártyánál azonos legyen, hiszen a kommunikáció mindig ugyanúgy zajlik, csak az adatok értelmezése eltérő. Ezért célszerű a kimenetfüggő kódrészleteket egy külön forrásfájlban rögzíteni. Ilyen kártyaspecifikus rutinok az inicializálás, a kimenet frissítése, illetve a bizonyos típusoknál szükséges interruptkezelő rutin.

A kártyák szoftvere a következő forrásfájlokból épül fel:

Main.c: ez a fájl tartalmazza a főprogramot. A főprogram elvégzi az adott kártya inicializálását, majd egy végtelen ciklusban várakozik új parancsra, mely beérkezése esetén a megfelelő rutint meghívja. Ez a fájl minden kártyánál azonos.

TWI_slave.c: ez a fájl tartalmazza az I2C kommunikációhoz szükséges rutinokat, szintén minden kártyánál azonos.

RS232.c: Az Atmega8 mikrokontrollerek nem rendelkeznek JTAG interfésszel, ami a programfejlesztésben nagy segítséget jelentene, ezért az egyébként nem használt soros port alkalmazható ilyen célra diagnosztikai üzenetek segítségével. A fájl a mikrokontroller soros portjának inicializálását, a küldést és a fogadást végző rutinokat tartalmazza.

EEPROM.c: Az Atmega mikrokontrollerek rendelkeznek egy beépített eeprommal is, ami kiválóan alkalmas olyan paraméterek tárolására, melyek lehetőleg a mikrokontroller újbóli felprogramozása nélkül változtathatóak kell, hogy legyenek. Ilyen paraméter például a kártya címe az I2C buszon.

CardSpecific.c: Ahogy a nevéből is kiderül, ez az egyetlen olyan forrásfájl, ami különbözik az egyes kártyáknál. Ez a fájl kötelezően tartalmazza az összes olyan kezelő rutint, amit a főprogram hívhat. Itt valósul meg a főprogramban kapott parancs végrehajtása, és a reset után a kártya megfelelő kimenet szerinti inicializálása is.

5. A szimulátor nyomtatott áramköri kártyáinak tervezése

A diplomaterv keretében feladat volt a megtervezett áramkörök nyomtatott huzalozású kártyáinak elkészítése is. A kapcsolási rajz és a PCB-terv is a PROTEL 99SE kombinált tervezőprogrammal történt.

5.1. Alaplap PCB terve

Az alaplap PCB tervének elkészítésekor a kapcsolási rajzban meghatározott összeköttetések megfelelő elkészítése mellett a mechanikai tulajdonságok megfelelő átgondolására is hangsúlyt kell fektetni: A PCB-nek illeszkednie kell a védelmére készített burkolathoz, ami a kártyák megvezetését is szolgálja, illetve elegendő számú és megfelelő elhelyezésű rögzítési ponttal is kell, hogy rendelkezzen, hogy a - főleg a fejlesztési fázisban elforduló - gyakori húzó és nyomó igénybevételnek, ami a kártyák ki- és berakásával jár, ellenálljon.

A terv elkészítése a következő főbb lépésekből állt:

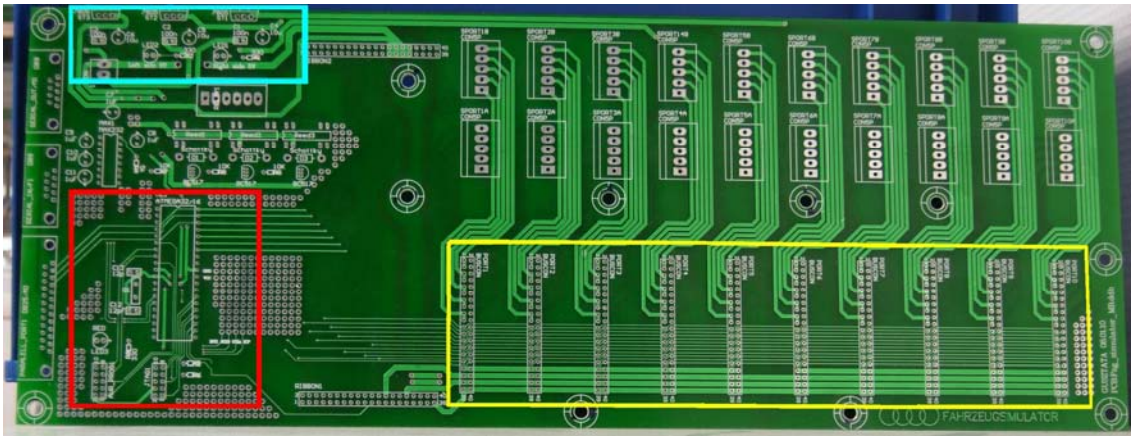
- Először az alkatrészek modulonkénti elhelyezése volt a cél. A kommunikációs interfészek a PC felé mindenképpen a kártya szélén kell, hogy elhelyezkedjenek, így azok pozíciója meghatározott. Ugyanígy a feszültségstabilizátor áramköröké is, mivel megfelelő hűtésről kell gondoskodni, ami legegyszerűbben a burkolaton kialakított hűtőfelületen oldható meg. A mikrokontroller logikailag a PC interfész és a szenzorkártyák között helyezkedik el, célszerű ezt a gyakorlatban is követni az összeköttetések könnyebb kialakítása érdekében. A szenzorkártyák csatlakozóinak helye foglalja el a nyák legnagyobb részét. A szimulált szenzorok jeleinek kivezetése szintén az alaplapon keresztül történik. Ez sokkal praktikusabb, mintha a kimeneti jeleket magukról a kártyákról vezetnénk a motorvezérlő felé, hiszen a vezetékezés sokkal átláthatóbban megoldható, illetve egy kártya sokkal gyorsabban kiszabadítható. A jelek így a szenzorkártya buszcsatlakozóján keresztül először az alaplapra jutnak, majd onnan közvetlenül egy másik csatlakozón keresztül jutnak a motorvezérlőhöz. A buszcsatlakozók elhelyezése sorban egymás után, párhuzamosan állítva a nyák egyik oldalán történt. A szimulált jelek elvezetésére szolgáló csatlakozók a hozzájuk tartozó buszcsatlakozók fölött kaptak helyet. A bővítőcsatlakozók a panel két szemközti

hosszabb oldalán kaptak helyet, hogy két egymás melletti alaplap minél könnyebben csatlakoztatható legyen.

- Az elhelyezés után, a vezetékezés előtt a felfogatási pontok elhelyezése következett. Ekkor a panel mérete már lassan véglegesíthető volt, 415x168mm-re adódott. A felfogatási pontok a sarkokban, a hosszabb oldalak mentén, és a buszcsatlakozóknál a középvonal mentén kerültek elhelyezésre.

- Ezután következett a vezetékezés. Sajnos az autorouter nem tudott megbirkózni a feladattal, melynek oka valószínűleg a busz miatti nagyszámú összeköttetés volt. Így maradt a kézi huzalozás az „Interactive Routing" segítségével. Szerencsére a buszcsatlakozók és a hozzájuk kapcsolódó szenzorcsatlakozók vezetékezését csak egyszer kellett megfelelően elkészíteni, és onnan azt mind a 10 csatlakozóhoz tovább lehetett másolni.

Az PCB terv alapján elkészített panel az alábbi ábrán látható:

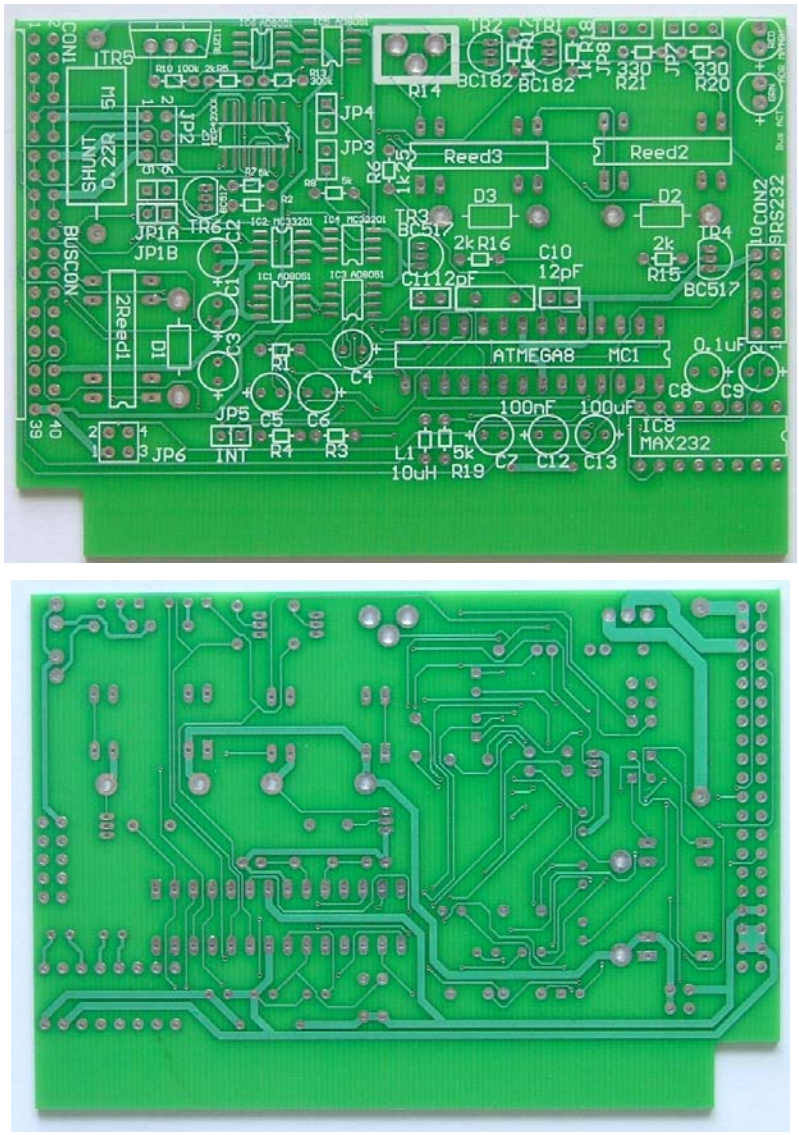


5.1. ábra Az alaplap PCB alkatrész oldala. Pirossal keretezett területen a mikrokontroller, balra a kommunikációs portok, felette a kapcsolórelék. Világoskékkel keretezett területen a feszültségstabilizátorok. Sárgával keretezett területen a kártyafoglalatok, felettük a szenzorkimenetek csatlakozói.

5.2. A szenzorkártyák PCB terve

Mivel a szenzorkártyák mindegyike ugyanazzal a mikrokontrollerrel van megvalósítva, a rendszerterv megalkotásánál az a döntés született, hogy a megtervezett kapcsolások egy univerzális PCB-n lesznek megvalósítva. Így elegendő egyetlen típus megtervezése, és az így egy kártyán megjelenő számos periféria segítséget is jelenthet bizonyos továbbfejlesztésekhez, ha egy kimenet tulajdonságain javítani lehetne. Ezáltal viszont a szenzorkártyák PCB tervének elkészítése az alaplapénál nagyobb kihívást jelentett, hiszen egy lehetőleg minél kisebb kártyán kellett megvalósítani az összes kívánt funkciót. Az indokolttá tette az SMD tokozású alkatrészek alkalmazását.

A terv alapján készült panel a következő ábrákon látható:



5.2. ábra A szenzorkimeneteket megvalósító univerzális PCB komponens és forrasztási oldala

Amint az ábrákon látható, a panelon néhány jumper is helyet kapott, amivel az adott kártyának megfelelő működéshez szükséges kapcsolatok állíthatók be:

JP6: felelős a relék típusának megfelelő behúzófeszültség kiválasztásáért. Erre azért volt szükség, mert a választott Reed relék készülnek 5V és 12V feszültségű behúzótekercccsel is, azonban mindig csak valamelyik típus beszerezhető, a rendelés pedig hosszú időt vesz igénybe.

JP5: A buszon az egyik jelvezeték az alaplap mikrokontrollerének külső interrupt lábára van kötve, ezáltal lehetőség van a kártyák felől interruptkérésre, melynek hatására például az alaplap olvasást kezdeményezhet. JP5 ezt a kapcsolatot hozza létre vagy szakítja meg, hogy a kártya mikrokontrollerének ezt a lábát másra is fel lehessen használni.

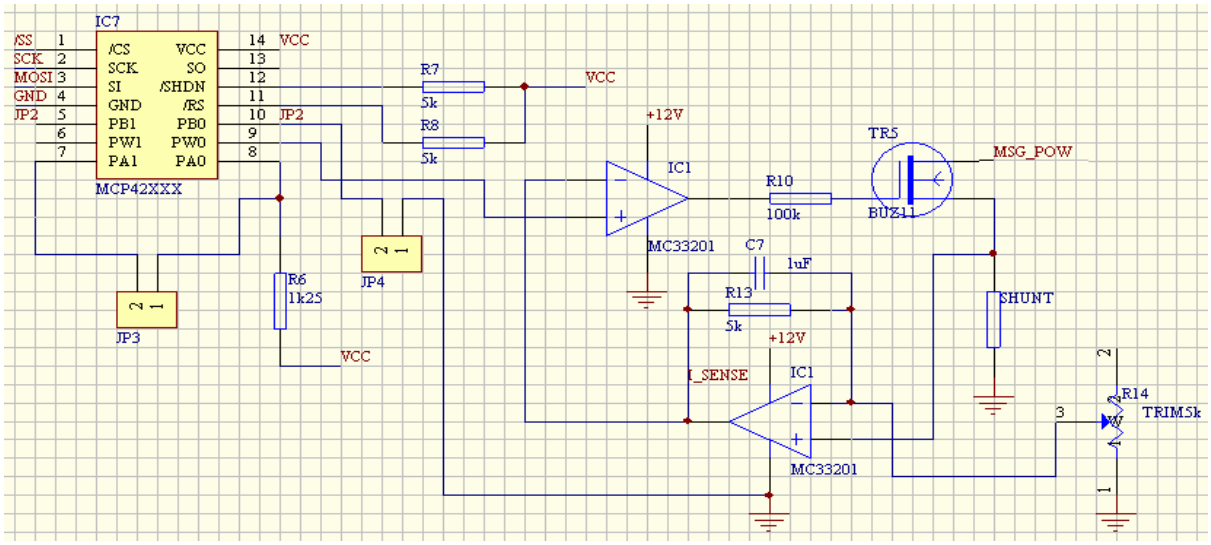
JP1A/B: Az analóg feszültségkimenetet a motorvezérlő tápfeszültségének megjelenése kapcsolja a szenzorcsatlakozóra a már ismertetett módon, ezen jumperek segítségével azonban a két analóg kimeneti csatornát, melyet egy kártya tartalmaz, közvetlenül is ki lehet adni a csatlakozókra.

JP7, JP8: A kártyán két led is helyet kapott, melyek a felhasználó felé szolgálnak jelzésül: a kártya megcímzését a piros led rövid felvillanása jelzi, hiba esetén, például ha az analóg feszültségkimenet kívánt feszültsége nem állítható be megfelelően, nagy fényerővel folyamatosan villog a hiba megszűnéséig. A mikrokontroller korlátozott lábszáma miatt azonban a digitális kimeneten nem lehet ugyanazokon a kivezetéseken a két led, mint a többi típusnál, így itt a meghajtó kimenet kiválasztása ezekkel a jumperekkel történik.

JP2: Az ellenálláskimenet kalibrálásához nyújt csatlakozási lehetőséget

JP3, JP4: A kártyán egy változtatható terhelés is helyet kapott, ami lehetőséget ad a szenzorok motorvezérlő által biztosított tápfeszültségének beállítható terhelésére, melyre azonban a helyes működéshez nincs szükség, inkább csak kísérleti célokat szolgál, ezért közlése itt, és kisebb részletességgel történik:

Kapcsolási rajza:



5.3. ábra A változtatható terhelést megvalósító áramszabályzás kapcsolási rajza

A kapcsolás egy áramszabályozást valósít meg, melynek alapjelét az egyébként ellenálláskimenetet megvalósító digitális potenciométerrel megvalósított változtatható feszültségosztó szolgáltatja. Mivel az ellenálláskimenetű szenzoroknál nincs külön tápfeszültség, így nincs szükség terhelésre sem, a többi szenzornál pedig nincs szükség a digitális potenciométerre, így ez a párosítás jól kihasználható. Szükség esetén azonban az analóg kimenet feszültsége is lehet alapjel. JP3 és JP4 az ellenálláskimenet és az alapjelgenerátor kapcsolások közötti váltást teszi lehetővé. A söntellenálláson eső feszültséget az IC1 műveleti erősítővel megvalósított változtatható erősítésű nem-invertáló alapkapcsolás erősíti. IC2 ennek és az alapjelül szolgáló feszültségnek a különbségét erősíti, mellyel a BUZ11 típusú FET gate-jét vezérli, így tehát egy arányos szabályzást megvalósítva. A negatív visszacsatolás a műveleti erősítő bemenetén valósul meg. Az áramszabályozás céltartománya a sönt értékének megfelelő megválasztásával állítható be.

6. PC oldali konfigurációs szoftver készítése

A szimulátorkártyák fejlesztéséhez, teszteléséhez, illetve a kártyák változtatható paramétereinek újraprogramozás nélküli beállításához szükség van PC oldalon egy felhasználói programra. A szimulációs környezet irányítását végző felhasználói program nem rendelkezik ilyen funkciókkal, mivel az óvatlan felhasználó így képes lenne a kártyák címének megváltoztatására, ami a rendszer működésképtelenségét okozhatja. A program fejlesztése LabView környezetben történt, ami a grafikus felület elkészítését egyszerűen teszi lehetővé, és számos, a feladathoz jól illeszkedő felhasználói felület elemet tartalmaz. Mindemellett a soros port kezelése is átgondolt, könnyen elsajátítható. A programban jelenleg az alábbi funkciók állnak rendelkezésre:

- kártyák kimenetének beállítása;
- kártyák eepromban tárolt paramétereinek változtatása;
- buszhoz csatlakozó kártyák paramétereinek lekérdezése.

A program alapvető eleme a PC és az alaplap közötti kommunikációt megvalósító blokk. Ezt a blokkot hívja különböző paraméterekkel mindegyik funkciót megvalósító eseménykezelő rutin, illetve ezt a blokkot használták fel a szimulációs felhasználói program kommunikációs moduljaként is. A modul feladata az alaplapnál tárgyalt kommunikációs protokoll megvalósítása, illetve fogadja az alaplaptól érkező esetleges hibaüzeneteket.

A kártyák kimenetének beállítása egy egyszerűsített felületen történik, ahol az adott kártya címét kell megadni, illetve a kimenet típusát, hogy a beírt érték megfelelő formátumban érkezzen az adott kártyához. Lehetőség van közvetlenül is a kiküldött bájtok értékének meghatározására, mely a fejlesztéshez elengedhetetlen.

Az eepromban tárolt paraméterek közül jelenleg még csak a kártya címének megváltoztatása került implementálásra. További fejlesztések keretében itt szeretném beállíthatóvá tenni a kártyák kimenetének kezdeti értékét, illetve a működési tartomány korlátozását is.

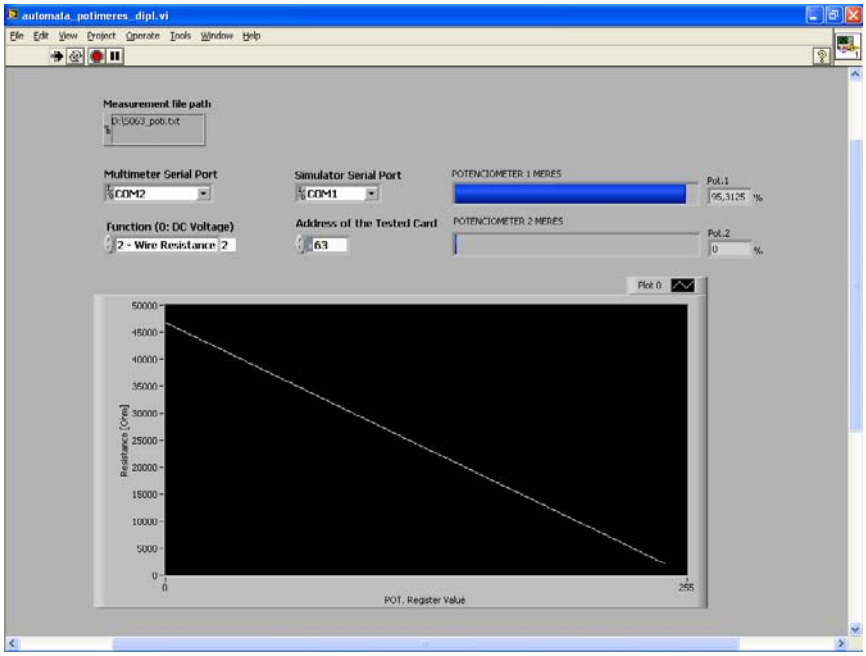
A kártyák lekérdezése a következőképpen történik: Program lekérdező üzenetet küld sorban az összes érvényes kártyacímre, és várja a válaszokat. Amennyiben a lekérdezett cím nincs jelen a buszon, az alaplap a „NO_ADDR” üzenetet küldi vissza. Ha egy cím jelen van, akkor az alaplap „DATAIN” üzenete után a kártya elküldi eepromban tárolt paramétereit, melyek jelenleg a következők: cím, kártyatípus, szoftververzió. A kártya típusát és a szoftververziót tartalmazó eeprom bejegyzés szándékosan nem változtatható a program által. Ezeket a paramétereket a mikrokontroller szoftver írja be az eepromban a felprogramozás utáni első indításkor, és az adott szoftver pontos azonosítására szolgál, ami az állandó fejlesztések miatt fontos.

7. Mérések és eredmények

A nyomtatott áramköri kártyák beültetése és „élesztése” után az elkészült áramkörök tesztelése következett. Ahhoz, hogy az analóg feszültség és az ellenálláskimenet működését a teljes tartományban ellenőrizni lehessen, meg kellett valahogyan oldani azok automatikus mérését, hiszen a karakterisztikát nagyobb felbontással „kézzel” végigmérni elég hosszadalmas lett volna.

Ezt a problémát egy Labview-ban írt program segítségével sikerült megoldani. A kártyák tesztelését végző program is Labview-ban készült, így a parancsok küldése már kész volt, csak automatizálni kellett. A méréshez rendelkezésre állt egy HP34401A digitális multiméter, ami vezérelhető soros portról, így kézenfekvőnek tűnt ennek alkalmazása. Az elkészült program felépítése a következő:

A soros port megnyitása, majd a műszer inicializálása után egy „for” ciklusban történik maga a mérés, ami az adott című kártyának először elküldi a kimenetet beállító parancsot, majd 0,5 másodperc várakozási idő után a multiméterről lekérdezi a kimenet értékét, amit egy vektorban tárol. A ciklus végén, azaz amikor a program végigért az előzetesen egy fájlból beolvasott alapjelértékeken, a mérési eredmények vektorát egy fájlban elmenti. Ennek tartalmát aztán Matlabban lehet feldolgozni. A program a következő felhasználói felülettel rendelkezik:



7.1. ábra Az automatikus mérést végző program felhasználói felülete

A mérési összeállítás, illetve a teljes szimulációs környezet a megépített elektronikával:

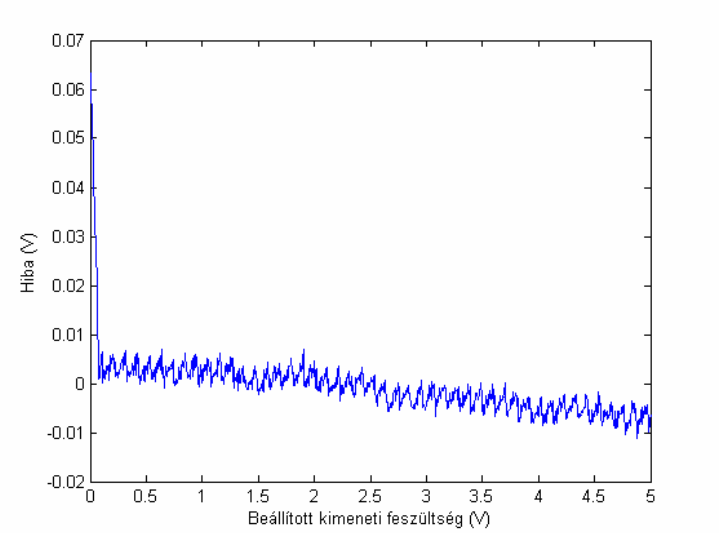
7.2. ábra A szimulátorrendszer

A szimulátorelektronika a két összekötött alaplappal, illetve a szenzorkártyákkal:

7.3. ábra Az összeállított szimulátorelektronika két összekapcsolt alaplappal, és a szenzorkártyákkal

7.1. Az analóg kimenet

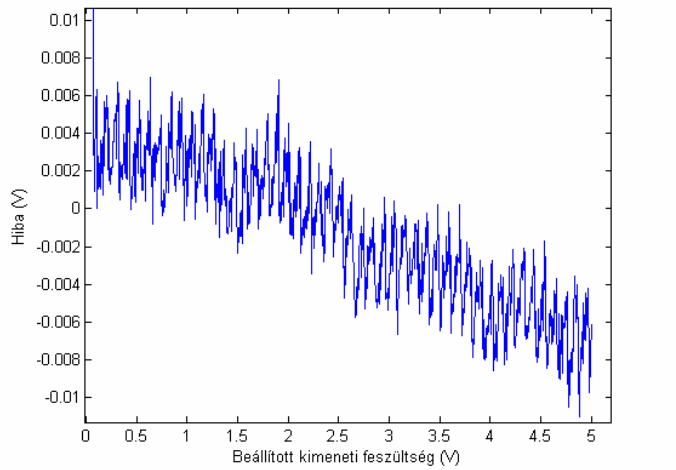
Az analóg kimenet mérése során a tesztvektor a 0-5 V feszültségtartományban 5 mV lépésközzel megadott 1001 elemű vektor volt. A kapott eredmény a következő:



7.4. ábra Az analóg feszültségkimenet hibája a feszültség függvényében

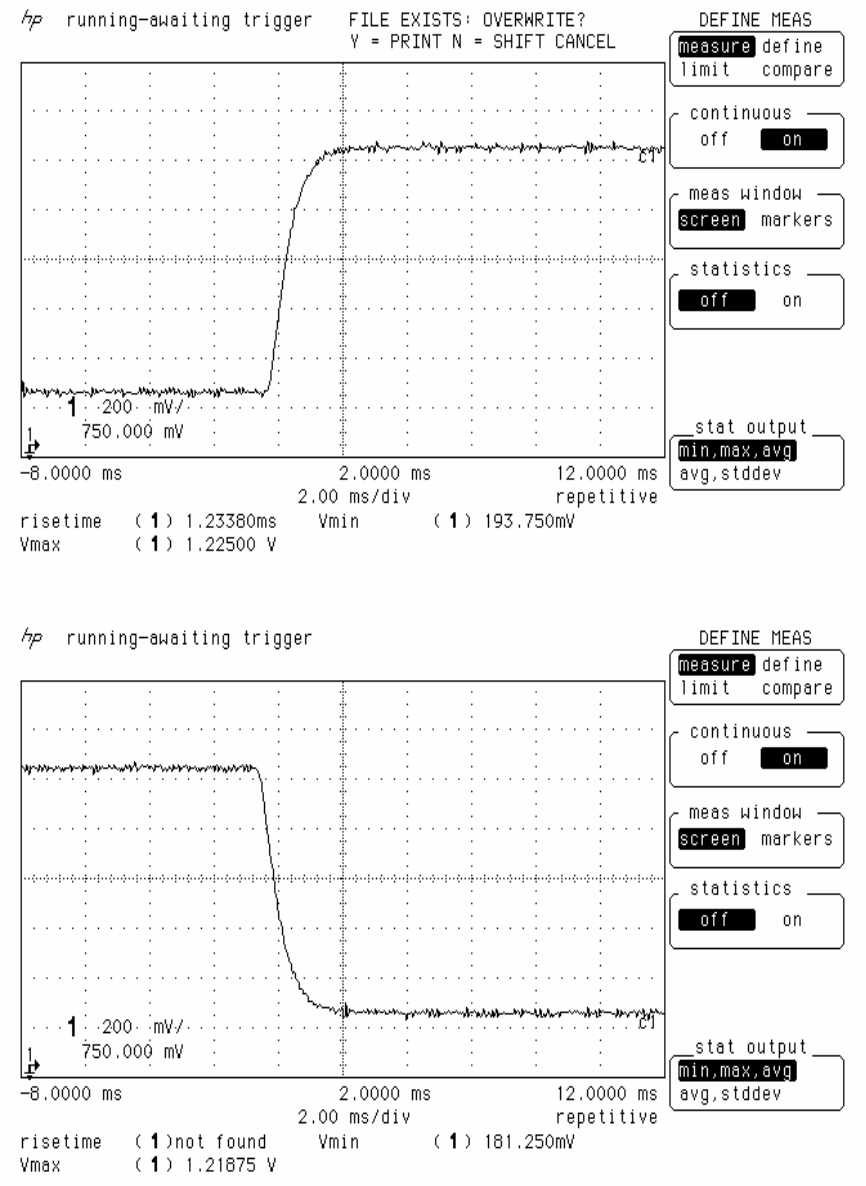
A 0 V közelében megjelenő nagy, 70 mV hiba annak tudható be, hogy bár a kimeneti erősítő rail-to-rail felépítésű, a 0 és12 V tápfeszültség miatt a 0 V-ot csak megközelíteni tudja, illetve a mikrokontroller PWM kimenetén az alacsony szint sem éri el a 0 feszültséget. Mivel azonban ez bőven a számunkra érdekes tartományon kívül van (0,5-4,5 V) így ez a hiba elfogadható.

A 7.5 nagyított ábrán látható, hogy a vízszintes tengelyen ezt a 70 mV értéket elérve a hiba nagysága a teljes tartományban 10 mV alatt marad. Látható, hogy a hiba legkisebb értéke 2,5 V körül van, ennek valószínűleg az az oka, hogy az ellenállásosztót megvalósító trimmerpotenciométerrel a kalibrálást 2,5 V-on végeztem.



7.5. ábra Az analóg feszültségkimenet hibája a feszültség függvényében

A kimenet 1V ugrásjelre adott válasza az alábbi oszcilloszkóp-képeken látható:



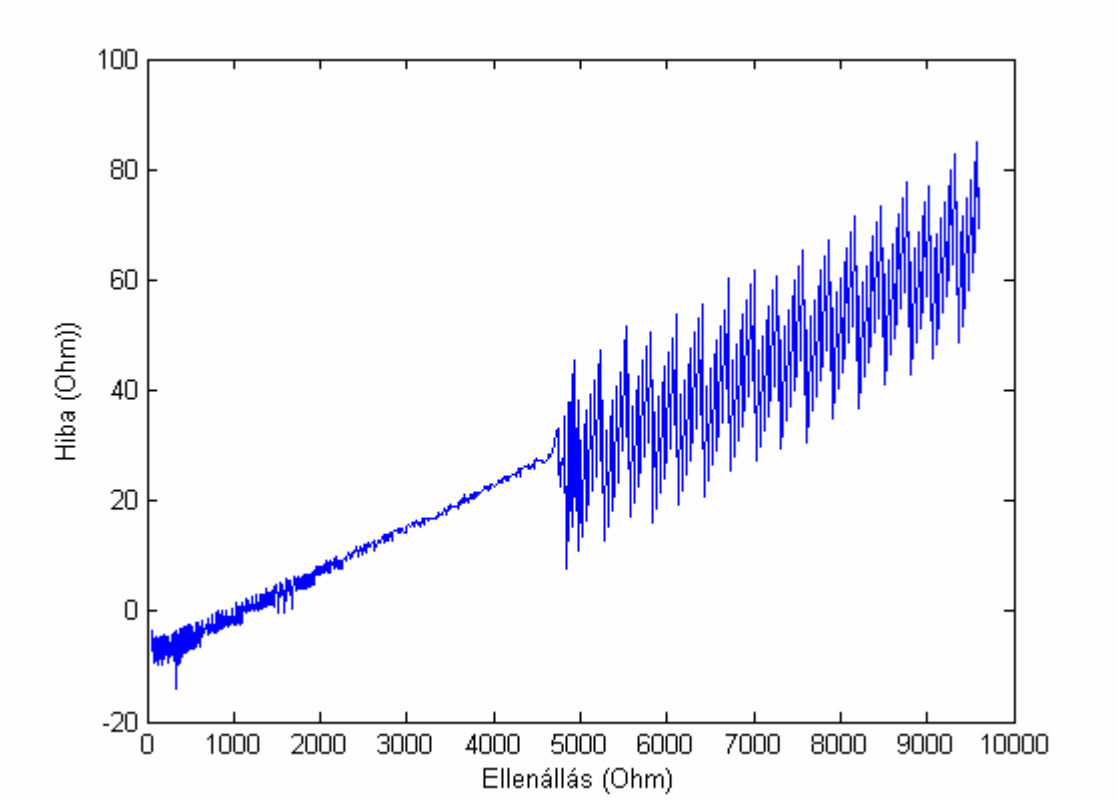
7.6. ábra Az analóg feszültségkimenet pozitív és negatív ugrásjelre adott válasza

7.2. Ellenálláskimenet

Az ellenálláskimenet mérése a feszültségéhez hasonlóan, automatikusan történt, azzal a különbséggel, hogy a generált tesztvektor egy növekvő lépésközű vektor volt, hogy csökkentsük a mérési pontok számát:

```
a=60:1:499;
b=500:5:1995;
c=2000:10:5000;
d=5020:20:9680;
tesztvektor=[a b c d];
```

Ez megegyezik azzal a vektorral, amihez a párhuzamos kapcsolásban az egyes tagok optimális regiszterértékei generáltuk, hiszen ezen értékek, mint parancs kiadásával lehet beállítani a kívánt értéket. A tesztvektor felbontása ennek megfelelően elég nagy, csak a felsőbb tartományba éri el a 20 Ω-ot, ami indokolt is, hiszen az ellenállástagok felénél nagyobb értékre párhuzamos kapcsolással nem tudunk felmenni, így onnan kezdve marad a 8 bit potenciométerfelbontáshoz tartozó 40 Ω-os pontosság. A kapott eredmény:



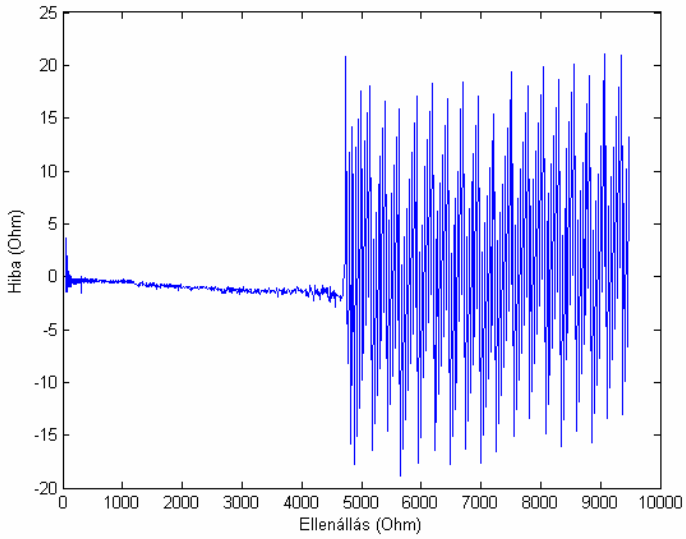
7.7. ábra Az ellenálláskimenet hibája az ellenállás függvényében

Ami rögtön feltűnik az ábrán, hogy mennyire elválik a párhuzamos kapcsolással megvalósított alsó féltartomány a felsőtől, illetve a növekvő hiba, ami egyértelműen kalibrációs problémára utal. A kalibrációs problémát valószínűleg az okozza, hogy a táblázat generálásához a potenciométer két paraméterét kellet felhasználni, az egyik a teljes ellenállása, a másik a „csúszka" ellenállása. Előbbi a két kivezetésen megmérhető, utóbbi csak számítható különböző beállításokhoz tartozó ellenállásértékek alapján, ha az előbbi érték már ismert. A teljes ellenállást viszont bekapcsolás után közvetlenül mértem, és későbbi tapasztalatokból kiderült, hogy ez csak a tápfeszültség rákapcsolása után néhány perccel áll be állandó értékre, azalatt kb. 50-100 Ω-ot nő, ami magyarázza is a növekvő hibát.
De ettől függetlenül megállapítható, hogy a kapott kimenet még így is bőven túlteljesíti a követelményeket, sőt, pontosabb mint maguk a szimulált szenzorok.

7.2.1. Automatikus kalibráció megvalósítása LabView-ban HP34401 digitális multiméter illesztésével

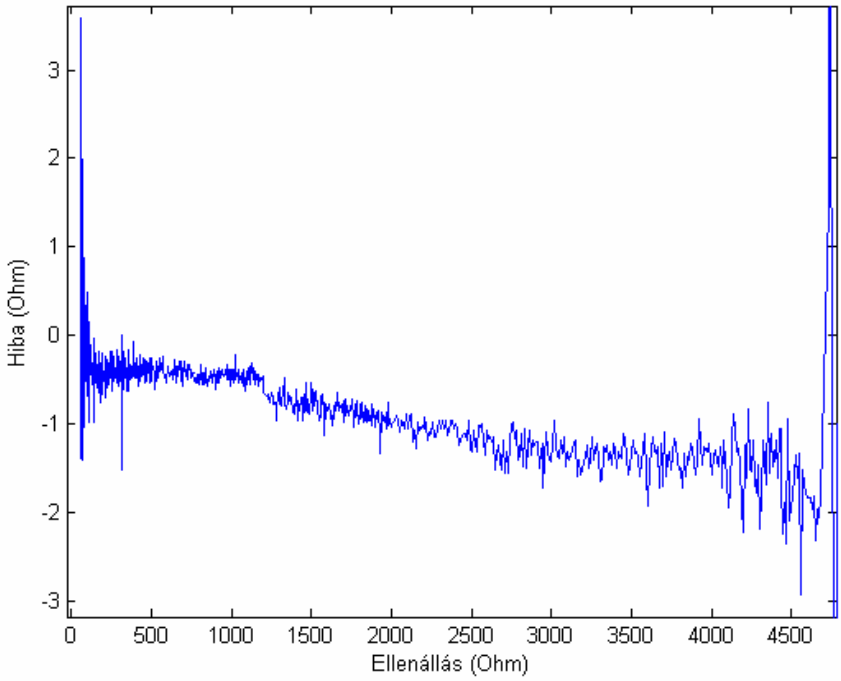
Hogy még pontosabb kimenet legyen elérhető, felmerült az ötlet, hogy a már meglévő automatikus mérőprogramot kalibrálásra is fel lehetne használni, amennyiben a tokban lévő két potenciométer ellenállását a lehetséges 256 pontban megmérem. Így a táblázat generálásához nem az összellenállásból számítva kell meghatározni az egyes beállításokhoz tartozó értékeket, melyekből majd az optimális párhuzamos párosítást meghatározzuk, hanem a mért értékekből.

Az így számított táblázattal a következő hibák adódtak:



7.8. ábra Kalibrált ellenálláskimenet hibája az ellenállás függvényében

Látható, hogy gyakorlatilag eltűnt a kalibrációs hiba, a párhuzamos kapcsolású tartományban az eltérés még 4 kΩ fölött is kisebb, mint 3 Ω:



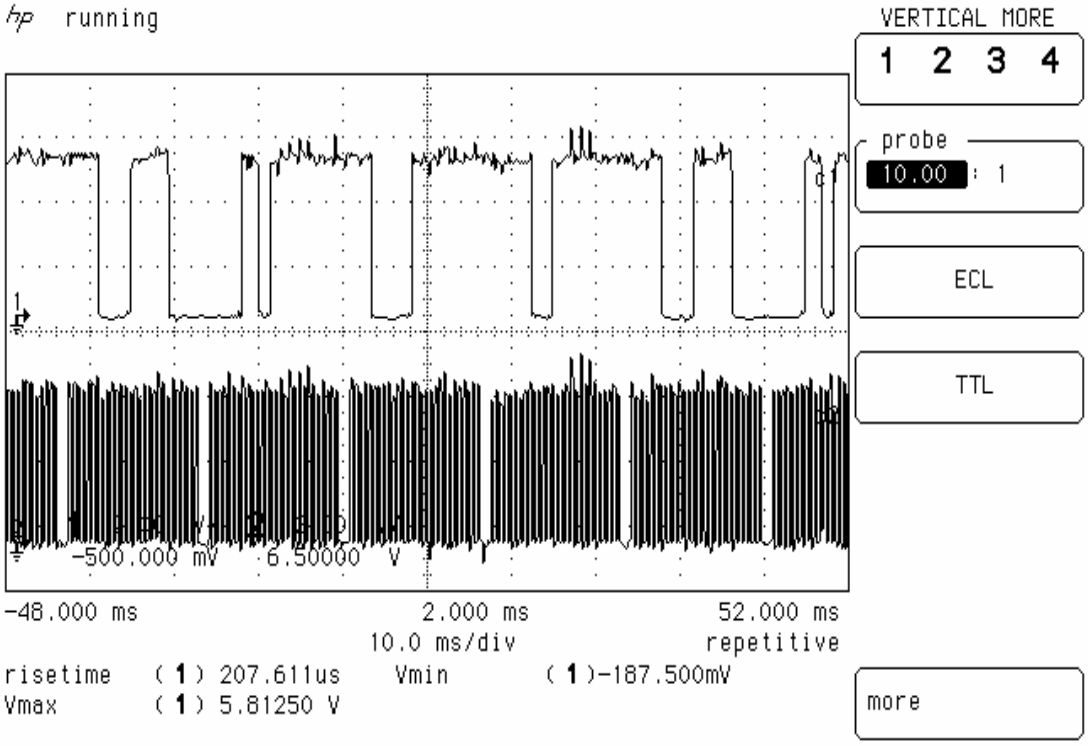
7.9. ábra Kalibrált ellenálláskimenet hibája az ellenállás függvényében (60-4700 Ω)

Érdekes kérdés, hogy milyen egy ilyen digitális potenciométernek a hosszúidejű stabilitása, erre azonban a későbbi tapasztalatok fognak választ adni.

7.3. Hullámforma kimenet

A hullámforma kimenetekről olyan felvételt, amelyen a teljes periódus hossza látszik, nem mindig sikerült készíteni, mivel megfelelő felbontás esetén a teljes jelalak már nem fér bele a képbe, de a működés mindenképpen megfigyelhető.

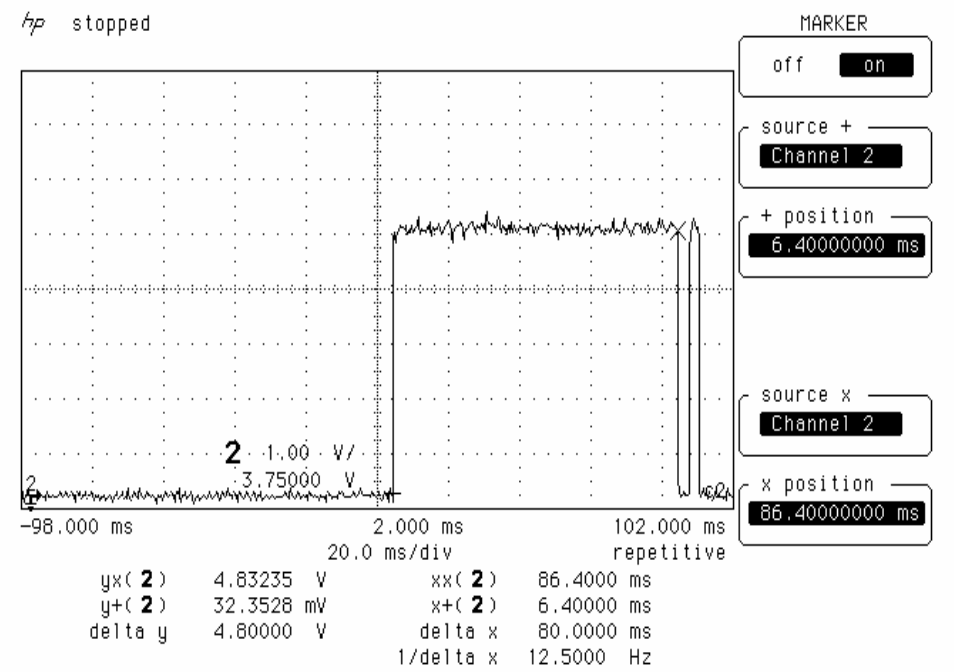
A fordulatszám-jeladók jelét a motorvezérlővel összekötve, rendeltetésszerű használat közben mértem, így a megfelelő felhúzó ellenállásokról nem kellett külön gondoskodni:



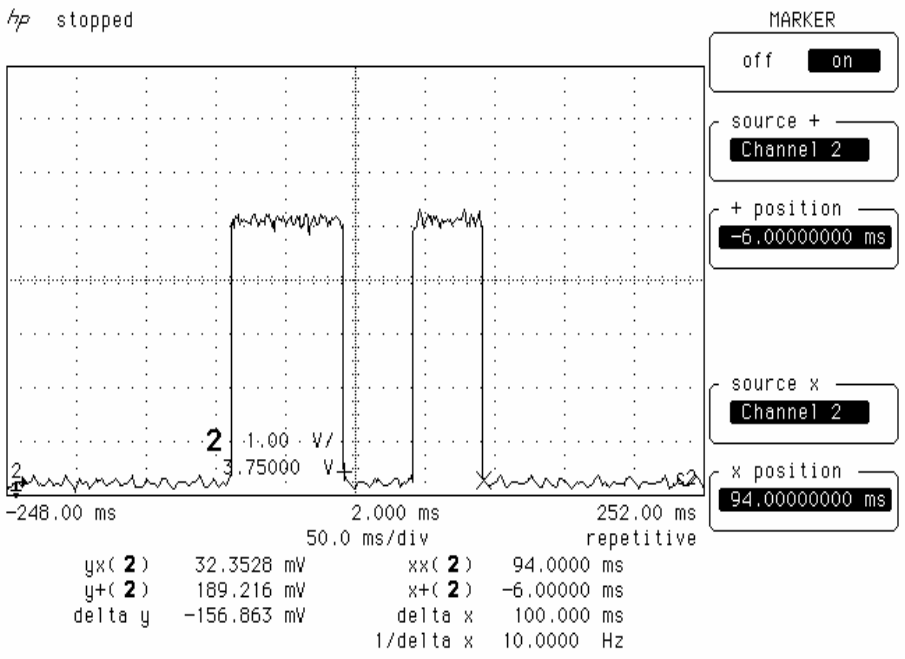
7.10. ábra Fordulatszámjeladók kimenetének jelalakja (rpm=1200 1/min)

Az ábrán valamivel több mint egy vezérműtengely-fordulat figyelhető meg. A helyes működést az igazolja, hogy a motorvezérlő a jel hatására a befecskendező-szelepeket megfelelően működteti. (hibás jel esetén egyáltalán nincs befecskendezés, hibaüzenetet kapunk.)

Az olajszint- és hőmérséklet-jeladó kimenetét egy, a szenzor vizsgálatára alkalmas eszköz segítségével teszteltem elsősorban. A teszter az olajszintet- és hőmérsékletet jelzi ki, illetve a mért pulzusszélességeket is kiírja. A kapott értékek megegyeztek a szimulációban beállítottakkal. A kimenet oszcilloszkóp képe a következő két ábrán látható, a ciklus első illetve második félperiódusa látszik:



7.11. ábra A PULS szenzor adatátviteli ciklusának első fele: 80ms széles impulzus (kurzorok) majd hőmérséklet-érték



7.12 ábra A PULS szenzor adatátviteli ciklusának második fele: 80 ms impulzus majd a referenciaidő, 100 ms (kurzorok)

Összefoglalás

A diplomaterv keretében megvalósítottam egy olyan rendszert, amely alkalmas az R4 PDTDI motormenedzsmentben jelenlévő szenzorok elektronikus szimulációjára, a szenzorok által szimulálni kívánt jelek paramétereinek PC-től való fogadására.

A feladat során megismertem a modern motormenedzsment felépítésével, az azt alkotó szenzorok és beavatkozók működésével. Megismertem a konkrét megvalósításban szereplő R4 PDTDI motortípus működését, szenzorainak kimeneti tulajdonságait. Megszereztem a választott mikrokontrollerek programozásához és önálló alkalmazásához szükséges ismereteket, illetve megismertem az I^2C busz működését. Megterveztem és megépítettem a szimulációt végző rendszert, implementáltam a mikrokontrollereket működtető szoftvereket.

Dolgozatomban bemutattam a motormenedzsment fejlődését, működését, a fedélzeti diagnosztika szerepét az alkatrészek analízisében. Ezután bemutattam a rendszer megvalósításához szükséges megfontolásokat, illetve az egyes szenzortípusok kimenetének részletesebb elemzése után azok megvalósítási tervére. Az elkészült rendszer működését mérési eredményekkel szemléltettem.

Elmondható, hogy sikerült olyan rendszert létrehozni, amely működése a kiírásban megfogalmazott céloknak megfelel. A végleges összeállításban sikerült a motor szimulált beindítása, a motorvezérlő megfelelő beállításokkal nem jelez hibát a szenzorok működésében.

A jövőben lehetőség van a ez egyes kimenetek működésének fejlesztésére, a szoftverben számos további bővítési lehetőség előkészítése megtörtént. A pozitív tapasztalatok alapján megkezdődhet a további motortípusokra történő tényleges kibővítés.

Irodalomjegyzék

[1] Dr. Frank Tibor, Dr. Kováts Miklós *„Benzinbefecskendező és motorirányító rendszerek”* 2004

[2] Dr. Kováts Miklós, Dr. Nagyszokolyai Iván, Szalai László *„Dizel befecskendező rendszerek”* 2002

[3] Tölgyesi Zoltán *„Fedélzeti doiagnosztika On-Board-Diagnostic”* 2005

[4] *„Der 2,0 l TDI-Motor”* Selbststudienprogramm, Volkswagen AG, Wofsburg 2003

[5] Vendégh Gábor (AUDI Hungária Moror Kft)*„Fahrzeugsimulator für Motormanagement R4 PDTDI”*

[6] „*Az I2C busz és Használata*” Philips Semiconductors 1995

[7] Atmega8 High-performance, Low-power AVR® 8-bit Microcontroller atmega8.pdf

[8] „*Using PWM Output as a Digital-to-Analog Converter on aTMS320F280x Digital Signal Controller*” Texas Instruments Application report February 2006

[9] Microchip MCP42XXX Single/Dual Digital Potentiometer with SPI™ Interface mcp42xxx.pdf