



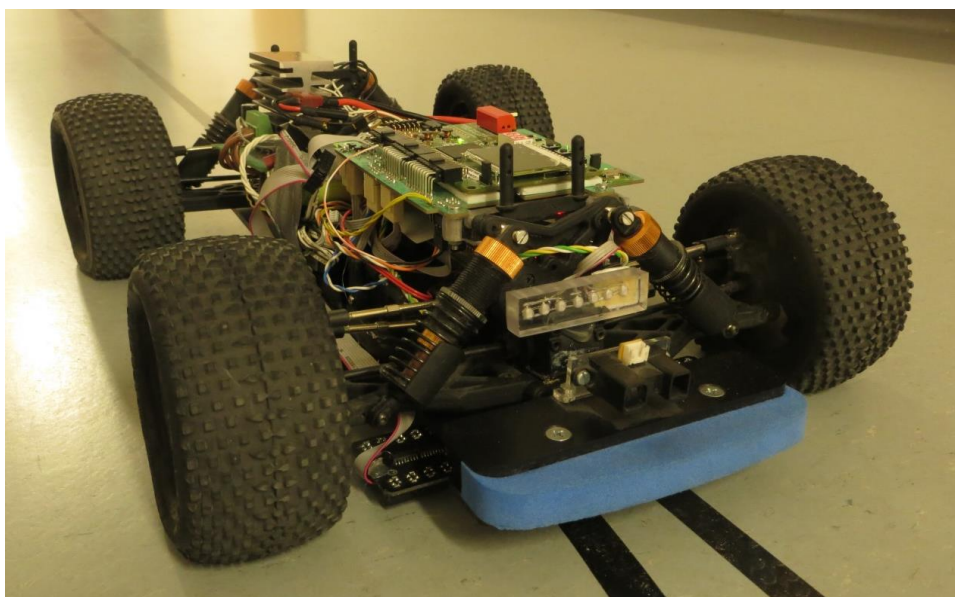
M Ű E G Y E T E M 1 7 8 2

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Himes Ádám

DIPLOMATERV

Helymeghatározás és pályarögzítés RobonAUT járművel



KONZULENS

Szalóki Dávid

BUDAPEST, 2014

Tartalomjegyzék

Összefoglaló	1
Abstract.....	2
1 Bevezetés	3
2 Feladat részletezése	4
2.1 Vonalkövetés	4
2.2 Helymeghatározás	4
3 Vonalszenzor	5
3.1 Célkitűzések.....	5
3.2 Optikai szenzor tervezése	5
3.2.1 Szenzor típus kiválasztása	5
3.2.2 Szenzorsor elhelyezése	6
3.2.3 Optikai elemek kiválasztása, elrendezése.....	6
3.2.4 Optikai elemek bekötése.....	8
3.2.5 Földtől való optimális távolság	8
3.2.6 Fototranzisztor sönt ellenállása	9
3.2.7 Analóg jelek mérése	10
3.2.8 Analóg jelek mérési stratégiája	10
3.2.9 Elektronikai kapcsolás	15
3.2.10 NYÁK tervezés szempontjai	18
3.2.11 Elkészült NYÁK terv és hardver	20
3.3 Jelfeldolgozás	21
3.3.1 Jelfeldolgozó egység.....	21
3.3.2 Feldolgozás lépései.....	22
3.3.3 Hardverkezelés	22
3.3.4 Felfutási idők	24
3.3.5 Szenzorok bemérési stratégiája	25
3.3.6 Szenzor bemérő logika, jeltovábbítás.....	26
3.3.7 Szenzor jelek vizsgálata, kalibrálás	27
3.3.8 Vonal látkép, kalibrálás ellenőrzése	29
3.3.9 Vonalfelismerés	30
3.3.10 Vonal középpontjának meghatározása	31

3.3.11	Vonalszenzor felbontása.....	34
3.4	Vonalak analízise.....	35
3.4.1	Követendő vonal kiválasztása	35
3.4.2	Vezérvonal szöghelyzetének felismerése	36
4	Helymeghatározás.....	37
4.1	Szenzorok	37
4.1.1	Enkóder.....	37
4.1.2	Inercia szenzor	37
4.2	Mérések	38
4.2.1	Próbamérés	38
4.2.2	Nagy felbontású mérés	39
4.2.3	Hosszú távú mérés	40
4.2.4	Kiértékelés	41
4.2.5	Giroszkóp adatok utólagos korrekciója	41
4.2.6	Hibalehetőségek.....	43
4.3	Pontosítás.....	44
4.3.1	Jobb szenzor	44
4.3.2	Új modell felírása	44
4.3.3	Kormányzóg becslése	45
4.3.4	Új modell összehasonlítása az előzővel.....	46
5	Térképkövetés	48
5.1	Követő algoritmus	48
5.1.1	Elgondolások	48
5.2	Virtuális vonalszenzor	48
5.2.1	Abszolút koordináta rendszerben	48
5.2.2	Relatív koordinátarendszer	49
5.3	Vonalkövető szabályzó	50
5.4	Motorvezérlő javítása	51
6	Pályarögzítés.....	52
7	PC szoftver.....	55
8	Továbbfejlesztési lehetőségek	56
	Irodalomjegyzék.....	57

HALLGATÓI NYILATKOZAT

Alulírott **Himes Ádám**, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózataán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2014. 12. 19.

.....
Himes Ádám

Összefoglaló

A BME Automatizálási és Alkalmazott Informatikai Tanszéke 2012 óta minden évben megrendezi RobonAUT nevű műszaki tervezési és konstrukciós versenyét. A verseny célja egy olyan autonóm robotautó tervezése és építése, ami egy előre részleteiben nem meghatározott pályát követ, és különböző ügyességi feladatokat képes önállóan megoldani. Diplomamunkám keretében egy ilyen autót fejlesztettem tovább, versenyfeladatokon túlmutató képességekkel felruházva. Feladatom a vezérlőegység hardverének és szoftverének továbbfejlesztése volt, ami magában foglalta a szükséges új hardverek megtervezését, és a szoftver implementálását.

Az új vonalszenzor feladata a jármű számára fekete vonallal kijelölt pályán való haladás javítása. Ehhez egy kétsoros 2x24 szenzort tartalmazó érzékelőt terveztem korábbi tapasztalataim alapján. Megépítése után, különböző szűrések és algoritmusok alkalmazásával javítottam a felbontásán, ami ennek köszönhetően fél mm pontossággal képes meghatározni a vonal középpontját, és egy fokos pontossággal annak szöghelyzetét, mindezt 4 kHz-es frissítés mellett. A jelek mérését és előfeldolgozását – ami során nagyfrekvenciás jelekkel kellett dolgoznom – egy Spartan6-os FPGA-val valósítottam meg. A további kiértékeléseket az autó főprocesszora, egy ARM Cortex-M4-es végzi.

Fő feladatom a helymeghatározás volt, melyet inerciális helymeghatározással (enkóder + giroszkóp) teljesítettem. Mérésekkel vizsgáltam a szenzorok és különböző modellek pontosságát, majd javaslatokat tettem annak javítására. Kombinálva a vonalkövető funkciót és helymeghatározást, leprogramoztam a pályarögzítő funkciót. A rögzített pályarajzot átküldi az autó Bluetooth-on keresztül a számítógépnek, amit egy szintén saját készítésű szoftver, grafikusán megjelenít. Az adatok letárolhatóak és bármikor visszatölthetőek a járműre, ami ezek után vonalszenzor használata nélkül is kis hibával képes megismételni az útvonalat. Az adatok külső bevitele esetén, pedig koordinátákkal is lehet útvonalat rajzolni az autó számára. Az implementált funkciók rövid és közepes távolságokon megfelelően működnek, ellátják funkciójukat.

Végezetül pedig összegyűjtöttem milyen előnyeink származhatnak a versenyen/bemutató során, ezeknek a kiegészítő funkcióknak a használatából, és felsoroltam továbbfejlesztési ötleteimet.

Abstract

The Department of Automation and Applied Informatics of the BUTE organizes the RobonAUT designing and engineering competition in each year since 2012. The task of the race is to design and build an autonomous robot car, that follows a path not detailed in advance, and can perform different objectives. As my final thesis I'm upgrading a car like this with features that are not included in the race's objectives. My task is to develop further the controller's hardware and software, including the design of the necessary new hardware and the implementation of the software.

The aim of the new line scanner is to improve the ability of following a path marked by a black line. For this task, I designed a dual line sensor (including 2x24 individual sensors), using my earlier experiences. After the production, I improved its resolution by different filters and algorithms, reaching a resolution of half mm in determining the midpoint of the line, one degree in determining the angle, and its update frequency can be up to 4 kHz. For measuring and preprocessing the signals – that includes high frequencies – I use a Spartan 6 FPGA. The post-processing is done by the car's main processor, an ARM Cortex-M4.

My main task was determining the location of the car. I completed it by using inertial localization (encoder + gyroscope). Then I compared the precision of the sensors and different models by measurements, and proposed solutions for improvements. With the combination of the line following ability and inertial localization I programmed the path recording function. The car transmits the recorded data for the PC via Bluetooth, and my own PC software displays this path graphically. The data can be saved and can be downloaded to the car anytime. By this map file, the car can repeat the same path with minimal inaccuracy without using the line sensors. The path also can be drawn by external coordinates. These implemented functions work well in short and medium ranged distances, it fulfills the requirements.

Finally I collected the advantages of these new, extra-abilities and what can we benefit from them during the race/show. I also listed some further development ideas.

1 Bevezetés

A BME Automatizálási és Alkalmazott Informatikai Tanszéke 2012 óta minden évben megrendezi RobonAUT nevű műszaki tervezési és konstrukciós versenyét. A verseny célja egy olyan autonóm robotautó tervezése és építése, ami egy előre részleteiben nem meghatározott pályát követ, természetesen a lehető leggyorsabban, és különböző ügyességi feladatokat képes önállóan megoldani, például útelágazások, akadálykikerülés, vezérvonal átmeneti hiánya, parkolási feladatok. A verseny részletei elolvashatóak a verseny hivatalos honlapján (<http://robonaut.aut.bme.hu>).

Háromfős csapatunk 2014-ben vett részt először a versenyen, amit eredményesen zártunk: a résztvevő 20 csapat közül 2. helyezést értünk el. A kiinduló alapként kapott távirányítós autó automatizálása során sok tapasztalatot szereztünk, és a tesztek, valamint a verseny során rengeteg továbbfejlesztési ötlet jutott az eszünkbe. Komancsik Norberttel együttműködve célunk ezeket a versenyfeladatokon túlmutató képességeket megvalósítani.

Diplomadolgozatom témája a robotautó azon részeinek továbbfejlesztése és gyártása, melyeket már a versenyre is én magam terveztem, kiegészítve az új funkciókkal. Feladatom a vezérlőegység hardverének és szoftverének továbbfejlesztése. Ez magában foglalja a hardvertervezés lehetőségeinek vizsgálatát, a szükséges új hardverek megtervezését, és a szoftver implementálását is.

A bővítő részegység feladata a jármű számára fekete vonallal kijelölt pályán való haladás során a pálya adatainak rögzítése. A járműnek végig kell tudnia haladni a korábban rögzített pálya mentén vonalszenzor használata nélkül is, illetve előre koordinátákkal megadott útvonalon.

2 Feladat részletezése

2.1 Vonalkövetés

A járművünknek egy fekete szigetelőszalaggal rajzolt vonalat kell tudnia követni. Ehhez szükségünk van egy optikai szenzorra, mely folyamatosan figyeli a robot alatt a talajt, és visszaadja a vonal középpontjának előjeles távolságát a jármű szimmetriatengelyéhez képest. Több vonal esetén a többi vonalat is érzékelnie kell.

2.2 Helymeghatározás

A járműnek végig kell tudnia haladni előre, koordinátákkal megadott útvonalon, illetve eltávolítani a szigetelőszalaggal jelölt pálya pontos rajzát. Ennek végrehajtásához elengedhetetlen, hogy a robot pontosan tudja magáról, hol helyezkedik el a térben. A versenyszabályzat értelmében a robot nem kommunikálhat a külvilággal, és nem helyezhetünk el tereptárgyakat sem, így a globális helymeghatározó rendszerek nem jöhetnek szóba. A feladat megvalósítását egy inercia szenzor beépítésével és egy, a kerék forgását figyelő, enkóder segítségével tervezem megvalósítani. A pontos pályarajz felméréséhez szükség van egy megbízhatóan működő vonalszenzorra is.

3 Vonalszenzor

3.1 Célkitűzések

Gyors rendszert szeretnék, ami nagy sebességnél is észreveszi a pályán elhelyezett keresztvonalakat, illetve a jelfeldolgozást követően a szabályzó legalább olyan gyors legyen, mint a rákapcsolt digitális szervó frissítési frekvenciája.

A vonalszenzor ne csak teljesen világos talajon legyen működőképes, ne kelljen minden alkalommal a talajhoz és fényviszonyokhoz kalibrálni, és legyen érzéketlen enyhébb szennyeződésekre, csempék közti fugára.

A vonal pozícióján kívül legyen képes annak szöghelyzetét is meghatározni az autó szimmetriatengelyéhez képest. Ezáltal egy jobb kormányzás valósítható meg, ami nagysebességű kanyarodás esetén képes ellenkormányozni az autót, kisodródó hátsókerék esetén.

3.2 Optikai szenzor tervezése

3.2.1 Szenzor típus kiválasztása

A vonal érzékelésénél tehát két fontos szempontot kell szem előtt tartani: a sebességet, és a felbontást. Piackutatást végezve a kapható szenzorokról, három fajta rendszer használatát gondoltam végig:

Reflexiós optocsatoló: Ezek a legegyszerűbbek, különálló optikai elemek (fotodiódák és fototranzisztorok) vannak egymás mellé helyezve. Előnye a nagy sebesség, kis alkatrész költség, de felbontása korlátozott.

Vonalszkenner: Működési elve hasonló az előző típushoz, sűrűn elhelyezett fényérzékelőkből áll, így jó felbontással rendelkezik, ám feldolgozó elektronikája korlátozott sebességű. Az ilyen típusú szenzorok költsége ráadásul meghaladja a rendelkezésre álló költségkeretet, így más megoldást kell keresni.

CCD kamera: Ez esetben egy, az autó tetején elhelyezett kamera figyelné az utat, így jó előre ismerné a pályát, azonban az olcsó CCD detektorok nagy sebességnél, képfrissítési frekvenciájuk és fényérzékenységük miatt elmosódott képet adhatnak. Alkalmazásuk komoly képfeldolgozási műveleteket, így nagy számításigényű hardvert követelnek meg.

Választásom a külön elemekből álló, saját tervezésű szenzorra esett, melynek felbontását és sebességét úgy választhatom meg, hogy az megfeleljen a feladatra.

3.2.2 Szenzorsor elhelyezése

A vonalszenzort célszerű minél előbbre helyezni a járművön, hogy előbb kapjunk visszacsatolást a vonal helyzetéről. A szenzor viszont lejjebb lóg az az autó alvázról, és akadályozhatja az emelkedőkön való átjutást. Ezért a szenzort a közvetlenül az első kerék tengelye alá helyezem. Ott a szenzor földtől való távolsága terepviszonyoktól függetlenül közel állandó. Ez az egyes optikai szenzorok jelének feldolgozása szempontjából is egy fontos feltétel.

A szenzor szélességét illetően a cél a maximális rendelkezésre álló hely kihasználása volt. A kerék elfordulásának akadályozása nélkül ez maximálisan 17,5 cm lehet.

3.2.3 Optikai elemek kiválasztása, elrendezése

Optikai elemekként első körben a szponzorációba kapható CNY-70-es reflexiós optikai szenzorok használatát fontoltam meg. Ennek fizikai szélessége 7 mm. A vonal pozícióját azonban ennél jobb felbontásban kellene ismerni, így gondolkoztam más szenzor használatában is. SMD tokozású infra LED-eket és fototranzisztorokat beépítve egymástól 2,5 mm-enként lehetett volna pontos információnk a vonalról. Teljes szélességben, ilyen felbontáshoz viszont 64 szenzorra lett volna szükség, melynek ára még infraszűrő nélküli fotodiódák használata mellett is jelentős részét elvitte volna a rendelkezésre álló költségkeretünknek, így ezt a verziót pénz hiányában elvetettem.



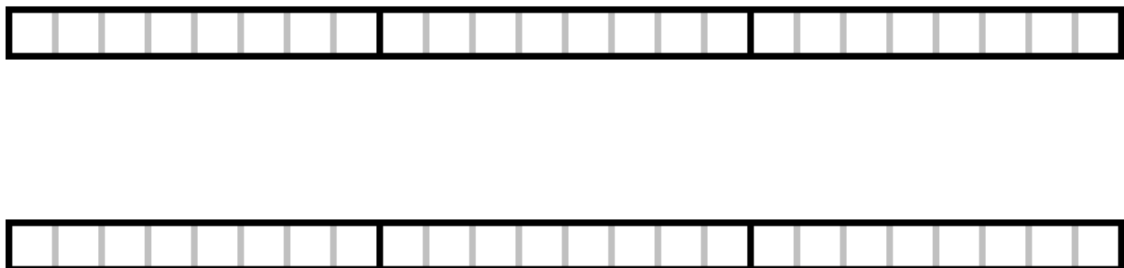
1. ábra: Szenzor elemek korábbi elrendezése

Választásom maradt a CNY-70-es elemeken, amiket végül két sorban helyeztem el. Az egyik sort az ábrán (1. ábra) látható módon 3,5mm-el eltolva, hogy növeljem a felbontást. Erre a felbontásra és finom korrekciókra a hosszú egyenesekben nagy sebességgel történő haladáskor van szükség, amikor feltehetően középen helyezkedik el a vonal, és mivel csak korlátozott számban álltak rendelkezésemre a kiválasztott

elemek, ezért a második sorban csak középen helyeztem el szenzorokat, és helyette szélesebbre terveztem az első sort.

Ennek az elrendezésnek van egy komoly hátránya, amit az új szenzor tervezésekor figyelembe kell venni. Az első sor 7 mm-el előrébb helyezkedik el, nem ugyan azt a vonalat látja, mint a hátsó sor. Ha az autó nagyjából egyenesen halad, akkor ez kiküszöbölhető azzal, hogy az első sor mérési eredményeit eltárolom, és az autó 7 mm-el való elmozdulása után mért, hátsó soros szenzoradatokkal együtt dolgozom fel.

A jelenlegi koncepció szerint a vonal szöghelyzetét is szeretném mérni, ehhez pedig az ilyen módon eltolt adatok felhasználása nem alkalmazható, mivel ez a késleltetős módszer csak akkor működik jól, ha közben az autó, kanyarodás nélkül, egyenesen előre halad. Én pedig pont a kanyarokban szeretném kihasználni a nagyobb felbontást és a vonal szöghelyzetéről való információt. Az új vonalszenzor tehát egy sorban elhelyezett elemek alapján fogja meghatározni a vonal középpontját. Az autó tengelyével bezárt szög meghatározásához két ilyen sort helyezek el az ábrán látható módon (2. ábra).



2. ábra: Szenzor elemek új elrendezése

A két látszólagos vonalközéppont, és a szenzorsorok távolságának ismeretében meghatározható lesz ez a szög. Ez annál pontosabban határozható meg, minél távolabb helyezkednek el a szenzorok, viszont annál kisebb lesz a maximális szögelfordulás, amit még detektálni tudunk. A szenzorok elhelyezésekor fontos szempont az emelkedőkön való akadálymentes áthaladás, ezért csak az első vagy hátsó tengely alatt helyezhető el szenzorsor. A tengelytáv túlságosan nagy ahhoz, hogy a tengelyek alatt elhelyezve a két sort, detektálni tudjuk a vonalferdeséget ellenkormányzáskor, ezért mindkét sor az első tengely alatt lesz felszerelve, egymástól néhány centiméteres távolsággal. Az így számolt szöginformáció felhasználhatóságához, igen pontos, nagyjából milliméteres felbontású szenzorsorra van szükség, amit az egyes szenzorok jeleinek interpolációjával kívánok elérni.

35 mm nyák = 21 mm hézag = 28 mm távolság

2°-os elfordulás = $28 \cdot \tan(2^\circ) = 0,98$ mm-es felbontást követel

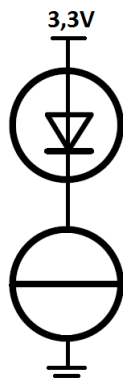
A szenzorsor számára 175 mm hely áll rendelkezésre, a szenzorok pedig 7 mm-esek, így 24 szenzor helyezhető el egy sorban kényelmesen. Azaz összesen 48 csatornát kell mérni.

3.2.4 Optikai elemek bekötése

A CNY-70-es szenzorok 2 elemből tevődnek össze. Egy infra LED-ből és egy villamosan független fototranzisztorból. A LED dióda fényt bocsájt ki, melynek fénye a falról/földről visszaverődve aktiválja a fototranzisztort, és a fénnel arányos árammennyiséget enged át.

A LED-ek kapcsolását tekintve (3. ábra) egyszerű a helyzet, a megengedhető maximális áramerősséggel hajtom meg azokat, a maximális fényerő érdekében, hogy a külső beszűrődő fényhez képest, minél erősebb legyen a visszavert fény.

A fototranzisztorok esetében az azokkal sorosan kapcsolt ellenállás feszültségét mérem (4. ábra), ami egyenesen arányos lesz az áramerősséggel. Az ellenállásérték meghatározásához méréseket végeztem.



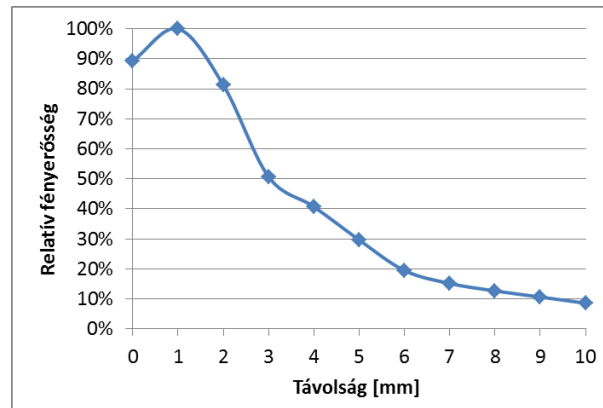
3. ábra: LED-ek kapcsolása



4. ábra: Fototranzisztor kapcsolása

3.2.5 Földtől való optimális távolság

Először megvizsgáltam hogyan alakul a szenzor érzékenysége a távolság függvényében. A visszaverődő fényerősséget mértem fehér papír esetén különböző távolságokból. Az eredmény az 1. grafikonon látható. 6 mm felett már nagyon elhalványul, nem érdemes távolabbról használni.



1. grafikon: Szenzor távolság karakterisztikája

Ezt követően a szenzor látószögét vizsgáltam. Ahhoz, hogy a szenzorok felbontását interpolációval megnöveljem, a szenzorokat a földtől úgy kell elhelyezni, hogy a látószöge a teljes maga alatti területet belássa, de lehetőleg a szomszédos szenzor alá már ne lásson. Ennek mérésére, a fehér papírlapra egy 7 mm széles szigetelőszalag darabot ragasztottam, ráhelyeztem a szenzort, és elkezdtem milliméterenként felemelni. A fekete szigetelőszalag szinte teljes mértékben elnyeli a fényt, így amikor elkezd növekedni a fototranzisztor árama, annál a távolságnál kezd el belátni a szomszédos szenzor alá. Ez a távolság 4-4,5 mm-re adódott, ami még az átlagos érzékenységtől tartományba esik, tehát a szenzort ilyen távolságban kell elhelyezni a földtől.

3.2.6 Fototranzisztor sönt ellenállása

A földtől való távolság ismeretében kimértem mekkora áramerősséget eredményez az infra LED fehér lapról visszavert fénye. Ezt követően megmértem, külső fény esetén mekkora a bevilágítás mértéke, ami tovább növelheti az áramerősséget, és nem jól megválasztott mérőellenállás esetén telítésbe megy a tranzisztor. A mérést reggeli alacsony napállás mellett végeztem, fehér lappal, melyet üveglapon keresztül besütő nap fénye világított meg. Ekkor a LED fényének többszöröse lett az alapérték, tehát jelentős a kívülről beszűrődő fény, számításba kell venni.

Az ellenállásértéket úgy választottam meg, hogy a mért beszűrődő nap, és a LED fénye együttesen se eredményezzen nagyobb feszültséget az ellenálláson, mint amennyit az ADC mérni tud. Ez az ellenállás érték: 360 ohm.

3.2.7 Analóg jelek mérése

Összesen 48 darab analóg csatornát kell mérnem, amihez szükség van néhány AD átalakítóra és analóg multiplexerekre. Az AD átalakítók ára relatív magas a többi alkatrészhez képest, ezért célszerű azok számát minimalizálni, és egy ADC-vel minél több csatornát mérni.

Analóg multiplexerekből a legelterjedtebb a 8:1-es változat, de létezik 16:1-es is. Én a 16:1-eset választottam, mert így már elegendő 3 AD átalakító. Természetesen lehetne több lépcsőben alkalmazni a multiplexereket és egyetlen AD átalakítóval mérni minden csatornát, de az már jelentősen megnövelné a teljes mérés időtartamát.

A multiplexerek kapcsolási ideje az AD konverzió idejéhez képest elhanyagolhatóan kicsi, így a továbbiakban csak az integrált áramkör árát vettem figyelembe a használandó multiplexer kiválasztásakor, melynek típusa: CD74HC4067SM96.

A multiplexer után következett az analóg digitális átalakító kiválasztása. Itt fontos szempont volt az ár, és a sebesség. Alapos piackutatás eredményeként találtam egy olyan AD átalakítót, ami 1 MSample sebességre képes, egyidejűleg két csatornát mér, és az ára is elfogadható. A kiválasztott ADC típusa: ADC122S706CIMT.

A kétcsatornás AD sebességének kihasználása érdekében a 48 csatornát inkább 4 egyenlő részre osztom fel, ezzel megrövidítve a teljes mérés idejét. 12:1-es multiplexerek nincsenek, így maradnak a kiválasztott 16:1-esek, csak nem lesz minden bemenetük kihasználva.

Mind az analóg multiplexerek mind a LED csoportok vezérléséhez célszerű shiftregisztereket használni, így spórolva a szükséges jelvezetékek számával. Erre a célra kiválasztott általános célú nagysebességű shift regiszter típusa: 74HC595D. Az analóg multiplexerek vezérlésére ebből 4 bitre van szükség.

3.2.8 Analóg jelek mérési stratégiája

Korábban már említettem, hogy a külső beszűrődő fény, jelentős feszültségnövekedést okoz a mérőellenálláson, ezért annak mértéke nem elhanyagolható a visszavert fény mérésekor. Ennek kiszűrésére kétszeri mérést alkalmazok: visszavert fény mérése felkapcsolt LED-ekkel, és egy sima fénymérés LED-ek nélkül. A szenzor

karakterisztikája nagyjából lineárisnak tekinthető, így a két mért értéket egymásból kivonva megkapom mit mért volna a szenzor külső fény nélkül.

Szükség van tehát a LED-ek kapcsolgatására, nem világíthatnak folyamatos üzemben. A folyamatos üzem energiafelhasználás szempontjából sem megengedhető, hiszen 48 db szenzor 50 mA-es fogyasztása 3,3 V-os feszültségen közel 8W-os fogyasztás jelent, ami hamar lemerítené az akkumulátort.

További probléma, hogy a szenzorsor földtől való távolságát a fototranzisztor látószögéhez igazítottam, de a LED látószöge ettől nagyobb, valamint vissza is verődik a talajról és bevilágít a szomszédos szenzoroknak. Ebből következik, hogy a LED-eket nem elég egyszerre fel-le kapcsolni, hanem blokkokra osztva kell kapcsolgatni minden valahányadik LED-et.

A tavalyi járműre elkészített szenzoron meghagytam a LED-ek tetszőleges sorrendben történő kapcsolgatásának lehetőségét. Ehhez a szintén szponzorációba kapott STP08CP05TTR típusú LED meghajtó IC-ket választottam, melyek további előnye, hogy beépített áramgenerátort is tartalmaz, nincs szükség áramkorlátozó ellenállások beépítésére.

A tavalyi szenzorral idén méréseket végeztem, egyesével felkapcsoltam a LED-eket, és kiolvastam a szomszédos szenzorok által mért fényerősségeket. A felkapcsolt LED-hez tartozó szenzor fényerejét a táblázatban 100%-nak vettem, ehhez arányosítottam a többi által mért fényerősségeket.

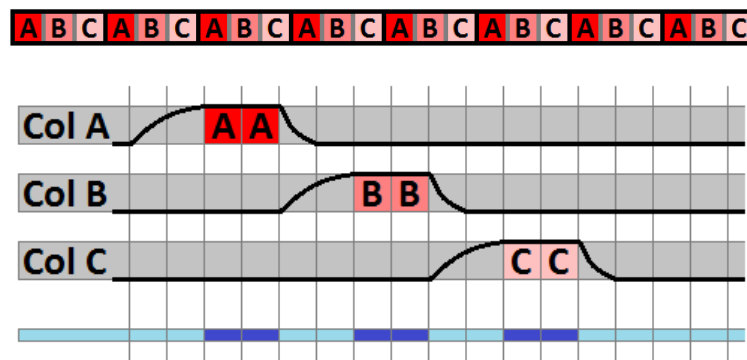
1. táblázat: LED bevilágítás

LED 1	LED 2	LED 3	LED 4	LED 5	LED 6
0,0%	0,0%	0,0%	2,7%	35,2%	100,0%
0,0%	0,0%	2,4%	27,7%	100,0%	22,1%
0,0%	1,9%	31,6%	100,0%	22,0%	1,7%
2,5%	27,5%	100,0%	24,2%	1,6%	0,3%
30,3%	100,0%	28,3%	1,2%	0,0%	0,0%
100,0%	19,8%	3,6%	0,0%	0,0%	0,0%

Az 1. táblázatból kiderül, hogy a szomszédos szenzoroknál erősen jelentkezik ez a bevilágítás, de a második szomszédnál viszont már igen csekély. Ha a szenzor sebessége megköveteli, hogy minden második szenzort egyszerre mérjünk, akkor minimális hibával felkapcsolható minden második LED, de ha erre nincs szükség, célszerű csak minden harmadikat.

Az új szenzor tervezésekor a LED-ek egyenkénti kapcsolgathatóságáról lemondok, és inkább előre megtervezett blokkokban kapcsolom azokat. Ezáltal egy egyszerűbb kapcsolást kapok, amivel a NYÁK mérete is csökkenthető.

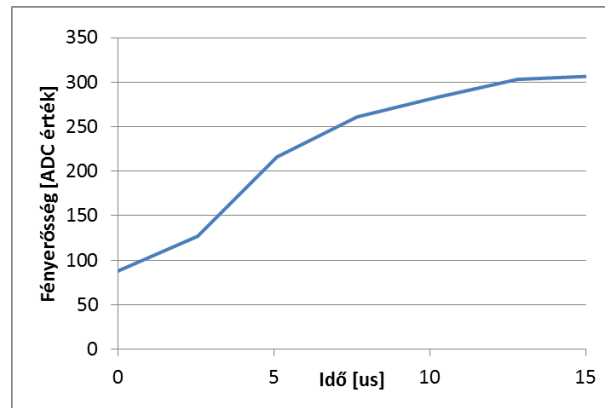
Az egymásnak bevilágító LED-ek miatt a mérés menete a következő. Vegyük az egy sorban elhelyezett LED-eket, és foglaljuk őket rendre A, B, C csoportokba. A mérés indításakor felkapcsoljuk az A csoportot. Ekkor meg kell várni, amíg a LED-ek elérik maximális fényerejükét és a fototranzisztorok árama is beáll. Ezután indíthatjuk el az analóg jelek mérését, mely valamekkora időt igénybe vesz, és ez idő alatt a többi LED-et még nem szabad felkapcsolnunk. Ha megtörtént az utolsó mérendő csatorna mintavételezése is, akkor lekapcsolhatjuk az A csoportot, és ezzel egyidőben felkapcsoljuk a B-t. Amíg a B csoport LED-jei felkapcsolnak, az alatt az „A” csoport LED-jeinek kikapcsolási tranziensei is lecsengenek, és időben indítható lesz a következő mérés.



5. ábra: Egy LED sor idődiagramja

Az idődiagramból (5. ábra) látszik, hogy amíg a kapcsolási tranziensek zajlanak, addig az AD átalakítók üresjáratban vannak. A két mérendő LED sor egymástól kellő távolságra van ahhoz, hogy egymást ne zavarják, így ilyenkor mérhetném a második sor szenzorjait. A kérdés az, hogy mekkora időszületet igényel a LED-ek kapcsolása, és mekkorát az egyidejűleg világító LED-ekhez tartozó fototranzisztorok jeleinek mérése.

Ehhez megmértem a jelenlegi rendszeren a LED-ek felkapcsolási idejét, ami 10 us-re adódott. A felkapcsolási tranziens a 2. grafikonon látható.



2. grafikon: LED felfutás

Egy sorban 24 LED található, 3 csoportra bontva ez 8 LED egyidejű mérését követeli meg. 4 darab AD átalakítóm van, így mindegyiknek két mérést kell egymás után elvégeznie. Egy mérés időtartama az átalakító adatlapja alapján számítva 1,6 us (@12Mhz).

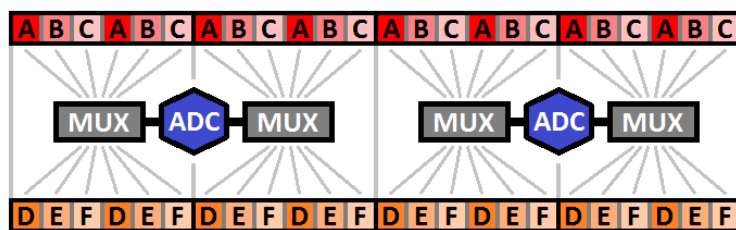
A csatorna mérési idejét meghosszabbítja, hogy az analóg multiplexer váltása után várni kell, amíg az AD átalakító bemenetén stabilizálódik az új feszültség. Erre újabb mérést végeztem. A csatornák közötti közel azonos jelszintek esetén ez a tranziens nem jelentős, viszont egy sötét szenzor mérése után, aminek kimeneti feszültsége 0V körüli, egy világító szenzor mérésekor meglepően hosszú időt vesz igénybe a beállítás, így a 2. táblázatban egy ilyen esetet mérési eredményeimet tüntettem fel. Látható, hogy 2,5 us után még mindig csak 95%-os a jelszint, de ez már elég jól közelíti a ténylegeset, így a továbbiakban ezzel számolok. Egy sorban és egy csoportban található szenzorok jeleinek bemérése tehát: $2 \cdot (1,6 + 2,5) = 8,2$ us időt vesz igénybe.

2. táblázat: ADC csatornaváltás

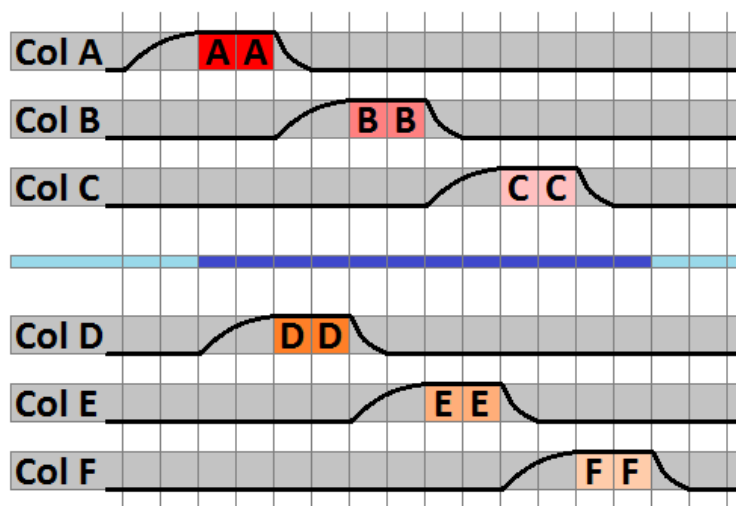
Idő[us]	1,44	1,76	1,92	2,24	2,56
Fényerő	<80%	88%	91%	94%	95%

Ez közel azonos a LED-ek felkapcsolási tranziensével, tehát megéri a két szenzorsor LED-jeit egymástól független blokkokba foglalni, időben késleltetve kapcsolni, és a második sor mérést az első sor tranziensei alatt mérni.

Ezek alapján a végleges kapcsolás (6. ábra) és vezérlési idődiagram (7. ábra):



6. ábra: LED-ek és szenzorok bekötése



7. ábra: LED-ek és szenzorok tervezett időzítése

3.2.9 Elektronikai kapcsolás

A vonalszenzor kapcsolási szempontból két különálló részre bontható: Egyik, ami a LED-eket és azok meghajtóit tartalmazza, másik pedig a fototranzisztorokat, az analóg jeleket feldolgozó multiplexerek és ADC, valamint a multiplexerek csatornakiválasztó shiftregisztere.

Tápellátás:

Az digitális elemek kivétel nélkül mind 3,3V-os feszültségről működnek. Erre már található egy 3A-es tápegység a járművünkön, így ez nem jelent különösebb problémát. A LED-ek meghajtásához is elegendő a 3,3V, mert nyitófeszültségük maximum 1,6V, és többet nem is érdemes rákötni, hiszen csak eldisszipálná a LED előtét ellenállása. Az egyszerűbb huzalozás érdekében a fototranzisztorokat is 3,3V-ra terveztem. Így a vele sorba kapcsolt ellenálláson elérhető maximális feszültség a tranzisztor nyitófeszültségével kevesebb (~0,7V-al), azaz nagyjából 2,6V. Az ADC referencia feszültségét ehhez igazítván 2,5V-nak határoztam meg. Ezt egy nagyon pontos LM385D-2.5 típusú referencia diódával állítom elő az ADC számára.

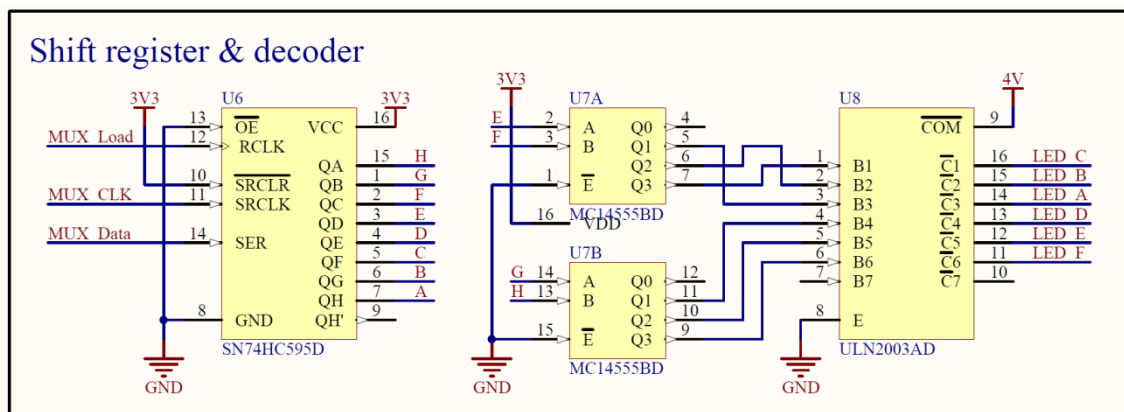
Eddig tehát mindenhez elég a 3,3V, az analóg digitális átalakítónak azonban szüksége van 5V-os tápellátásra is, az analóg részei számára. Ugyan 5V-os tápegység is van már az autónkon, de azon több digitális áramkör is van már, melyek zavarjeleket tesznek a tápfeszültségre, és ezzel zajossá tennék méréseinket. Ezért az ADC számára külön tápegységgel állítom elő a feszültséget közvetlenül az akkumulátor feszültségéből. Erre a célra egy speciális kisáramú (50 mA-es), de pontos, low drop típusú táp IC-t választottam, mely 0,5%-os hibával képes előállítani a kívánt feszültséget.

LED-ek:

A LED-ek 6 csoportjának kapcsolgatásához legalább 350 mA-es tranzisztorokra van szükség. Erre az ULN2003-as típusú 7db darlington tranzisztort tartalmazó IC-t választottam, mely 500 mA-t képes kapcsolni. A LED-ek áramát (50 mA) egy-egy sönt ellenállás korlátozza elemenként.

Az analóg multiplexerek bemeneteit kiválasztó shiftregiszter 8 bites, melyből csak 4 bitre volt szükségünk. További jelvezetékeket megspórolva a LED-ek kapcsolgatásához a maradék 4 bitet fogom felhasználni.

Az 14. oldalon található vezérlési diagram (7. ábra) alapján, egyszerre csak két LED csoport felkapcsolására van szükség. Egyik csoport az A-B-C valamelyike, másik pedig a D-E-F egyike. LED soronként 2-2 bitet felhasználva ez a 4 bit pont elegendő számomra, 2 biten ugyanis 4 állapot tárolható: a 3 csoport egyike, plusz a lekapcsolt állapot. Ehhez csak egy dekóderre van szükség, mely a bináris számot visszafejti és meghajtja a tranzisztorok bemeneteit. Erre a célra az MC14555DB típusú IC-t választottam, mely két ilyen 4:1-es dekódert tartalmaz, azaz egy db IC elegendő a feladatra. A kapcsolási rajz részletet a 8. ábra tartalmazza. (A shiftregiszter kimenetiből az ABCD lábak vezérlik az analóg multiplexereket, a későbbi 10. ábra szerint).



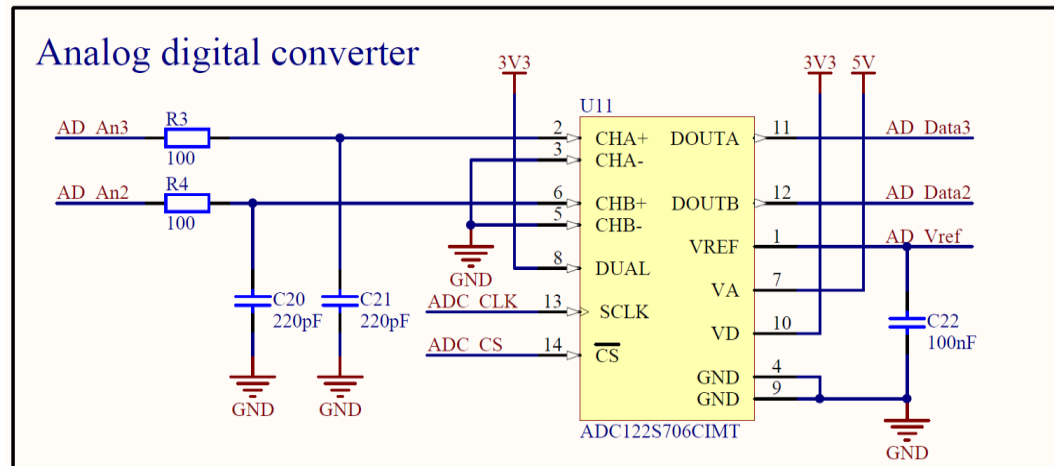
8. ábra: Shift regiszter és dekóder kapcsolási rajza

Analóg jelek:

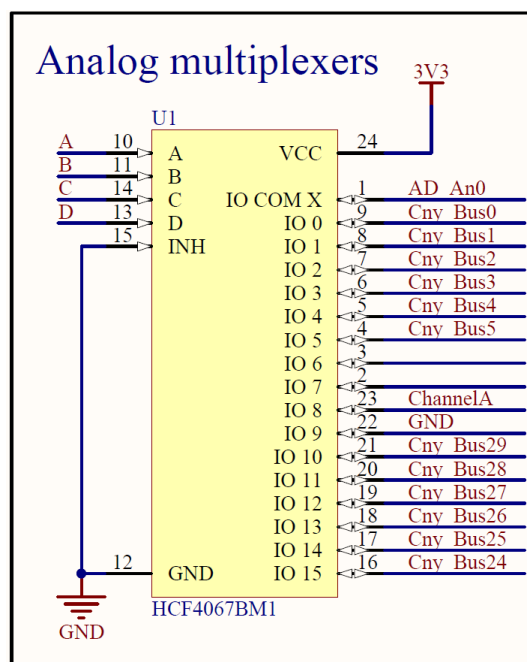
Az analóg jelek mérését tehát 2 darab 2 csatornás ADC végzi (9. ábra), melyek csatornáinak bemeneteire egy-egy 16:1-es analóg multiplexer van kötve (10. ábra). A mérési stratégia megvalósításához nincs szükség a multiplexerek különálló vezérlésére, így azok csatornakiválasztó bemeneteit bitenként közösíthetem.

ADC-k szükséges adatvezetékei: ADC_CLK, ADC_CS, ADC_DATA[4]

Shift regiszter vezetékei: MUX_CLK, MUX_DATA, MUX_LOAD



9. ábra: ADC kapcsolási rajza



10. ábra: Analóg multiplexer kapcsolása

A jelvezetékek csökkentése végett, ezek közül igyekszek összevonni, amit lehet. A shift regiszter maximális működési órajel frekvenciája jóval magasabb az ADC órajelénél, ezért megtehetem, hogy közös órajelről vezérlem a két IC-t, azaz $MUX_CLK = ADC_CLK$. Ezen kívül már csak az ADC_CS és MUX_LOAD vonható esetlegesen össze, a többi vezeték ugyanis már különálló információkat tartalmaz. Az ADC_CS jele úgy működik, hogy amíg logikai 1 jelszinten van, az ADC energiatakarékos, készenléti módban van, és amikor 0-ba vált, elkezd mérni. A mérés befejeztével újra logikai 1-be kell állítani, különben újabb mérés veszi kezdetét. A kiválasztott shift regiszter $LOAD$ lába érzékeny, tehát számára mindegy milyen jelszint van a bemenetén, csak a 0-ból 1-be váltás esetén változtatja meg a kimenetét.

Ha az ADC_CS és MUX_LOAD lábat összekötöm, akkor az AD konverzió befejeztével történik meg a shift regiszter betöltése, tehát az analóg csatorna kiválasztása, és az a következő mérés végezetéig úgy is marad. Ez a fajta működés számomra teljesen megfelelő, ezért össze is kötöm azokat.

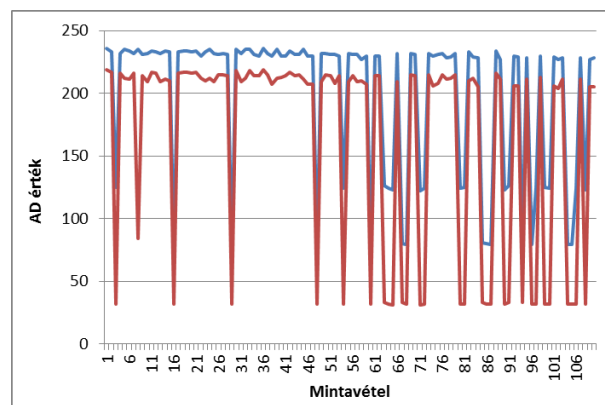
A szükséges adatvezetékek (7 db) tehát: MUX_CLK, MUX_LOAD, MUX_DATA, ADC_DATA[4]

3.2.10 NYÁK tervezés szempontjai

A NYÁK tervezésekor a legfontosabb szempont a nagyfrekvenciás jelek megfelelő elvezetése, és a millivoltos analóg jelek pontos méréséhez megfelelő árnyékolás használata. Az árnyékolást úgy oldottam meg, hogy amennyire lehetett próbáltam egyoldalasra tervezni a NYÁK-ot, a másik réteget pedig földdel töltöttem ki. Ezen kívül kerültem, hogy az analóg jelvezetékek közelében nagyfrekvenciás digitális jelek legyenek.

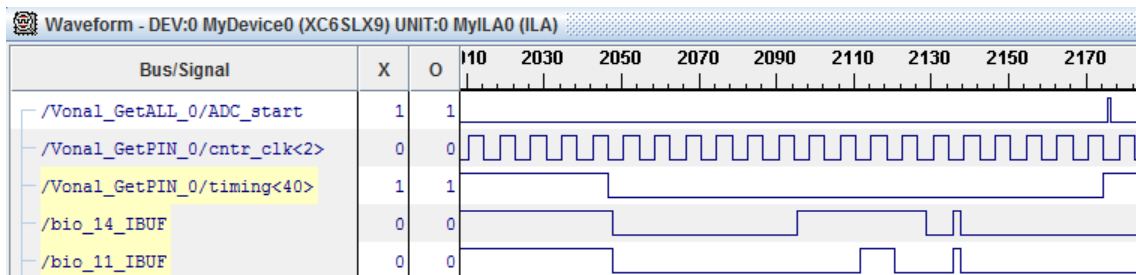
Nagyfrekvenciás jelek:

Ami több nehézséget okozott, azok a nagyfrekvenciás jelek voltak. Ezeket a jeleket egyrészt célszerű minél rövidebb úton eljuttatni a cél IC-hez, másrészt kerülni kell a 90°-os töréseket a nyákon, amik visszhangot kelthetnek a jelekben. A korábban készített vonalszenzor nem volt képes maximális frekvencián üzemelni, mert ott a működése nagyon bizonytalanná vált, ezért fontosnak tartottam a hiba okát megtalálni, és orvosolni. Bizonyos csatornákat problémamentesen mért, egyes csatornák pedig tele voltak hibákkal. A két legmegbízhatatlanabb csatornának egymást követő mintavételezéseit a 3. grafikon mutatja.



3. grafikon: ADC zaj 12Mhz

A mért jel egy konstans feszültség volt, mégis keletkeztek benne fele, negyed, illetve nyolcad akkora tüskék. Ezek 2 hatványai, digitális hibára vallanak. Ráadásul összefüggés is megfigyelhető a két csatorna között, a hibák egyszerre keletkeztek. Mivel a csatornák 80%-a hibamentes volt, ezért tudtam, hogy zavarjelek produkálják a jelenséget, és nem arról van szó, hogy az AD átalakító működési sebessége vagy a szalagkábel hossza korlátozná a maximális átviteli frekvenciát.



11. ábra: ADC hibadetektálás

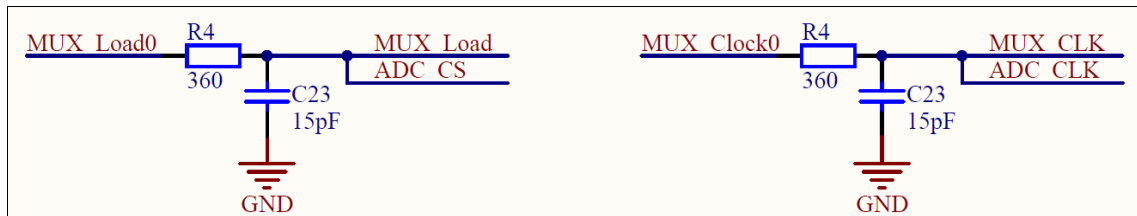
Digitális analízátorral megvizsgáltam a jelalakokat, melynek eredménye a fenti idődiagramon (11. ábra) látható. Észrevehető, hogy vannak negyed órajel hosszú visszaérkező jelek az AD átalakítótól. Ezt követően több jelet nem is adott az AD, valamiért leállt. Arra gondoltam, hogy a ADC_CS jelen keletkezhet ilyenkor valami zavar, de ez esetben nagyimpedanciás állásba kerül a kimenet, és az a szórt kapacitások miatt, fel/lehúzó ellenállás hiányában megjegyezné korábbi állapotát. Feltételezve, hogy az AD átalakító hibátlan, már csak az órajellel lehettek problémák. Terveztem rá egy RC szűrőt, hogy az esetleges zavarokat kiküszöbölve elkerüljem, hogy az órajel élváltásakor esetlegesen több órajelet érzékeljen. Ez megmagyarázná miért lép olyan gyorsan tovább a következő bit-re, és fogynak el a további shiftelendő bitek idő előtt.

Az RC szűrő időállandóját úgy választottam meg, hogy az órajel fél periódusának 1/3-a alatt érje el a logikai magas szintet. Ezt nagyjából 2 időállandónyi idő alatt eléri. A 16Mhz-es órajel periódusideje 62,5ns, a választott időállandó tehát ennek 1/12-ed része, azaz 5,2ns. Az ellenállás megválasztásakor igyekeztem minél kisebb értéket választani, így nagyobb kondenzátorra van szükség, és kevésbé módosítják számításaimat az egyéb jelenlévő szórt kapacitások. Az ellenállás minimális méretét a maximális kimeneti áram szabja meg, amit 10mA-re választottam. Ez még nem jelent különösebb problémát az FPGA-nak.

Az ellenállás minimális értéke: $R = U/I = 3,3/0,01 = 330 \text{ ohm}$.

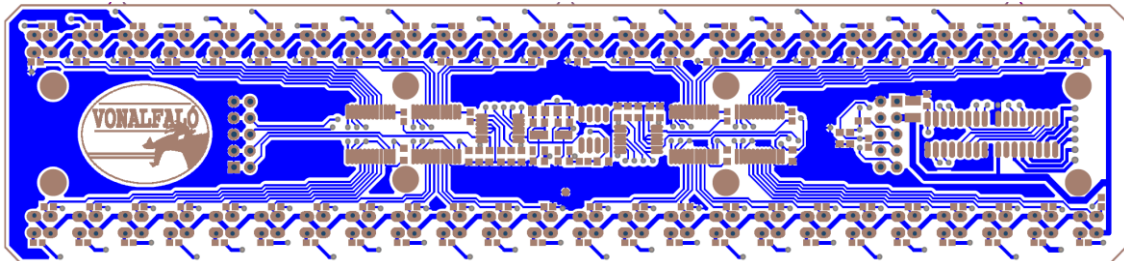
Hozzá tartozó kapacitás: $C = \tau / R = 5,2 / 330 = 15,7 \text{ pF}$.

A választott kapható értékek végül 360 ohm és 15 pF. Biztos, ami biztos alapon nem csak az órajelre, de a LOAD és CS lábakra is tettem egy ilyen szűrőtagot, kiegészítve ezzel a kapcsolási rajzot a 12. ábra szerint.

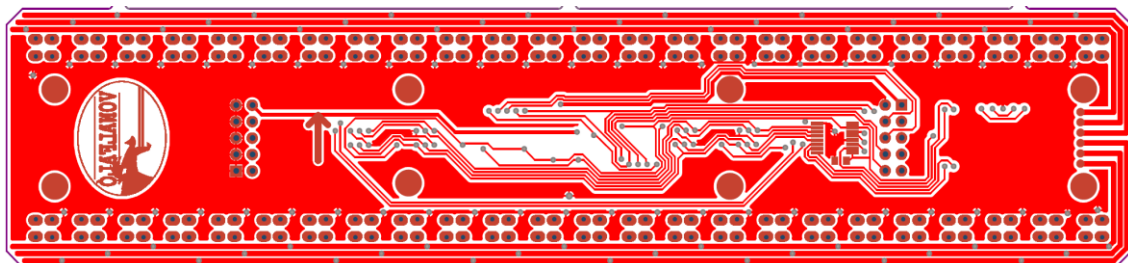


12. ábra: Digitális szűrő kapcsolása

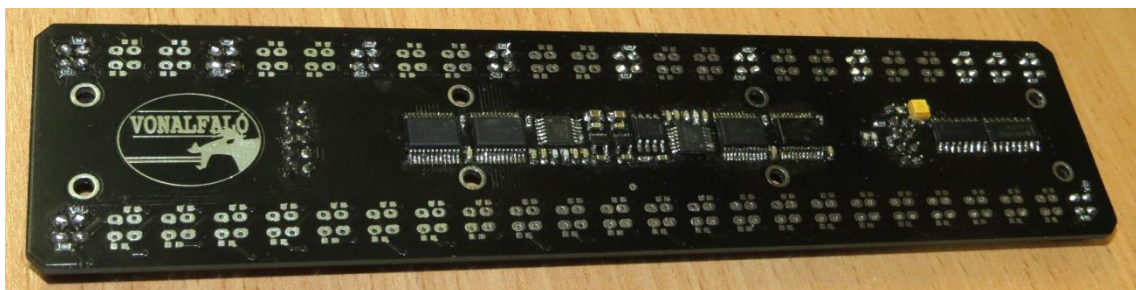
3.2.11 Elkészült NYÁK tervek és hardver



13. ábra: Vonalszenzor NYÁK – Top Layer



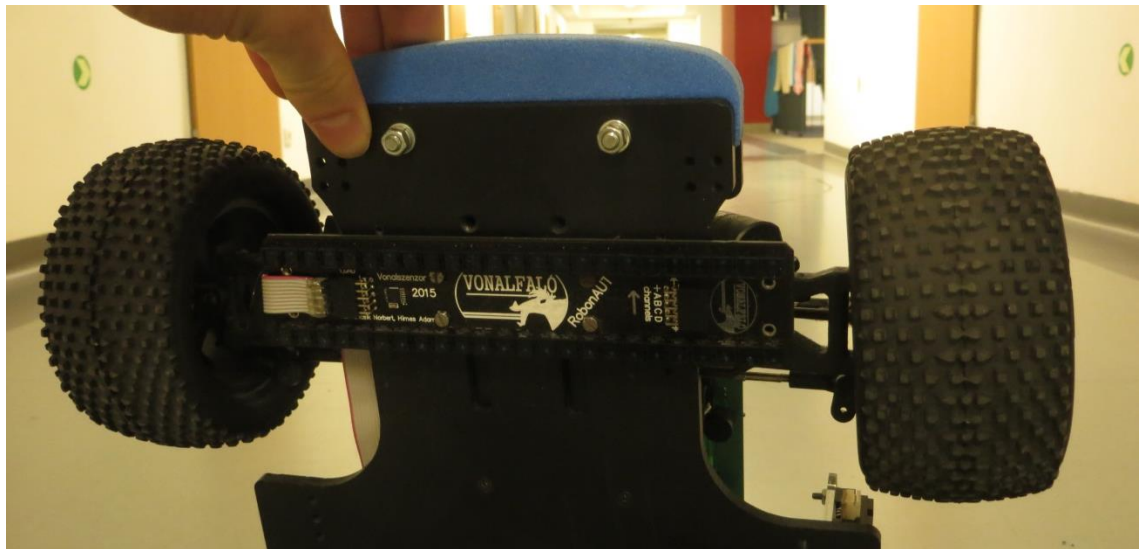
14. ábra: Vonalszenzor NYÁK – Bottom Layer



15. ábra: Vonalszenzor – Felső oldal



16. ábra: Vonalszenzor – Alsó oldal



17. ábra: Vonalszenzor – Felszerelve

3.3 Jelfeldolgozás

3.3.1 Jelfeldolgozó egység

A szenzor shift regisztereinek és AD átalakítójának maximális sebessége 16Mhz. Ahhoz, hogy ezt ki is tudjam használni, legalább 32Mhz-es mintavételezés, de inkább annak is a duplája szükséges. Ez a sebesség már a mikrokontrollerek határát feszegeti ráadásul ilyen gyakorisággal el is tárolni, feldolgozni a többcsatornás jeleket, elvinné a processzor számítási teljesítményének nagy részét. A vonalszenzor hardverének kezelésére ezért egy FPGA-s megoldást választottam.

Az FPGA-k több száz Mhz-en képesek működni, és mivel logikai áramköröket tartalmaznak, az ebben kialakított egységek párhuzamosan működnek. Sebességük független a bemenetek számától, mérendő jelek mennyiségétől. Erre a célra a MIT tanszéktől kölcsönkért Logsys Spartan 6-os FPGA-s kártyát választottam. Ennek legfőbb oka a költséghatékonyság volt, de az is szerepet játszott a kiválasztásában, hogy

korábbi tantárgyaim keretében már programoztam ezt a típusú panelt, ezért gyorsan tudtam fejleszteni rá, Verilog programozási nyelvet alkalmazva.

Az FPGA-s kártyán a 3.2.9-es fejezetben felsorolt 7 db nagysebességű I/O portra van szükség a szenzor teljes kezeléséhez, melyből 3 kimenet, 4 pedig bemenet. Feladata a LED-ek ciklikus kapcsolgatása megfelelő sorrendben, és a fototranzisztorok által mért feszültségek beolvasása. A mért eredményeket eltárolja, majd miután megmérte minden egyes szenzor jelét le-, illetve felkapcsolt LED-del is, elvégzi a szükséges kivonásokat (világos mínusz sötét mérés), ezzel előkészítve az adatokat a további kiértékelésre.

Az adatok további feldolgozását FPGA-n megvalósítani már hosszadalmas tervezési feladat lenne, és a jelfeldolgozás ezen fázisa már nem követeli meg az FPGA alkalmazását, ezért a további lépéseket az autó főprocesszorán fogom futtatni. Ez a processzor a szponzorációba kapott STM32F4DISCOVERY fejlesztői kártya, mely elegendő számítási teljesítményt nyújt a feladat elvégzésére. A két kártya SPI buszon fog kommunikálni egymással.

3.3.2 Feldolgozás lépései

- kommunikáció a hardverrel
- szenzorok bemérése a LED-ek kapcsolgatásával
- világos, sötét mérés közti különbségképzés
- szenzoronkénti kalibráció
- mérési adatok számának duplázása két szomszéd átlagolásával
- referenciaképzés (komparálási feszültségszint meghatározása)
- komparálás a referenciával
- bináris adatsorban vonalkeresés
- vonal középpontjának meghatározása az analóg értékekből
- vonal szöghelyzetének megállapítása a második szenzorsor használatával
- egyéb felismerendő vonaltulajdonságok vizsgálata

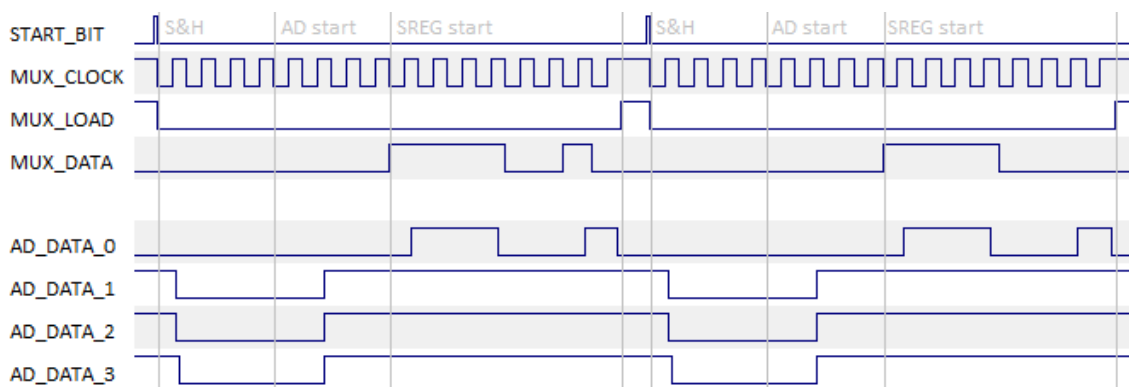
3.3.3 Hardverkezelés

Első lépésként egy olyan logikai modul tervezése szükséges az FPGA-ban, ami a vonalszenzor hardvereit kezeli. Azaz kapcsolgatja a LED-eket, kiválasztja a mérendő analóg csatornát és fogadja az AD átalakítók jeleit.

Az általam kiválasztott ADC 4 db jelvezetékkel rendelkezik. Szükség van egy maximum 16 Mhz-es AD_CLOCK órajelre, egy AD_CS (ChipSelect) jelre, ami a konvertálást engedélyezi, egy és csatornánként van 1-1 kimeneti adat (AD_DATA[4]). Üresjáratban az AD_CS lábat magas szinten kell tartani, és annak lehúzásakor kezdődik a konverzió. Az első 4 órajel alatt a Sample&Hold mintavételezés történik, majd a következő 12 órajel alatt megy végbe a konverzió és egyben az érték küldése az AD_DATA kimeneteken, melyeket fogadnunk kell. Az adatok negatív órajelre változnak, tehát felfutó élnél kell mintavételezni azokat.

A LED-ek kapcsolgatását és az analóg csatorna kiválasztását egy közös shift regiszter végzi. Ennek is kell egy órajel, aminek a felfutó élére mintavételezi az adatot, és kell egy MUX_LOAD jel, aminek felfutó élére átmásolja a bemeneti buffer tartalmát a kimenetre. A MUX_CLOCK és AD_CLOCK valamint a MUX_LOAD és AD_CS jelek közösítve lettek a hardvertervezésnél a jelvezetékek spórolása végett, ezért az adatok betöltése az AD konverzióhoz kötött, annak végén fog aktiválódni shift regiszterünk kimenete. A 8 adatbitet tehát a konverzió utolsó 8 órajelében kell elküldenünk a MUX_DATA lábon.

Az ezek alapján elkészített időzítési diagram két egymást követő mérésre:



18. ábra: Vonalszenzor hardverének időzítési diagramja

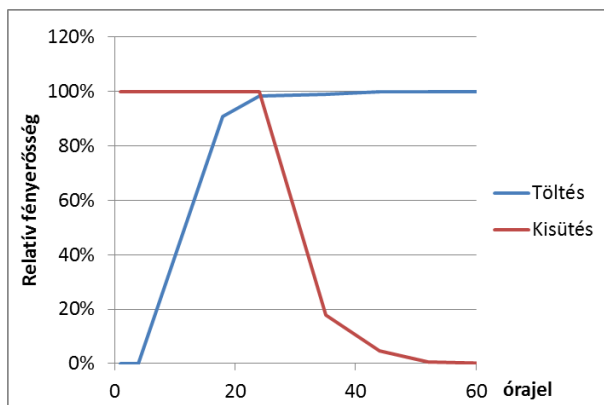
A modulnak kell egy FPGA órajel, egy start jel, a mérendő csatorna sorszáma, és a LED-ek kívánt állapota. Utóbbi két adat, nem a most elindításra kerülő mérésre vonatkozik, hanem majd az azt követőre. A konverzió végeztével a modul generál egy DONE jelet, ekkor van lehetőség a beolvasott értékeket tároló regiszter átadására egy másik modul számára.

Az időzítési diagramot a logikába épített ChipScope-al generáltam, így ez a ténylegesen megvalósított hardver működését mérte és rajzolta ki. Látható, hogy az

AD_DATA csatornákon a jel nem negatív órajelre vált, ahogy annak lennie kellene, hanem 3/4 órajellel csúszásban van. Ez 50 ns jel késleltetést jelent, ami a szalagkábel hossza és az órajelre rakott RC szűrő miatt keletkezik. A feldolgozás szempontjából ez csupán annyi módosítást igényel, hogy az eredetileg pozitív órajelre történő mintavételezés helyett a negatív élre kell mintát a jelekből mintát vennem.

3.3.4 Felfutási idők

Az AD csatorna kiválasztásakor kell egy kis idő, amíg a multiplexer átkapcsolja a kimenetét. Ezen kívül az AD bemeneti lábán egy RC szűrőt is elhelyeztem, ami további késleltetést visz a rendszerbe. Ahhoz, hogy megállapíthassam mennyi idővel a csatornaváltás után lehet elindítani az új mérést, méréseket végeztem. Ehhez két csatorna között váltottam át a multiplexert, melyből az egyik egy sötét dióda fénye volt, a másik pedig maximális fényerőhöz tartozó értéket adott. A csatornaváltás után X órajellel később elindítva a mérést feljegyeztem a mért értéket, amit a 3. táblázat és 4. grafikon tartalmaz.



4. grafikon: Szenzor felfutás

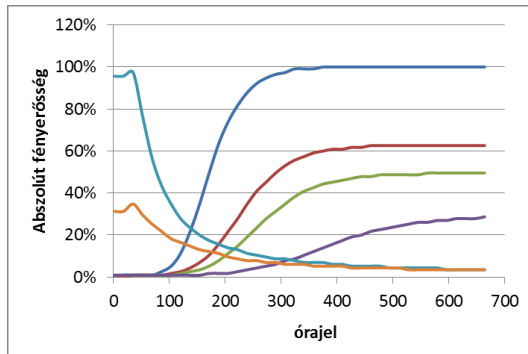
órajel	töltés	kisütés
1	0%	100%
4	0%	100%
18	91%	100%
24	98%	100%
35	99%	18%
44	100%	5%
52	100%	0%
64	100%	0%
69	100%	0%
84	100%	0%

3. táblázat: Szenzor felfutás

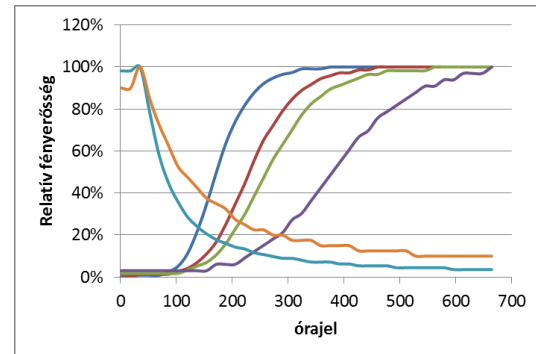
A jel gyorsan felfut, annak lefutása az időigényesebb. Néhány %-os hiba már nem okoz problémát, ezért a következő konvertálást a 0-5%-os maradó értéknél indítom el, 48 órajellel a csatornaváltás után. A mérés időtartamát hosszászámolva kereken 64 órajelenként lehet mérést indítani.

Ezt követően a LED-ek felkapcsolási idejét és CNY-70-es szenzorok reakció idejét kezdtem el vizsgálni együttesen. Az AD csatornáját fixáltam, így nem szükséges annak beállítására várni, elég, ha 1 órajelnyi szünetet hagyok a két konverzió között, szinkronizálási okok miatt. A LED fel- vagy lekapcsolását követően elkezdtem naplózni hogyan változik az analóg jel értéke, majd táblázatba foglaltam azokat.

A mérést több visszaverő felületen is elvégeztem, különböző távolságokból, és azt tapasztaltam, hogy a szükséges idő nagyban összefügg a jel amplitúdójával. Az 5. grafikon az abszolút jeleket mutatja, míg a 6. grafikon a relatív fényerőket.



5. grafikon: LED felfutás (abszolút)



6. grafikon: LED felfutás (relatív)

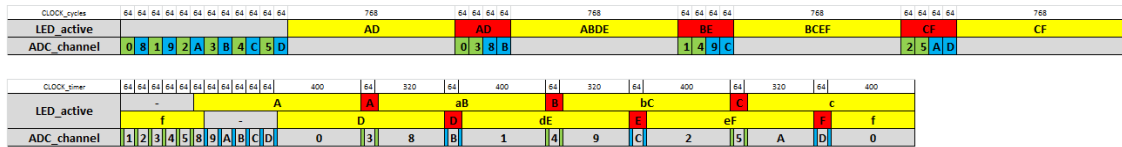
A visszavert fény csökkenésével növekszik a beállási idő. A CNY-70-esek fototranzisztorjai áram kimenetet szolgáltatnak, kisebb beérkező fény esetén kisebb árammal tölti fel a kapacitásokat, hosszabb időt vesz igénybe.

A grafikon legalacsonyabb értékét egy olyan felületen mértem, aminél csak jobbra lehet számítani, ezért a várakozási idő meghatározásánál azt vettem alapul, és 768 órajelre választottam. Ez 48 us-nek felel meg, ami ötszöröse a tavalyi szenzorral mért időnek. Ennek oka, hogy akkor LED-enként külön áramgenerátor hajtotta meg azokat, idén pedig egy tranzisztor közösen kapcsol 8 LED-et, és nem is áramgenerátoros a táplálásuk. Ezek szerint az ULN2003-as IC nem volt jó választás, gyorsabban is el lehetne végezni a kapcsolásokat, de ezen már utólag nem szeretnék változtatni, ha így is teljesíti az elvárásaimat.

3.3.5 Szenzorok bemérési stratégiája

Az 3.2.8-as fejezetben eredetileg tervezett mérési stratégiát újra át kell gondolni a mérési eredmények láttán. A LED-ek állapotának megváltoztatása után relatív sokat kell várni a mérés időtartamához képest, ezért nem biztos, hogy szükség van a bonyolult vezérlésre.

Egyszerűbb lenne, ha az első és hátsó sor LED-jeit egyszerre kapcsolgatnám, és az AD mérések indítása előtt pedig fixen 64 órajelet várnék minden csatornánál. Az így elkészített időzítést a diagram (19. ábra) felső sora szemlélteti. A teljes mérés 4068 órajelből áll, azaz 3,5kHz-enként kapunk információt a vonalakról.



19. ábra: LED-ek és szenzorok megvalósított időzítése

Kíváncsi voltam, és elkészítettem a bonyolultabb verziót is, amit eredetileg terveztem, hogy összehasonlíthassam a két módszert egymással. Itt először az első sor sötét méréseivel kezdek, és utána kapcsolom is fel annak LED-jeit. A két sor egymástól kellő távolságban van, így ez nem zavar be a második sor sötét méréseinek. Ezzel $6 \cdot 64 = 384$ órajel spórolható. Ennek másik előnye, hogy a világos méréseknél egyszerre csak 2 csatorna mérése szükséges, így további $3 \cdot 2 \cdot 64 = 384$ órajelet megspórolok. A világos értékek mérésekor az első AD konverzió egyből indítható, hiszen a csatornát az előző mérés végeztével már átkapcsoltam, így az csak 16 órajelet igényel. Ezzel, kicsivel még tovább lehetett csökkenteni a teljes mérés ciklusát, ami végül 3696 órajel lett. Ennek időzítését a 19. ábra alsó fele mutatja.

A két módszerrel 3,5 kHz vagy 4,3 kHz lehet a maximális frissítési frekvencia, tehát 25%-os gyorsítás érhető el a bonyolultabb időzítést alkalmazva. Mivel a célom a minél gyorsabb szenzor tervezése volt, és a kapcsolást is ehhez terveztem, ezért a hosszú LED felfutási idők ellenére kitartok eredeti tervemnél, és a gyorsabb verziót valósítom meg.

3.3.6 Szenzor bemérő logika, jeltovábbítás

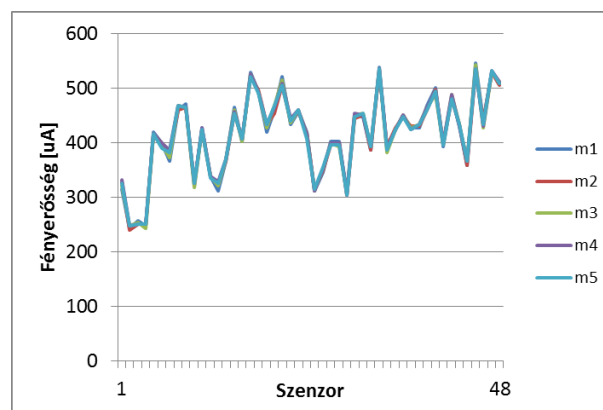
Az időzítések már adóttak, ezek alapján létre kell hozni egy modult, mely a hardver kezelőt paraméterezetten meghívva elvégzi a méréseket. Ezt a legegyszerűbb FPGA-ban úgy megvalósítani, hogy létrehozok egy időzítőt, melynek periódusa állítható, továbbá egy ciklusváltozót, ami számon tartja, melyik fázisban vagyunk. Ezen kívül szükséges egy BlockRAM, amit a ciklusváltozóval címezve megkapjuk a következő fázis periódus idejét, beállítandó LED állapotokat, és a kiválasztandó analóg csatorna sorszámat.

Az így megvalósított modul, a mérések elvégzése után még egy kivonást is képez a világos és sötét mérések között, valamint megkeresi a maximum fényértéket. A beolvasott értékeket a modul egy – az előzőtől különböző – BlockRAM-ban tárolja el, amihez modulon kívülről történő hozzáférést biztosít SPI buszon keresztül.

A SPI kommunikációs interfészt annyival egészítettem még ki, hogy +1 vezetéken generálok egy megszakítást a főprocesszor felé, amikor az analóg jelek mérése befejeződött. Ezt követően történik meg az értékek kiolvasása a buszon.

3.3.7 Szenzor jelek vizsgálata, kalibrálás

A hardver és az azzal való kommunikáció megvalósítása után nekiálltam különböző méréseket végezni a vonalszenzorral, és kitalálni annak további jelfeldolgozását. A szenzort egy fehér lapra helyeztem, attól 4,5 mm-re, ahogy azt korábban előírtam, és beolvastam a 48 szenzor jelét egymás után 5x. Az eredmény a 7. grafikonon látható.



7. grafikon: Nyers szenzoradatok

Az egyes szenzoroknak sokkal nagyobb szórása van, mint vártam, a legérzékenyebb és legérzékletlenebb szenzor között 2x-es áramerősség különbség van. Az így kapott jelből nagyon nehéz lenne pontosan meghatározni a vonal pozícióját, ezért mindenképpen szükség van szenzoronkénti kalibrációra. Ami viszont biztató, hogy az 5 mérés során (m1...m5) mindig ugyan azt mértem, tehát nem zajról, hanem különböző szenzor érzékenységek okozzák az egyeletlen jelalakot.

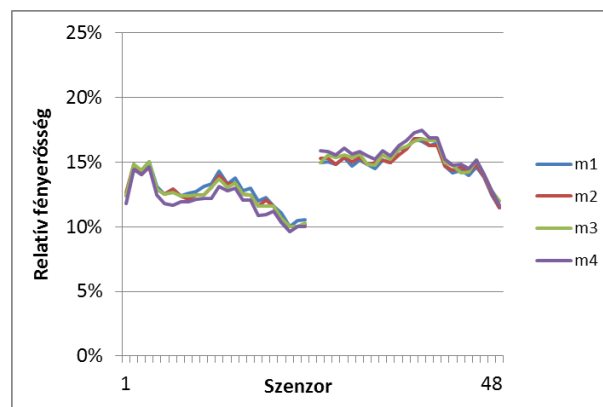
Az 5 mérés értékeit kiátlagoltam, és ezt vettem kalibrációs referenciának. A jelfeldolgozás során a mért értékeket elosztom ezzel a kalibrációs paraméterrel, így megkapom százalékosan minden szenzor fényerejét.

Elvégeztem még egy olyan mérést is, ahol a maximális értéket kerestem. Ehhez különböző felületű tárgyakat helyeztem alá, és eltérő távolságokból vizsgáltam. Nem meglepő módon egy tükrös, fémes felületről verődött vissza a legtöbb fény, ilyenkor a legérzékenyebb szenzor áramerőssége 1,9 mA volt. Ez az AD referenciájához képest 25%-os érték, azaz biztosan nem fog túlcsoordulni 3x akkora, külső forrásból beszűrődő

fény esetén sem, tehát helyesen választottam meg a mérőellenállásokat. Külső fény nélkül, normál felület esetén, a legjobb szenzorok kimeneti jele a referenciának az 1/16-od részét képezik. Az AD átalakító 11 bites (+előjel bit), ebből a tartalékok miatt 4-et elvesztve is marad még 7, mely 128 részre, gyengébb szenzorok esetén 64 részre osztja fel a tartományt. A szenzor 7 mm széles, szóval még így is tized milliméteres tartományban maradhat a felbontása. A pontosság szempontjából célszerű lenne kisebb referenciát választani, de ezzel csökkenne a szenzor megbízhatósága, hiszen ha a külső forrásból származó fény (pl. bevilágító Nap) éri a szenzort, akkor telítésbe mehet, és nem fogja látni a vonalat. Ha az így megválasztott ellenállásokkal és referenciával is kellően pontos marad a szenzor, nem korlátozom fölöslegesen a működési tartományát.

Ismervén a maximális fényerőt és a szenzorok érzékenységét, a dolgozatomban további részében, a kalibrálást elvégezve százalékosan fogom feltüntetni a fényerősségeket.

A kalibrálás helyességét ellenőrizvén végeztem egy újabb mérést (8. grafikon) a szenzorral egy sötétebb árnyalatú felületen, és megvizsgáltam ott is közel egyenletes fényt kapok-e, vagy a szenzor érzékenysége nem lineáris, és többlépcsős kalibrálásra lenne szükség.



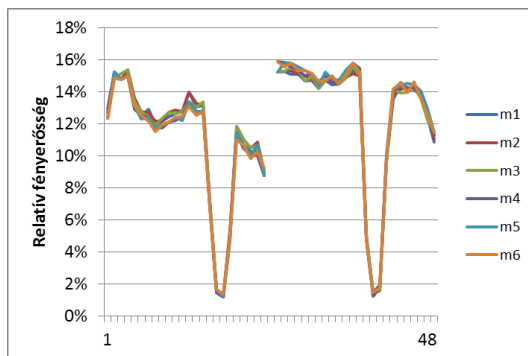
8. grafikon: Kalibrált szenzoradatokat (sötét felület)

A grafikonon a szakadás előtti rész az első-, az azutáni pedig a hátsó szenzorsor adatai. Mindkét vonal jobbra lejt, azaz valószínűleg ferdén helyeztem el a szenzort a felületen (két végén a távolság a felülettől nem volt azonos). Sajnos ettől a ferdeségtől eltekintve, az egymáshoz közel eső szenzorokat vizsgálva sem kaptam egyenletes kalibrált fényerősséget, de az eddig maximálisan 100%-os eltérés helyett 25-30%-ra csökkentettem a hibát.

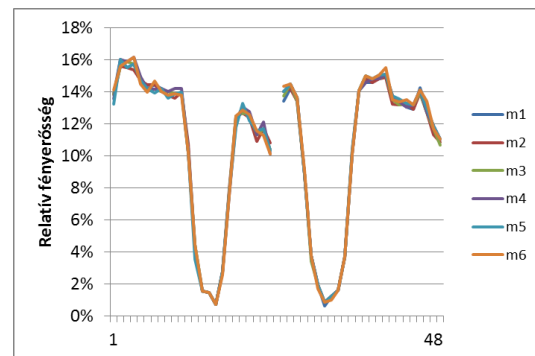
Összevetve a korábbi, bekalibráló mérésekkel, (különösen az első 5 szenzort vizsgálva) megfigyelhető, hogy az érzéketlenebb szenzorok így kisebb fénynél túlkalibráltak lettek, és pont azok mutatnak most kimagasló értéket. Ez azt jelenti, hogy a karakterisztikájuk nemlineáris, vizsgálni kellene az érzékenységet több fényerősség mellett. Ez egyrészt sok beméréssel jár, másrészt bonyolult táblázatkezelést és számításokat követel a mikrokontroller részéről. Ráadásul az is lehet, hogy nem csak a szenzorok, de a LED-ek is eltérőek, és azok karakterisztikáját is számításba kellene venni. Ezért a kalibrációt egyelőre elfogadom jelenlegi állapotában, és majd ha a vonal közepének meghatározása nem lesz kielégítő pontosságú, visszatérek erre a pontra.

3.3.8 Vonallátkép, kalibrálás ellenőrzése

A kalibrálás után felragasztottam egy vonalat a sötét felületre, és megnéztem mit lát belőle a szenzor. A teszteket szándékosan végeztem sötét felületen, hiszen ha ezen kielégítően jó eredményt kapok, nem kell attól tartanom, hogy a versenypályán, a nemideális körülmények mellett bizonytalan működést okozzon. Az első mérésnél (9. grafikon) merőleges volt a szenzor a vonalra, míg a második esetben (10. grafikon) kicsit szögbe állítottam.



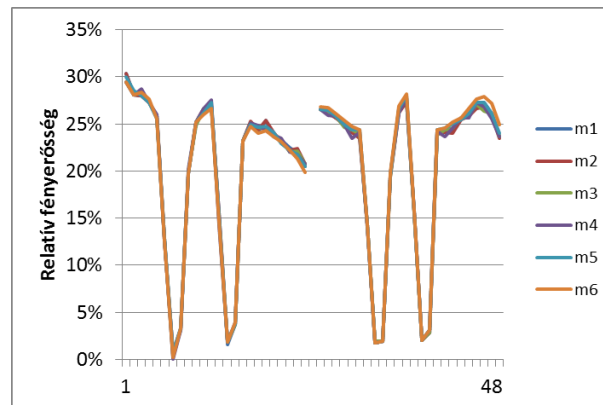
9. grafikon: Szenzorjel merőleges vonal esetén



10. grafikon: Szenzorjel ferde vonal esetén

A vonalak egyértelműen megtalálhatóak, jelentős fényerő csökkenést okoznak. Szépen látszik az is, hogy szögben nézve a vonalat szélesebbnek látszik. Pozitív eredmény, hogy a fehér részek mindkét esetben ugyan olyan alakúak, mint a kalibrálás utáni egyenletes felületen (8. grafikon). Ez azt jelenti, hogy adott felületre történő kalibrálással nagyon jó eredményeket lehetne elérni, de még így is elfogadhatóan jók, hogy a kalibrációt más körülmények között végeztem. A jobbra lejtése továbbra is megvan, tehát nem a kalibrálás után, hanem a kalibráló mérés elvégzése alatt lehetett ferde a panel.

Végeztem egy harmadik mérést is (11. grafikon), ahol két, egymással nem párhuzamos vonalat ragasztottam fel világos felületre. A két vonal távolságának különbözősége, a két sor méréseit összehasonlítva, a grafikonon is megfigyelhető.



11. grafikon: Szenzorjel két merőleges vonal esetén

A felület miatt a fényerősség majdnem duplájára nőtt, ami így közelebb áll a fehér lapos kalibrációhoz, és kevésbé zajos a jel. A ferdeség az első szenzor esetében továbbra is megvan, most már egészen biztosan a kalibrálásnál volt rossz a beállítás. A második sor nagyjából vízszintesnek mondható, amiből arra következtetek, hogy nem simán ferdeség okozta a hibát, hanem talán hullámos lehetett a kalibráló papírlap.

Mindenképpen szükséges lesz újrakalibrálni a szenzorokat a további mérések elvégzése előtt. Ehhez előbb fel fogom szerelni mechanikusan a szenzort a járműre, mert egyrészt annak kerékfelfüggesztései nem egyformák, másrészt az akkumulátorok miatt, az egyik oldalon nagyobb a rá nehezedő súly és emiatt lejjebb süllyed. A jármű alvázának síkja tehát nem párhuzamos a talajjal, és az ebből adódó hibát is magába kellene foglalnia a kalibrációnak. Továbbá ezt a végső kalibráló mérést a verseny helyszínén kellene elvégezni, hogy a visszavert fényerősség is olyan legyen, amire majd a versenyen számítani lehet.

3.3.9 Vonalfelismerés

Rendelkezésemre áll tehát egy kalibrált visszavert fényerősség minden egyes szenzorról. Ebben kell megtalálnom a vonalat. Az első verzióban ezt úgy valósítottam meg, hogy kiválasztottam a legvilágosabb értéket, ennek vettem a felét, és az így meghatározott fényerősséghez komparáltam az egyes szenzorokat. Amelyik ennél nagyobb volt, azt fehérnek, ami alatta, azt feketének (vonálnak) vettem. Ez elég megbízhatatlanul működött, ezért megpróbáltam javítani rajta. Első lépésként a

komparálási határt kezdtem változtatni, hogy ne a fele legyen. Kipróbáltam több változatot is 25-75% között, de az egyébként is erős kontraszt miatt ez nem változtatott sokat. A mérések közben észrevettem, hogy néha előfordul egy-egy kiugró szenzor, amelyik jelentősen nagyobb jelet ad, mint a többi világos jelű. Ebben az esetben a komparálási határt ez a hibás szenzor szabja meg, ezért az egyszerű maximumkereső függvény helyett más kellett kitalálnom. A későbbiekben csökkenő sorba rendeztem a szenzorok értékeit, és kiválasztottam mindig a 4. legnagyobb értéket, és abból határoztam meg a referenciát. Így már akár 3 kiugró érték is belefért, nem jelentet problémát.

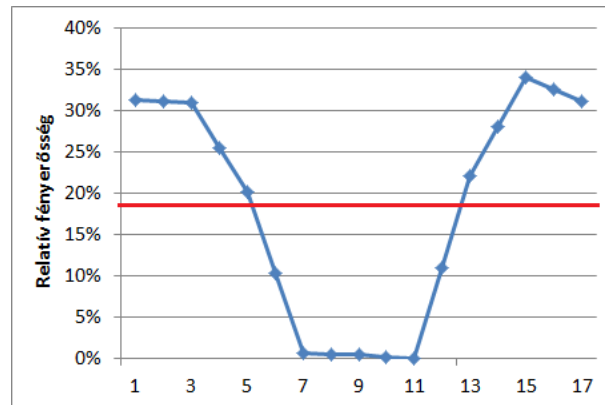
A komparálás elvégzése után van egy bináris adatsorom, amiben meg kell találni a vonalat. Itt egyszerűen legalább 3 feketét kerestem egymás mellett, mert ez a minimális szélessége a szigetelőszalagnak. Az ennél keskenyebb vonalakat figyelmen kívül hagytam. Később ki kellett egészíteni azzal, hogy ha a fekete adatsorba bekerül 1db fehér, azt még tekintse feketének, mert többnyire a komparálási határ környékén volt az értéke.

3.3.10 Vonal középpontjának meghatározása

A vonal középpontjának meghatározására két módszert vizsgáltam. Az egyik megoldás az volt, hogy a fekete szenzorok közül a középsőket teljes feketének veszem, a két szélsőnél, ami átmenet a sötét és világos között, ott pedig a telítettsége alapján meghatározom, hogy az alatta lévő terület hány százaléka vonal. Ebből kiszámítható a szigetelőszalag két oldalának koordinátája, és a kettő átlaga a közepe. Sajnos ez a modell nagyon érzékeny volt arra, hogy a szenzor pont 4,5 mm távolságra legyen a talajtól, ha egy kicsit is közelebb vagy távolabb került, nagyon pontatlanná vált. Ezt ugyan lehetne mechanikailag biztosítani, de nem egy rugalmas megoldás.

Ezt követően egy új modell kidolgozásába fogtam. A vonal szélességére valójában nem volt szükségem, csak az fontos hol van a közepe. Ezt egy súlyozott átlaggal is meg lehet határozni, ahol a súlyok a fényerősség, az értékek pedig az egyes szenzorok koordinátái. Legjobban a fekete szenzorokat kell súlyozni, ezért a fényerősségeket meg kellett fordítani. Ehhez kiválasztottam a legmagasabb értéket, és ebből vontam ki az összes többi szenzor értékét. A súlyozáshoz ki kellett választani egy tartományt, amire ezt alkalmazom. A tartománynak tartalmaznia kell a fekete szenzorokat, és azok néhány szomszédját. A közvetlen szomszédokra mindenképp

szükség van, mert ezek képezik az átmenetet. Előfordulhat, hogy az érték a komparálási határ közelében van, és ilyenkor az egyel távolabbi szenzorra is szükség van. Ugyanakkor túl távoli szenzorokat vizsgálva előáll az a probléma, hogy egymás mellett lévő két vonal esetén már beleszámítom a másik vonal kezdetét is. Ezért jobbnak találtam finomítani a felbontást, minden két szenzor közé elhelyeztem egy virtuálisat, aminek fényerejét interpolációval határoztam meg két szomszédjából. Így már sokkal könnyebben el lehetett dönteni mely szenzorokat kell felhasználni a súlyozott átlaghoz.



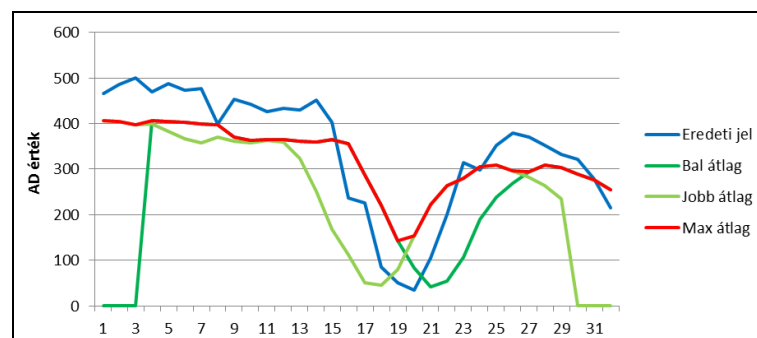
12. grafikon: Szenzorjel kiválasztott tartománya súlyozott átlaghoz

A 12. grafikon a vonal környéki szenzorok értékeit, és az interpolációval elhelyezett köztes értékeit is tartalmazza. A vízszintes vonal a meghatározott komparálási határt jelöli. A grafikont leolvasva a súlyozáshoz a sötéteket, és azon kívül 2-2 szomszédos szenzort célszerű figyelembe venni. Az azt követő pont már egy kiugró érték. A súlyok invertálásához, a vizsgált intervallum maximális fényerejéből vontam ki a többi értéket, de ez azt eredményezte, hogy a legvilágosabb szenzor nullás súlyt kapott, tehát kiesett. A finomított felbontásnak köszönhetően gond nélkül kibővíthettem a tartományt további 1-1 szenzorral, még az is kellő távolságra lesz azoktól a szenzoroktól, amik már feltételesen mérhetik a vezérvonal két oldalán elhelyezett segédvonalakat. Ebben a kibővített intervallumban, szinte mindig valamelyik szélsőszenzor határozza meg a maximumot, ami a nullás súlyozás miatt kiesik. Éppen ezért a túloldali párját is kiejtem, hogy ne „húzza” el a középpontot abba az irányba. Tehát a kibővítésre csak a maximum meghatározásának erejéig van szükség, a súlyozásnál továbbra is marad az eredeti 2-2 szomszédot figyelembe vevő tartomány.

Ez a számítás álló helyzetben való méréseknél már nagyon jó, pontos eredményeket adott, de a rázkódó autón, kanyarokban mégis sokat hibázott. A menet közbeni mérések annyiban különböznek az asztalitól, hogy a szenzor és a föld távolsága

változó. Kanyarban megbillen az autó, így az egyik oldali szenzorok világosabb, másik oldal sötétebb értéket mutat. A rázkódások miatt pedig időnként közel kerül a földhöz, és mivel a kalibrálás csak egy fix távolságra lett meghatározva, ezért ilyen esetekben a gyengébb szenzorok egy kiugróan alacsony vagy magas jelet adnak. A komparálásnál ezek mind komoly problémát jelentenek, ezért az egyszerű referenciaképzés helyett, kifejlesztettem egy megbízhatóbbat, ami lokális referenciát képez.

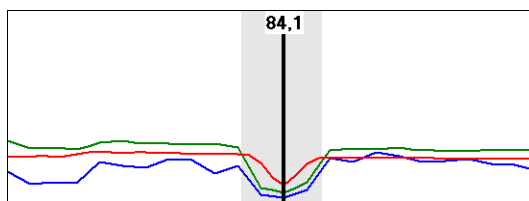
Minden ponthoz külön komparálási szintet határoztam meg, a következőképpen. Minden pontban kiszámoltam a kiválasztott pont szomszédjainak speciális átlagát (a pont önmaga nem számít bele saját értékébe). A komparálási határ megválasztásakor a fekete vonal melletti világos jelek értékére van szükség, nem beleszámítva a feketét. A komparálás elvégzése előtt nem ismerem, hogy mi számít sötétnek, és mi világosnak, de azt tudom, hogy ha a pont éppen egy határátmeneti pont, akkor az egyik oldalán világos jelek lesznek, másik oldalán pedig sötétek. (Ha egy átlagos pontról van szó, ami egyértelmű fehér, vagy fekete, ott nem igazán számít mekkora komparálási határt választunk, könnyen eldönthető lesz, ezért ezt az esetet nem szükséges vizsgálni). Szóval a jobb és baloldali szenzorok átlagát külön kell kezelni. Számoltam egy súlyozott átlagot úgy, hogy a ponttól 1-el és 2-vel balrább lévő szenzort teljessel, a 3-ra lévő pedig 50%-os súllyal vettem figyelembe. Osztáskor viszont 2,5 helyett 3-al osztottam el az összeget, így biztosítva azt, hogy ez a speciális érték minden esetben alacsonyabb legyen, mint a tényleges átlaguk lenne. Hasonlóképp számoltam egy jobboldali átlagot is. A kettő közül a magasabb lesz a kiválasztott komparálási érték, ezért egy maximumképzéssel határoztam meg a ponthoz tartozó lokális komparálási határt. A számított értékek a 13. grafikonon láthatóak, egy zajosabb, ferde szenzorhoz tartozó mérési eredményen alkalmazva.



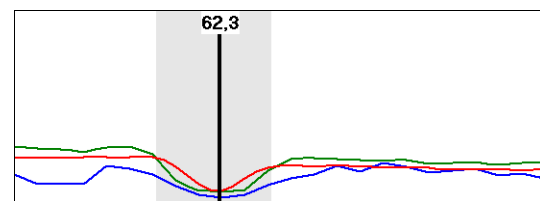
13. grafikon: Lokális komparálási határ

3.3.11 Vonalszenzor felbontása

Az ismertetett módszerrel már mindenféle körülmény és felület mellett megbízható, pontos, nagy felbontású értéket mutatott a szenzor a vonal középpontját illetően. Írtam egy saját PC szoftvert, ami valós időben kirajzolja a szenzorok jeleit (kék), a kalibrált értékeket (zöld), a lokális komparálási határt (piros), és azt, hogy a pont éppen világosnak, vagy sötétnek számít-e (szürke háttér jelenti a sötétet). A programból kivágott képernyőket a 20. ábra és 21. ábra tartalmazzák. Az felsősorok az első szenzorsor értékeit mutatják, az alsók a hátsóét.

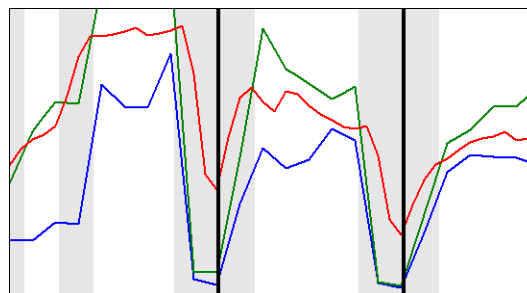


20. ábra: Szenzorkép jobbra ferde vonal



21. ábra: Szenzorkép balra ferde vonal

A képekről is látszik, hogy a középvonalat nagyon szépen berajzolja középre, és hogy a ferdebb vonal (jobboldali ábra) szélesebbnek látszik. Bekalibrált felületen, (ezek a mérések ilyenek), csupán $\pm 0,1$ tized millimétert ugrál a középpérték a mintavételezések során. A vonal minimális elmozdítása esetén – amennyire kicsit kézzel mozdítani tudok rajta – egyből meg is változik a középponti koordináta. Ezzel a mérési és jelfeldolgozási algoritmusommal tehát bőven túlteljesítettem az eredetileg kitűzött célokat, ami 1 mm-es felbontást követelt meg a 2° -os vonalferdeség észleléséhez.



22. ábra: Szenzorkép zajos jel esetén

Ami pedig a megbízhatóságát illeti, leteszteltem extrém körülmények között is a szenzort. Aláhelyeztem egy fehér, hullámos papírlapot, sokkal közelebb a szenzorhoz, mint annak lennie kellene. Emiatt a kalibrálás teljesen elmászott (22. ábra) és egyeletlenné vált, mert nagyjából 3x-os volt így a fényerősség, de meglepő módon még így is megtalálta az algoritmus a két vonal közepét, nem mért be téves értéket. A felbontásáról ilyen körülmények között nincs pontos információ, de az 1 mm-t így is biztosan tudja.

A megvalósított vonalszenzorom tehát normál körülmények között 0,5 mm-es felbontással biztosan képes meghatározni a vonal közepét, 1°-os pontossággal annak ferdeségét, és másodpercenként több mint 4000 mintavételezésre képes.

3.4 Vonalak analízise

3.4.1 Követendő vonal kiválasztása

Előfordulhat, hogy a vonalszenzor több vonalat is észlel. Lehet ez akár téves mérésből, például csempék közti fuga, vagy a vezérvonal mentén, annak két oldalán elhelyezett segédvonalak. Ilyenkor a szenzornak tudnia kell, melyik vonalat kövesse a sok közül. Éppen ezért eltárolom, hogy hol látta a vezérvonalat korábban, és mikor az új szenzor jelét dolgozza fel, megvizsgálja talál-e olyan vonalat, ami ehhez képest maximum 15 mm-t mozdult el valamelyik irányba. Ha igen, akkor az a vezérvonal.

Ezen kívül szükség van még egy irányító regiszterre, hogy ha a vonalat elveszítettük, vagy több vonal van, és szeretnénk váltani, akkor ezt megtehessük.

Többféle utasítást programoztam le:

- a) mindenképpen a megadott szabály szerintit követi:
 - baloldalt
 - jobboldalt
 - középsőt (páros szám esetén bal középsőt)
 - középsőt (páros szám esetén jobb középsőt)
- b) előző vonal követése, elvesztése esetén másik vonal kiválasztása a) pont szerint
- c) kizárólag az előző vonal követése, elvesztése esetén hibajelzés

3.4.2 Vezérvonal szöghelyzetének felismerése

A hátsó szenzorsor nem vesz részt közvetlenül a vonalak felismerésében, ennek funkciója kizárólag a vezérvonal szöghelyzetének meghatározása. Ha korábban nem volt követett vonal, akkor megkeresem azt a vonalat, amelyik a legközelebb van a vezérvonalhoz, és úgy tekintem, hogy ez a párja. A két mért középpontból, ismerve a szenzorsorok távolságát, egy egyszerű arkusz tangenssel meghatározható az autó szimmetriatengelyével bezárt szög. Ilyen esetben csak 30 foknál kisebb ferdeségű vonalat fogadok el. Ha korábbi mérések alapján ismert volt a vonal szöghelyzete, akkor viszont korlátozás nélkül, mindent elfogad, amennyiben az hasonló szögben áll, mint korábban. Így akár 75 fokos szög is mérhető, ha a vonal megfelelően helyezkedik el, és nem csúszik le a szenzorsorokról.

4 Helymeghatározás

4.1 Szenzorok

A jármű térbeli elhelyezkedésének felderítése két szenzor alapján fog történni: egy kiválasztott kerékre (vagy annak meghajtómotorjára) szerelt enkóderből, és egy inercia szenzorból.

4.1.1 Enkóder

Az autó elmozdulásának meghatározására jelenlegi autónkon enkódert használunk. Az irányítandó autónk egy 4 kerék-hajtású jármű, melynek van egy motor által hajtott főtengelye, annak két végén 1-1 differenciálmű, amik továbbítják a hajtást a kerek felé.

Az alkalmazott nagy felbontású inkrementális jeladót a differenciálművek közötti főtengelyen helyeztük el, így lényegében egyfajta kiátlagolt értéket kapunk az egyes kerek elfordulásáról, azaz a jármű elmozdulásáról. Felbontása 0,45 mm / inkrement. Az autó haladási sebességét is ebből az értékből határozzuk meg időmérés segítségével.

4.1.2 Inercia szenzor

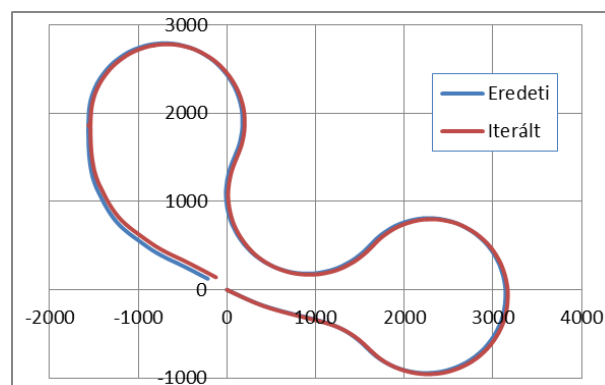
Inercia szenzorként egy F3-Discovery kártyát használunk a jelenlegi autónkon. Kiválasztásának oka az volt, hogy ha időnk engedi, bonyolultabb szűrőalgoritmusokat alkalmazzunk majd a giroszkóp (és esetlegesen gyorsulásmérő, iránytű) jeleinek feldolgozására. Erre végül nem került sor, és csak a giroszkóp egyik tengelyének adatait használtuk fel a tájékozódáshoz.

Legpontosabbnak azt a megoldást találtuk, és végül ezt is alkalmaztuk, hogy az autó nyugalmi helyzetében folyamatos kalibrációt végzünk, és azzal korrigáljuk a giroszkóp adatait. A kalibrációt olyankor végzi a rendszer, amikor a gyorsulásérzékelők és giroszkóp adatok alapján régóta nyugalmi helyzetet érzékel. Ekkor az utolsó 511 mérés átlagát offset-nek veszi, amit a szöghelyzet integrálásakor figyelembe vesz.

4.2 Mérések

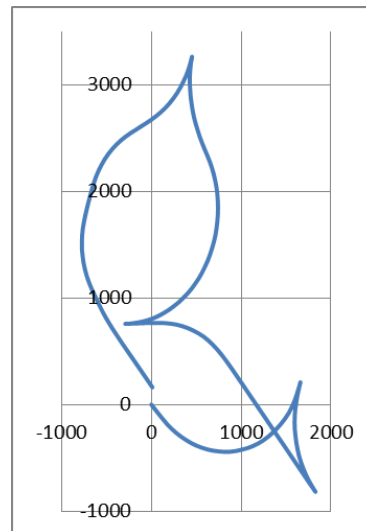
4.2.1 Próbamérés

A szenzorok kipróbálásaként először a rendelkezésre álló rendszerrel végeztem néhány mérést, hogy felmérjem mégis mennyire lesz használható helymeghatározásra. A rendelkezésre álló RAM és meglévő programstruktúra csak korlátozott mennyiségű adatok tárolását tette lehetővé, ezért egy kisméretű pályán, ritka mintavételezéssel (64 mm-es lépésközzel) végeztem a mérést. Az így kapott adatokból szépen kirajzolódott a bejárt útvonal, amit a 23. ábra szemléltet, de látszik, hogy nem tökéletes az eredmény, ugyanis nem egyezik meg a kiindulási és a végponti koordináta, pedig az autót ugyan oda és ugyan olyan irányban irányítottam vissza ahonnan elindítottam. A hiba nagysága 25 cm, melyet a lépésköz utólagos finomításával (adathalmazon iterációt végezve) 19 cm-re sikerült lecsökkenteni. Az iteráció során két mérési pont közé további 15 pontot helyeztem el, melynek tovább finomítása nem vezetett pontosabb eredményre.



23. ábra: Próbamérés interpolációval

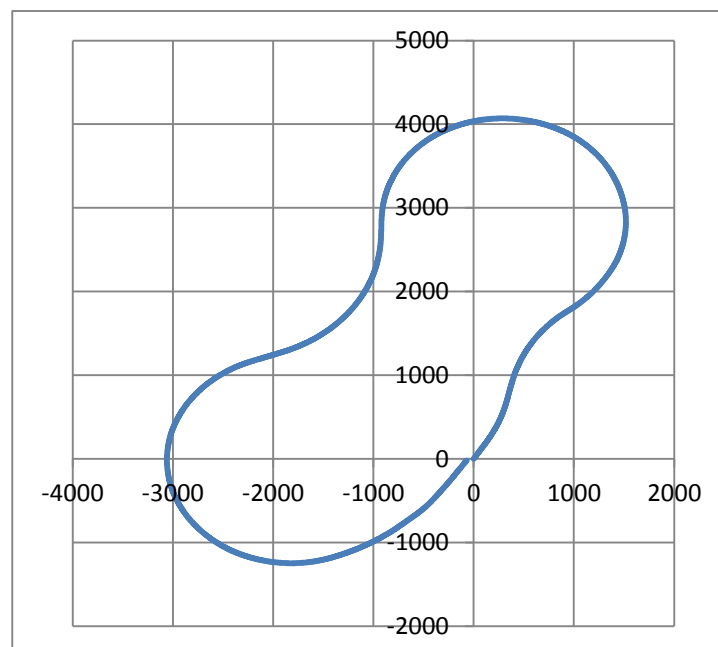
Kíváncsi voltam mennyire változik a hiba nagysága, ha az autóval nem csak előre, hanem hátrafelé is haladok. A tolatásokkal kiegészített pályarajzot a 24. ábra mutatja. A hiba nagysága 21 cm-re adódott, ami nagyjából megegyezik a tolatás nélküli hibával (sőt picit jobb is), így a továbbiakban nem szükséges külön vizsgálni a hátramenetet.



24. ábra: Próbamérés tolatással

4.2.2 Nagy felbontású mérés

Lássuk be, hogy egy ekkora autónál a 64 mm-es lépésköz elég durva közelítő mérésnek számít, ezért átalakítottam a programot, hogy több adat eltárolására is képes legyen, majd megismételtem a mérést 4 mm-es felbontással (25. ábra). Az eredmény nagyon biztató, 15 méteres út megtétele után mindössze 10 cm eltérés adódott, ami fele akkora, mint az interpolációval javított első körös eredmény.



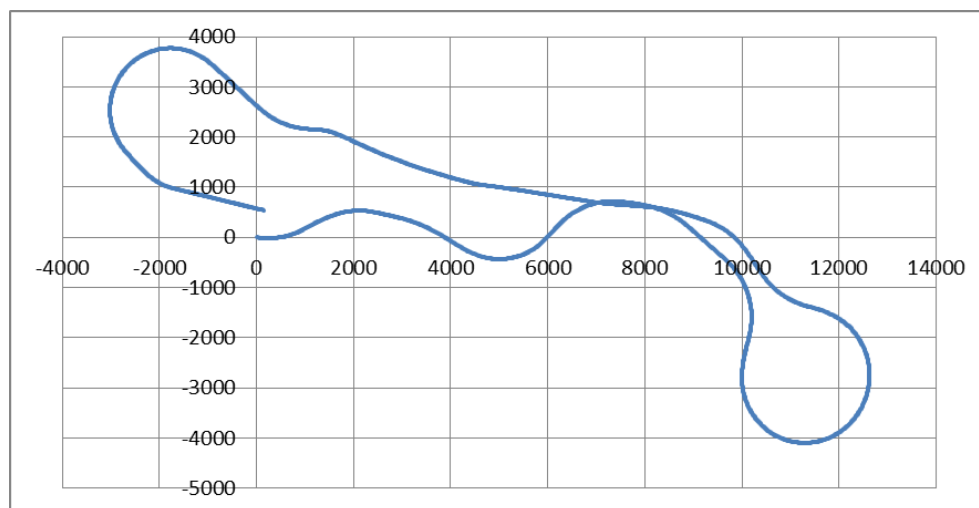
25. ábra: Nagyfelbontású mérés

A 4 mm-es lépésközt már kicsit túlzásnak éreztem, ennél ugyanis a rendszer mechanikai kotyogása is nagyobb, ezért újra megismételtem a mérést egy köztes, 16

mm-re megválasztott lépésközzel. Az eredmény csupán 5%-al okozott nagyobb hibát a 4 mm-eshez képest, így a továbbiakban 8/16 mm-es lépésközzel fogom végezni a méréseket a rendelkezésre álló memória és mérendő úthossztól függően

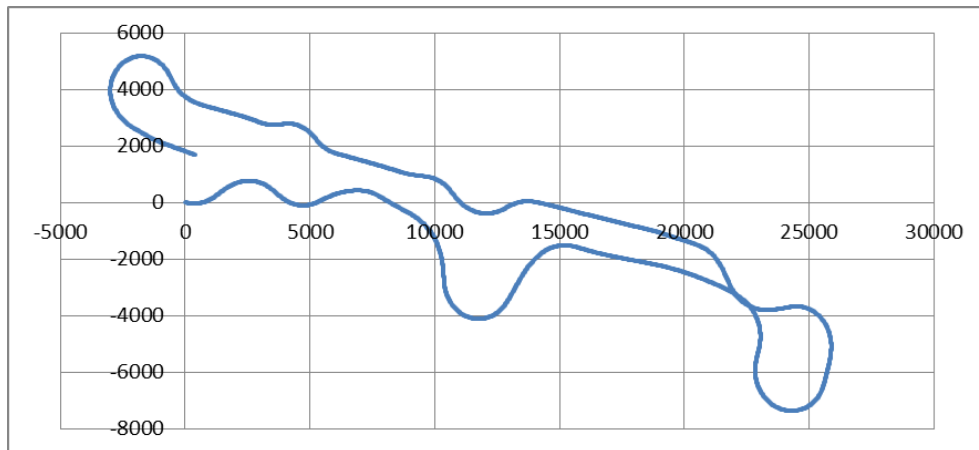
4.2.3 Hosszú távú mérés

Azt már láttuk, hogy kisebb nagyobb hibával kirajzolja egy minimális 15 méteres pálya térképét, de a versenyen ennél sokkal nagyobb távokat kell megtenni, ezért végeztem néhány hosszabb mérést is az autóval (26. ábra), hogy kiderüljön mennyivel nő az abszolút hiba. Ilyen hosszon már komoly hiba adódott, a kezdeti és végkoordináta között 56 cm eltérés mutatkozott. A kirajzolt pálya érintőit tekintve is már $5,8^\circ$ -os eltérés van a kezdeti és végpont érintője között, ami kezdett gyanússá válni, hogy nem csupán a szemmértékkel történő párhuzamos parkolás hibája, hanem feltételezhetően a giroszkóp hibája is kezd jelentőssé válni.



26. ábra: Hosszú távú mérés 40m

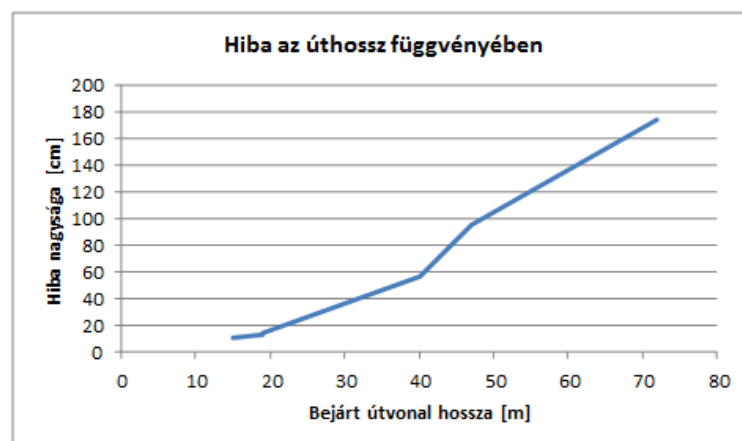
Hogy megbizonyosodjak arról, hogy a hiba lineáris összefüggésben van a megtett út hosszával, végeztem egy 72 m hosszú mérést is (27. ábra). A hiba 173 cm-re nőtt. Ekkora tévedéssel már egészen biztosan nekimenne valaminek. A versenypálya hossza 50-60 m körül van, és szélességét tekintve 30 cm hibánál nagyobb eltérés nem engedhető meg. Ezen felül az érintők eltérése is $8,9^\circ$ -ra nőtt, tehát biztosan a giroszkóp pontatlanságából származik.



27. ábra: Hosszú távú mérés 72m

4.2.4 Kiértékelés

Eddigi méréseim abszolút hibáit a mérés során bejárt útvonal hosszának függvényében ábrázolva a 14. grafikont kapjuk. Az eredményekből nagyon jól kivehető a várt lineáris karakterisztika.



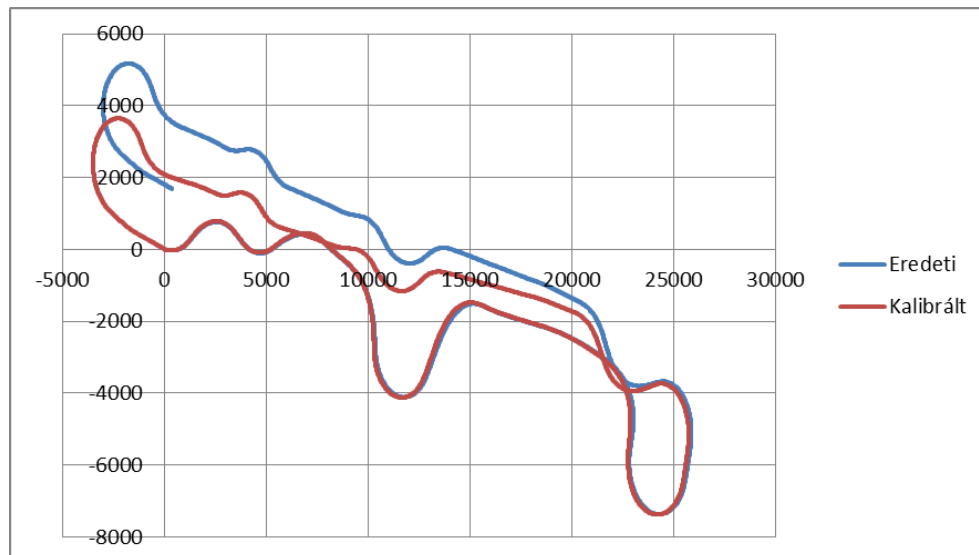
14. grafikon: Hiba az úthossz függvényében

A hosszú távú méréseknél megfigyelhető, hogy a hiba nagy része az autó szimmetriatengelyére merőleges. Ebből következik, hogy az enkódernek nincs jelentős változó hibája, ugyanakkora utat mér odafelé, mint visszafelé. A hiba a giroszkóp jel kétszeres integrálásából adódik.

4.2.5 Giroszkóp adatok utólagos korrekciója

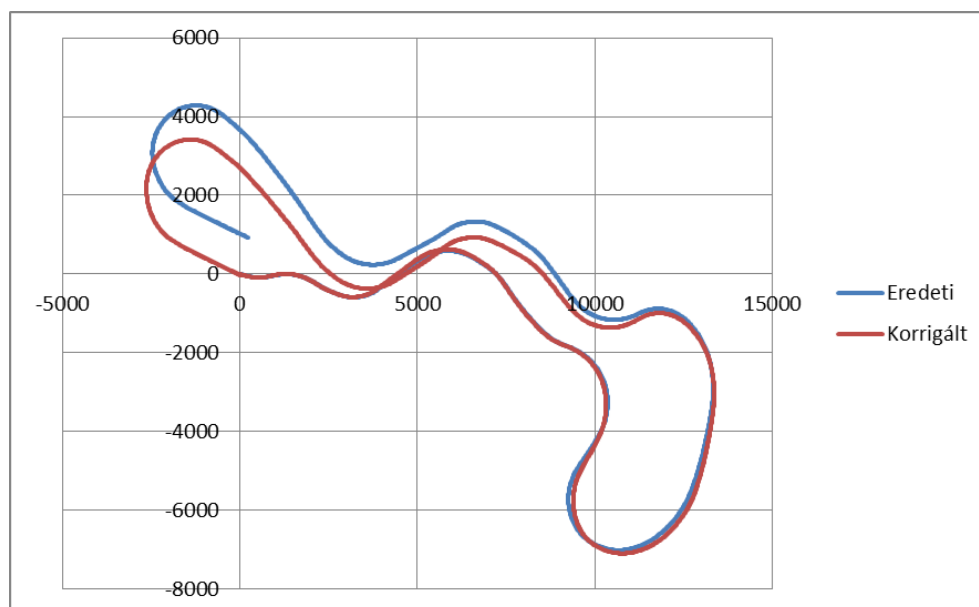
Felmerült egy gondolat, hogy mi van akkor, ha a vízszinteshez képest picit szögben áll a giroszkópos panel, vagy bármilyen okból kifolyólag valami statikus hibája van. Ennek vizsgálatára egy korrekciós tényezővel egészítettem ki az excel

táblázatot, melynek értékét addig állítottam, amíg egy zárt pályarajzot nem kaptam (28. ábra). Ezzel sikerül 13 cm-re csökkentenem a hibát, ami egész elfogadható lenne, ha ez a korrekciós együttható a többi mérési eredményt is kijavítaná.



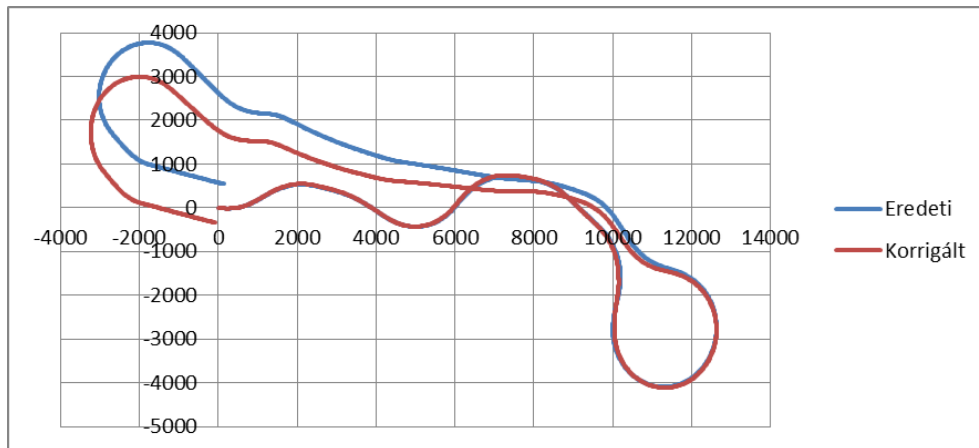
28. ábra: Kalibrálás a 72m-es mérés alapján

Ráhúzva ezt a korrigálást a közvetlenül előtte mért 47 méteres mérésre egész bízható eredményeket kaptam, zárt térképet eredményezett (29. ábra).

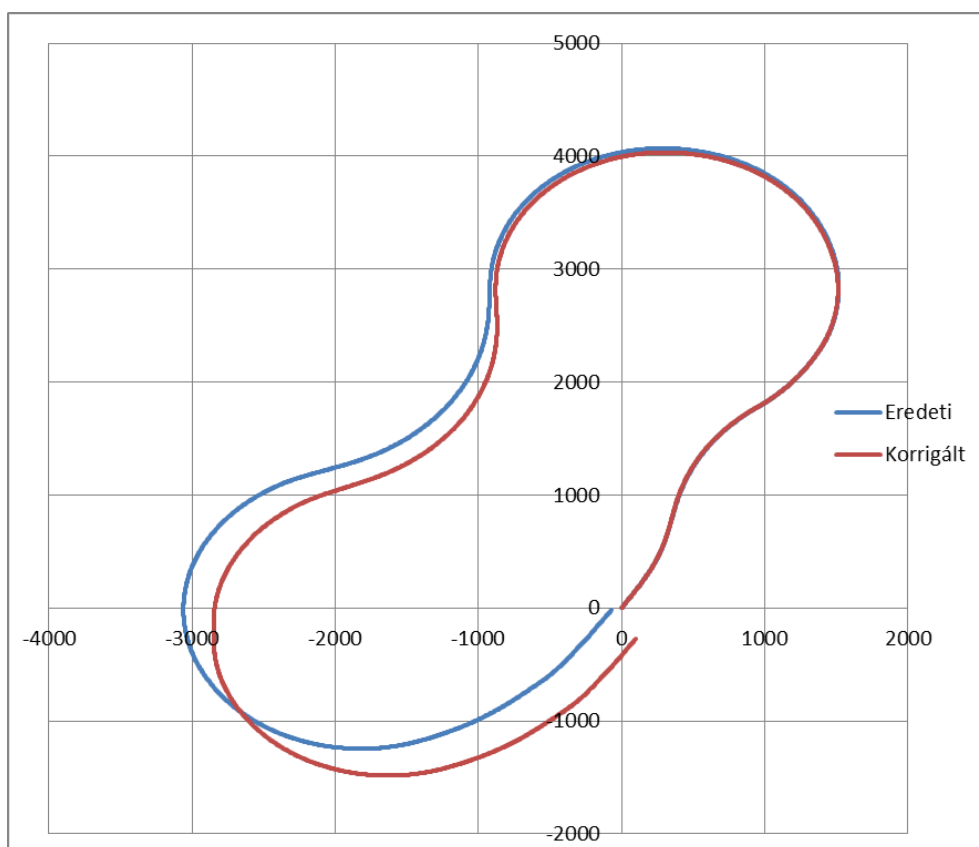


29. ábra: Kalibrált 47 méteres pálya

További 3 korábbi mérésen is elvégezve azonban csökken a hatékonysága, sőt bizonyos esetekben elrontja az eredményt. Ezeket a következő 30. ábra és 31. ábra szemlélteti.



30. ábra: Kalibrált 40 méteres pálya



31. ábra: Kalibrált nagyfelbontású mérés

A korrekció minden mérésen a helyes irányba módosította az adatokat, de a rövidebb méréseknél erős túllövással tette azt, ezzel komoly hibákat generálva. Az utólagos korrigálás tehát nem működik, okát máshol kell keresni.

4.2.6 Hibalehetőségek

Több mérést is elvégezve próbáltam megkeresni a tévedés okát, de nem sikerült megfejtennem mi okozhatja azt, hogy az egyik mérés után a giroszkóp $<1^\circ$ -os hibával

ugyan azt az értéket mutatja a kiindulási helyre való visszaparkolás után, mint induláskor, másik esetben pedig akár 5-7°-os eltérések is adódtak, hol az egyik, hol a másik irányba.

Próbáltam összefüggéseket keresni, hogy sebesség függő-e (nem), rázkódások hogyan befolyásolják (egyhelyben rázva az autót kis mértékben eltérült, de 1°-on belül maradt), nagy motoráramok, kefeszikrázások okozzák-e a hibát (ez volt a legszámottevőbb, de finoman, közel állandó sebességgel haladva a járművel sem szűnt meg a hiba), de ezek egyike sem segített előrébb.

4.3 Pontosítás

4.3.1 Jobb szenzor

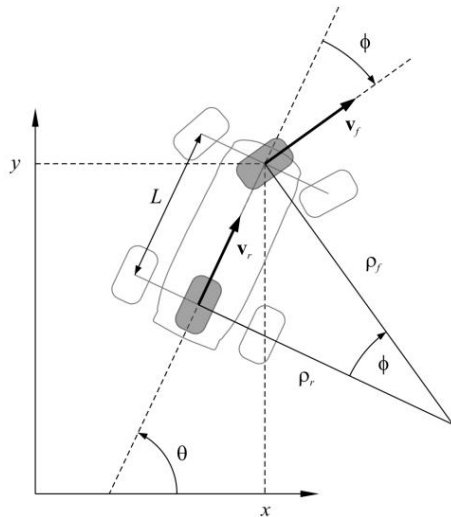
A mérések során alkalmazott giroszkóp tehát nem elég pontos, másikat kell helyette keresni, vagy valami komolyabb szűrő algoritmust alkalmazni a kimenetén. Az ilyen algoritmusok kellően bonyolultak tudnak lenni, és mivel diplomamunkám témájának csak egy kis részét teszi ki a giroszkóp jelének feldolgozása, ezért ebbe nem kívántam mélységeiben beleásni magamat. Kerestem egy olyan szenzor modult, amely ezeket a szűréseket elvégezve egy pontos szöghelyzetet biztosít számomra a mérésekhez. Éppen idén került sorozatgyártásra egy ilyen 9 szabadságfokú rendszer (iNEMO-M1), mely tartalmaz egy 3 tengelyű giroszkópot, 3 tengelyű gyorsulásmérőt és 3 tengelyű magnetométert. A giroszkóp jelét a másik 2 szenzor segítségével korrigálja, és Kálmán-szűrést alkalmazva kiszámolja az Euler-szögeket. A termék újdonsága miatt, sajnos ez nehezen beszerezhető, és a hozzá kapcsolódó dokumentációk is hiányosak, remélem ezek ellenére azért lehetőségem lesz kipróbálni mire képes, megrendeltem.

4.3.2 Új modell felírása

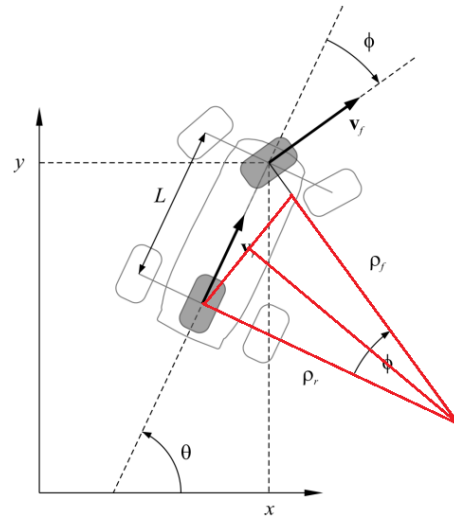
A szenzor hibái ellenére bizonyos mérések kellő pontosságúak ahhoz, hogy a szenzorra várás ideje alatt is továbbfejleszthessem az algoritmusomat. Az eddigi mérések során egyszerű modellel dolgoztam, ahol csak az elmozdulást és az autó szöghelyzetét vettem figyelembe:

```
coordX += dDist * cos(gyroRad);  
coordY += dDist * sin(gyroRad);
```

Ha ismert lenne a kormányzott kerék szöghelyzete, akkor ezen a modellen lehetne finomítani. Kanyarodáskor, az autó kerekeinek állását a 32. ábra mutatja.



32. ábra: Kormányzott autó elmozdulás
modellje 1. [1]



33. ábra: Kormányzott autó elmozdulás
modellje 2. [1]

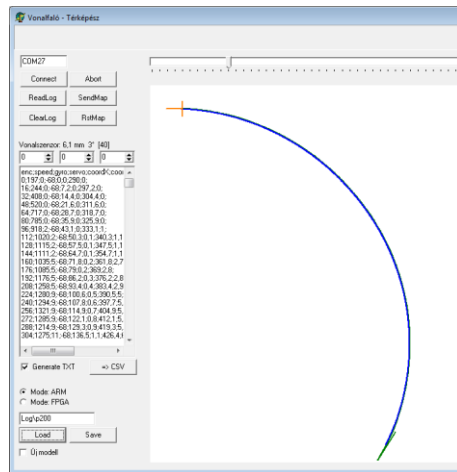
Az autó ilyenkor egy ρ_r sugarú köríven halad, ami jól közelíthető a jobboldali ábrán (33. ábra) berajzolt egyenlő szárú háromszöggel. Az új modell alapján a koordináták számításakor a θ szöghöz hozzá kell adni a ϕ kormányyszög felét, hogy a tényleges elmozdulást kapjam.

4.3.3 Kormányyszög becslése

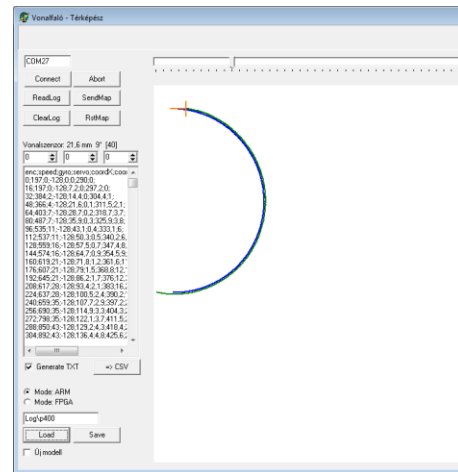
A kormányyszögről ugyan nincs pontos visszacsatolása a rendszernek, de egy kellően gyors szervó kormányozza a járművet. Elkerülve a hirtelen kormánymozdulatokat, a kerék állása jól becsülhető a szervónak kiadott jelből. A szervó és kormánymű együttes karakterisztikájának mérését első körben szögmérővel próbáltam felvenni, de leolvasáskor túl nagy volt az emberi tényező szerepe, ezért más megoldást kerestem. A 33. ábra szerint a kormányyszög számítható az autó fordulókörének sugarából és a tengelytávból az alábbi egyenlet szerint:

$$\phi = \arctan\left(\frac{D}{\rho_r}\right)$$

A fordulókör sokkal pontosabban mérhető, sőt felhasználva a már meglévő helymeghatározó algoritmusomat, egyszerűen elvégezhető. A szervónak fix jelet adtam ki, elindítottam az útvonal napolózását, és megtettem legalább egy negyed kört az autóval. A mérési adatokból (34. ábra és 35. ábra) az x tengely maximumát leolvasva meg is kaptam mekkora az autó fordulókörének sugara.



34. ábra: Fordulókör (szervójel 200)



35. ábra: Fordulókör (szervójel 400)

Az eredmények pontosságában nem lehettem teljesen biztos, ezért az egyik szervóálláshoz tartozó rádiuszt mérőszalaggal is lemértem, és a hibát arányosítva korrigáltam a többi szervóálláshoz tartozó eredményeket (4. táblázat).

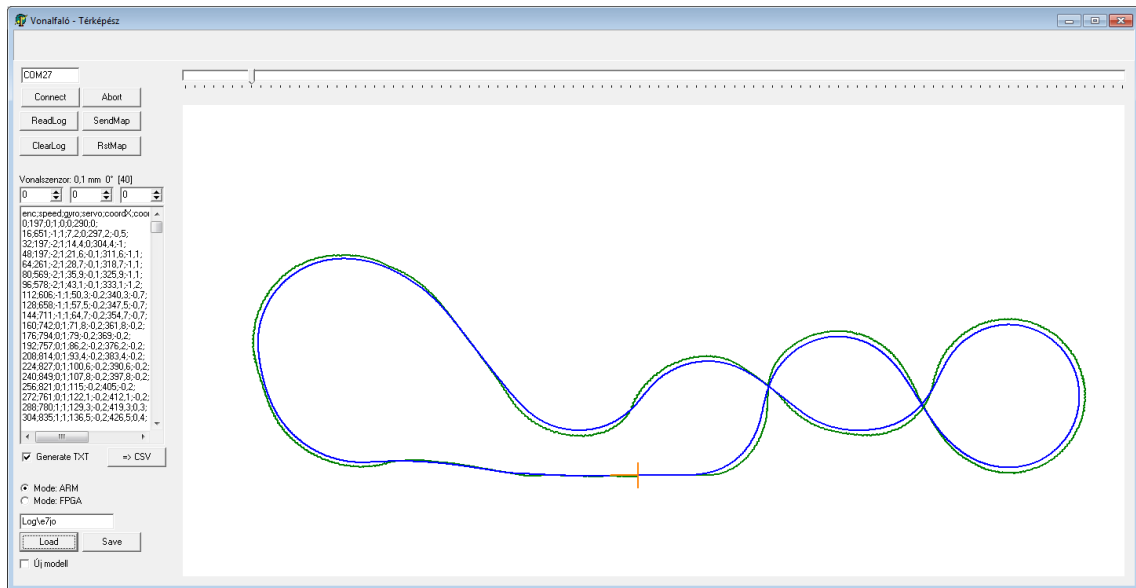
4. táblázat: Szervó jel és kormányzőög kapcsolata

Szervó pozíció	Sugár (naplózott)	Sugár (korrigált)	ϕ
-800	675	608	25,5
800	750	675	23,2
-400	1550	1395	11,7
400	1600	1440	11,4
-200	4460	4014	4,1
200	4250	3825	4,3

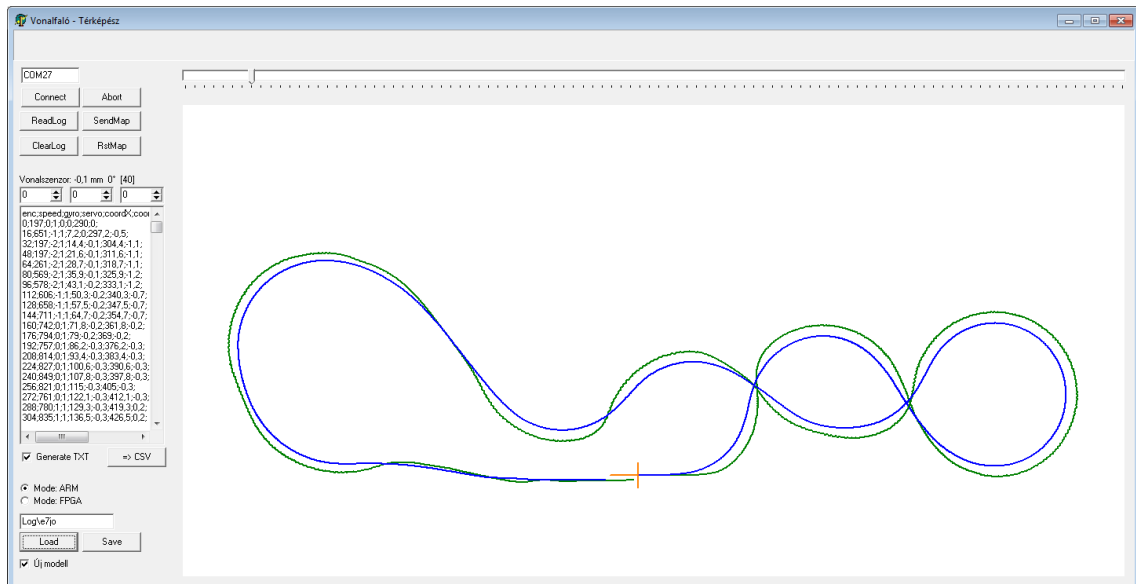
A karakterisztika az elején kicsit érzéketlenebb, de nagyjából lineárisnak tekinthető. A mérési pontokat felhasználva 3 részre bontva a tartományt beírtam a függvényt a járműbe, és a továbbiakban a becsült kormányzőöget is naplóztam a méréseim mellé.

4.3.4 Új modell összehasonlítása az előzővel

A modellek összehasonlítását számítógépen végeztem. Az autóban letároltam az elmozdulásokhoz tartozó giroszkóp és kormányzőögeket, majd ezeken az adatsorokon mindkét modell szerint kiszámítottam a pályát. A régi modell eredményeit a 36. ábra, az újat pedig a 37. ábra mutatja. A kék vonal a hátsó tengely középpontjának koordinátája, a zöld pedig az első tengelyé.



36. ábra: Útvonalrögzítés a régi modellel



37. ábra: Útvonalrögzítés az új modellel

Ez a mérés meglepően jól sikerült, a giroszkóp mindössze fél fokos hibával ugyan azt az értéket mutatta, mint elinduláskor, és a bejárt útvonal is 1cm-es eltéréssel zárt görbét alkot. Az új modell pedig csak rosszabb eredményeket adott, mint az eredeti, itt 5cm-es lett a hiba. Látható, hogy a hátsó és első tengely középpontjai által leírt pályák közt is megnőtt a távolság. Összehasonlítva a fordulókörök rádiuszait a valósággal, itt is az eredeti modell bizonyult pontosabbnak, így az új modell alkalmazását a jövőben mellőzőm. Pontos giroszkóp működés esetéig kellően pontos eredményt adott az eredeti.

5 Térképkövetés

5.1 Követő algoritmus

5.1.1 Elgondolások

Az autó vonalat követni már tud, így a legegyszerűbbnek azt a megoldást találtam, hogy a betáplált térkép alapján szimulálok egy vonalszenzor jelet, és ezt kötöm a vonalkövető szabályzóra. Ez egyrészt gyorsan megoldásra vezet, másrészt könnyen tesztelhető, hogy miként befolyásolja a vonalkövető tulajdonságokat a vonalszenzor elhelyezése, illetve milyen extrákat nyújt a vonal ferdeségének ismerete. A feladat tehát adott, meg kell határoznom, hogy az autó pillanatnyi síkbeli helyzete mellett, a térképet, mint virtuális vonalat tekintve, mit adna kimenetül a vonalszenzor.

5.2 Virtuális vonalszenzor

5.2.1 Abszolút koordináta rendszerben

A betáplált térkép nem más, mint a naplózás során eltárolt koordináták halmaza egy abszolút, derékszögű koordinátarendszerben. Az autó szöghelyzetének és annak egy pontjának koordinátáit ismerve felírható a virtuális szenzorsor képzeletbeli vonalának egyenlete (ami mentén a szenzorok elhelyezkednek).

Ehhez egyrészt szükség van az egyenes meredekségére, illetve egy pontjának koordinátájára. A szenzorsor merőleges az autó szimmetriatengelyére, aminek meredeksége ismert, abszolút koordináta rendszerben ez $\tan(\Theta)$. A szenzorsor erre merőleges, így annak meredeksége $\tan(\Theta + 90^\circ)$.

Az első tengely – ami mentén a virtuális vonalszenzor elhelyezkedik – középpontja is könnyen számolható a hátsótengely koordinátáinak (coordX , coordY) és a tengelytávnak (290 mm) ismeretében.

```
coordXhead = coordX + 290 * cos(Θ);  
coordYhead = coordY + 290 * sin(Θ);
```

Az egyenest „ $y = a x + b$ ” alakban keressük.

```
a = tan(Θ + 90°);  
b = coordYhead - a * coordXhead;
```

Hasonló módszerrel felírható a térkép koordináták közül kiválasztott 2 egymás utáni pont közti szakasz egyenlete is. Az egyenletrendszer megoldásából megkapjuk a szakasz és a vonalszenzor egyenesének metszéspontját, amit meg kell vizsgálni, hogy a szakaszunk egy pontjáról van-e szó, vagy kívül esik azon. Utóbbi esetben tovább kell lépni, és 2 másik ponttal elvégezni ugyanezt a vizsgálatot, mert azt a szakaszt keresem, ami a vonalszenzor alatt van, tehát metszi annak egyenesét. Ha ez a szakasz megvan, akkor már csak a metszéspont fejkoordinátától való távolságát kell meghatározni, illetve megállapítani, hogy az autó szimmetriatengelyétől balra, avagy jobbra esik a pont (például vektoriális keresztszorzás segítségével). Ezt az előjelhelyes távolságot kell továbbadni a vonalkövető szabályozónak.

Miután megvalósítottam ezt az alapjel generátort, kipróbáltam, és már első kísérletre is egész jól végigment az autó a betáplált útvonalon. Azonban néhány furcsaságra lettem figyelmes. Bizonyos helyzetekben hirtelen kormánymozdulatokkal „kerülgette a kátyúkat”. Rájöttem, hogy az abszolút koordinátarendszer nem volt jó választás, itt ugyanis előfordulnak olyan helyzetek, amikor közel végtelen meredekségű egyenesekkel kell számolnia a mikrovezérlőnek, ami hibás eredményeket hozhat.

5.2.2 Relatív koordinátarendszer

Át kellett tehát ténnem relatív koordinátarendszerre. Az x-tengelyt az autó szimmetriatengelyének választottam, az y-t pedig a vonalszenzor egyenesének. Így a fejkoordinátám lett az origó, és a kimeneti alapjel pedig egyszerűen a vonal szakasz és az y-tengely metszéspontjának y koordinátája. A geometriai számítások így nagymértékben leegyszerűsödnek, cserébe minden vizsgálandó pontot át kell transzformálni a relatív koordináta rendszerbe a következő kód szerint:

```
px = mapX[i] - coordXhead;  
py = mapY[i] - coordYhead;  
map2x[i] = px*cos(gyroRad) + py*sin(gyroRad);  
map2y[i] = -px*sin(gyroRad) + py*cos(gyroRad);
```

Transzformáció után egyszerűen megállapítható, hogy a vizsgált szakasz a vonalszenzor előtt / mögött helyezkedik-e el vagy éppen metszi azt, hiszen elegendő a pontok x-koordinátáját megvizsgálni az alábbi feltétel alapján:

```
if ((map2x[i]<0)&&(map2x[i+1]<0)) { /*mögöttünk van a szakasz*/ }  
else if ((map2x[i]>0)&&(map2x[i+1]>0)) { /*előttünk van a szakasz*/ }  
else { /* metszi a vonalszenzor egyenesét */ }
```

Ez abból a szempontból is előnyös, hogy a vizsgálattal egyúttal azt is megkapom, melyik irányba találok a keresett szakaszt. Az autó haladásakor elegendő mindig az előzőleg vizsgált szakaszt újraellenőrizni, hogy továbbra is metszi-e még a szenzor vonalát, és nem szükséges a térkép összes környező szakaszát megvizsgálni.

A szakasz és az egyenesünk metszéspontjának origótól való távolsága:

```
a = (mapy[i+1]-mapy[i]) / (mapx[i+1]-mapx[i]); // meredekség  
y = mapy[i] - a * mapx[i]; // távolság
```

A vonal és az autó szimmetriatengelye által bezárt szög:

```
 $\alpha$  = arctan(a); // vonal ferdeség
```

Az alapjel kimenetek: y és α .

A relatív koordinátarendszer használatával megszűntek a véletlenszerű hibák, jól működött a térképkövetés.

5.3 Vonalkövető szabályzó

Az eddigi vonalkövetés egy PD szabályzóval volt megvalósítva, ami csak az y irányú vonaltávolságot vette alapul, mert a korábbi egysoros szenzorom csak ezt tudta biztosítani. A térképadatokból ezen felül a vonal szöghelyzete is ismert, és ezen információ birtokában sokkal jobb szabályzót lehet felírni. A diplomám keretében tervezett új vonalszenzor is (hasonlóképpen a térképhez) képes mérni a szöghelyzetet, így ez a szabályzó mindkét szempontból átalakítandó.

Egy gyors összehasonlító vizsgálatot végezve, létrehoztam egy olyan szabályzót, ami a kormányzott kerekeket párhuzamosra állítja a vonallal. Ehhez felhasználtam a 4.3.3-as fejezet (Kormányaszög becslése) keretében végzett méréseket. Ez önmagában kevés az autó vonalon tartásához, ezért egy P szabályzóval egészítettem ki, mely a vonal távolságától függően módosít a kormányaszögön. Az így megvalósított vonalkövetés, már a szabályzó paraméterek hangolása nélkül is észrevehetően hatékonyabb volt az eddiginél. Sebesség függő P használatával pedig már elegendően jó eredményt sikerült elérnem.

5.4 Motorvezérlő javítása

A tesztek közben megadta magát a motorvezérlő végfok. Éppen pozíciószabályozás volt érvényben a kerekekre (blokkoló kerekes fékezés), majd egyszer csak elindult hirtelen padlógázzal előre. Felkaptam az autót a földről, elengedtem a biztonsági engedélyező kapcsolót, mire padlógázzal elindult hátramenetbe. Újraindítottam a rendszert, de utána már, ha gázt adtam, csak ugrott egyet a jármű, és sehova tovább. Mivel a mikrokontroller újraindítása nem oldotta meg a problémát, sejtettem, hogy hardveres hibáról van szó, és elkezdtem méregetni a végfokot. Kiderült, hogy a H híd 4 FET-je közül egy teljesen összesült rövidzárra, egy másik konstans 100 ohmot képvisel, kettő viszont látszólag ép maradt.

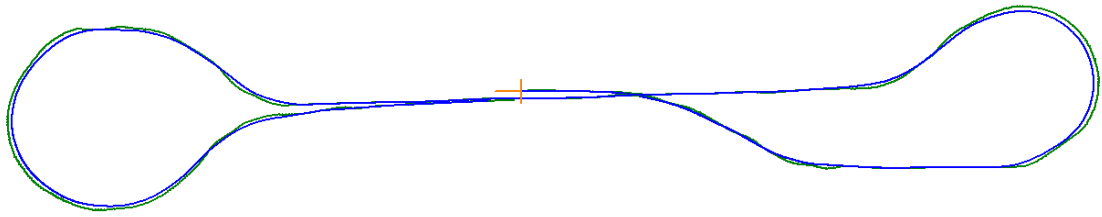
Sajnos ez nem volt egy megbízható áramkör, várható volt, hogy egyszer megadja magát, mivel egy másik csapat is ezt a típust használta, és nekik még a tavalyi verseny közben elfüstölt. Mivel ugyan ilyen típusú meghajtó rendelése heteket vesz igénybe, más megoldást kellett találnom, hogy folytathassam diplomamunkámat. Új motorvezérlő beszerelése időigényes folyamat, és ezt követően valószínűleg a szabályzókat is újra kellene hangolnom, ezért ez sem egy gyors megoldás. Elkezdtem gondolkodni, hogy mit tudok kihozni a félig tönkrement végfokból. A 100 ohm a motor ellenállását tekintve szakadásnak tekinthető, van tehát egy fixen földre nyitó félhidam, és egy működőképes. Ha a működőképes FET-ekből a pozitív ágat nyitom, akkor még hajtható a motor az egyik irányba (ami pont a hátramenet volt, de a motor pólusait megcserélve, most már előremenet). Átprogramoztam a mikrokontrollert, hogy ne is adjon ki ettől eltérő jeleket, amivel rövidzár gyanús viselkedést produkált a jármű (lehetséges, hogy a 100 ohmos FET még továbbra is képes nyitni). Ennek köszönhetően az autó újra működőképes lett, tudtam folytatni a méréseimet, de ha egyszer felgyorsult, nagyon messzire vitte a lendület. Alaposabban átgondolva a helyzetet, rájöttem, hogy ha a működő félhídnak a negatív felét nyitom ki, azzal rövidre tudom zárni a DC motor pólusait ezzel egy motorféket megvalósítva. A várthoz képest meglepően jó lassulást sikerült elérni, ami már a sebességszabályozáshoz is elegendő volt, tudtam folytatni a méréseimet.

6 Pályarögzítés

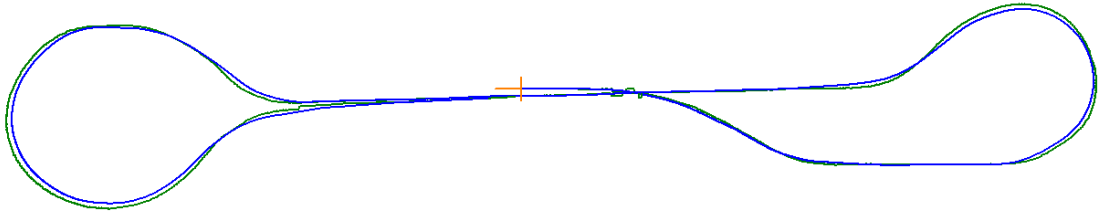
A pályarögzítő funkcióhoz immár minden adott, csak kombinálni kell a meglévő program modulokat. A pálya rögzítéséhez át kell adni a kormány szervót a vonalkövető szabályzónak, és hagyni kell, hogy az autó végigmenjen a szigetelőszalaggal jelölt pályán. Eközben a helymeghatározó algoritmus az eddig megszokott módon eltárolja merre járt az autó. A pontos pályarajz felméréséhez még annyi kiegészítés szükséges, hogy a helymeghatározás közben nem a hátsó vagy első tengely koordinátáját kell tárolni, hanem a vonalszenzor által mért vonalközéppontot. Ezzel kiküszöbölhető a vonalkövető szabályozás hibája. A vonalközéppont koordinátáját a vonalszenzor megadja az autóhoz viszonyított koordináta-rendszerben, ami az 5.2.2-es fejezetben ismertetett módon (annak inverzével) transzformálható vissza az abszolútba.

A pályarögzítő funkció teszteléséhez beállítottam az autót vonalkövetésre, majd hagytam végigmenni a pályán lassú sebességgel. A feltérképezett útvonalat a következő oldalon található 38. ábra jeleníti meg. Itt a zöld vonal az autó orra által bejárt utat jelenti. Látszik, hogy a szabályzó belengése miatt kanyarok elején nem követi a hátsótengely által leírt ívet. A 39. ábra ugyan ezt a rögzített útvonalat tartalmazza, de ezúttal számoltam a vonalszenzor által visszaadott jellel. Itt a zöld jel már nem az autó orra által bejárt utat, hanem magát a pályarajzot tartalmazza. Látható, hogy a kanyarok elején lévő bizonytalanságot nagyon jól kompenzálta, viszont belekerült némi zavar azokon a részekben, ahol két vonal is jelen volt közel egymás mellett. A vonalszenzor jelének átlagolásával ez a hiba is kiküszöbölhető.

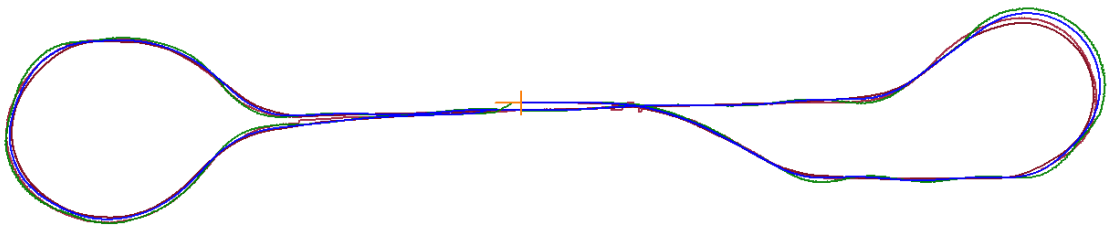
Sajnos a giroszkóp továbbra is pontatlan, mert az iNemo szenzor nem érkezett meg időben, hogy maradjon időm a felprogramozására, beüzemelésére és kommunikáció megvalósítására, ezért csak a régi giroszkóppal tudtam befejezni a munkát. Ennek hibája a naplóból is jól látszik, a visszaérkezési egyenes szögben áll az elindulási egyeneshez képest (5° -os hiba), pedig a valóságban az egy egyenes vonal volt. Ekkora szöghibát integrál össze a giroszkóp az út megtétele során.



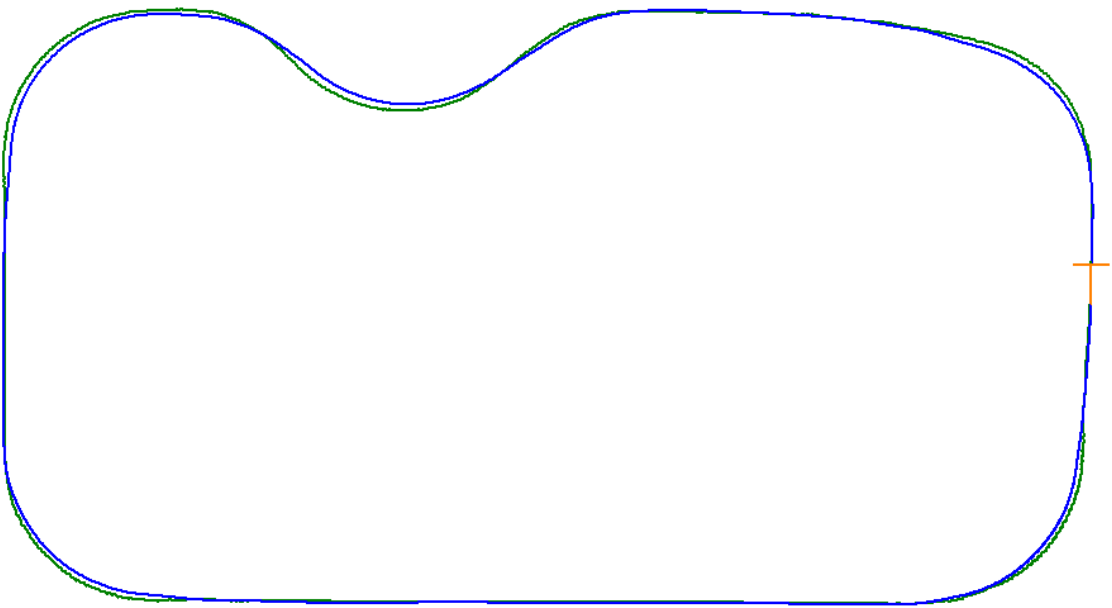
38. ábra: Pályarögzítés fejkoordinátával



39. ábra: Pályarögzítés vonal koordinátával



40. ábra: Rögzített pálya újra bejárása



41. ábra: Pontos pályatérkép

Ezt követően visszatöltöttem az autóra a frissen rögzített térképet, és megnéztem hogyan követi le emlékezetből. Sajnos a giroszkóp pontatlansága miatt egyre nagyobb és nagyobb távolságra kerül az autó a vonaltól, és mire visszaérkezik, már 10 cm-el elcsúszik oldalirányban. Hogy ez valóban a giroszkóp hibája-e vagy a térképkövető algoritmusé, az a naplófájlok összehasonlításából kideríthető, feltételezve, hogy a giroszkóp hibája a két mérés során ugyan akkora maradt (a felvételt és a lejátszást közvetlenül egymás után indítottam, a rendszer újraindítása nélkül). A két pályát a 40. ábra hasonlítja össze, ahol a piros az eredetileg rögzített, a kék-zöld pedig a visszajátszás közben rögzített bejárt útvonal térképe. A két görbe nagyon jól fedi egymást, ezért is gondolom, hogy egy pontos giroszkóppal valóságban is ugyanott haladna végig a jármű. Az első kanyarban, a visszajátszás során az autó szemmel láthatóan letért a vonalról (túl szűk volt neki az ív a nagyobb sebesség mellett). Ez a naplóban is megjelenik, ott ugyanis jelentősen külön válnak a piros és kék vonalak. Ennek ellenére a kanyar utáni egyenes szakaszon egyből visszatalált az eredeti pályára, tehát a pályarögzítő algoritmus megfelelően működik.

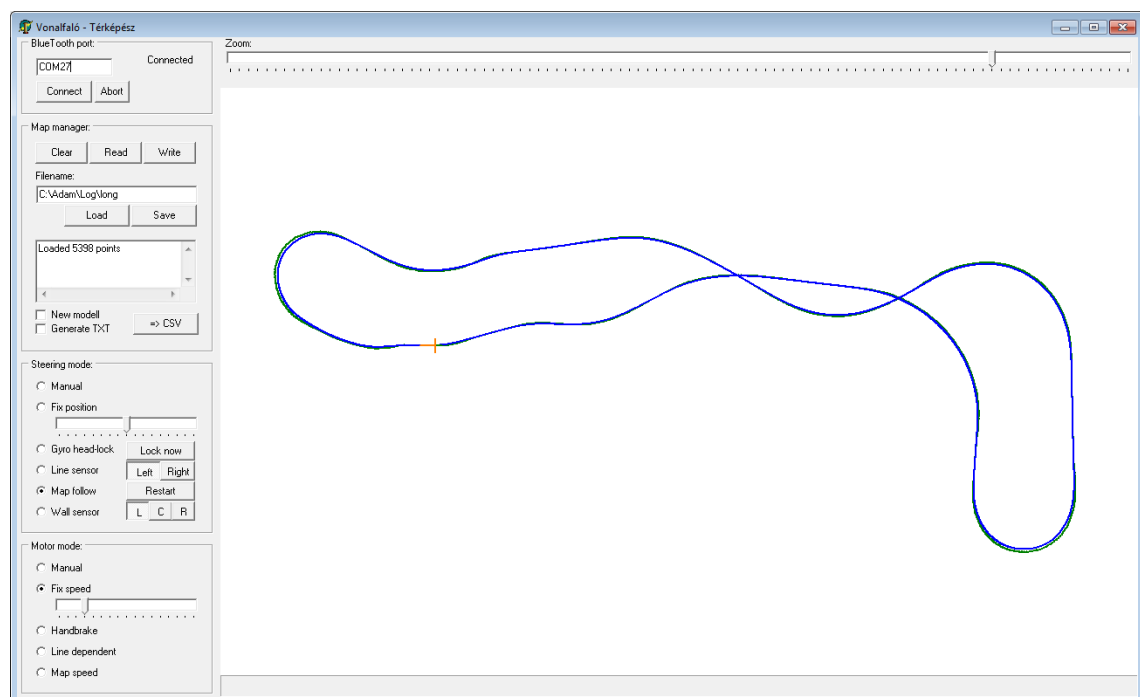
Sok mérést elvégezve, sikerült kifogni olyan ideális körülményeket, amikor a giroszkóp megfelelően kis hibával működött. Az így feltérképezett 41. ábra szerinti pálya hibája (kiinduló és végpont közti távolság) kevesebb, mint 5 mm-re adódott.

7 PC szoftver

A sok üzemmód között kezdett egyre nehezebb lenni a váltás, ezért a mérésekkel párhuzamosan írtam egy PC alkalmazást Delphi környezetben (42. ábra), ami lehetővé teszi a parancsok kiadását, a mérési adatok feldolgozását, és a térképek kezelését.

A helymeghatározás során naplózott adatokat kiolvassa a jármű processzorából Bluetooth-on keresztül, és megjeleníti a bejárt pályát grafikusán. Emellett lehetővé teszi az így feltérképezett pálya elmentését is. Ezt később betölthetjük, és ugyanúgy Bluetooth-on letölthetjük az autóra, hogy megismétlje az útvonalat.

A szoftverbe új térképet bevinni egyelőre csak a koordináták beírásával lehetséges, de a jövőben tervezem kiegészíteni egy egérrel grafikusan rajzolható pálya beviteli lehetőségével.



42. ábra: PC szoftver programképernyője

A szoftver egyben lehetőséget nyújt a különböző vezérlési módok közti váltásra:

Kormány szervó módjai: kézi vezérlés / fix pozíció / iránytartás giroszkóp alapján / vonalkövetés / térképkövetés / fal mellett haladás.

Motor módjai: kézi vezérlés / fix sebességre szabályozás / pozíciószabályozás / vonalfüggő sebesség / térképpel tárolt sebesség. Utóbbi kettő még fejlesztés alatt áll.

8 Továbbfejlesztési lehetőségek

Járműelektronika továbbfejlesztése:

- Szükséges az iNemo beüzemelése, ezzel biztosítva körülményektől függetlenül a megfelelő pontosságot a térben való tájékozódáshoz.
- Hasznos funkció lenne kombinálni a vonalszenzoros és térképes algoritmust. Elgondolásom szerint alapvetően térkép alapján közlekedne az autó a versenypályán, de a vonalszenzor jelét felhasználva szakaszonként korrigálná síkbeli szöghelyzetét és koordinátáit. Így az esetleges megcsúszások esetén keletkezett hibák kiküszöbölhetők. Ez a módszer lehetőséget ad arra is, hogy a szigorú vonalkövetés helyett az ideális íven vegye a kanyarokat.
- Első körben térképezze fel a pályát biztonságos sebességgel, majd a feltérképezett útvonalból számítsa ki, melyik szakaszon mekkora sebességgel tud végighaladni. Ehhez fel kell ismernie a kanyarok sugarát, és rendelkeznie kell függvényvel, amivel megszabja, milyen rádiuszhoz milyen sebesség tartozik.
- Lehet kísérleteket tenni a kanyarok farolva történő bevételére, de ehhez mindenképpen erősebb motorra lesz szükség, és előnyt jelentene az első és hátsótengely független meghajtása is. Ezt a funkciót majd a készülőben lévő új autóban próbáljuk megvalósítani Komancsik Norberttel.

PC szoftver továbbfejlesztési ötletei:

- Egyéb belső változók megjelenítése az oldalsávon: például sebesség, megtett út, szenzorok jelei, áramfelvétel, akkumulátorok állapota, stb.
- Amennyiben a Bluetooth átviteli sebessége győzi, rajzolja folyamatosan a térképet számítógépen, hogy merre jár éppen.
- Egérrel, grafikusan történő térképrajzolás lehetősége.
- Integrálni a vonalszenzorhoz készített, annak jelalakjait megjelenítő programot.

Irodalomjegyzék

- [1] Kolumbán Sándor, Kiss Domokos: RobonAUT Szabályozástechnikai szeminárium (2014. október 17.)
- [2] John J. Craig: Introduction to Robotics: Mechanics and Control 2004 (3. kiadás), ISBN-13: 978-0201543612
- [3] Novák István: Fejezetek a nagyfrekvenciás áramkörök elméletéből 1954, Felsőoktatási Jegyzetellátó Vállalat
- [4] BME-MIT: LOGSYS SPARTAN-6 FPGA Kártya: Felhasználói útmutató (v2.1) http://logsys.mit.bme.hu/sites/default/files/page/2009/09/LOGSYS_SP6_FPGA_Board.pdf
- [5] Dr. Vajda Ferenc: Intelligens Robotok - Mobilis robotok navigációs kérdései Budapesti Műszaki és Gazdaságtudományi Egyetem, Digitális célrendszerek kutatócsoport (2011. szeptember 12.)
- [6] Xavier Pacheco: Borland® Delphi™ 6 Developer's Guide, 2001

Függelék

- Vonalszenzor kapcsolási rajza
- Új (építés alatt lévő) autó főpaneljének kapcsolási rajza
- CD melléklet
 - Hardver projektfilek
 - Szoftver projektfilek
 - Képek és videók