Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Horváth Gergely

AUTONÓM AUTÓMODELL FEJLESZTÉSE

BELSŐ KONZULENS:

Dr. Sujbert László

KÜLSŐ KONZULENSEK:

Baranyai Zoltán

Dr. Balogh András

BUDAPEST, 2016

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Horváth Gergely**, 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 hálózatán 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, 2016. 05. 30.

.....
Horváth Gergely

Kivonat

Napjainkban egyre jobban teret hódítanak a robotika és mesterséges intelligencia alkalmazásai. A technológiai innováció lehetővé teszi, hogy a szakemberek eszköztára egyre jobban kiszélesedjen. A járműipar fejlődése idővel egyre több funkciót integrál a rendszereibe, melyek eredményeként létrejönnek az autonóm járművek. Egy ilyen magas komplexitású rendszer megtervezése számtalan kihívás elé állítja a mérnököket. A valós idejű jelfeldolgozás, szenzor fúziók, szabályozástechnika mind-mind önmagukban is egy külön szakterületet képviselnek, amikhez még hozzáadódnak az automotiv követelmények. A diplomaterv témája egy korlátozott képességű robotautó kifejlesztése, amely jó kiindulási alapot jelenthet az autonóm járművek megtervezéséhez. A modellautó képes egy előre nem ismert pályán kamera és infraszenzorok alapján (redundánsan) vonalkövetést megvalósítani, valamint elkerülni az útvonalon előforduló váratlan akadályokkal való ütközést. Az autó számos funkciót redundánsan valósít meg, amely hibatűrőbbé teszi a rendszert a menet közben felmerülő problémákkal szemben.

A dolgozat röviden ismerteti a korábban megalkotott modellautó főbb paramétereit. Ezután áttekintjük a robotautó irányítását, ami alapján felmérhető a megtervezendő hardver és szoftver bonyolultsága. A rendszertervben a képfeldolgozó egység (Raspberry PI 3), illetve a hozzá kapcsolódó kisebb számítási teljesítményű vezérlőegység (STM32F4) kiválasztása történik. A dolgozat részletesen bemutatja a hardvertervezés főbb lépéseit a tápegységektől kezdve egészen a nyomtatott áramköri lap megtervezéséig. A diplomamunka ezt követően a jelfeldolgozási-, vezérlési algoritmusokat, illetve a hozzá kapcsolódó diagnosztikák szoftveres megvalósítását taglalja. A dolgozat tartalmazza a robotautó megalkotásához, megépítéséhez szükséges lépéseket és megfontolásokat, valamint a validálja a modellt a teszteredmények alapján. Ennek kapcsán vázolja az egyes részegységek működését, az akkumulátor menedzsmentet, a kommunikációt, a távolságérzékelést, illetve összehasonlít két vonalkövető megoldást, majd levonja a konklúziót. A modellautó még számos továbbfejlesztési lehetőséget hordoz magában. Ezek közül a legfontosabbak az ABS és ASR funkciók, illetve a képfeldolgozó algoritmusban is jelentős potenciál rejlik. Ezeket a tapasztalatokkal ötvözve a dolgozat végén tárgyaljuk.

Abstract

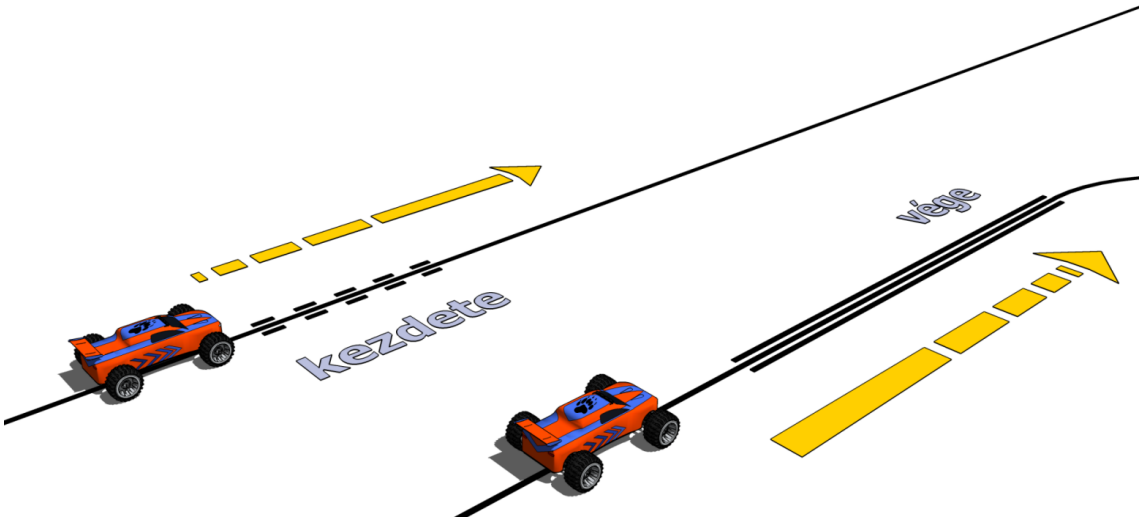
Over the past couple of years robotics and artificial intelligence have been increasingly gaining popularity. Innovation in technology gives more tools to the experts to work with. The car industry integrates more and more features into their products getting closer and closer to autonomous vehicles. Designing such complex systems poses a lot of challenges to engineers. Real-time signal processing, sensor fusions and control theory are separate fields of expertise on their own, not to mention the automotive requirements. This paper demonstrates the creation of an autonomous car with limited abilities. The car is capable of following a line without prior knowledge about the track. The car implements several functions with redundancy to make it more fault tolerant as a system against problems it might encounter on the track.

This thesis gives a short introduction to the parameters of the RC car used to create the robot car. After this, we go through the control of the car, which gives us an idea about the complexity of the hardware and software we need to design. In the system design we choose the image processing unit (Raspberry PI) and the less powerful controller unit (STM32F4). In following section we go in depth into the steps of hardware design from the power supply unit to the design of the PCB. After this, the paper explains signal processing and control algorithms along with the implementation of diagnostics of the software. The thesis incudes the steps and considerations needed to create such a system, furthermore, it validates the model based on test results. The paper discusses the unit part by part as battery management, communication, detection of the distance, respectively compare two line follower solutions, then jump to conclusion. The car has a lot of potential for further improvements, most important of which are the ABS and ASR features as well as the image processing algorithm. These are discussed at the end of the paper along with a summary of my experiences.

1 Bevezetés

A mai modern, digitális világban egyre kevesebb időt töltünk várakozással, hogy eljussunk A pontból B pontba. A felgyorsult technológiafejlődés lehetővé teszi, hogy a mindennapi tevékenységekeinkre fordított időt csökkentsük, legyen az a parkolás, vagy egy adott időpontban felvesz minket a saját autonóm autónk. Néhány éven belül (~2025) az is elképzelhető, hogy a sofőr a munkából hazamenet pihenésképpen megnézze az egyik kedvenc sorozatát. Ezen ok(ok)ból kifolyólag megpróbáltam felmérni, hogy mekkora erőforrásra van szükség egy ilyen feladat elvégzéséhez. A rendszer teljes megvalósítását, tesztelését sokkal bonyolultabb lett volna megvalósítani egy gépjárművel, mint egy kis RC autóval.

2013-ban a Budapesti Műszaki és Gazdaságtudományi Egyetem sokadszorra szervezte meg az autonóm robotok versenyét, ahol háromfős csapatok mérhették össze a tudásukat. A verseny célja, hogy a csapatok egy hétköznapi RC modell autót képessé tegyenek egy előre ismeretlen pályán való végig haladásra, az akadályok megfelelő felismerésére és kimanőverezésére, mindezt emberi beavatkozás nélkül. Az 1.1. ábrán a pálya gyorsulási szakasza látható, mely kezdetét a szaggatott vonalak jelzik, míg a végét a három párhuzamos, egyenként 19 mm széles vonal határozza meg.



1.1. ábra: A gyorsulási lassulási szakaszok jelzései [1]

Az akadályok a hétköznapi forgalomban adódó szituációkra hasonlítottak, úgy, mint a parkolás, a garázsbeállás (ajtónyitással), megfordulás egyenrangú útkereszteződésben stb.

A verseny második fele már nem az ügyességről, hanem a gyorsaságról szólt. Melyik versenyautó tudja a leggyorsabban teljesíteni a négy kanyarból álló gyorsasági pályát. Az autók névleges végsebessége elérheti a 40 $\frac{\text{km}}{\text{h}}$-t is, de a versenypálya korlátozott hossza miatt ez körülbelül 20-25 $\frac{\text{km}}{\text{h}}$-ra csökkenhet.

A diplomamunkám kertében ezt az autót fogom tovább fejleszteni, mind hardveres, mind szoftveres szempontból. Az eddigi vonalkövetés, motor-, kormányvezérlés alapjaiban nem fog megváltozni, de a(z) (intelligens) képfeldolgozó kamera alkalmazásával képes lesz hamarabb felismerni a különböző forgalmi helyzeteket, illetve redundánssá válhat a vonalkövetés, az akadályok percepciója is. Célom, hogy létrehozzak egy működőképes rendszert és mindeközben felmérjem, hogy mekkora számítási kapacitás szükséges a vonalkövetéshez vagy az akadályok észleléséhez.

1.1 A modell autó architektúrája

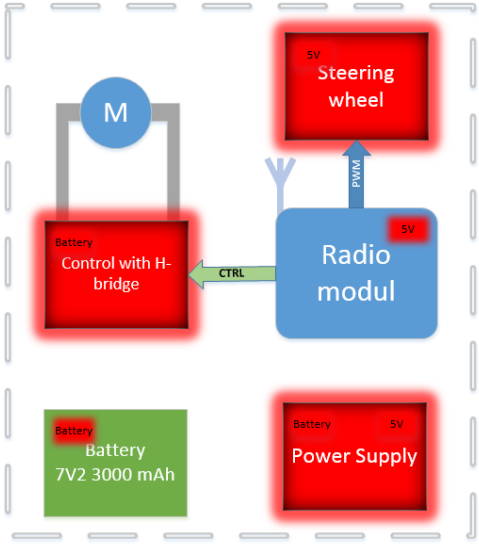
Az eredeti RC autó négy kerék meghajtású, differenciál zárral és (olajos) teleszkópokkal is rendelkezik, ahogy azt az 1.2. ábra is szemlélteti. Az utóbbit a vonalszenzorok miatt mellőznöm kellett (lásd 3.2).

1.2. ábra: Az eredeti RC modellautó

Az RC autó architektúráját az 1.3. ábra mutatja, ahol a rendszer tápfeszültséget egy 7,2 V-os Lítiumion-akkumulátor biztosítja. A kocsi longitudinális mozgását egy DC motor vezérlésével oldották meg, amelynek három állapota van: a teljesen bekapcsolt (előremenet), a kikapcsolt vagy a teljesen bekapcsolt, de ellentétes polaritással (hátramenet). A rádiós modul segítségével fogadja a felhasználó által elküldött

kormányszög, illetve „gázpedál” állásait. Az elindulás pillanatában az akkumulátor a H hídon keresztül rákapcsolódik a nagy teljesítményű DC motorra, amely akár 100 Ampert is képes felvenni, viszonylag hosszabb időn keresztül (200-300 ms). A gyári akkumulátor feszültsége ekkora áramcsúcs hatására akár a névleges érték 10%-ra is lecsökkenhet (körülbelül 1 V-ra). A vezeték nélküli kommunikáció megszűnésekor a kisautó elkezd lassulni a H hídba épített FET-ek body-diódája miatt, ami jelentősen növeli a disszipációt.

Ha a motort egy adott fordulatszámon szeretnénk tartani, tehát, hogy állandó sebességgel haladjon, akkor ez az architektúra alkalmatlan, hiszen a szabályozóval nem lehet az áramot a felső, illetve az alsó hídágban folyatni, amely a sebesség lassú változását eredményezné. A későbbiekben látni fogjuk, hogy mennyire csökken le a disszipáció értéke (lásd 4.1.2.1).



1.3. ábra: A modellautó blokkvázlata

A robotautó első kerekeinek a kormányzását az analóg szervó biztosítja. A rádiós modul a felhasználó által elküldött információt átalakítja egy 50 Hz-es PWM jellé, amelynek a kitöltési tényezője tartalmazza a szöginformációt.

A blokkvázlat által illusztrált megoldás alkalmas arra, hogy megvalósítsa az autó hozzávetőleges koordinációját, de kiírás teljesítéséhez, sokkal precízebb szabályozásra van szükség. Sajnos a RobonAUT verseny napján több példát is láttam arra, hogy a FET-ek body-diódájának vesztesége, rövid időn belül túlmelegítette/tönkretette a „motorvezérlő IC-t”.

2 A robotautó irányítása

Az ismereteimet tovább mélyítve megpróbáltam feltérképezni, hogy milyen eszközökkel lehetne megvalósítani a kitűzött feladatot. Pont ilyen, de sajnos még ehhez hasonló eszközt sem találtam, amely forgalomban lenne. Néhány hasonló külföldi versenyt leszámítva, ahol szintén egy robotot építettek meg képfeldolgozó egységgel. Ha a részegységeket külön-külön meg lehetne venni, abban az esetben is, meg kellene oldani ezek integrációját a kocsival, amelyet a vonalkövető szenzorok nem tettek volna lehetővé.

Milyen alapvető egységek kellenek, hogy teljesüljön a kiírás?

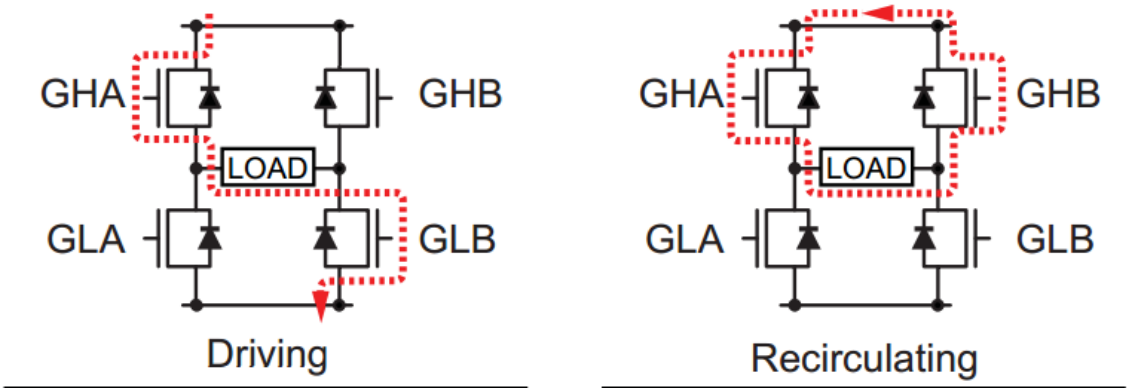
Egy DC motorvezérlőre van szükség, illetve egy kis kormányszervóra, hogy elmozdítsuk az x-y koordináta-rendszerben a kocsit. Ezen felül tudnunk kell érzékelni a vonalat, illetve előnyt jelente, ha egy két akadályt is fel tudnánk ismerni (vonal- és távolságérzékelő szenzorok, kamera). Ezeket az adatokat fel kell dolgozni, átalakítani és vezérelni a perifériákat.

2.1 A longitudinális mozgás megvalósítása

Amint arról már korábban is szó esett, egy nagy teljesítményű DC motort vezérlését kell hatékonyan megoldani. A RobonAUT versenyen egy másik IC-t használtam (Allegro A3941), de sajnos a versenyre készülés során többször is tönkrement a logikai része (pedig a logikaivezérlő egység (1 kΩ), illetve a FET-ek (22 Ω) is egy-egy soros ellenálláson keresztül kapcsolódtak a másik komponensekhez).

A DC motor három típusából egy kefés motort fogunk (brushed motor) irányítani egy H-híddal. Induláskor jóval nagyobb áramot vesz fel, mint állandó terheléskor, hiszen ilyenkor a motor nem más, mint egy soros R-L kör. A PWM vezérlés nem csak ezt a problémát oldja meg, hanem azt is, amikor állandó sebességen akarjuk tartani robotautót (nyilvánvalóan kell legalább egy P szabályozó). A kocsi lassításához háromféle megoldás közül választhatunk: Az első, hogy lekapcsoljuk a híd FET-jeit, majd azok body-diódái vezetik az áramot (generátoros üzem). A másik, hogy a DC motort dinamikusan fékezzük, ami azt jelenti, hogy a híd két felső vagy két alsó ágában a tranzisztorok vezetnek, tehát a motort rövidre zárjuk. A harmadik és egyben leggyorsabb az ellenáramú (irányváltásos) fékezés (plugging), amikor a híd másik két félvezetőjét használjuk arra, hogy csökkentsük az armatúra áramot, pont ellentétesen a 2.1. ábra

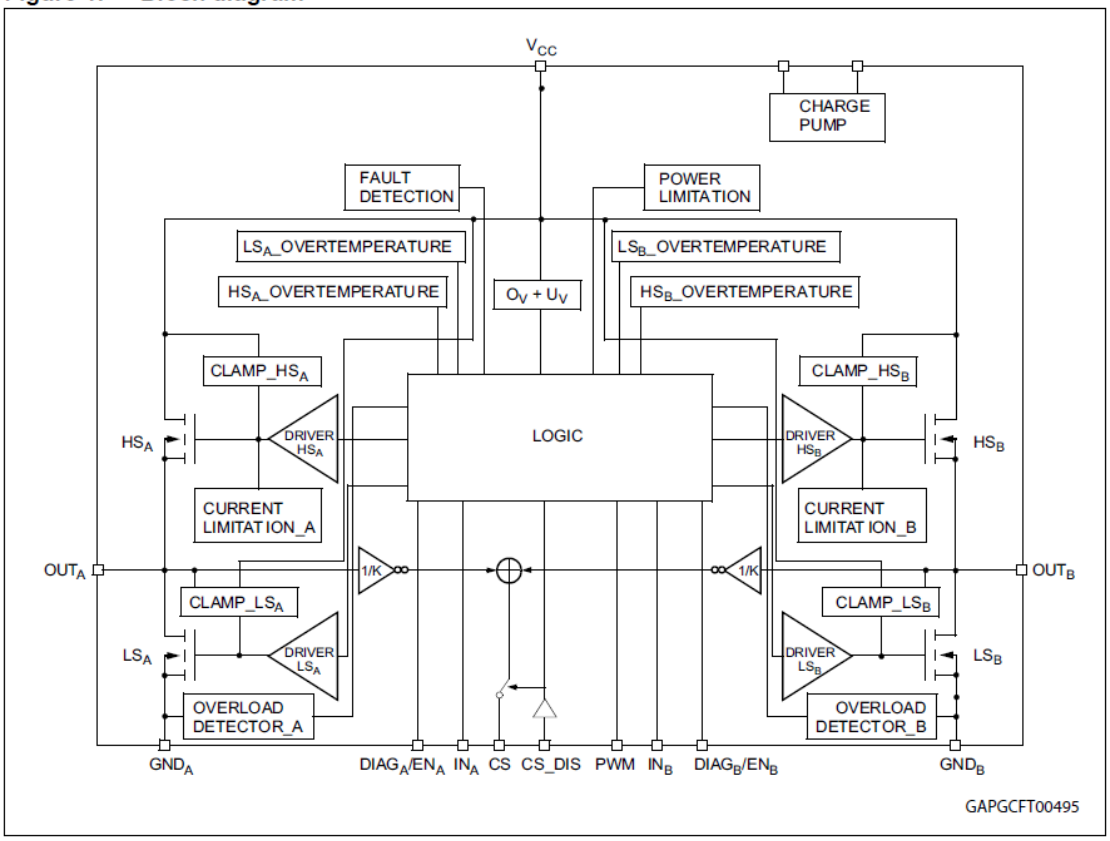
Driving üzemmódjával. Ennek mértékét a felhasználó szabja meg, de ilyenkor a névleges terhelés áramának 50-szerese is megindulhat! Amint majd a későbbiekben látni fogjuk, a fékezésre a generátoros üzemet, vagy a PWM kitöltési tényezőjének megváltoztatását használjuk [7] [36] [37].



2.1. ábra: DC motor vezérlés (driving, dinamic brake) [2]

Az új IC előnye, hogy a H-hídat is tartalmazza, amely lehetővé teszi, hogy több diagnosztikát implementáljanak a komponensbe, ezáltal megbízhatóbbá válik. Továbbá a motor áramát nem kell egy külső műveleti erősítővel és egy shunt ellenállással mérni, mindezt megtehetjük ha két ellenállást csatlakoztatunk az IC-hez (CS pin), mert a motor áramát egy áramgenerátoron keresztül arányosan kivezetik. Ha a motor árama 30 A, akkor az ellenállásokon átfolyó áram ennek körülbelül a hétezrede, azaz 4,3 mA. A funkcionális blokkdiagramot a 2.2. ábra illusztrálja, melyen az előző áramgenerátoros részt az $\frac{1}{K}$ szimbólum jelöli. Az ST VNH5019A-E egy tejesen automotiv DC motor vezérlő IC, amelyet még hővédelemet is tartalmaz [2][3].

Figure 1. Block diagram



2.2. ábra: A motorvezérlő IC blokkdiagramja [3]

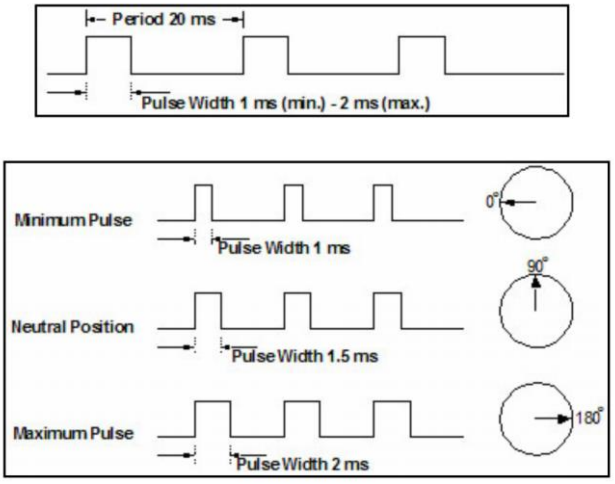
A DC motor sebességét egy 16 kHz-es PWM-mel állítjuk be, hogy az emberi fül kevésbé hallja a kapcsolási frekvenciát. Ekkora frekvencián a kapcsolási veszteségek még nem jelentősek, de a motor áramhullámossága csökkenthető, ha növeljük a frekvenciát (ennél az IC-nél maximum 20 kHz-ig). Az INA és INB vezérlő lábakkal lehet beállítani, a DC motor négy féle vezérlését (a híd négy állapota), míg a DIAGA-val és DIAGB-vel azt deríthetjük ki, hogy a híd melyik ágában történt hiba. A probléma típusának megállapításához nem lesz elég információnk, de azt tudjuk, hogy a motort már nem lehet irányítani. Ekkor ezt a kontrollerrel detektáljuk, majd töröljük a hibát a motorvezérlő IC-ből.

A hátránya az előző IC-vel szemben a disszipáció, míg a külső FET-ek (AUIRF2804) tipikus ellenállása 1,5 mΩ, addig az integrált verzióé 18 mΩ, mely páronként értendő, egy a high side-ra és egy a low side-ra. A disszipáció 6-szorosa az előző verzióhoz képest, ami rendkívül soknak tűnik, de a hardvertervezés fejezetben látni fogjuk, hogy elvezethető ez a hőmennyiség (4.1.2) [3] [4].

2.2 Laterális mozgás megvalósítása

Az autó gyári tartozéka egy kis 50 grammos analóg szervó, aminek a tápfeszültsége 5 V, de ennek értéke egy kisebb skálán változhat (4,8 V-6 V). A nyomaték és a beállási idő a tápfeszültség függvényében változik az előbbi nő, míg utóbbi csökken, ha a feszültséget növeljük (4,8 V -> 6 V).

A kis analóg szervót maximum 50 Hz-es PWM jellel vezérelhetjük, amelynek az impulzus szélessége meghatározza, hogy éppen milyen szög értékre kell elfordulnia. A minimum és maximum értékeket leolvashatjuk a 2.3. ábráról, illetve észrevehetőjük, hogy a 180 °-os tartományt mindössze 1 ms képviseli.



2.3. ábra: Szervo motor vezérlése PWM-el [5]

Amint a szervó megkapja az új pozíció értéket, megpróbál a lehető leggyorsabban beállni (ami terheletlenül $\frac{0{,}25\ s}{60\ °}$)[1]. Folyamatos PWM esetén, bármilyen külső erő hatására a szervó megpróbálja az adott pozícióban tartani az első kerekeket. Értelemszerűen, egy nagyobb külső erő esetében, amelyet az adatlapokban nyomatékként adnak meg, már nem képes a kormányszöget kiaktuálni. A gyári szervó esetében ez az érték annyira kicsi, hogy ha az autó parkolás közben (körülbelül nulla sebességnél) nem képes elfordítani megfelelő szögben a kormányt[2], csak valamilyen szöghibával.

[1] Az eredeti kis szervó közelítő értéke

[2] Ekkor egy búgó hangot hallunk, és az áramfelvétele megnövekszik 100 mA környékére

Eddig csak az analóg szervóról beszéltünk, de léteznek digitális szervók is, melyek előnyei messze túlszárnyalják a hátrányait. A vezérlésben, a tápfeszültségben, a motorban és a fogaskerekekben nincs (számottevő) eltérés, de a visszacsatoló részben már igen.

A digitális szervókban egy mikrokontroller végzi az információfeldolgozást és a szabályzást is. A kis MCU lehetővé teszi, hogy egy egyenletrendszert oldjon meg, ami egy előre beprogramozott matematika modell. Típustól függő, de a számítási idő körülbelül 0,1 ms és 2 ms közötti.

Vessünk egy pillantást a következő ábrákra (2.4. ábra, 2.5. ábra), ahol a sárga jel az 50 Hz-es PWM, míg a zöld a szervók áramértékei. Az analóg szervó áramcsúcsai a PWM jel periódusával szinkronban van, míg a digitálisnál egység, két periódus között többször változtatja az áramfelvételét, tehát utóbbi gyorsabb, mint a vezérlő jele.

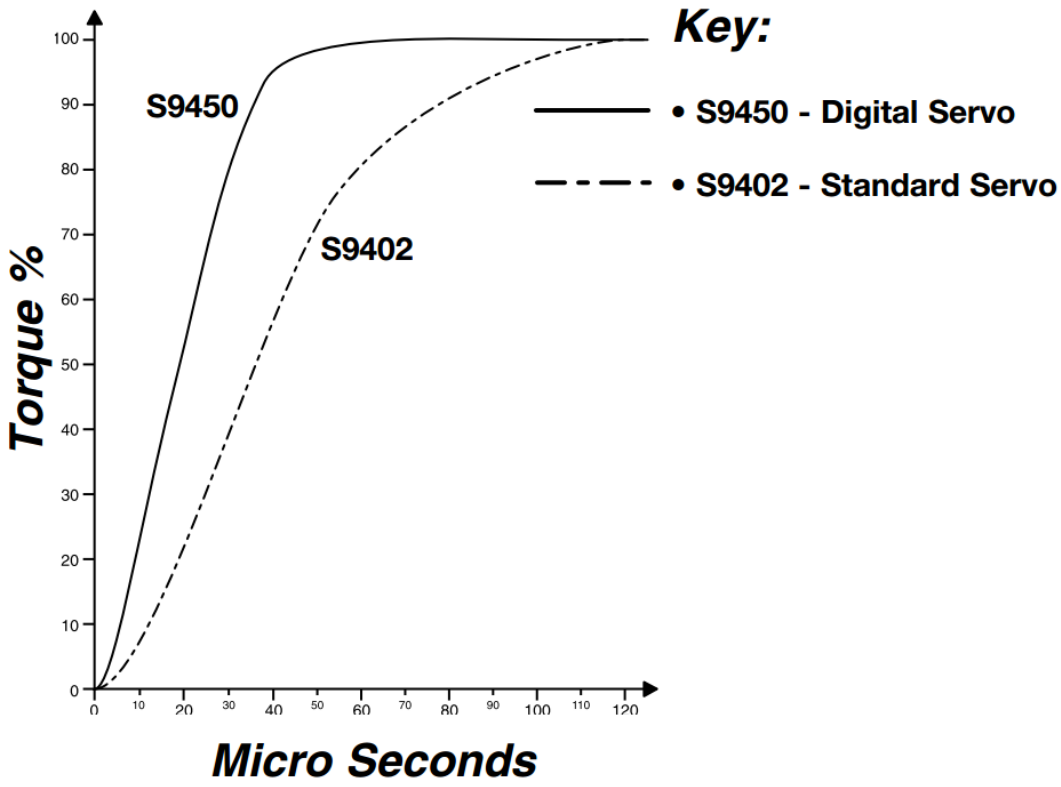
2.4. ábra: Analóg szervo áramcsúcsai [8]

2.5. ábra: Digitális szervo áramcsúcsai [8]

Nagyobb terheléseknél a digitális szervók teljesítményét nem befolyásolja a PWM frekvenciája, ellentétben az analóggal, ahol az információfeldolgozást követően hosszabb ideig vesz fel áramot. A mikroprocesszor a szervó teljesítményét optimalizálja, egyenletesebbé teszi. Az analóg szervók áramfelvétele mindössze néhány 100 mA (100-800), míg a digitális esetén ez az érték elérheti akár a 4 Ampert is. Ha egy adott rendszerhez sok ilyen szervó tartozik, akkor gyorsan megnövekedhet az áramfelvételünk.

A digitális résznek köszönhetően a PWM vezérlési frekvenciája 50 Hz-ről 300 Hz-re emelkedhet, ami 6-szor gyorsabb, mint az analógé. A holtsávja is kisebb, az előző két ok miatt, amivel nagyobb pontosságú szabályozást érünk el. Néhány digitális szervónál azt is be lehet állítani, hogy az adott pozícióba tartsa, miután a PWM vezérlés megszűnt. Ez fontos lehet egy emberi robotnál, ahol a „kéz szerepét” ez a kis egység tölti

be. Az előnyei közé tartozik még, hogy 100%-os nyomatékot, már fele annyi idő alatt eléri, mint az analóg, ahogy a 2.6. ábráról leolvasható.



2.6. ábra: Analóg és digitális szervo nyomatékainak beállási ideje [6]

A digitális szervó használata az alábbiakat eredményezi [6][7][8][9]:

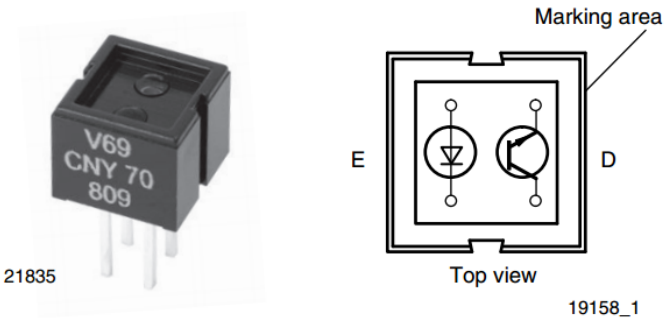
- nagyobb lesz a felbontásunk (kisebb a holtsávunk)
- kisebb lesz a válaszidőnk (gyorsabb lesz a beállási időnk)
- a szervó mozgatása közben konstans lesz nyomatékunk
- álló helyzetben (pl. parkolás) nagyobb erőt tudunk kifejteni
- a tápegységünket fel kell készíteni a nagyobb áramcsúcsokra

2.3 A vonalkövetés

Az elsődleges jel felismerése nem más, mint egy fekete vonal, ami egy körülbelül 19 mm-es fekete szigetelőszalag. Ebből nyilvánvalóan lehet több is, hiszen a gyorsuló és lassuló szakaszokat is detektálni kell.

2.3.1.1 IR szenzorokkal

Az előző modellben ezt a funkciót egy CNY70-es optikai szenzorral terveztem meg, melyet a 2.7. ábra illusztrál. Itt semmilyen probléma nem merült fel, úgyhogy akár át is emelhettem volna a korábbi verziót, de ehhez képest megvizsgáltam, hogy van-e jobb, gyorsabb szenzor. Első körben nézzük meg, milyen paraméterei vannak a CNY70-esnek, majd néhány alternatívát is szemügyre veszünk.



2.7. ábra: A CNY70-es optikai szenzor [10]

Az optikai szenzorban található dióda infravörös tartományban fényt bocsájt ki, amely az adott felületről visszaverődve a fototranzisztor nyitásifeszültségét állítja be. A fehér felületről a visszaverődés mértéke körülbelül 90%, míg a fekete felületről ez az érték nagyjából 10%. Ez a fénysáv (~950 nm) lehetővé teszi, hogy akár éjszaka, világítás nélkül is megvalósuljon ez a funkció. A vonalkövetés minőségét leginkább a szenzor és a felület távolsága befolyásolja. A korábbi verzióban sikerült 5 mm-re fixálni ezt az értéket, amely nagyon jó jel-zaj viszonyt eredményezett [10].

A második befolyásoló tényező a szenzor áramfelvétele, amely a fel- és lefutási időket akár többszázszorosára is változtathatja. Néhány mA-es áramerősség esetén (1-2 mA) ez az érték elérheti akár a 100 µs-ot, míg 10 mA-es esetben 200 ns-ra csökkenhet. A nappali fényviszonyok megváltozása a szenzor „hatásfokát” redukálhatja, ha egy emelkedőn felfele megyünk, akkor a napsugarak beesési szöge módosulhat, vagy éppen egy világos helyről egy árnyékos részre érkezünk. Szerencsére a CNY70-es rendelkezik egy nappali fényszűrővel, amely megkönnyíti a fekete vonal érzékelését [10].

A szenzorok hátránya, hogy a szélességük, illetve a hosszúságuk is 7-7 mm, míg a vonal névleges szélessége 19 mm. Egyszerű számolás alapján bebizonyítható, hogy a 3 érzékelő közül egy, biztosan teljesen feketének fogja látni az adott felületet, míg a másik

kettő csak részben. A hardvertervezés fejezetben több ilyen IR szenzort fogok összehasonlítani.

2.3.1.2 Kamerával

A kamera felhasználásával az előző szenzort kiváltva, vagy éppen párhuzamos feldolgozással lehetőség van egy másfajta vonalkövetés megvalósítására. Egy digitális kép beolvasásának számtalan mondja lehet, a modern technológia miatt: digitális kamera, szkenner, CT[3], MRI[4]. Egy HD minőségű kép felbontása 1280x720, amihez még a pixelértékek is hozzátartoznak. A képet értelmezhetjük, úgy, mint egy mátrixot, amelynek a felbontása határozza meg a sorai és az oszlopai számát, illetve a mátrix értékei a pixelek lesznek. Az utóbbi értékeket különbözőképpen tárolhatjuk el, melyekhez segítséget nyújtanak az eltérő színterek. A színtér nem más, mint a színek tárolásának egy matematika reprezentációja. A leggyakoribb az RGB formátum, mely a három alapszínt használja fel, pont úgy, mint az emberi szem, hogy megalkossa a színes képet. Negyedik paraméterként hozzáadhatjuk az átlátszóságat, amit α-val szokás jelölni. Azonban számos más színrendszer elérhető más-más előnyökkel és hátrányokkal [29] [44]:

- RGB: Nagyon egyszerű additív színrendszer, de nem túl effektív[5] (közvetlenül nem férünk hozzá a pixelértékekhez), illetve sajnos nem-lineáris.
- CMY(K): Meglehetősen egyszerű implementálni, de nehezen transzformálható (pl. RGB-ből). Kivonás alapú színtér, ami szintén nem-lineáris. A negyedik komponens fokozza a színskálát és a sűrűséget (pl. nyomtatóknál).
- HSL[6]: Lineárisan transzformálható az RGB formátumból és nagyon egyszerű módosítani a színárnyalat alapján a szaturációt és az intenzitást.

[3] Computed Tomography / számítógépes tomográfia

[4] Magnetic Resonance Imaging / mágneses magrezonancia

[5] Intezitás módosítás folyamata: pixel kiolvasása, módosítása, pixel felülírása

[6] Hue Saturation and Lightness / Színárnyalat telítettség és fényesség

- YIQ, YUV, YCbCr[7], YCC: Az emberi szem számára sokkal szebb minőséget nyújt, ugyanannyi bit felhasználásával, mint az RGB formátumnál a fényesség és a színtelitettség miatt.
- CIE: Kétfajta színtere létezik, amelyek lineárisok (CIELuv, CIELab), ,illetve könnyen mérhető a színek közötti távolság

Mindegyik pixel értékhez tartozik egy adat típus. Ha az RGB formátumnál maraduk, akkor 24 bites színmélység 8 bitenként feloszthető a három alapszín intenzitásértékeire.

Tételezzük fel, hogy másodpercenként legalább 30 HD képet készítünk, illetve hogy a színtere RGB888. Ekkor körülbelül 632 Mbit-et kell átvinni a kamera és a feldolgozó egység között. Ez hatalmas adatmennyiség, amelyet nem szabad figyelmen kívül hagyni. Ha csak egy monokróm kép információit kell továbbitani, akkor az csak 211 Mbit másodpercenként. A 2.8. ábra egy-egy kép mátrixértékeit illusztrálja egy 3x3-mas mátrixban. A kép jobb oldali része a fentebb említett RGB888-es formátumot reprezentálja, míg a baloldali egy fekete-fehér kép mátrixértékeit.

Gray scale image

	Column 0	Column 1	Column 2
Row 0	128	18	85
Row 1	67	244	116
Row 2	65	21	28

RGB image

	Column 0			Column 1			Column 2		
Row 0	74	21	116	200	172	250	222	49	6
Row 1	104	182	171	97	22	96	210	48	99
Row 2	221	245	170	32	109	124	47	153	53

2.8. ábra: Monokróm és az RGB kép mátrixai

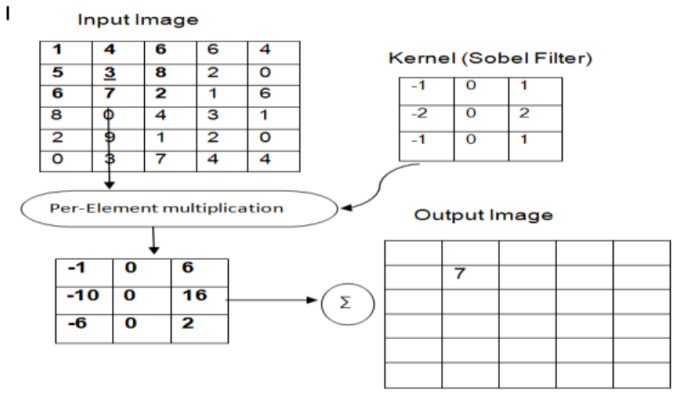
2.3.1.2.1 Éldetektálás

Mi egy fekete vonalat keresünk, amelyhez elegendő egy monokróm kép is. Általában a képen éleket keresünk, amelyek a képfeldolgozás létfontosságú elemei közé tartozik, mellesleg a legtöbb komplex problémára is megoldást nyújt, úgymint, akadályfelismerés, vonalkövetés, szegmentálás, vagy éppen a képrekonstrukcióban [32].

Sokféle algoritmussal lehet megvalósítani a fenti feladatot, melyek közül válasszuk ki kettőt: a Sobel operátort, illetve a Canny féle éldetekciót.

[7] Y: Luminance (fényerősség), Cb: chrominance blue (színtelitettség kék) Cr: chrominance red (színtelitettség piros)

A Sobel operátor kettő 3x3-mas maszk mátrixot használ (2.9. ábra), ahol az egyikkel az x irányú gradienst, míg a másikkal az y irányút számolja ki. Az algoritmus mindegyik képponton meghatározza az intenzitás értékeket, amelyek az éleket reprezentálják. A Sobel algoritmus nem más, mint egy konvolució [32].

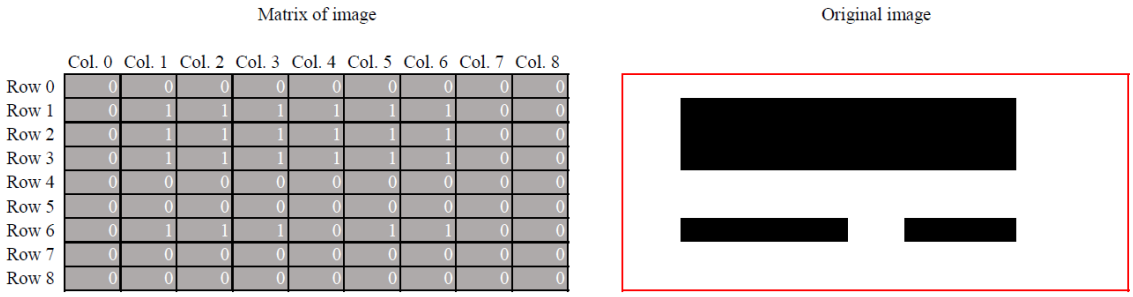


2.9. ábra: Sobel operátor konvolúciója [33]

A Sobel operátor lépései a következők:

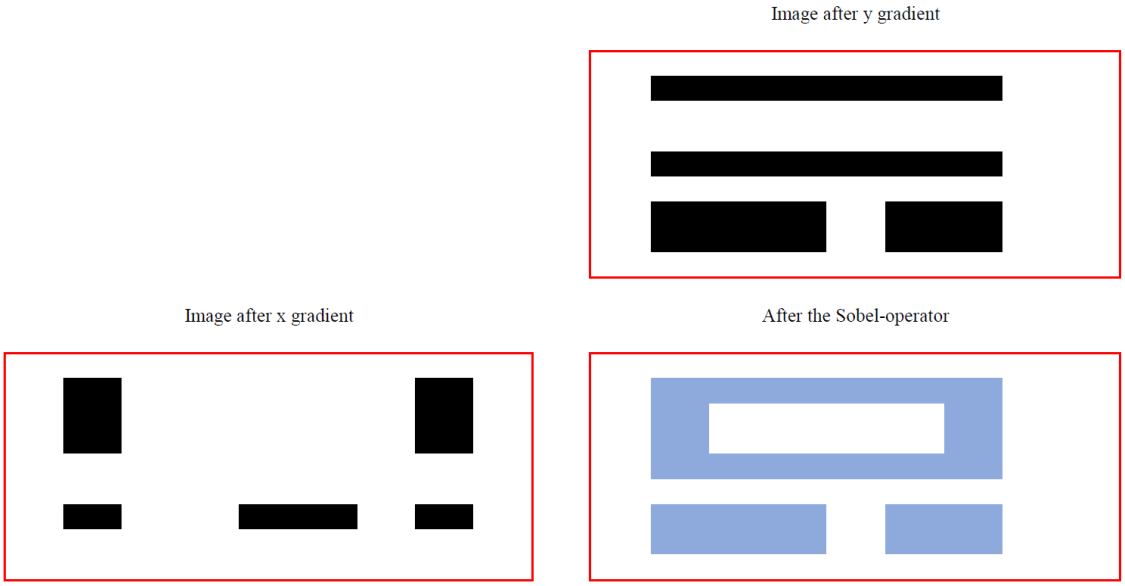
1. A bementi kép elfogadása.
2. A 3x3-mas mátrixok (Gx, Gy) alkalmazása a képre.
3. Sobel él detekció és gradiens alkalmazása
4. Gx, Gy maszkok kezelés külön-külön a bementi képre.
5. Abszolút maximum meghatározása.
6. Az abszolút nagyság lesz a kimeneti kép.

Az algoritmus ismertében nézzünk egy példát arra, hogyan is működik a Sobel operátor a gyakorlatban, melyet a 2.10. ábrán vizsgáljunk meg.



2.10. ábra: A kép és a bináris mátrixa

Az egyszerűség kedvéért a kép mátrixát bináris értékekkel elemezzük. Induljuk el a mátrix bal felső sarkából, majd oszloponként haladjunk végig az x irányú gradienskereséshez. Az első és a második oszlop között az felső sorban nincs értékváltozás, ezért nem jelölünk semmit. A második sorban már igen, hiszen az $M_{1,0}$ és az $M_{1,1}$ között, intenzitásváltozás történt, amit jelölnünk kell, ahogy az a 2.11. ábrán is látszik. Hasonlóképpen kell eljárnunk a második, harmadik és a hatodik sorban is. A következő oszlopot megfigyelve, sehol nem történt változás a másodikhoz képest. Tovább haladva, a negyedik oszlopban az $M_{6,3}$ és az $M_{6,4}$ között szintén egy értékváltozás következőbb be, amelyet jelölünk. Tovább folytatva az algoritmust, meghatározhatjuk az x irányú gradienseket. Ha az y irányból is elvégezzük az előző lépéseket, akkor kapjuk az y irányú intenzitásértékek változásának a helyeit. Az x és y mátrixokat egy közös képpé formálva, a Sobel operátor eredményét olvashatjuk le. Az eredeti képpel összehasonlítva az látható, hogy az algoritmus megtalálta az éleket, melyeket kiolvashatunk a mátrixból, ha végig böngésszük a pixeleit.



2.11. ábra: A Sobel operátor rész- és végeredményei

A Canny éldetektálás az egyik leggyakrabban használt képfeldolgozási eszköz, amely 5 lépésből áll:

1. Simítás: A kép elhomályosítása, hogy eltüntesse a zajt (Gauss szűrés, x-y)
2. Gradiens keresése: Az éleket a nagy gradiensek jelölik.
3. Nincs maximum elnyomás: Csak a lokális minimumok lesznek jelölve

4. Kettős küszöbszint: A várható éleket a küszöbszintek határozzák meg (hiszterérzis)
5. Élkövetés: Végső éleket meghatározni, úgy, hogy amelyikek nem kapcsolódnak egymáshoz, azokat eldobja.

Tegyük fel, hogy a képünk már zajmentesített. A gradiens keresés a Sobel operátor alapján történik, de ezen felül még azt is meghatározza, hogy milyen az él iránya, amit a második deriváltból származtat. Ez azért fontos, hogy a küszöbszinteknél egymáshoz tudjunk viszonyítani, ezáltal a tényleges határvonalat kapjuk meg, úgy ahogy a bináris élkeresésnél is.

2.12. ábra: Canny vs eredeti vs Sobel

A 2.12. ábra prezentálja az algoritmusok eredményét, ahogy látszik is, a bal oldali rész sokkal zajmentesebb és jobban kitűnnek az élek. Az emberi szem első ránézésre is el tudja dönteni, hogy ugyan azt látja, csak az egyik „makulátlan”.

A 2.1. táblázat foglalja össze az algoritmusok előnyeit, illetve hátrányait.

Operátor	Előnyei	Hátrányai
Sobel	Egyszerű, gyors Könnyen detektálja az éleket és azok orientációit	Kissé pontatlan (75%) Zajérzékeny
Canny	Jó a lokalizációja, illetve a pontossága (87,5%) Remek a zajelnyomása	Bonyolult lehet valós idejű alkalmazásokban az teljesítményigénye miatt

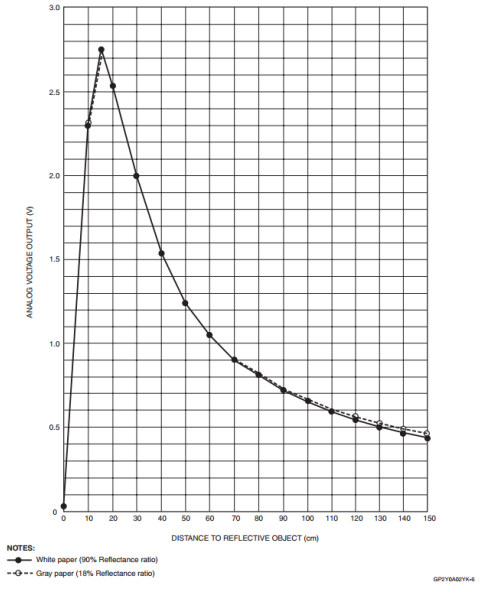
2.1. táblázat: A Sobel és a Canny algorimusok összehasonlítása [32] [45]

Számtalan más él detektáló algoritmus létezik, úgymint, Prewitt (pontatlan), Roberts (lokációs problémák), melyek nagyon hasonlóak a fentebb említettekhez, de a gyakorlatban ezeket a használják a legtöbbször [34].

2.4 Távolságérzékelés

2.4.1.1 IR[8] szenzorral

Alapvetően a kocsi körüli távolságérzékeléshez három szenzorra lesz szükség, egy előre, és kettő oldalra. Ezzel a konfigurációval tolatás közben nem tudjuk érzékelni az akadályokat, de ez nem is volt része a feladatnak. Az első szenzornak legalább 80-90 cm-es tartományt kell lefednie, míg a két oldalsónak 30-40 cm-t. Gondoljunk csak bele, hogy a modellautó 5 $\frac{m}{s}$-os sebességgel megy és kevesebb, mint egy méter a féktáv, akkor minden valószínűséggel neki ütközik a falnak. Ez persze a legextrémebb eset. A cél az, hogy legyen egy jól működő szenzor, amellyel biztosan meg tudjuk mondani, hogy van-e előttünk akadály vagy sem. Ezen felül még azt is, hogy az milyen messze is van, hiszen ez alapján lehet eldönteni, hogy fékezzünk, vagy pedig kikerüljük. A két oldalsó szenzornak azt kell érzékelnie, hogy van-e a közelben valamilyen test vagy nincs. Az utóbbi két szenzor nem feltétlen szükséges a feladatok maradéktalan végrehajtásához.



2.13. ábra: 20 és 150 cm közötti távolságérzékelő [14]

[8] Infra-Red

Az előző architektúrában két analóg szenzort használtam fel, amely hasonló elven működik, mint a korábban bemutatott CNY70-es. A különbség, hogy itt egy visszahajló karakterisztikát kapunk a fekete és a fehér felületekről, ahogy a 2.13. ábrán látszik.

A Sharp GP2Y0A02YK távolságérzékelő szenzor 20 és 150 cm közötti tartományban jelzi az előttünk lévő objektumokat. A tipikus válaszideje 39 ms, az áramfelvétele 33 mA, a pontosságát elsősorban az ADC felbontása határozza meg [14]. 150 cm-ről 20 cm-ig a tipikus feszültségváltozása értéke 2,05 V, ami egy 12 bit-es A/D[9] esetén 0,5 cm-es reális felbontást valósíthat meg.

A távolságot nem csak infra szenzorokkal érzékelhetjük, hanem ultrahanggal is, mely logikailag hasonlóan működik, mint az előző alkatrész, csak nem fénnyel, hanem ultrahanggal. A Parallax 28015-ös szenzor 18,5 ms alatt méri meg a 3 méteres távolságot, ami a legrosszabb esetben is kétszer gyorsabb, mint az előző. Az áramfelvételeik és a fizikai kiterjedésük körülbelül megegyezik, de a hőmérséklet, a páratartalom hatására megváltozik a hangterjedési sebessége (V_S), ami miatt korrekcióra van szükség. A 2.1-es egyenlet tartalmazza ezt, ahol $V_0 = 331{,}3\ \frac{m}{s}$. Kompenzáció nélkül, ha egy tárgy 2 méterre van és a hőmérséklet 30 °C-ot változott (T), akkor az abszolút hiba 0,1 m lesz, míg a relatív hiba 5,2%, ahogy ezt a 2.2-es egyenlet eredménye igazolja [15].

$$V_S = V_0 + 0{,}606T \qquad (2.1)[15]$$

$$\frac{\Delta V}{V_S} = \frac{V_S - V_0}{V_S} \qquad (2.2)$$

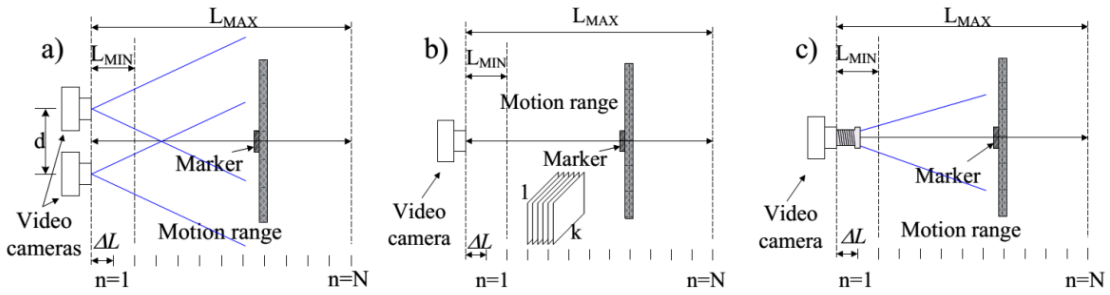
Az infra szenzorhoz képest nem egy ADC-n keresztül nyerjük ki az információt, hanem egy adott hosszúságú PWM impulzussal, amelyet egy időzítőre lehet kötni. Ezt követően mindig triggerelni kell, ha mérni szeretnék, míg a másik ezt automatikusan elvégzi. Sajnos csak nagyobb objektumokat képes érzékelni a hang alapú szenzor, illetve mindig merőlegesen kell beesni a hullámoknak, különben nem térnek vissza az érzékelőhez. A távolságérzékeléshez az infra szenzorokat fogom használni az egyszerűségük és robusztusságuk miatt.[15]

[9] Az A/D referenciafeszültsége 3,3 V.

2.4.1.2 Kamerával távolságmérés

Alapvetően ahhoz, hogy távolságot mérjünk két kamera szükséges, amelyet sztereó képfeldolgozásnak neveznek. Ha tudjuk a kamerák paramétereit és a relatív pozíciójukat, akkor háromszög módszerrel meghatározható az általuk látott objektum. A két felvevő egység elhelyezkedésest, 2.14. ábra a része mutatja.

Lehetőség nyílik az előző algoritmus továbbfejlesztésére, illetve hardveres költségcsökkentésre, ha csak egy kamerát használunk fel. Ekkor, a kamerával két különböző helyen kell képet készítenünk, amely egy mozgó rendszerben könnyedén teljesíthető. Ez után az inverz perspektív transzformációt felhasználva, megkapjuk a távolságértéket, ha meghatároztuk a karakterisztikáját az objektumnak. A harmadik megoldásban az autófókuszt használjuk ki, melyet a 2.14. ábra c része illusztrál [47].



2.14. ábra: Távolságmérés kamerával [47]

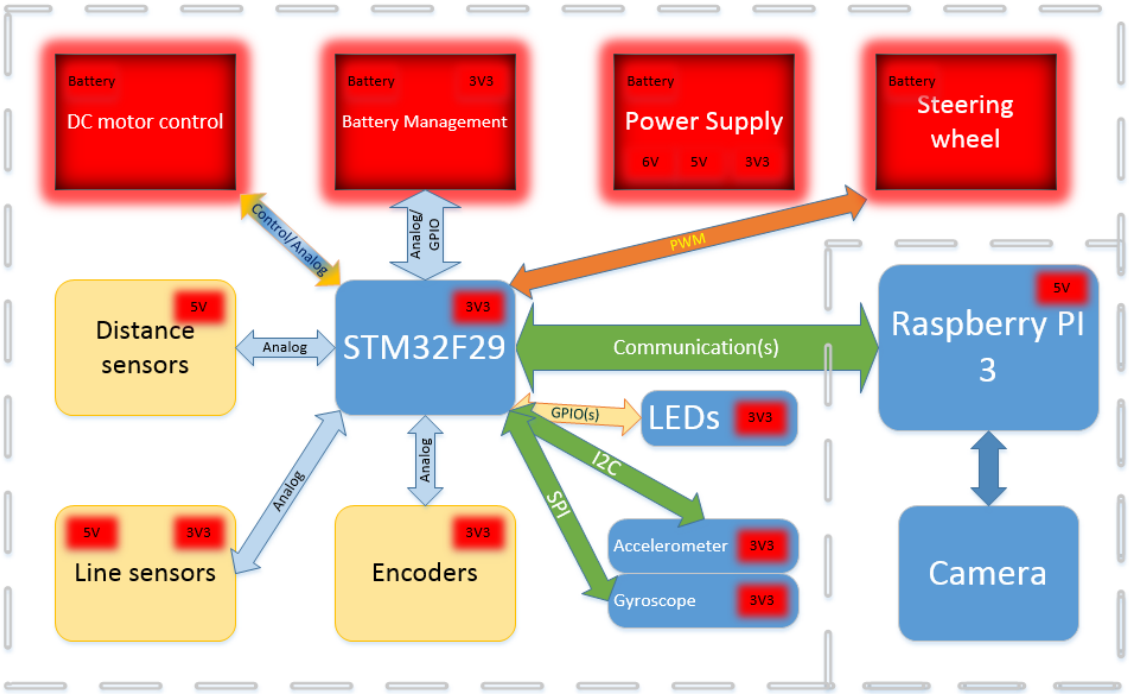
Sajnos a sztereó képfeldolgozást nagyon nehéz megoldani valós idejű rendszerekben, mert nagyon nagy a számítási teljesítmény igénye, de ennek ellenére, ha találunk egy optimális frissítési frekvenciát, mely függ a rendszertől, illetve annak használatától, akkor alkalmazhatjuk [46].

3 Rendszertervezés

Egy ilyen nagy bonyolultságú rendszerhez elengedhetetlen készíteni egy rendszertervet, amelyből már körvonalazódik az, hogy hogyan is képzeltem el az architektúra működését. A tervezésekor az átláthatóságot és a modularitást tartottam szem előtt. Ez a blokkdiagramm segített a feladatok, folyamatok párhuzamosításában, illetve, az egyes részek bonyolultsága is könnyebben megérthető. A Microsoft Visio nevű szoftverét teljesen alkalmasnak találtam erre a feladatra.

3.1 A rendszer blokk diagramja

A rendszer két nagy processzoros részre osztható (3.1. ábra), az első, amely az autó jeleinek feldolgozásával foglakozik, úgymint a vonalkövetés (IR), távolságérzékelés (IR), DC motor kontroll, míg a második a képfeldolgozásra fordítja a figyelmet. A modularitás miatt az utóbbi egység semmi máson nem fog dolgozni, így akár a későbbiekben nem csak szoftveresen, de hardveresen is lehet fejleszteni.



3.1. ábra: A rendszer blokkvázlata

Az elsődleges egység előállítja a szükséges tápfeszültségeket (3V3, 5 V, 6 V), amelyből 5 V-ra mind a kettő résznek szüksége van. A Raspberry PI 3-mat választottam a képfeldolgozáshoz, egyrészt azért, mert a kis mérete ellenére felveszi a versenyt a

néhány évvel ezelőtti asztali számítógépek teljesítményével, másrészt azért, mert meglehetősen kevés az áramfelvétele és mindamellett nagyon olcsó is [35].

Az elsőleges kontroller kiválasztásánál az volt a célom, hogy az STM32-es család legnagyobb processzorát használjam, amely a lehető legtöbb ADC-vel rendelkezik, illetve ezen technológia maximális órajelén üzemeltessem az MCU-t[10].

A modellautó irányításához elengedhetetlen a kormányszög, melyet két tényező határoz meg. Az egyik a kocsi első tengelyének a közepéhez képest, hol helyezkedik el ez a vonal, amit nevezzük ofszetnek (p). A másik, hogy ez a vonal milyen szöget zár be a kocsi orientációjával, tehát mennyire párhuzamos a robotautó és a szigetelőszalag, amelyet nevezzük egyszerűen a vonal szögének (δ).

A Raspberry PI elsődlegesen a szög (δ) és az ofszet (p) értékét küldi el STM32-es kontrollernek, amely megvalósulhat UART-on, illetve SPI-on. Az utóbbit elég mm-en pontosan megadni (3 byte), míg az előbbinél tizedes pontossággal (5 byte). A 3.1-es egyenlet segítségével nézzük meg mennyi bitet és mekkora sávszélességet kell a rendszernek biztosítani, hogy ezeket az adatokat feldolgozhassa az ST-s kontroller. A $Q_{LiveCounter}$ értékei 0-tól 999-ig terjedhetnek, tehát 3 byte szükséges a tovább küldéshez. A $Q_{Package}$ legyen 7 byte, amely a start, stop, illetve az elválasztó jeleket tartalmazza.

$$Q_{Sum} = Q_{Angle} + Q_{LiveCounter} + Q_{Offset} + Q_{Package} \tag{3.1}$$

A 3.1-es egyenlet illusztrálja, hogy SPI esetében, csak 144 bit-re van szükségünk amit, ha UART-on szeretnénk átküldeni, akkor 180 bit-re nő[11]. 100 ms-os frissítési idő mellett a sávszélesség 1,8 Mbps-os értéket éri el, azonban ez a szám nem lehet nagyobb, mint 45 Mbps, hiszen akkor az alsó kontroller nem tudná fogadni, nem hogy feldolgozni az adatokat. Ez körülbelül 10 FPS-nek megfelelő képfeldolgozást jelent, amely első ízben elég is lehet, hogyha kellően pontos adatokat kapunk/küldünk. Gondoljunk csak bele, amikor autót vezetünk, akkor nem minden esetben fokuszálunk a terelővonalra, néha muszáj a táblákra is, így az emberi jelfeldolgozás e része csökken.

Az akkumulátor menedzsmentet (battery managemnet) egy kis hatlábú PIC végzi, amely az elsődleges akkumulátor feszültség percepciója után 5 másodperc múlva

[10] 90 nm-es technológiával 500 MHz-es órajel is elérhető, de itt a maximum csak 180 MHz [43]

[11] A start a stop, illetve a paritás bitek megnövelik a sávszélességet

bekapcsolja a rendszer többi részét, vagyis a tápfeszültségeket. Erre két okból is szükség van, egyrészt, hogy a kocsi még véletlenül se induljon el azonnal, másrészt, ha véletlen a csatlakozónál nem sikerül a tökéletes kontaktot kialakítani. Sajnos az utóbbi következménye lehet, hogy a csatlakozó egy idő után „besül”[12], másrészt előfordulhat az az eset is, hogy még éppen kalkulál az ST-es MCU, de a valóságban az akkumulátor már nincs ott, de rövid időn belül ismét csatlakoztatva, azonnal elindul az autó.

A kommunikációt hardveresen meg sokszoroztam (UART, SPI, USB 1.1 és USB 2.0), hogy akár párhuzamosan vagy redundánsan is el lehessen küldeni az adatokat. Ezek az információk nem igényelnek titkosítást, hiszen csak vezetékes hálózaton keresztül férhetnének hozzá a robotautóhoz.

3.1.1 A kisebb teljesítményű kontroller

Az eredeti koncepciónál 3000 mAh-ás akkumulátorral körülbelül 10 perces rendszeridőt lehetett elérni, amely 15-20 A-es (átlag) áramfogyasztást jelent. A battery management feladata, hogy ezt az időt megsokszorozza, úgy, hogy a rendszer nem egy, hanem összesen 2-3 akkumulátorral rendelkezik, melyek kapacitásai eltérhetnek (3000-5000 mAh). Az intelligens szabályzások miatt, az üzemidő remélhetőleg eléri az egy órát, ami a teszteléseknél kulcsfontosságú.

A DC motor vezérlésre a Raspberry PI is alkalmas[13], bár kissé körülményes a pontos időzítés. Nem tartalmaz valósidejű órajel generátort (real-time clock)[14], illetve a feladatkezelőben a taszkok prioritását szem előtt kell tartani. Ezért, és a fentebb említett modularitás végett, a DC motor irányítását az STM32-re bíztam, amelyet egy P(ID) szabályzó fog irányítani. A motor fordulatszáma egyenesen arányos a kapocsfeszültségével, míg a nyomatéka négyzetesen függ az áramától. A szabályozónak szüksége van egy bemenetre, ami jelen esetben a motor árama és a kocsi sebessége. Előbbit a motorvezérlő IC szolgáltatja, míg az utóbbit az enkóder(ek). Alapvetően egy enkóder már elég ahhoz, hogy meghatározzuk a sebességet, kettőnél ha felhasználhatjuk a négyszeres kiértékelést, akkor a kalkuláció hibája a felére csökkenhet.

---

[12] Leginkább a nagy link kapacitások árama okozza

[13] Rendelkezik PWM modullal

[14] Wi-Fi-n keresztül frissíti az órajelét,

Az első tengelyen a kerekek szögelfordulását egy állapot visszacsatolásos szabályozóval irányítja a kontroller, míg az állandó sebességért a P(ID) szabályozó felelős. Mint minden nagyobb rendszerben, itt is lesznek diagnosztikák, amik a tápfeszültséget, a távolságérzékelő szenzorokat ellenőrzik. Az autó élesztése elején ezeket az információkat el is küldte a hozzá csatlakozó PC-nek. Az egyik legérdekesebb diagnosztika a kis szervóé, ami 6 V-os tápfeszültségről üzemel, de ha egy meghatározott feszültségszint alá süllyed (pl. 4,8 V), akkor a kerekeket nem tudja a kívánt szögben elfordítani, úgymond beragadt. Ebben az esetben a szervót a korábbi vagy a nulla állapotba kell visszaállítani. Ha ezt nem kellő gyorsasággal (néhány 20 ms alatt) vesszük észre, akkor az egész rendszer újraindulhat.

A vonal- és a távolság mérőszenzorok analóg jeleket állítanak elő, amelyeket szintén az STM32-es MCU ADC-i dolgoznak fel a megfelelő mintavételi frekvencia megválasztásával (lásd 5.2.4.1). A távolságmérő szenzorokat sokkal ritkábban fogja lekérdezni, mint a vonalszenzorokat, hiszen amíg 1 méter alatt elegendő kétszer megnézni, hogy van-e előttünk vagy mellettünk valamilyen akadály (a parkolási ciklust leszámítva[15]), addig a vonalszenzoroknak körülbelül 1-2 ms-onként megfelelő információt kell szolgáltatni, hogy a szabályozókör ne boruljon föl.

A diagramon még látszik egy LED-es blokk, amelyek a későbbiekben felhasználhatók az autóban a szokásos fényjelzésekre úgymint, a világítás, fékezés, irányjelzés, de a tesztelések során remek állapot- és hibavisszajelzők voltak.

A rendszerbe opcionálisan elhelyeztem egy gyorsulásérzékelőt, ami az enkóderekkel kooperálva javíthatja a kocsi pozitív, illetve negatívgyorsulási értékeit. Ezzel a megoldással elinduláskor a maximális nyomtatékot tudjuk ráadni a kerekekre, anélkül, hogy azok kipördülnének (ASR[16]), fékezéskor ezt megfordítva, ilyenkor ABS[17]-ként fog működni a szenzorfúzió. Tegyük fel, hogy az autó nyugalmi állapotban van. Ekkor a DC motor vezérlésével elindítjuk az autót, amely jelen esetben nulla sebességgel halad, mert a kerekek a nagy nyomatéktól kipördültek. A gyorsulás érzékelő értéke ekkor közel nulla értéket mutat, a sebesség érzékelő pedig már 2 $\frac{m}{s}$-t mér, körülbelül 100 ms

[15] A parkolási ciklusnál a centiméteres pontosság eléréséhez, gyakrabban kell mintavételezni

[16] Anti-Slip Regulation

[17] Anti-lock Braking System

alatt. Mondhatjuk, hogy egy sebesség ugrás jött létre, amelyet az erre felkészített szabályozó érzékel, így a nyomatékot csökkenti. Ez akkor lehet nagyon hasznos, ha apró kavicsokon, vagy éppen a vizes, havas, jeges felületeken próbálunk hirtelen sebességet változtatni.

Nem csak gyorsulás érzékelőt érdemes a rendszerbe implementálni, hanem egy giroszkópot is, amely megadja kocsinak az orientációját, amely főként a kanyarokban lehet hasznos, mert a gyorsulásérzékelővel ezeket az értékeket nem triviális meghatározni.

3.1.2 A képfeldolgozó egység

A Raspberry PI nem más, mint egy bankkártya méretű eszköz, amelyet Nagy-Britanniában fejlesztettek ki, hogy a tanulók játékosan elsajátíthassák az alapvető programnyelveket. Az első változata 2012-ben jelent meg a termékpalettán, amelyet hamarosan követett a 2-es (2015) és 3-mas (2016) verzió is. Mindössze négy év alatt a teljesítménye megtízszereződött. Az általam választott eszköz processzora négymagos, 1,2 GHz-es 64 bites SOC (Sytem On Chip). Rendelkezik HDMI kimenettel és számos perifériával (UART, USB, SPI, kamera), illetve egy 250 MHz-es VideoCore IV-es GPU-val.

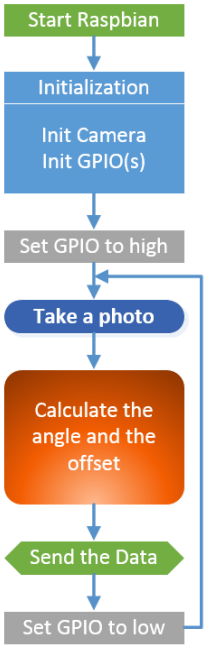
A Cortex-A53-mas (20 nm-es technológiával készült) processzor egyensúlyt biztosít a teljesítmény és az energiagazdálkodás között. Nem meglepő módon ezeket a processzorokat telefonokban is gyakran alkalmazzák a magas teljesítményük és az alacsony fogyasztásuk végett. Maga a RPI[18] 3 300 mA körüli fogyasztást generál tétlen állapotban (idle), azonban 100%-os CPU használat mellett (650 mA) és egyéb perifériákat kihasználva, ez az érték felemelkedhet 2,5 A-re is.

A szoftveres támogatása hasonló, mint az STM32-es processzornak, rengetek videóban és fórumban lehet szembesülni a saját kérdésünkkel. Az Open CV[19], ami egy nyílt forrású valósidejű képfeldolgozó könyvtár struktúra, mely lehetővé teszi, hogy a képfeldolgozási algoritmusokat könnyedén felhasználhassuk. Nem csak C, hanem C++, Python és még Java nyelven is elérhetőek ezek a függvények. Amint arról már korábbi

[18] Raspberry PI

[19] Open Source Computer Vision

fejezetben is szó esett, az éldetektáláshoz, a gradienskereséshez nagy segítséget nyújtanak egy előre megírt és letesztelt függvények. Az Open CV nem csak az RPI-n képes szoftveres támogatást adni, hanem a Windowson, vagy akár IOS-en [16].

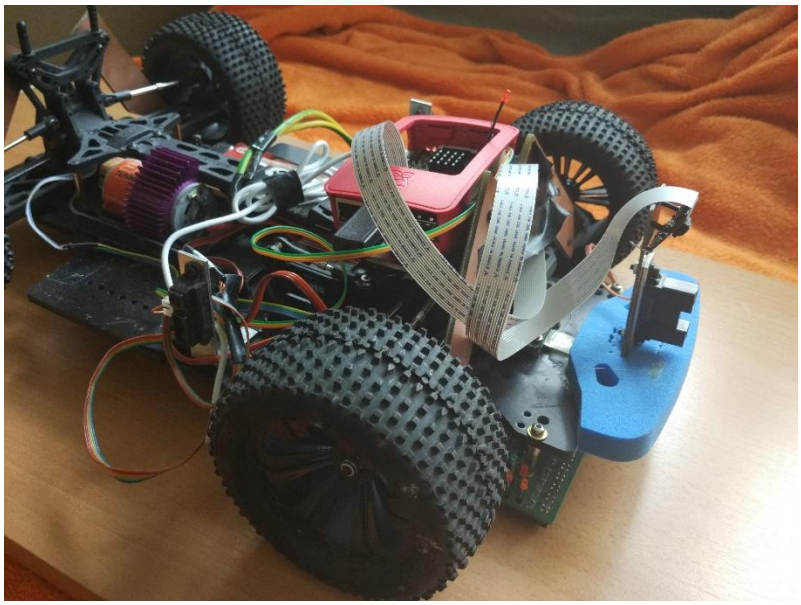


3.2. ábra: A Raspberry PI blokkdiagramja

Az elsődleges mikrokontroller tápfeszültség stabilizálása után elindul a Raspberry PI 3, amelyen egy Raspbian operációs rendszer fut. A Debian rendszer továbbfejlesztett változata, mely Linux alapú. A képfeldolgozó programot önműködően elindítja, amely inicializálja a kamera, illetve a kommunikációs interfészek portjait, ahogy ezt 3.2. ábra is illusztrálja. A fő ciklusban mindig egy monokróm képet készít, majd abból a képfeldolgozó függvények segítségével meghatározza a vonal ofszetjét, illetve szögét. Ahogy kiszámolta ezeket az adatokat, a kontroller elküldi az STM32-es MCU-nak, majd kezdi elölről a fő ciklust. Az diagrammon még látható két GPIO parancs, melyet a későbbi mérésekhez, visszajelzésekhez felhasználhatunk.

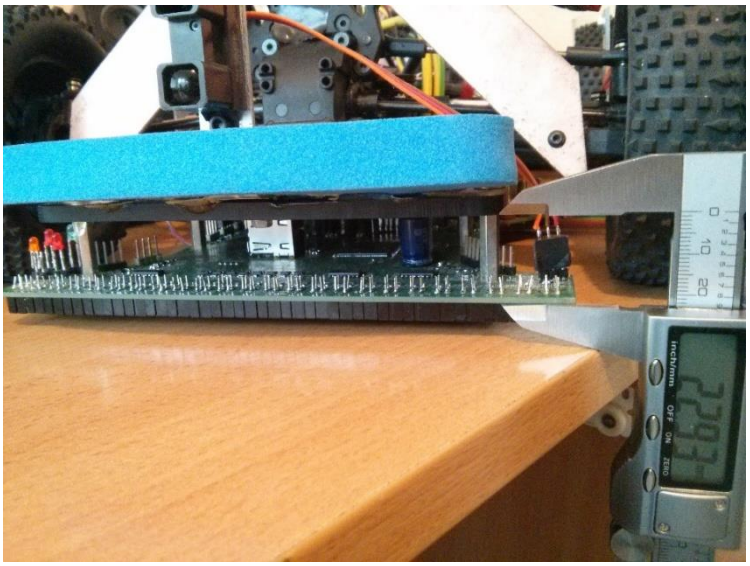
3.2 Mechanikai kialakítás

A rendszer tehát két részből fog állni, amelyeket el is kell helyezni az autón. A Raspberry PI 3 az első tengely felett foglalt helyet, míg a hozzá tartozó kamera a távolságszenzorral került egy konzolra, ahogy a 3.3. ábra mutatja.



3.3. ábra: Raspberry PI 3 és a kamera stádiuma

Az STM32-es NYÁK-nak a modell autó alján kell elhelyezkedni a vonalérzékelő szenzorok miatt. Első körben vizsgáljuk meg, hogy tényleg lesz-e elég hely a szenzoroknak, a motorvezérlőnek és mikrokontrollernek, hogy egy panelon helyezkedjenek el. A korábbi verzióban ez a feltétel már teljesült.

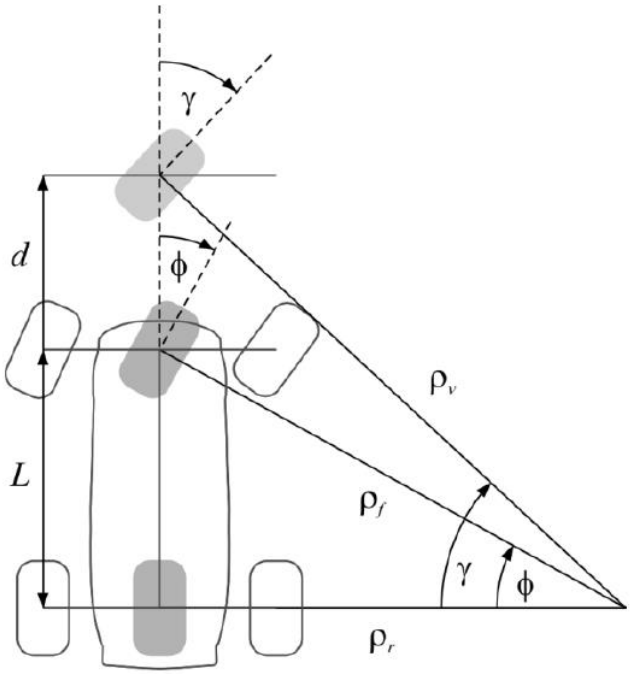


3.4. ábra: A bumper és NYÁK bottom oldalának a távolsága

Az IR szenzorok és az érzékelendő felület között, a beállított távolságnak 5 mm-nek kell lennie, ezért az autó gyári teleszkópjait el kellett távolítani és fixen rögzíteni (3.3. ábra). Maga a szenzor 6 mm magas, amihez még a PCB tipikus 1,6 mm-es magassága is hozzáadódik. A távtartó 20 mm magas, vagyis a NYÁK felső oldalán a legmagasabb

alkatrészek ettől csak kisebbek lehetnek. Az utóbbi elemek az 5 mm vastag műanyag „alvázhoz” (bumperhez) csatlakoznak, ami eredetileg az autónak a legalacsonyabb pontja volt. E pont és a futófelület között a távolság 37,6 mm. Amint a 3.4. ábra is prezentálja nem sok hely maradt a magas alkatrészeknek.

A kétszenzorsor alkalmazása esetén az eredeti kormányszöget meg tudjuk növelni, ha az első sort a kerekek elé helyezzük el, így már hamarabb érzékeljük a vonal paraméterváltozásait, miközben a hátsó sort az első tengely mögé célszerű letenni [39].



3.5. ábra: Szenzorsorok elhelyezkedésének hatása [39]

A 3.5. ábra ρ_f-el szemlélteti az eredeti fordulási sugarat, míg az újat a ρ_v-vel. Az egyenleteket felírva a *tan γ -ra*, és *tan Φ* –re, akkor a 3.2-es egyenlet adódik az összerendezésük után:

$$\tan\gamma = \tan\Phi \times \frac{L+d}{L} \tag{3.2}$$

A d-t próbáltam úgy megválasztani, hogy a szenzorok lehetőleg ne legyenek távol a rögzítési pontoktól, így ha az eredeti kormányzási szög 30° volt, akkor az új körülbelül 36 °-os lesz, ami 20%-os többletet jelent [39]!

Az előző modellben az akkumulátor csatlakozókat mindig alá kellett támasztani, hogy ne károsodjon a PCB. Ezek az alkatrészek úgymond „szorulnak”, így majd a panel tervezésnél figyelni kell arra, hogy a mechanikai erőhatásokat több furaton keresztül

egyenletesen osztodjon szét (háromszög módszer), különben a belső rétegen akár a réz fólia meg is repedhet/szakadhat. A további szenzorokat, programcsatlakozókat bonthatóvá és könnyen hozzáférhetővé kell tenni, ha cserélni vagy valamit módosítani kellene az autón. Az autó mozgásakor a vonalszenzorok z irányú mozgása meghamisítja a méréseket, melyek elkerülése érdekében, a lehető legtöbb helyre mechanikai rögzítéseket kell tenni (körülbelül 50 mm-enként).

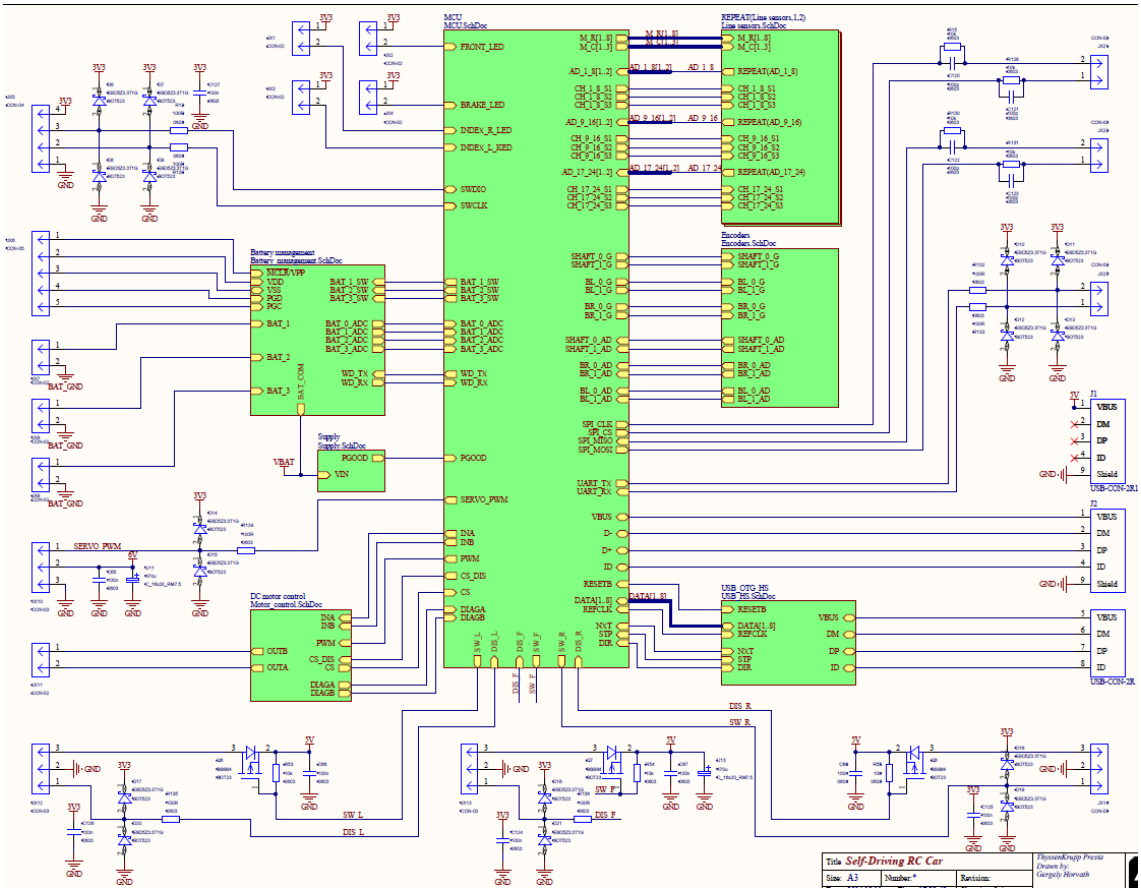
A mechanikánál még fontos megemlíteni a kormány összetartást, illetve, hogy a mechanika és az elektronika szinkronban legyen. Ha a kormány szervó mechanikailag nincs középen, akkor azt még szoftverből ki lehet korrigálni, de nem minden esetben. Ezért fontos beállítani a kormányösszetartást. Általánosságban elmondható, hogy az első kerekek dőlésszöge negatív, így kanyarodáskor a negatív dőlésszögű külső kerék nagyobb leszorító- és oldalerőt képesek felvenni, ami nagyobb kanyarsebességet eredményez. A hátsó kerekek dőlése semleges lehet, tehát párhuzamos az autó függőleges tengelyével, másrészt, ha ezeket nem állítjuk be, akkor különböző rádiuszon fordul meg az autó, ráadásul a kerekek még „nyolcas” alakban is járhatnak.

4 Hardvertervezés

A hardvertervezést az Altium Designer CAD szoftverrel készítettem el, amelyben felhasználtam a beépített SVN modult, hogy verziónként követhető legyen a fejlesztés. A blokkdiagramm logikáját megpróbáltam átemelni a kapcsolási rajz készítésekor, amint azt a későbbiekben is látjuk.

4.1 A kapcsolási rajz

A kapcsolási rajzot hierarchikusan rajzoltam, tehát az egyes egységeket (pl. motorvezérlő) egy-egy külön oldalon ábrázoltam, a könnyebb érthetőség végett, ahogy ezt a 4.1. ábra is mutatja.



4.1. ábra: A kapcsolási rajz blocksheet oldala (függelék [2])

A blocksheet oldalon találhatók a be- és kimeneti csatlakozók, melyek a kritikus pontokon ESD védelemmel biztosítottam a feszültségimpulzusok ellen (bővebben lásd 4.1.9). A távolságérzékelő szenzorokat egy-egy P csatornás FET-tel lehet vezérelni, hogy

ezzel is elősegítsem az rendszer hosszabb működésének időtartamát, mivel a három szenzor átlagos fogyasztása az 5 V-ról 100 mA.

Jobban megnézve a kapcsolási rajz fő oldalát, észrevehető, hogy az egyik USB csatlakozó (J1), adatbuszaira nincs kötve semmi, amely a layout miatt szükséges. Ez egy duplamagas USB csatlakozó, amelyen keresztül a Raspberry PI 3 az 5 V-os tápfeszültséget kapja meg.

4.1.1 A tápfeszültségek

A rendszer működéséhez négyféle tápfeszültség szükséges, amelyből csak az akkumulátor feszültségét nem kell átkonvertálni a DC motor vezérléshez. A kormányszervó irányításához körülbelül 6 V kell, nélkülözhetetlen az 5 V a Raspberry PI-nek, a multiplexernek, távolságérzékelő szenzoroknak és opcionálisan az USB-s interfésznek. A legtöbb komponens azonban a 3,3 V-os feszültségről üzemel úgymint, az STM32, az enkóderek, a vonalszenzorok, a giroszkóp és a gyorsulásérzékelő is. Végezzünk egy gyors számolást, hogyha minden perifériát használunk, akkor mennyi lehet az áramfelvétel az egyes feszültségszinteken, ahogy ezt a 4.1. táblázat bemutatja.

6V: 2200 mA		**5V: 2702,12 mA**		**3V3: 345,165 mA**	
Kormányszervó [9]	2200 mA	Raspberry PI[20] [17]	2500 mA	Gyorsulásérzékelő [21]	0.165 mA
		Multiplexerek [20]	0,12 mA	Giroszkóp [25]	5 mA
		USB[21] [18,19]	52 mA	Vonalszenzorok [10]	120 mA
		Távolságérzékelők [14]	150 mA	Enkóderek [10]	120 mA
				STM32F4[22] [24]	100 mA

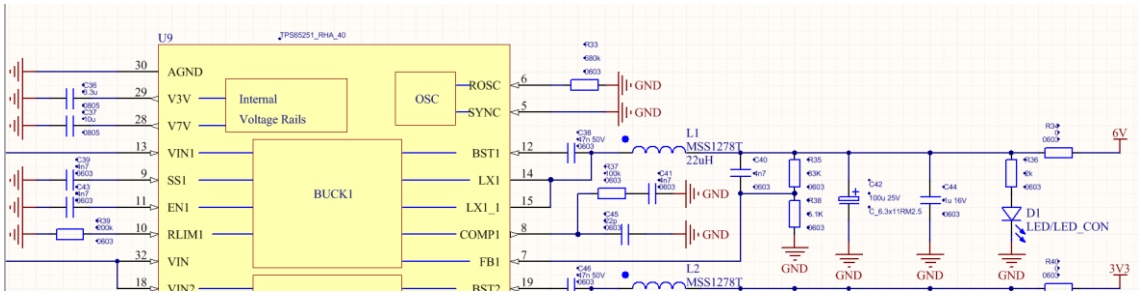
4.1. táblázat: A perifériáknak becsült áramfelvétele

Ha a tápfeszültségek előállításának hatásfoka 90%-os, akkor az akkumulátorból 7,2 V-os névleges feszültség mellett 4,25 Ampert fogyasztunk (27,8 W), amihez természetesen még hozzáadódik a DC motor áramfelvétele is (pl. 10A) (72W). Mindösszesen néhány W-ra vagyunk a 100W-os határtól, ha mindent maximálisan kihasználunk, amely csak néhány pillanatig állhat fent!

[20] Ha nem használjuk az USB-ket (pl. külső SSD), akkor 1200 mA lehet a maximális áramfelvétel

[21] Az USB-nél nem számoltam bele az esetleges töltőáramot

[22] Az STM32F4-re vonatkozó adatok csak becslések



4.2. ábra: A 6 V-os tápfeszültséget előállító IC BUCK 1 kapcsolása

Az MSC-s önálló laboratórium során megismertem a TI-os TPS65251-es IC-t amit ki is választottam a tápfeszültségek előállítására [23]. Három különálló tápfeszültség kimenettel rendelkezik, amelyek külön-külön kapcsolhatók (enable), időzíthetők (soft-start), illetve rendelkeznek túláram-, és hőmérsékletvédelemmel is. A 4.2. ábrán látható a kapcsoláson egy feszültszintet visszajelző LED, illetve egy 0 Ω-os ellenállás is, diagnosztikai célokra. A PGOOD egy open drain kimenetű láb, amely 85% fölötti hatásfok esetén logika 1 értékű, de ha ez csökken, akkor jelzi az MCU-nak egy lefutó élel. Ez a funkció hihetetlenül hasznos, mert mindegyik tápfeszültség elvileg kiskéseltetéssel ugyan, de közel egyszerre fognak elindulni, illetve még nagyban függ a kimeneti pufferkondenzátorok nagyságától a bekapcsolási idő. Az MCU csak akkor kezdi meg az inicializálásokat, ha a PGOOD logikai egy szinten van, illetve a tápfeszültségek a névleges működési tartományon belül vannak (10%).

A Buck 1 (3V3) folyamatosan 3 Ampert képes szolgáltatni, míg a Buck 2 (5V), illetve a Buck 3 (6V), csak 2 Ampert, míg a csúcsáramuk 0,5 Amperrel több, amellyel lefedtem a fentebb említett 27,8 W-os teljesítménycsúcsot.

4.1.2 A DC motor vezérlése

Az ST-s motorvezérlő IC már 5,5 V-tól képes a DC motorunkat vezérelni, melyhez összesen 5 jelre van szükség. Egy a PWM, kettő az irányitás és még kettő a diagnosztika/engedélyezés. A ki- és bemeneti feszültségtartomány 3V CMOS kompatibilis, ami a mi rendszerünkbe teljes egészébe megfelel. A 16 kHz-es jel impulzus szélességével állíthatjuk be az autó sebességét. A kitöltési tényező 0%-tól egészen 6,4%-ig nem fogja elindítani a kocsit, mert a maximális fel- és lefutási ideje az alsó FET-nek 4 µs. Ugyanez az érték korlátozza a 100%-os teljesítményt is. Az IC fogyasztása elenyésző a maga 6 mA-ével, ahhoz képest, hogy 30 Ampert tud vezérelni (folyamatosan) a

kimenetén. Az IC-nél kettő darab 1000 µF-os kapacitást is találhatunk, hogy a feszültséghullámosságot a motorvezérlő IC közelében redukálhassuk.

4.1.2.1 A motorvezérlő IC disszipációja

Az integrált FET-ek tipikus ellenállása (R_{DSon}) 18 mΩ. A motor árama körülbelül 10 A, ha 1-1,5 $\frac{m}{s}$-os sebességgel halad az autónk, de 5 $\frac{m}{s}$-nál elérheti[23] a 20-22 A-et is (I_{AVG}). Az átlagos disszipáció értéke 1,8 W, ahogy ezt 4.1-es egyenlet eredménye prezentálja a behelyettesítést követően.

$$P_{FETs} = R_{DSon}\, I_{AVG} \tag{4.1}$$

Az inverterhídba integrált FET-ek belső ellenállása 25 °C-nál még csak 18 mΩ, de 150 °C-on ez az érték 38 mΩ-ra változik, ami reális, hiszen ökölszabályként elmondható, hogy a FET-ek ellenállása 125 °C-on a fentebb említett érték (18 mΩ) közel kétszerese.Tételezzük fel, hogy a robotautót a maximális (5 $\frac{m}{s}$-os) sebességgen üzemel, amikor a disszipáció értéke nem több mint 15,2 W (P_{DMax}), ha módosítjuk a 4.1-es egyenlet paramétereit.

Ezt a hőmennyiséget már nehezen lehet eldisszipálni a PCB-n, de a megfelelő layout segítségével kivitelezhető. A sebesség növekedésével, egy pozitív hatás is hozzájárul a disszipáció csökkentéséhez, a menetszél. 20 $\frac{km}{h}$-nál ez körülbelül 5,55 $\frac{m}{s}$, amely olyan, mintha egy ventilátorral hűtenénk a panelt, illetve az IC-t. Az IC junction to case adatai 1,7 és 3,2 $\frac{°C}{W}$. Az előbbi a híd felső FET-jéhez, míg utóbbi az alsóéhoz tartozik. A folyamatos előrehaladás közben a felső FET állandóan bekapcsolt állapotban van, míg az alsó félvezetőt a PWM kitöltési tényezőjével kapcsolgatjuk. A NYÁK-kal körülbelül 8 $\frac{°C}{W}$-os hővezetést tudunk megvalósítani. Ha felrajzoljuk a helyettesítőképet, akkor a két érték (1,7 és 3,2) a NYÁK-kkal sorba, majd ezután egymással párhuzamosan kapcsolódnak. Ekkor a junction to ambient értéke (R_{JA}), körülbelül 5,2 $\frac{°C}{W}$. Az IC tok hőmérséklete nem haladhatja meg a 150 °C-ot (T_{JMax}). A junction hőmérséklete nyilván több lehet, de mivel az adatlap semmit nem írt róla, ezért ezzel az értekkel érdemes számolnunk a 4.2-es egyenletben.

[23] A motor áramat egy lakatfogóval mértem, amit különböző sebességeknél leolvastam.

$$T_{AmbientMax} = T_{JMax} - R_{JA}\, P_{DMax} \tag{4.2}$$

Az autó disszipációja azért nem kritikus, mert menetszél nélkül sem lépi túl az IC maximális megengedhető belső hőmérsékletét, ha a környezete nem lesz melegebb, mint 70,9 °C. Szobahőmérsékleten használva teljesíti ezt a feltételt. Ha logikusan végiggondoljuk, akkor az 5 $\frac{m}{s}$-os menetszél nem éri 100%-osan az IC-t, illetve a PCB-t, mert az előbbinél ott lesznek a link kapacitások, az utóbbinál viszont a szenzorok. Így csak egy elméleti becslést tudunk arra adni, hogy mekkora lehet a maximális környezeti hőmérséklet. Ha 100%-os menetszéllel számolhatunk, akkor ez az érték 131 °C, de 20%-os esetén még mindig csak 107 °C. A standard NYÁK hőmérséklet nem lépheti túl a 140 °C-ot, különben delaminálódik [3][27].

4.1.3 Szervókormány vezérlése

Mint korábban már említettem a digitális szervót maximálisan 333 Hz-el lehet irányítani, de a valóságban nem ez határozza meg az áramfelvételét, hanem az általa leadott nyomaték nagysága. A táp IC 6 V-os feszültségének maximális árama 2,5 A, de a digitális szervó csúcsáramfelvétele elérheti akár a 4 A-t is. A szükséges energiát a pufferkondenzátorból nyeri, melyet úgy kell megválasztani, hogy kapocsfeszültsége egy perióduson belül (pl. 10 ms), ne csökkenjen 0,5 V-nál többet. Ilyenkor a beállási idő és a nyomatéka változik, de nem olyan drasztikus mértékben hiszen, 4,8 V-ról is képes működni. Olyan szervóegységet próbáltam választani, amit a tápfeszültségről is üzemeltethetek (7,2 V), a baj csak az volt, hogy amikor az akkumulátorom teljesen feltöltött állapotban van, akkor a kapocsfeszültsége elérheti akár a 8,4 V-ot is. A legnagyobb tápfeszültségű szervó amit találtam, maximum 7,4 V-os feszültséget tolerált.

$$It = Q = CU \tag{4.3}$$

Az 4.3-mas egyenletet átrendezve, illetve a paramétereket kitöltve nézzük meg, hogy mekkora kondenzátorra lesz szükség. Az áramerőssége a fent említett 4 A, míg az áramtüske szélessége 1 ms, addig a C kapacitás értéke legyen 4700 µF. A pillanatnyi nagy fogyasztás miatt a tápfeszültségen körülbelül 0,8 V-os lesz a hullámosság, ahogy a 4.4-ös egyenlet eredménye mutatja. Ezt az energiát 2 ms alatt visszatölti a kondenzátorba a szabályozó IC, ha 2,5 A-es maximális árammal számolunk.

$$\Delta U = \frac{It}{C} \tag{4.4}$$

Az elsődleges gond, hogy a pufferkondenzátor magassága a tűrésekkel együtt meghaladja a maximális 20 mm-t (22 mm). Az 5 mm-es tartót ugyan még ki lehetne vágni (süllyeszteni), de ehelyett inkább alkalmazzuk azt a megoldást, hogy elfordítjuk 90°-al. Ekkora áramcsúcsoknál érdemes megnézni, hogy a kondenzátor mennyire melegszik, illetve, hogy az élettartama, hogyan alakul. Az előbbi az ESR-től függ ($R = 17\ m\Omega$), míg utóbbi az áramhullámosságtól (ripple current) [26].

$$P = I^2R \tag{4.5}$$

A 4.5-ös egyenlet alapján, a disszipáció itt sem jelentős (0,272 W), de ha jobban megnézzük az adatlapban a current ripple értékét, akkor 3,3 A-t olvashatunk le 105 °C-on. Így a kondenzátor élettartama redkuálodni fog, de nem 105 °C-os környezeti hőmérsékleten használjuk, hanem jellemzően szobahőmérsékleten. Egyébként az alkatrészek élettartama a hőmérséklet növekedésével csökken (körülbelül 10 °C-onként megfeleződik). [26]

A probléma a kormányszervónál akkor léphet fel, ha nem tudja elforgatni a kerekeket, mert túl nagy a felületi tapadás, vagy éppen egy falra erőhatást gyakorol. Sajnos semmilyen védelemmel nem látták el a szervót, ezért a tápfeszültséget előállító IC-t használtam fel arra, hogy tájékoztassa az STM32-es MCU-t, ahogy már erről korábban is szó volt.

4.1.4 Az analóg és digitális szenzorok

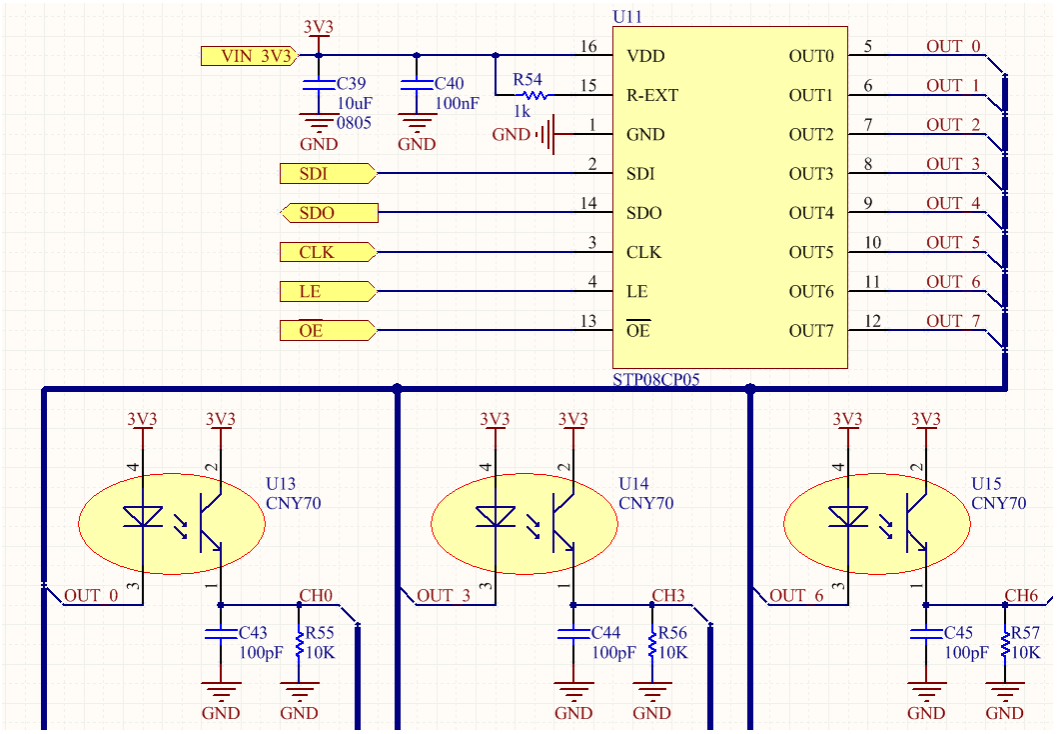
A fentebbi pontokban említést tettünk az autó külső jeleinek a méréséhez szükséges szenzorokról, melyek részben analóg és részben digitális komponenseket tartalmaznak, de alapvetően az utóbbi van majoritásban.

4.1.4.1 A vonalérzékelő szenzorok

A ROHM RPR-220, amely 4,5x6,4 mm, tehát a felbontásunkat növelni tudjuk a korábban alkalmazott szenzorhoz képest (7x7 mm). A távolság és kollektor áram karakterisztika csúcspontja 7 mm-nél található, míg a CNY70-es esetében ez 0,5 mm. Sajnos a tipikus kollektor árama gyengébb 0,3 mA, mert az eredeti szenzornál 1 mA. Jobban megnézve egy másik görbét, ha a LED áramerőssége 20 mA és 6 mm az érzékelési magasság, akkor a kollektor árama ~0,6 mA. Az előző karakterisztika miatt nem ezt választottam, de érdekes lehet a modell továbbfejlesztéséhez, a jobb felbontás miatt.[11]

A CNY70-es szenzor nagyobb testvére, a TCRT5000, amely 10,2x5,8 mm, tehát itt a felbontásunk redukálodik. Az előző szenzorhoz képest dupla akkora, de ami miatt mégis érdekes, hogy a kollektor árama 1 mA, illetve a karakterisztika csúcspontja 2,5 mm-es távolságérzékelő pontba esik. A felbontást nem akartam lecsökkenteni, másrészt nincs rajta védőburkolat, ami fontos, ha egy emelkedőn megyünk fel, akkor a szenzorok ne érjenek le, vagy ha mégis, akkor sem sérülhetnek meg az fényérzékelős felületek. (A CNY70-esnél 1 mm-es a védőtest) [13]

Az egyik legérdesebb szenzor az (TT) Optek OPB733TR, 7,6x4,1 mm, vagyis a felbontásunk ismét növelhető, illetve az adatlap szerint az optimális távolságérzékelés tartománya 10 és 25 mm közötti, de a szokásos karakterisztikát megnézve a tetőpont körülbelül 3,8 mm-nél van. A kollektor árama az adatlapban bizonytalanul van megadva. A normalizált áram értéket nem tartalmazza, úgyhogy a karakterisztikákból kiszámolva a csúcspontnál 0,35 mA, ami nagyon kevés. A fel- és a lefutási időknél a kollektor áram 1 mA[24][12].



4.3. ábra: CNY70-es áramgenerátoros vezérlése

[24] Az adatlap egy apróbetűs részre hivatkozott, hogy tervezett, de nem valós szám az 1 mA.

A hiányzó adatok, illetve a CNY70-es szenzor korábbi ismeretei miatt, evvel az optikai alkatrésszel terveztem meg az IR vonalkövetést. A mozgás megfelelő kontrollálásához elengedhetetlen feltétel, hogy a 19 mm-es vonalat detektáljuk. A két soros vonalszenzor miatt összesen 48 darab CNY70-es optikai alkatrészt kell vezérelni, de sajnos ennyi szabad ADC csatornája nincs a mikrokontrollernek, illetve nem szükséges folyamatosan bekacsolni a szenzorokat. Használjuk valamilyen multiplexeres megoldást, ahogy azt már korábban is tettem. A megoldást 4.3. ábra szemlélteti, ahol a 8 IR komponenst áramgenerátoros shiftregiszterrel vezéreljük. E kapcsolás előnyei közé tartozik, hogy összesen egy darab ellenállásra volt szükség ahhoz, hogy a 8 darab LED egyforma áramerősséggel működjön (azaz azonos fényerősséggel világítson). A maradék 5x8 darab LED áramerősségeit, csak az ellenállások gyártási szórásai befolyásolták[25] (± 1%).

A megoldás hátránya, hogy nem lehet tetszőleges időben megváltoztatni a CNY70-es szenzor állapotát, illetve a disszipáció is csak egy helyre korlátozódik, amely körülbelül 400 mW, ha az adott shift regiszter összes LED-ét 20 mA-el bekapcsoljuk. Az IC footprint-je nem tartalmaz thermal pad-et, ami megnehezíti a hőelvezetést, de szerencsére ez az érték még nem kritikus ($0{,}335\ W$), ha megnézzük a 4.6-os kalkulációt, ahol U_{LED} 1,205 V[26].

$$P_D = 8(U_{3V3} - U_{LED})I_{20} \tag{4.6}$$

Ezzel a megoldással 6 shiftregiszter IC-re, 6 ellenállásra volt szükség, illetve 10 GPIO lábra, hogy megfelelően vezéreljük a vonalszenzorokat.

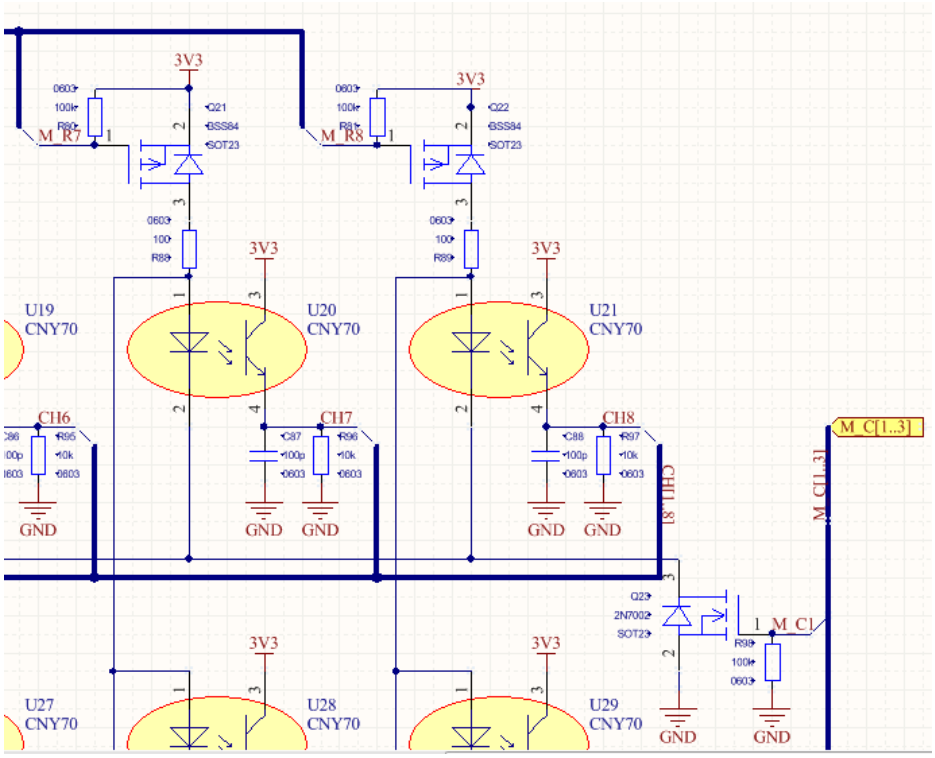
A második generációs megoldással el tudjuk érni, hogy bármelyik időpillanatban bármelyik LED-et be-, illetve ki lehessen kapcsolni. A mátrixos LED vezérlést a 4.4. ábra jeleníti meg, amihez összeégésében 2x11 darab tranzisztor kell. A kapcsolás egyik hátránya, hogy nem áramgenerátorosan vezérli az optikai egységeket, vagyis a tápfeszültség (3V3) változásával az áram is egyenesen változik. Azért, hogy ez ne következzen be nagymértében[27], arról a tápfeszültséget előállító IC gondoskodik (±2%).

---

[25] Tekintsünk el az egyéb paraméterektől, amit a worst-case számításoknál alkalmaznak

[26] 20 mA-nál digitális multiméterrel mérve

[27] Így körülbelül csak 0,6 mA-t változik a LED-ek áramerőssége.



4.4. ábra: CNY70-es tranzisztoros áramgenerátoros vezérléssel

A kapcsolás másik hátránya, hogy ha több LED-et akarunk egyszerre bekapcsolni, egy azonos ágban (pl. Q21 tranzisztornál), akkor az áramerősség a bekapcsolt LED-ek számával arányosan megváltozik[28]. A kapcsolás azonban tökéletesen alkalmas ehhez a vonaldetektáláshoz, mivel a két szenzorsor 24-24 optocsatolót tartalmaz, amelyből egyszerre csak 3-3 darabot kell bekapcsolni. Ennek következtében az első és a hátsó szenzorsor egyszerre vezérelhető az alábbiak szerint: elsőként az első, a kilencedik és a tizenhetedik LED fog világítani, másodikként a második, tízedik és a tizennyolcadik. Értelemszerűen az utolsó, azaz a nyolcadik periódusban a nyolcadik, tizenhatodik és a huszonnegyedik infra szenzor kapcsol be. A szenzorok a beolvasáskor így elkerülhetjük az átvilágítást, hiszen egymást zavarnák, illetve mérést is meghamisítanák a mi nézőpontokból. A nagyobb frekvenciás zavarok elnyomása érdekében (> 1 MHz) a CNY70-es tranzisztorok kimenetén egy-egy aluláteresztő szűrő található.

[28] Kettőnél felére, háromnál harmadára redukálódik

Az áramkör tervezésekor figyelembe kell venni, hogy a tranzisztor a kollektorán keresztül mennyi áramot képes átengedni az adott fényerősség mellett. A korábbi mérések azt bizonyították, hogy ha a fekete vonalat 5 mm-es távolságból érzékeljük, akkor a kimeneti feszültség 3,3 V körüli, míg egy barna vagy fehér parkettán ez a feszültség érték közel 300 mV. Ezek az értékek szoftveres szempontból megkönnyítik a dolgunk, hiszen rendkívül távol van egymástól a két tartomány, illetve a jel-zaj viszony is sokkal kedvezőbben alakul. Az optikai komponensekből effektíven 6 darab folyamatosan be lesz kapcsolva, ha végiggondoljuk az előző vezérlést, akkor ez 120 mA-es áramfelvételt jelent.

A 48 darab vonalszenzor értékét 6 multiplexerrel olvassuk vissza, mint a korábbi RobonAUT-os versenyen. Az IC csatornánkénti ellenállása 5 V-on körülbelül 20 Ω, míg 3 V-on akár 150 Ω is lehet, de mivel itt csak a mérőáram folyik rajta keresztül, ezért elhanyagolható a rajta eső feszültség (ami körülbelül 100 µV).

4.1.4.2 Sebességérzékelő szenzorok

A CNY70-es szenzorokat enkóderként is felhasználhatjuk, hogy a kocsi sebességét megmérjük. Az IR működését kihasználva, a motor tengelyére felragasztottam egy kört, mely alapterülete 6-6 darab fekete és fehér (egyenletes) körcikkekből állt. A motor forgása következtében a szenzor kimenetén egy „négyszögjelet" látunk az oszcilloszkópon (7.2. ábra), amit az STM32-es MCU feldolgoz. Hasonló módon a kerekek belső felére is fel lehet ragasztani egy vízálló fekete fehér csíkot. A rendszer összesen hat CNY70-es szenzort használ, melyek tápfeszültségei kapcsolhatók, az áramfogyasztás optimalizálásához, ahogy ez észrevehető a 4.5. ábrán. A bekapcsoló FET-ek gate-source-ához egy 100 kΩ-os ellenállást kapcsolódik, ezzel elkerülve, hogy vezérlés nélkül instabil állapotban legyenek a tranzisztorok. Az áramkörben minden egyes bekapcsolásakor lejátszódik ez a jelenség, mivel az STM32-es MCU-t inicializálni kell, továbbá, ha a NYÁK-on elszakadna ez a vezeték, akkor sem kapcsolna be a FET. (A PCB-n a lehető legközelebb kell elhelyezni ezeket alkatrészeket.)

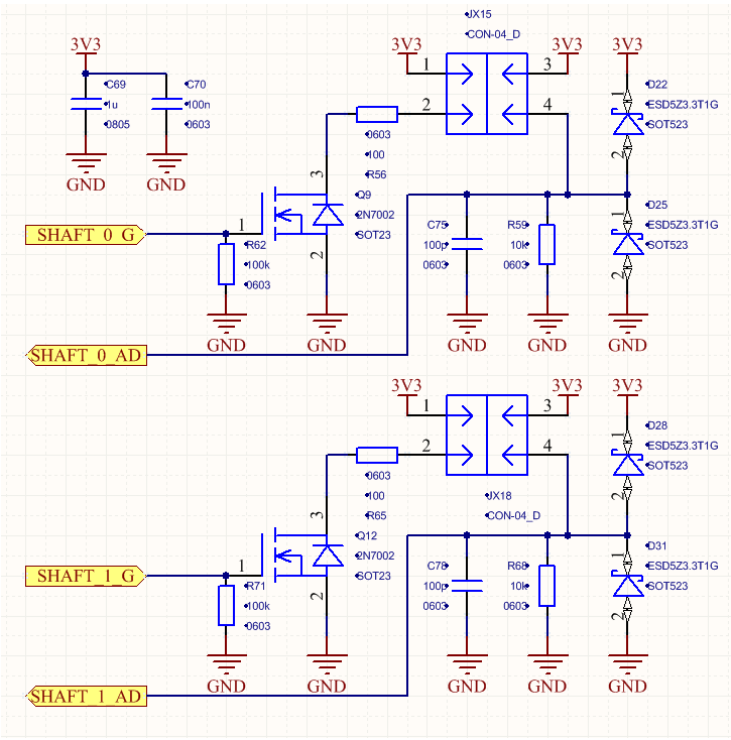
Tegyük fel, hogy a kocsi maximális sebességgel halad, amikor a kerekek maximális fordulatszámát a 4.7-es, 4.8-as és a 4.9-es egyenletekkel határozhatjuk meg. A kerek névleges átmérője 110 mm, illetve q a felbontás, melynek értéke 7 mm.

$$n_{max} = \frac{V_{max}}{2\pi r} \tag{4.7}$$

$$F_k = \frac{K_{Kerék}}{q} = \frac{D\pi}{q} \tag{4.8}$$

$$f_{max} = F_k * n_{max} \tag{4.9}$$

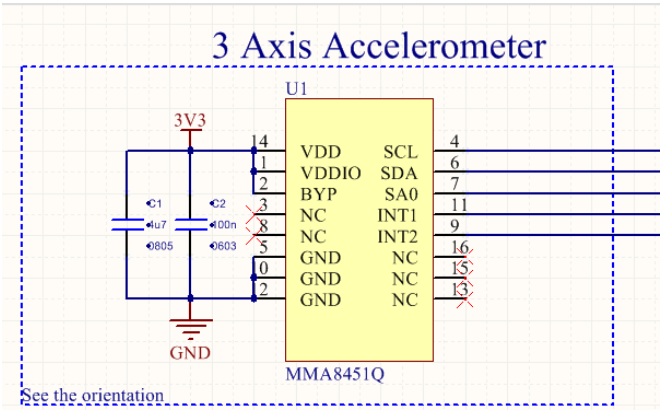
A maximális frekvencia (a 4.9-es egyenlenből) körülbelül 800 Hz, amely 1,25 ms-os periódus időt jelent, ha a kerekekre ragasztanánk fel a fekete fehér jelöléseket.



4.5. ábra: Enkóder bekötése

4.1.4.2.1 A gyorsulás érzékelő

Egy háromtengelyes gyorsulásérzékelő is helyett kapott kapcsolási rajzon (4.6. ábra), amely az adatokat I^2C-n keresztül küldi el, a megfelelő időzítésekkel. A IC regisztereit úgy is be lehet állítani, ha egy adott irányban elér egy gyorsulási értéket, akkor megszakítást küld az MCU-nak. Egy előre meghatározott pillanatban az MCU generálhat interrupt-ot az IC-nek, ha szeretné kiolvasni a gyorsulási értékeket, majd az érzékelő standby állapotba kerül [21]. A gyorsulás érzékelő kapcsolási rajzát a 4.6. ábra szemlélteti.

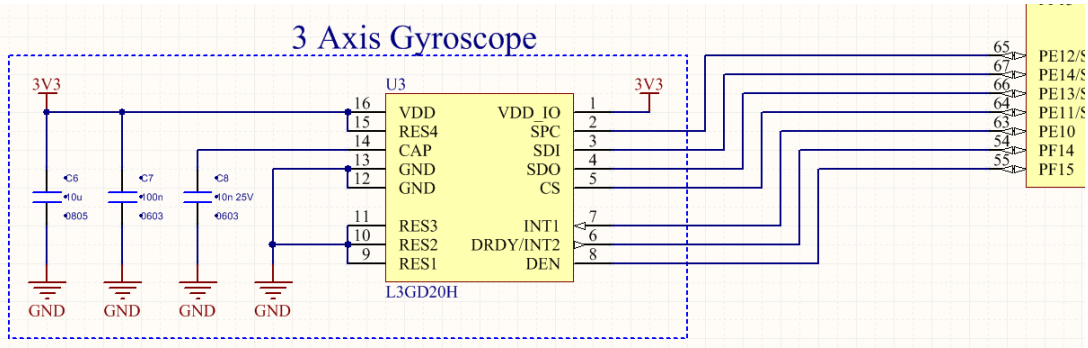


4.6. ábra: Gyorsulásérzékelő

A gyorsulásérzékelőt főként az elindulásoknál ASR-ként, illetve a lassításoknál ABS-ként lehet használni, minden egyéb esetben standby módban üzemel.

4.1.4.2.2 A giroszkóp

A giroszkóp segít az orientáció meghatározásában, amelyek főként kanyarodáskor jelenthet nagy segítséget. A gyorsulásérzékelőből is meg lehetne határozni, de az integrálás miatt veszítünk az információ értékéből. Ezért inkább ezt a szenzort használtam, melynek a kapcsolási rajzát a 4.7. ábra reprezentálja [25].



4.7. ábra: Giroszkóp kapcsolási rajza

4.1.4.3 Távolságérzékelő szenzorok

A rendszer három darab analóg távolságérzékelő szenzorral rendelkezik, a távolságok méréséhez, de ahogy arról már korábban is szó volt, az alkatrésznek visszahajló karakterisztikája van. Nézzük meg a 2.10. ábrát, ahol 1 V-os feszültséget két pontban is mérhetünk: 5 cm-nél és 63 cm-nél. A karakterisztikát illeszteni kell az STM32-es MCU-val, amelyet egyszerűen meg lehet valósítani, ha néhány pontban linearizáljuk. A szenzorokat függőlegesen kell elhelyezni a kocsin, ellenkező esetben veszítenek a pontosságukból. A konverziós idejük 39 ms, ezért alacsonyabb frekvenciás aluláteresztő

szűrőt tervezhetünk[29] az STM32-es bemenetére. A három szenzor maximális együttes áramfelvétele 150 mA.

4.1.5 Kommunikációs interfészek

A kontrollernek még maradt néhány szabad lába, amiket opcionálisan kivezettem, ezért négyféle kommunikációs interfésszel kapcsolódhatunk az STM32-es MCU-ra: UART, SPI, USB 1.1 és USB 2.0. Nyilvánvaló, hogy az USB-t lehet a legnagyobb távolságra elvinni, míg az SPI-t 30-40 cm-nél távolabb semmilyen körülmények között nem vinném[30]. Ha a Raspberry-n kívül egy másik eszközzel is szeretnénk összekötni a kontrollerünket, akkor azt tetszőlegesen megtehetjük, ha a DMA-nak még marad szabad erőforrása.

4.1.5.1 UART

Az UART maximális sebessége 5,62 Mbit/s, ami többszöröse a korábban említett minimális (1,8 Mbps-os) adatátviteli sebességhez képest. Mindegyik interfész közül ez a legegyszerűbb, illetve a legolcsóbb, hiszen mindösszesen csak az Rx és a Tx vezetéket kell fordítva összekötni a másik eszközzel. A két MCU 3,3 V-ról üzemel, de a jeleket mégis illeszteni kell egymáshoz, amit a 4.1.8-es alfejezetben részletesebben olvashatunk. Normál működés közben erre nem feltétlen lenne szükség, de a kaszkád hibákra is jobb előre felkészülni.

4.1.5.2 SPI

Az SPI körülbelül tízszer gyorsabb, mint az UART, a maga 45 Mbit/s-os sebességével. A 4 vezetékes összeköttetést akár 3 vezetékre is redukálhatjuk, ha kijelöljük, hogy melyik kontroller lesz a Master, mert amikor az órajelét elindítja, akkor rögtön elkezdődik az adatcsere. Még ha elenyészően is, de ez kedvezőbb hatással van az az áramfogyasztásra. A maximális sebességnél az SPI hullámhossza 6,66 m. Ökölszabályként elmondható, hogy a hullámhossz 10%-áig nem (feltétlen) kell foglalkozni a reflexiókkal, de a jelillesztést végig kell gondolni, mert azon a ponton megváltozik az impedanciája.

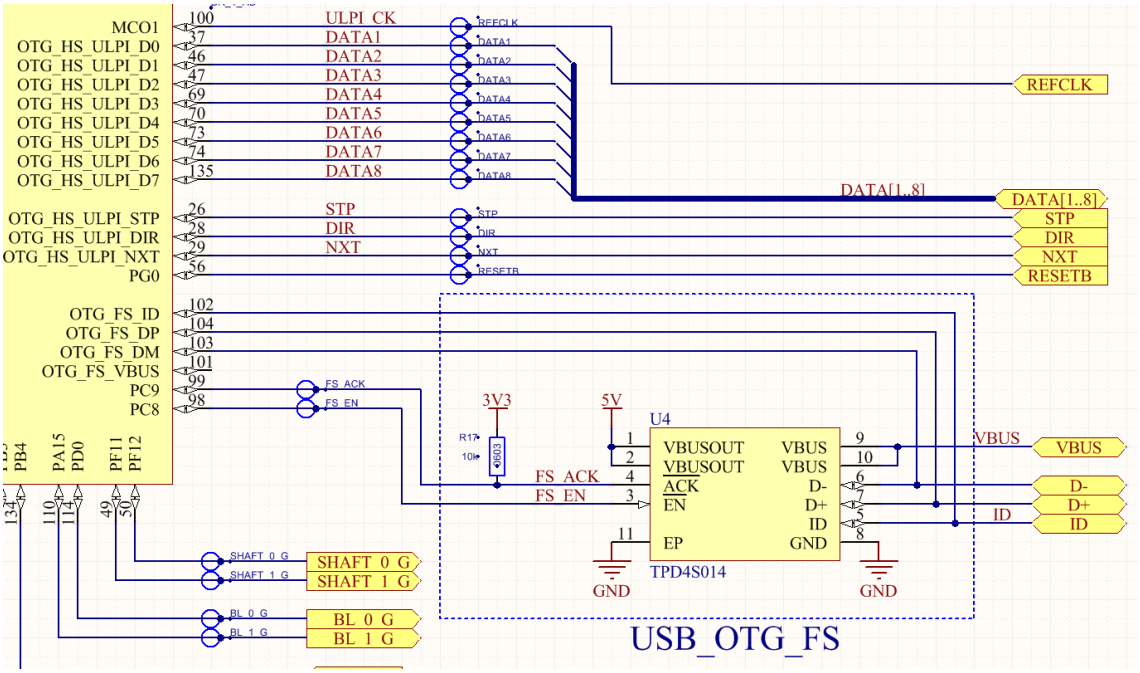
---

[29] Jelen esetben a vágási frekvencia 15,91 kHz

[30] Többszörös jelintegritással működhet!

4.1.5.3 USB 1.0 és 2.0

Az STM32-es kontroller mind a kettő USB változatot támogatja, míg a kisebb 12 Mbit/s, addig a 2.0-ás 480 Mbit/s-ra képes. Ez előzőhöz egy kis illesztő IC-t tettem, amely az 5 V-ról üzemel és 1,5 A-es töltőáramot biztosít az USB-s eszköznek. Akár elérhetjük azt is, hogy egy külső hordozható adattárolót kössünk össze és időbélyegesen rögzítsük az adatokat. Talán érdekesebb a továbbfejlesztett USB, hogy hogyan érjünk el 480 Mbit/s-os adatsebességet, amikor a kontroller maximális órajele 180 MHz.



4.8. ábra: USB 1.0 és USB 2.0 az MCU-n

Az adatokat nem sorban, hanem egy 8 bit széles adatbuszon keresztül, egy transceiver IC alakítja sorossá, amelyhez egy 60 MHz-es órajel szükséges. Ezt az MCU állítja elő, ahogy azt a 4.8. ábra is jelöli, az ULPI_CK felirattal. A pontos időzítések miatt kellett alkalmazni egy külső 16 MHz-es kvarc kristályt, egyébként elegendő lett volna az MCU belső R-C oszcillátora is.

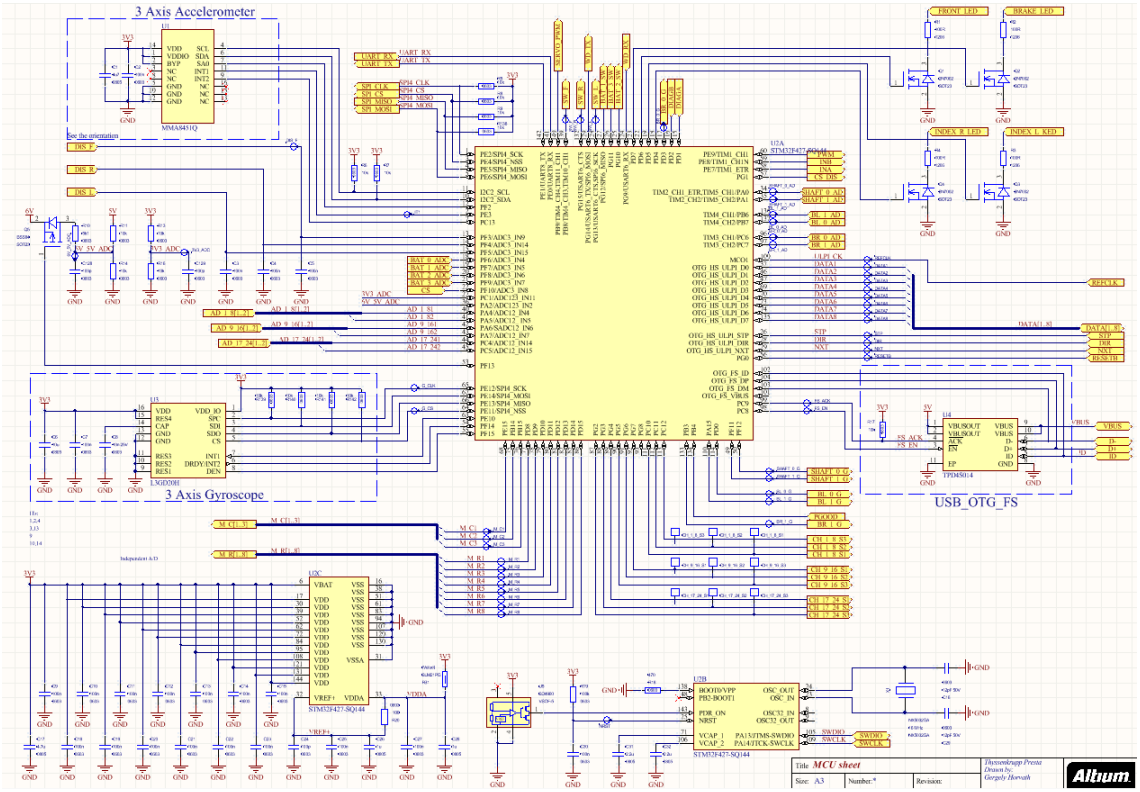
4.1.6 STM32F427 kontroller

Az STM32F4-es mikrokontroller családban csak egy eszköz rendelkezett külön analóg földeléssel, illetve három független ADC modullal. Soroljuk fel milyen analóg feszültségeket kell megmérni:

- 48 darab analóg vonalszenzort
- 3 darab távolságérzékelő szenzort

- 1 db DC motoráramot
- 6 darab tápfeszültséget

Összesítve 58 darab analóg feszültséget kell feldolgozni a kontrollernek, amely nem rendelkezik ilyen sok kivezetéssel. Eredetileg 24 analóg bemeneti csatornából, csak 16-ot ált rendelkezésemre, miután a nagy sebességű USB interfész ezekre a GPIO-kra csatlakozott. A mérendő jeleket multiplexálva, illetve a három ADC modult maximálisan kihasználva, meg tudjuk mérni az összes analóg jelet. A 4.9. ábra bal felső részén a Q5-ös P csatornás FET-el mintavételezhetjük az 5 és 6 V-ot. A referencia feszültség a 3,3 V-ból egy ferrit bead-en és egy aluláteresztő szűrőn keresztül valósul meg. Az analóg feszültségeket nem szükséges csak pár mV pontosan mérni, ezért nem láttam értelmét egy plusz referencia feszültségforrás használatának.



4.9. ábra: STM32-es MCU kapcsolási rajza (függelék [2])

Az STM32F427ZI 144 lábú tokozással rendelhető, maximálisan 180 MHz-ig használható a Cortex M4-es processzormag 2 MB flash memóriával, amely bőven elegendő a szoftver megírásához. Az STM32-es processzor családnak csak 3,3 V-os tápfeszültségre van szüksége a normál működéshez, ellentétben más processzorcsaládokkal, amelyeknél külső alkatrészekkel kell biztosítani a

magfeszültséget (például 1V2). A nem használt perifériákat tetszőlegesen lehet konfigurálni, így a lehető legoptimálisabb fogyasztást érhetjük el, miközben a kapacitási teljesítményből semmit sem veszítünk. Két DMA egységgel rendelkezik, amely közül az első, főként a kommunikáció lebonyolításában segít, míg a második, az analóg jelek automatikus beolvasását végzi, amit így nem a processzornak kell végrehajtani [24].

A CPU-t az előző STM32F4-es discovery kit-el összekötve serial wire debuggerként alkalmazva (SWD) programozható és debuggolható. Az SWD interfészéhez összesen 3 GPIO-t (SWDIO, SWCLK, NRESET) kell kivezetni az MCU-ból, ami kevesebb, mint a JTAG-et a maga 6 vezetékével. Persze itt még nem számoltuk bele a tápfeszültséghez szükséges további két csatlakozást [31].

4.1.7 Akkumulátor menedzsment

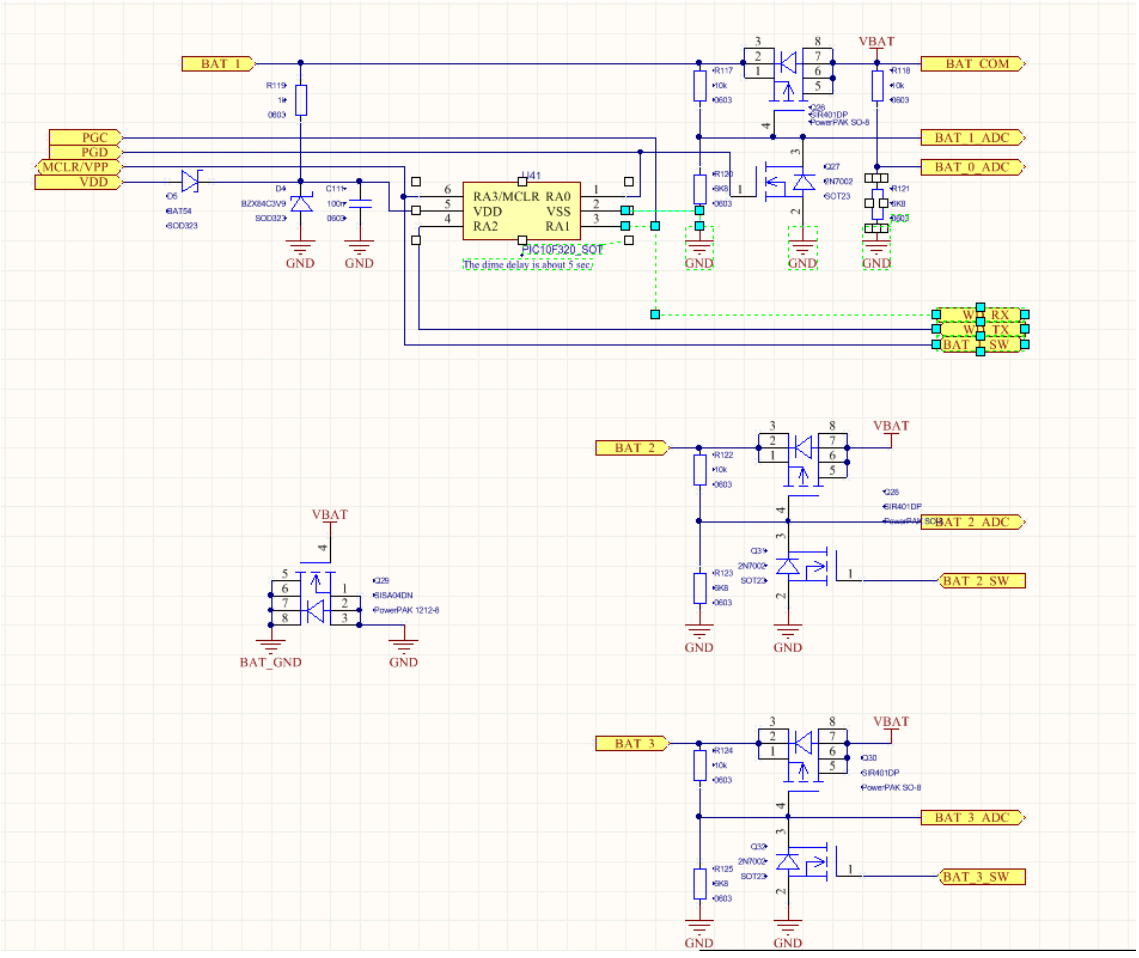
Az akkumulátor menedzsment a három feszültségforrásos rendszer felügyeletét végzi, illetve gondoskodik arról, hogy az első akkumuláttelepet egy 5 másodperces szoftveres időzítéssel kapcsolja a rendszerhez, amely egy biztonsági funkció.

Első körben hardveresen próbáltam megoldani az időzítést, azonban 5-7 komponensből tudtam volna egy olyan megoldást kialakítani, amely körülbelül 4-5 másodperc után kapcsolta volna be a 4.10. ábrán látható FET-eket[31]. A kondenzátor töltődése miatt szükség lett volna egy komparátorra, amely egy bizonyos feszültségszint fölött ugrásszerűen bekapcsolja a félvezetőket, amely egy pillanat alatt feltölti a nagy pufferkondenzátorokat, elkerülve ezzel a szikrázást az akkumulátor csatlakoztatása közben. A kapcsolás hátránya főként az akkumulátortelepek átkapcsolásánál jelentkezett, mivel valahogy le kellett volna választani a komparátor kimenetét, vagy az időbeállító kondenzátort kisütni. Az utóbbi, azért nem előnyös, mert ha hirtelen vissza akarjuk kapcsolni, akkor meg kell várni, míg a kondenzátor ismét feltöltődik (4-5 másodperc). A 4.10. ábra kapcsolása mindössze 3-4 alkatrészből áll, és tetszőleges időzítést lehet vele meg valósítani, ráadásul sokkal precízebben.

Az akkumulátor a katódja BAT1-hez kapcsolódik, míg az anódja a BAT_GND-hez. A fordított polaritás elleni védelmet egy P csatornás teljesítmény FET-tel valósul meg (Q26), amely ilyenkor záró irányba feszíti elő a body-diódáját. A maximális gate-

[31] A Q26, Q27, Q29 FET bekapcsolása szükséges

source feszültsége nem haladhatja meg a ±10 V-ot, amely szintén teljesül a 7,2 V-os akkumulátoroknál. Normál üzemi körülmények között a BAT1-hez kapcsolódó 1 kΩ-os ellenálláson keresztül folyik az áram egészen a GND-ig, ahol az N csatornás FET-en a body-diódáján keresztül az elektronok visszajutnak az akkumulátor anódjához. A PIC 3,9 V-os tápfeszültségét egy Zener-dióda garantálja, majd a beállított időzítés leteltével bekapcsolja a 2N7002-es N csatornás tranzisztort, amely majd bekapcsolja a P csatornás FET-et. Ebben a pillanatban a BAT_GND és a GND között az N csatornás FET gate feszültsége felugrik a tápfeszültségre, így a bekapcsolás pillanatában a kondenzátorok feltöltődésekor a kapcsolási veszteség minimális lesz. (Ellenkező esetben a diódákon egy pillanatra megnövekedne a disszipáció.)



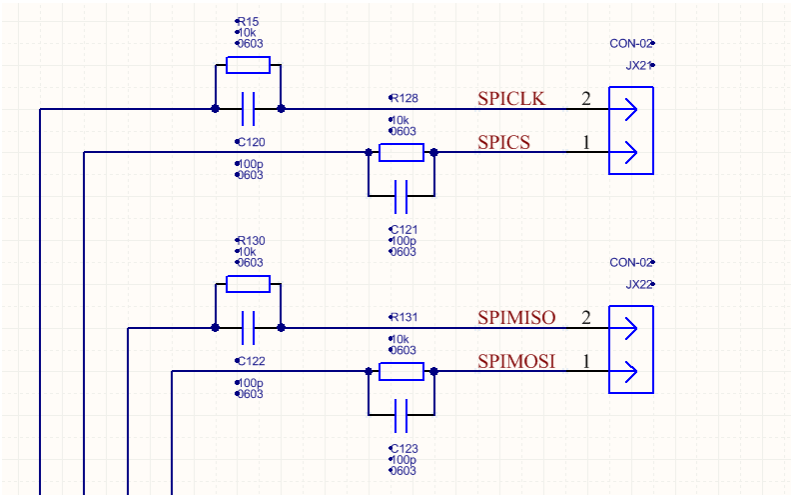
4.10. ábra: Az akkumulátor menedzsment kapcsolási rajza

A bekapcsolás után, ha az első akkumulátorról át akarunk kapcsolni egy másikra, akkor azt jelezni kell a PIC RA3-mas lábán. Ezt egy megszakítással generáljuk, melynek hatására megváltoztatja a Q27-es FET állapotát, majd alvó üzemmódba kerül. Az

energiafogyasztása minimális, illetve a layout helyigénye is (SOT-363[32]). Ha a költségeket nézzük, akkor a PIC még mindig olcsóbb, mint egy több komponensű hardveres megoldás. A PIC programozása az autó bekapcsolt állapotában is lehetséges lenne, ha a D6-os diódán kívül, még egy soros 1 kΩ-os ellenállást is tartalmazna a PIC és a BAT1_SW között.

4.1.8 Jelszintek illesztése

A Raspberry PI tápfeszültségét (5 V-ot) a táp IC állítja elő az akkumulátorból. Az UART, illetve az SPI kommunikációnál eltérő potenciál különbségek lehetnek (3V3, 3V3), amelyek feleslegesen terhelik az áramköröket. Ennek elkerülése érdekében az adatvonallal sorosan használjunk egy párhuzamos R-C kört, ahogyan a 4.11. ábra mutatja. A nagyfrekvenciás komponenseket a kondenzátort átengedi, míg DC szempontból az ellenállás minimalizálja az áramerősséget (néhány nA-re). A legrosszabb eset, amikor az STM32-es MCU megpróbál kommunikálni a másik eszközzel, aminek a tápfeszültsége még 0 V. Ekkor az adott komponens, a body-diódáin keresztül elkezd feltöltődni. Ezen felül még előfordulhat olyan eset, hogy a két MCU kivezetése kimenetre van felprogramozva, amitől szintén tönkremehetnek az MCU-k. A 45 Mbps-os SPI-nál még figyelni kell arra, hogy az R-C kör ne vigyen bele túl nagy jelterjedési időt, mert akkor hamis SPI üzeneteket generálunk. A leghosszabb vezeték esetében ez az érték nem lehet nagyobb, mint 6,66 ns.

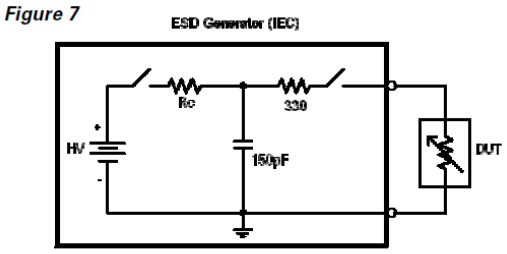


4.11. ábra: Feszültségszintek illesztése

[32] A SOT-363-s footprint körülbelül 2x2,1 mm alapterületű

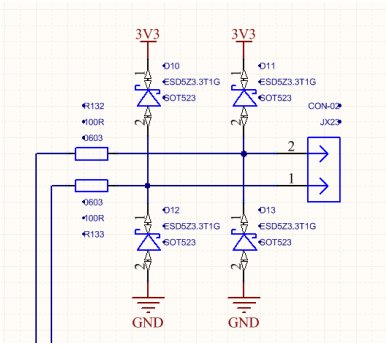
4.1.9 ESD védelem

Nagyon fontos, hogy az áramkört felkészítsük, az ESD[33] védelemre. Az emberi test kapacitása nagyon eltérő, de általánosságban elmondhatjuk, hogy az értéke 150 pF. A „kondenzátor” feszültsége elérheti a 8 kV-ot is, amelyet már egy átlagos IC nem tud károsodás nélkül elviselni, ezért az átlagos 2 kV alatti értéken kell tartani.



4.12. ábra: ESD generátor helyettesítő modellje [30]

Amint a 4.12. ábra illusztrálja az ESD generátor helyettesítőképét, vagyis az emberi test kapacitásának egyik modelljét. Az 330 Ω-os ellenállás leolvassa után rájöhetünk, hogy nem csak a feszültség, de az áramerősség is tönkreteheti az áramkörünket. Akár 30 A-es áramerősség is előfordulhat 1 ns-os ideig, 100 ns után ez az érték alig néhány 100 mA-e csökken. A NYÁK-on a tervezésnél kiemelkedően fontos, hogy milyen messze helyezzük el a TVS diódákat a csatlakozótól, melyek szimbólumait a 4.13. ábra prezentálja. A panelon a réz vezeték, úgy viselkedik, mint egy induktivitás, mely az IC kivezetéséig megsokszorozhatja a feszültséget. A további védelem érdekében alkalmazhatunk soros ellenállásokat, de úgy, hogy az adatátviteli sebességeket, illetve feszültségszinteket, reflexiókat csak kismértékben befolyásolja [30].



4.13. ábra: Az UART ESD védelme

[33] ElectroStatic Discharge

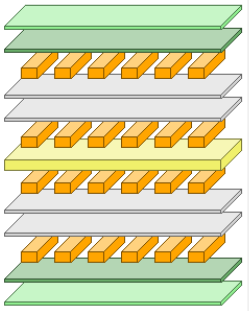
4.2 A NYÁK terv

A NYÁK tervezést szintén az Altium Designer CAD programmal végeztem el. A layout-ot két részre oszthatjuk fel, úgymint, teljesítményelektronika, és analóg/digitális részek. Az előbbihez az akkumulátor menedzsment, illetve a motorvezérlő IC-s részek tartoznak, utóbbihoz pedig a „nagy" sebességű kommunikációs adatvonalak (USB, UART), illetve az analóg feszültségek visszamérései.

A PCB az autó alatt fog elhelyezkedni, amelyet körülbelül 11 mm-re kell rögzíteni a kerekek futófelületétől. A rajzolati osztályt elsősorban az alkatrészek footprintje határozza meg, amely így a 6-os osztályba esik[34], tehát a vezetékek távolságai, illetve szélességei 0,15 mm-nél (6 mil) nem lehetnek kisebbek. A furatozási osztályt pedig a C kategóriába választottam, így a legkisebb (kész) furatátmérő 0,25 mm (10 mil). A motorvezérlő IC hőelvezetését segítette volna, ha nagyobb réz vastagsággal rendelem meg a NYÁK-ot (például 105 µm), de a disszipációs kalkuláció és később a valóság is igazolta, hogy elegendő a standard 35 µm-es rézvastagság. [28]

4.2.1 A NYÁK rétegfelépítése, dimenziói

Az áramkör bonyolultságát, illetve az adatkommunikációk sebességét, impedanciáját figyelembe véve, egy standard négyrétegű NYÁK megfelelő erre a célra. Két rétegen az impedanciákat sokkal nehezebb tartani, hiszen nincsenek kijelölt föld síkok vagy csak egy. A 4.14. ábra egy négyrétegű panel belső felépítését mutatja, ahol a TOP és a Mid Layer 1 távolsága 0,360 mm, míg a Mid Layer 1 és Mid Layer 2 közötti távolság körülbelül a duplája, azaz 0,71 mm.



Layer Name	Type	Material	Thickness (mm)	Dielectric Material	Dielectric Constant	Pullback (mm)	Orientation
Top Overlay	Overlay						
Top Solder	Solder Mask/Co...	Surface Material	0.04	Solder Resist	3.5		
Top Layer	Signal	Copper	0.018				Top
Prepreg1	Dielectric	Prepreg	0.18	FR-4	4.8		
Prepreg2	Dielectric	Prepreg	0.18	FR-4	4.8		
Mid Layer 1	Signal	Copper	0.035				Not Allowed
Core	Dielectric	Core	0.71		5		
Mid Layer 2	Signal	Copper	0.035				Not Allowed
Prepreg1	Dielectric	Prepreg	0.18	FR-4	4.8		
Prepreg2	Dielectric	Prepreg	0.18	FR-4	4.8		
Bottom Layer	Signal	Copper	0.018				Bottom
Bottom Solder	Solder Mask/Co...	Surface Material	0.04	Solder Resist	3.5		
Bottom Overlay	Overlay						

4.14. ábra: Standard 4 rétegű NYÁK rétegfelépítése

[34] Az Eurocircuits Kft osztálybesorolási táblázata alapján

A rétegfelépítés, illetve a főbb jelcsoportok a következőképpen alakultak:

- TOP: Nagyfrekvenciás jelek
- Mid Layer 1: GND és AGND
- Mid Layer 3: Tápfeszültségek (3V3, 5 V, 6 V, akkumulátor(ok))
- BOTTOM: Analóg jelvezetékek, tesztpontok

A két belső rézréteg 35 µm vastagságú, míg a két külső eredetileg 18 µm-es, de a galvanizáció miatt szintén 35 µm-esek lesznek.

A két soros vonalszenzorok alapvetően meghatározzák a NYÁK fizikai dimenziót, míg egymástól körülbelül 150 mm-e helyezkednek el, addig egy sorban 24x7 mm-et foglalnak el (168 mm-t). A kerekek maximális elfordulásakor sem érhetnek hozzá a PCB-hez, amelynek szélei lekövetik ezt a sugarat (függelék [2]). A NYÁK fizikai dimenziói végül 158,4 mm x 190 mm-es értékekkel rendelkeztek.

A panel összesen 8 darab M3-as csavarhoz való 3,2 mm-es furatot tartalmaz, melyeket próbáltam úgy elhelyezni, hogy az erőhatások egyenletesen érjék a NYÁK-ot. A szenzorsorok két végénél található 1-1, míg középen, a kerék tengelyével párhuzamosan szintén 1-1, illetve a motor és USB csatlakozónál is észrevehetők a rögzítések (függelék [2]).

4.2.2 Alkatrészek elhelyezése

A lehető legtöbb alkatrészt a TOP oldalra helyeztem el, mert a BOTTOM oldalon mindenfajta szöszöket, illetve koszokat szedhet össze, ami ugyan nem befolyásolja az áramkör működését, de esztétikai szempontból (számomra) fontos.

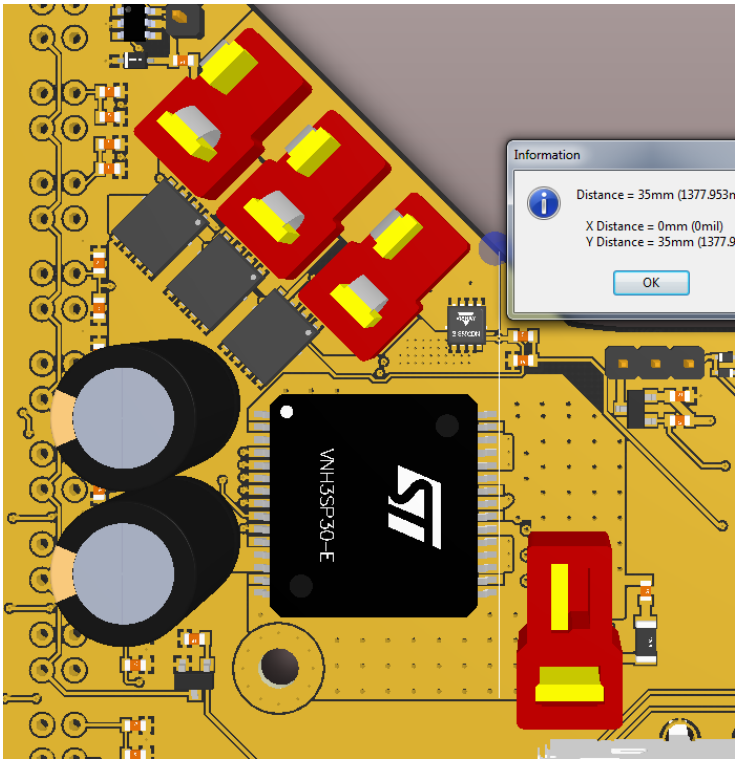
A másik szempont a csatlakozókhoz hozzáférhetősége, különös tekintettel az USB-kre. Az USB mini A, illetve B felületszerelt csatlakozók, amelyet egyikét sem lehetett volna úgy elhelyezni a panelon, hogy az ne ért volna hozzá a szenzorokhoz vagy a kerekhez. Ennek elkerülése végett alkalmazhatunk 2 soros USB csatlakozót, amelynél figyelembe kell venni, hogy a ház előtt körülbelül 5 cm-es távolságban egyetlen alkatrészt sem helyezhettünk el, különben az alsó USB-t nem tudjuk felhasználni. Másrészt azt ott lévő alkatrészek egy idő után leszakadtak volna vagy a bosszantóbb, ha éppen eltörik a forrasztás, és amikor az autó mozog, akkor vagy érintkezik, vagy nem. Nem is egy, hanem kettő ilyen csatlakozóra volt szükség, mert külön kábelen kell megtáplálni a Raspberry

PI-t. Erre nem egy dedikált tápcsatlakozót használtam (például NEB21R), hanem az előbb említett USB-t, mert így felhasználhattam a hozzá kapott USB kábelt.

4.2.2.1 Teljesítményelektronikai szempontok

4.2.2.1.1 Motorvezérlő IC

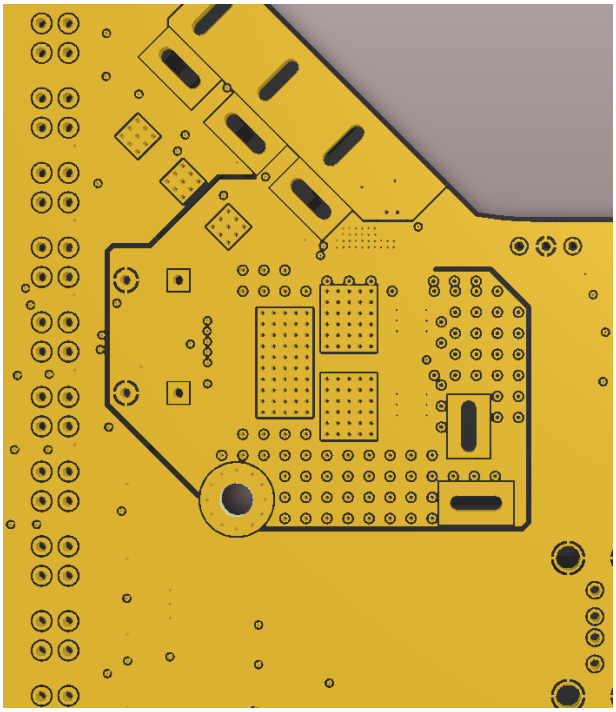
A három akkumulátor csatlakozónak egészen a NYÁK szélére kell kilógnia, hogy könnyen hozzáférhetővé váljon. Talán a legnagyobb erőhatást ez a három egység gyakorolja a panelra, amelyet minél közelebb kell elhelyezni egy furathoz, hogy a mechanikai erőhatásokat ne a panelre terjesszük ki. Ezek a csatlakozók több százszor lesznek mozgatva az első beüzemeléstől kezdve, másrészt ahhoz, hogy az áramhurok minimális legyen az akkumulátor és a DC motor között, a lehető legközelebb kell elhelyezni egymáshoz. A 4.15. ábra mutatja, hogy sikerült 35 mm x 45 mm-es területen összezsúfolni a teljesítményelektronikát, ami kisebb, mint a korábbi verziónál.



4.15. ábra: Teljesítményelektronikai rész 3D nézete (TOP)

Ha megnézzük, az ST-s IC környezetét, akkor egy nagyobb réz réteg át van viázva, hogy a jobb hő átvitelt biztosítsunk a belső rétegek és az alsó oldal felé. A poligon pour-ok nem csak a felső (top), de a többi rétegen is ugyan így szerepelnek, az áram és a hőelvezetés céljából. A belső rétegen a föld síkot ketté osztottam, hogy a motor árama ne

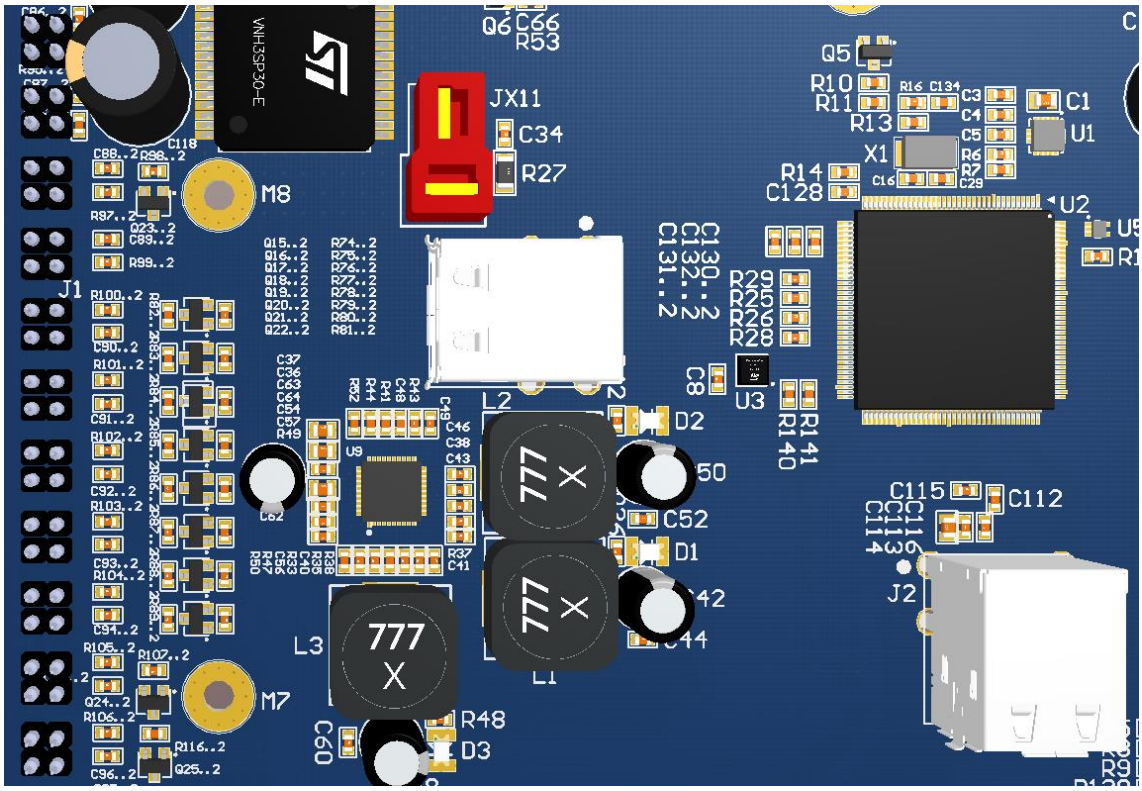
befolyásolja az analóg mérést. Ha 15 A-es a motoráram és a réz ellenállása 10 mΩ, akkor máris egy 150 mV-os ofszetet viszünk a mérésünkbe. Ezt elkerülhetjük, ha végiggondoltan tervezzük meg az áramutakat. Így csak a belső kialakított GND szigeten folyik ez az áram, míg a táp IC-hez szükséges 1 A, nem az analóg szenzorok mellett, hanem a másik oldalon folyik, ahogy a 4.16. ábra szemlélteti. Így hosszabb lesz a GND hozzávezetés a táp IC-hez, de kevésbé befolyásolja az analóg szenzorokat.



4.16. ábra: A GND elválasztása belső rétegen

4.2.2.1.2 Tápfeszültséget előállító IC

A panel további teljesítményelektronikai részéhez közel kell elhelyezni a tápfeszültséget előállító komponenseket, amely így biztosítja a kis áramköri hurkokat. A 4.17. ábrán észrevehető a TI-os táp IC alkatrészeit megpróbáltam minél közelebb elhelyezni egymáshoz, hogy a későbbiekben a kapcsolási frekvenciát nagyobbra választhassam a jelenlegi 300 kHz-ről 2,2 MHz-re. Felülről lefelé haladva elsőként az 5 V jelenik meg, majd a 3,3V és végül a 6 V a kormányszervónak. Az 5 V-os USB csatlakozó így nagyon közel került az IC kimenetéhez.



4.17. ábra: Tápfeszültséget előállító komponensek elhelyezkedése

4.2.2.2 Az analóg és digitális szenzorok elhelyezéseinek szempontjai

A vonalszenzorok helye -, mint ismert - dedikált, amelyet maga a mechanika határozott meg[35]. Az egyik szenzorsort mindenféleképpen az első tengely előtt kell elhelyezni[36], míg a másik tengely mögött, lehetőleg a rögzítési pontokhoz közel. Innen jön, hogy a szenzorsorok közötti távolság 146,95 mm.

A többi szenzor elhelyezésével sokkal szabadabban gazdálkodhatunk, hiszen a távolságérzékelőknek csak a csatlakozójukat kellett elhelyezni.

4.2.2.3 Csatlakozók

A legtöbb csatlakozó a panel szélén a félkörívben helyezkedik el, hogy szerelés esetén könnyen bontható legyen a kapcsolat. Próbáltam a gyakorisági szempontok alapján elrendezni, ezért a panel bal alsó sarkára került az SWD programcsatlakozó, utána az

[35] Nem csak a mechanika, hanem „nagyobb kormányszög létrehozása”

[36] A kerekek közé nem lehetett volna elhelyezni az első szenzorsort a fizikai dimenziók miatt

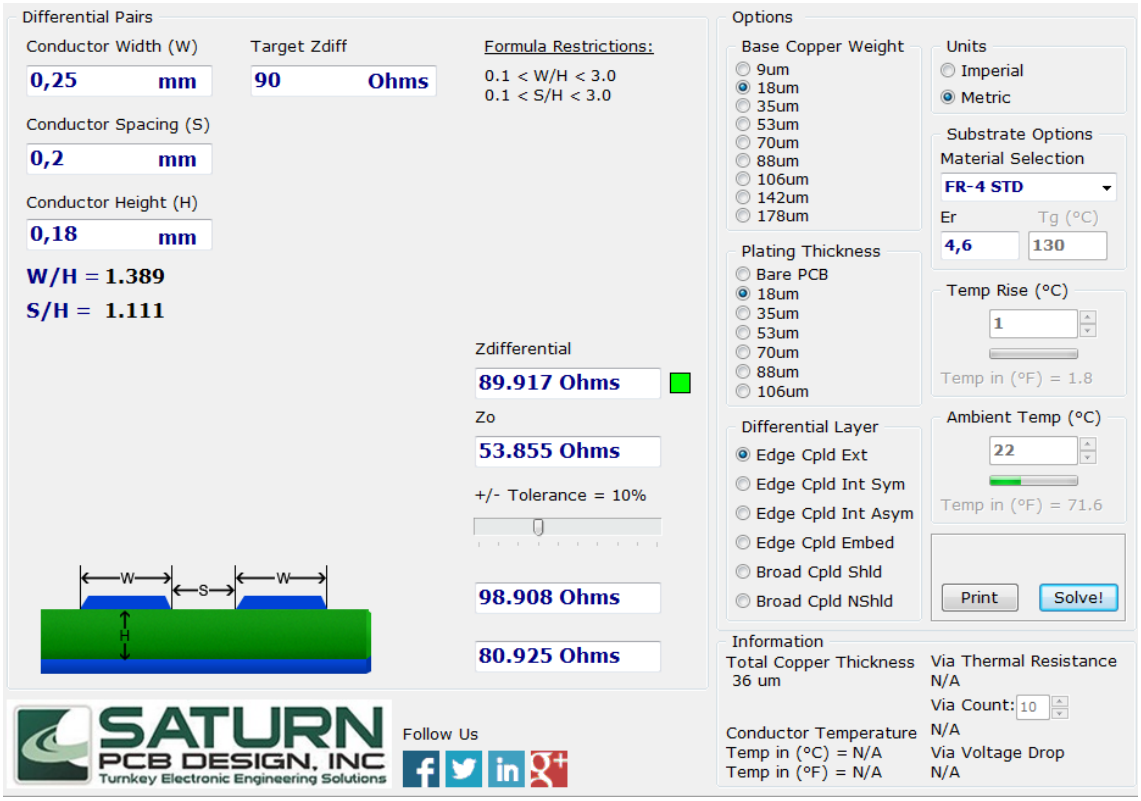
UART, az SPI, és végül a kormányszervó csatlakozója. A tápfeszültséget biztosító USB csatlakozót egy rögzítési ponthoz közel helyeztem el.

A kommunikációs USB jeleit a lehető legrövidebb úton kíséreltem meg elvezetni a csatlakozóhoz. Az STM32-es IC sajnos ezeket a buszvezetékeket nem egymás mellé vezette ki, illetve az USB csatlakozóknak külön 4 mechanikai rögzítési pontjuk is volt.

A csatlakozókhoz a lehető legközelebb kellett elhelyezni az ESD kiküszöbölésére alkalmazott TVS diódákat, illetve hozzájuk 2-3 mm-es sugarú körön egy viszonylag kis kondenzátort (pl. 100 nF), amely elnyeli a hirtelen nagy impulzusokat (8 kV, 1 ns) [30].

4.2.3 Impedancia illesztett kommunikációs interfészek

A nagy sebességű USB-nél a buszvezetékeket impedancia illesztve kellett elvinni az MCU-tól a transceiver IC-ig, majd onnan szintén illesztve a csatlakozóig. Utóbbit, mindkettő USB interfésznél el kellett végezni, vagyis beállítani a 90 Ω-os differenciális ellenállást, amit a Saturn PCB tool-al számoltam ki, ahogy ez a 4.18. ábra prezentálja.

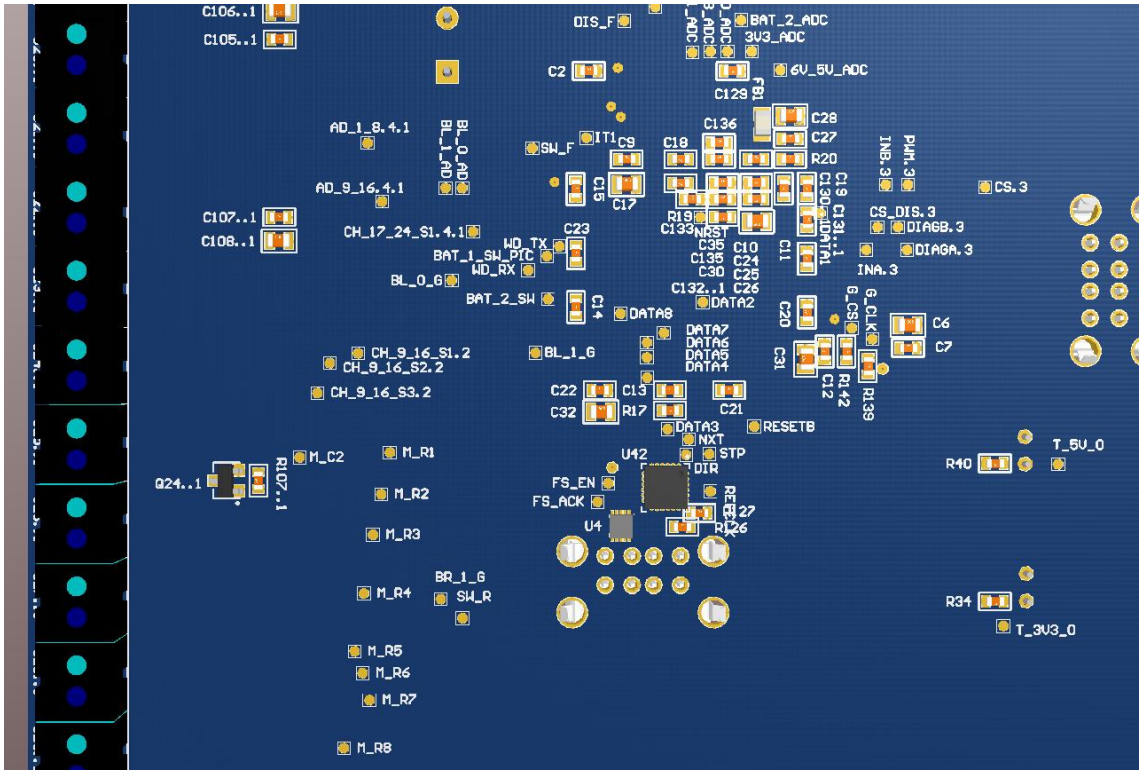


4.18. ábra: Impedancia számítás

A szükséges paramétereket (W, S, H) beállíthatjuk az Altiumban, mint DRC szabályok, így betarthatjuk a NYÁK tervezés követelményeit az USB-re vonatkozóan.

4.2.4 Tesztpontok elhelyezése

A tesztpontok a mérések elengedhetetlen kellékei, melyek egy részét a 4.19. ábra mutatja. A mérésekhez az autót nem szükséges szétszedni, mert minden tesztpont az NYÁK alsó oldalán helyezkedik el, pontosan a hozzáférhetőség miatt. Az STM32-es mikrokontroller minden egyes kivezetésén tudunk mérni, illetve ezeket a pontokat a szita rétegen (Bottom Overlay) egyértelműen elnevezhetjük (pl. BAT_2_ADC), így nem feltétlen szükséges a kapcsolási rajzból kikeresni. A tesztpontok mérete 1 mm, amelyre kényelmesen rá lehet tenni az oszcilloszkóp próbe-ját, vagy akár forraszthatunk is rá. Utóbbival vigyázni kell, mert ha nem tartalmaz egy viát, akkor könnyedén felszakítható a réz fólia és akár a kapcsolat megszakadásához vezethet.



4.19. ábra: Tesztpontok elhelyezkedése a NYÁK alsó oldalán

5 Szoftverfejlesztés

A kapcsolási rajz után, de a NYÁK terv készítésével párhuzamosan elkezdhetjük a szoftvertervezést, hogy minél hamarabb kizárhassuk az esetleges design hibákat. A három fő egységnek a folyamatábráit eleinte az Enterprise Architect-tel alkottam meg, de később áttértem a korábban használt Microsoft Visio-ra az egyszerűsége végett.

Elsőként a kis PIC-re írtam meg a néhány soros assembly kódot, majd az STM32-es eszközzel folytattam egy eval board-on, hogy az autó élesítése közben az esetleges hardver hibákat kirekeszthessem. A hardver elkészültéig az ST által kiadott két gyári függvénykönyvtárt is használtam, úgy, mint STM32 HAL[37] és SPL[38]. A HAL-os verzió annyi hibát tartalmazott, hogy rövid időn belül a korábbi verzióhoz kellett visszatérnem (SPL). Az előbbi könyvtár előnye, hogy átjárhatóságot biztosított a különböző rendszerek között, mert amit C nyelven létrehozunk, módosítás nélkül használhatjuk fel FreeRTOS-es környezetben.

A Raspberry PI fejlesztőkártyával akkor kezdtem el foglalkozni, amikor a robotautó képes volt az IR szenzorok alapján követni a vonalat, melynek szoftverét C++-ban implementáltam.

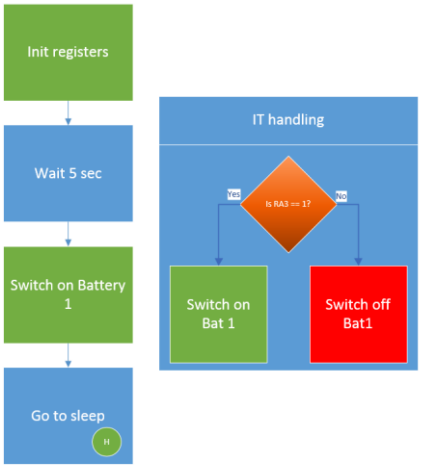
5.1 A PIC-es mikrokontroller

Amint arról a 4.1.7 alfejezetben szó volt, a PIC hajtja végre a bekapcsoláskor az 5 másodperces késleltetést. A tápfeszültség stabilizálását követően a regisztereket inicializálja, majd egy tetszőleges késleltetést beállítva, (ami jelen esetben 5 másodperc,) a „rendszerhez kapcsolja az akkumulátort”.

A parancsok végrehajtása után alvó módba kerül, amiből egy IT, vagy a WDT ébresztheti fel. A megszakításban az RA3-mas bemeneti láb függvényében dönt az akkumulátor „kapcsolási állapotáról”, majd a döntést követően ismét sleep állapotban lesz, ahogy ezt az 5.1. ábra diagrammja mutatja.

[37] Hardware abstraction layer

[38] Standard Peripheral Libraries



5.1. ábra: A PIC szoftverdiagramja

A szoftvert az egyszerűsége miatt nem C-ben, hanem assembly nyelven írtam meg a Microchip Mplab IDE szoftverkörnyezetében. Az áramfogyasztás minimalizálása érdekében, az időzítéshez a saját watchdog-ját használhatjuk fel. Az MCU sleep állapotba kerül a regiszterek inicializálást követően (wait 5 sec), majd ha a WDT regisztere túlcsordult (1 sec), akkor ellenőrzi, hogy várakozási feltétel teljesült-e vagy sem. Ha a feltétel teljesült, akkor a main függvénybe lép (go to sleep) amiből periodikusan felébreszti a watchdog, vagy az RA3-mas IT [22].

A szoftver működését a Proteus Szimulációs programmal ellenőrizhetjük, melyben a kapcsolási rajzot megrajzolva a mikrokontrollerre betölthető a hex fájl. Ha a szimulációs modell hibátlan, akkor az autóban is hibamentes működést kell produkálni, ahogy azt a valóságban is bizonyította.

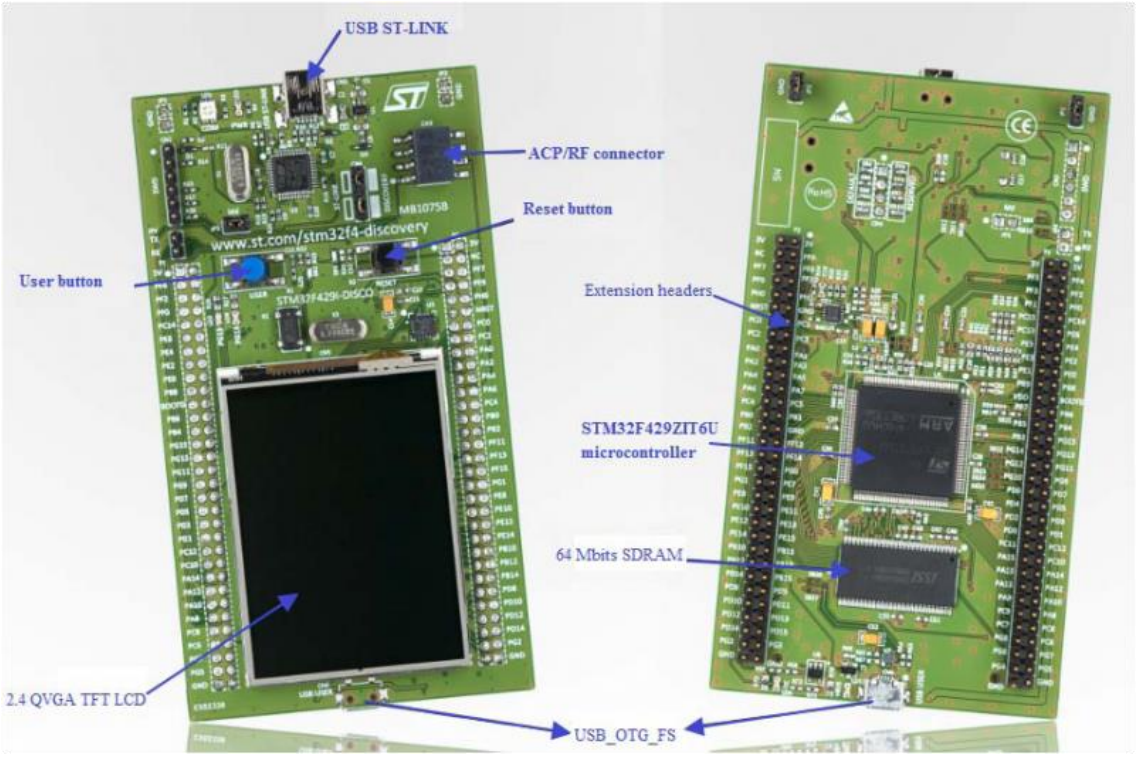
5.2 STM32F4 mikrokontroller

A szoftverkódot eleinte az Eclipse C/C++ szoftverrel próbáltam megírni, de sokszor nehézséget okozott a regiszterek értékeinek kiolvasása, ezért áttértem az Atollic-ra, amelyet az STM32-es rendszerre (is) optimalizáltak. A verziókövetés itt sokkal hatékonyabban megvalósítható, hiszen ha van egy működő kódrészletünk (pl. ADC mérés), akkor készíthetünk belőle egy tag-et.

5.2.1 STM32F429 főciklusa

A fentebb említett STM32-es fejlesztői kártyával (5.2. ábra) a lehető legtöbb szoftveres egységet megpróbáltam létrehozni, azonban hardveres korlátokba ütköztem.

Első körben a 16 kHz-es, illetve az 50 Hz-es PWM jelekkel foglalkoztam, majd ezt követően rátértem az ADC modulokra. Az utóbbit egy-egy ellenállás osztóval ellenőriztem, míg az előzőt egy asztali oszcilloszkóppal, hogy a valóságos eredményeket mérek-e vagy sem. Ezzel validáltam, hogy a processzor belső órajeleit is jól állítottam-e be.

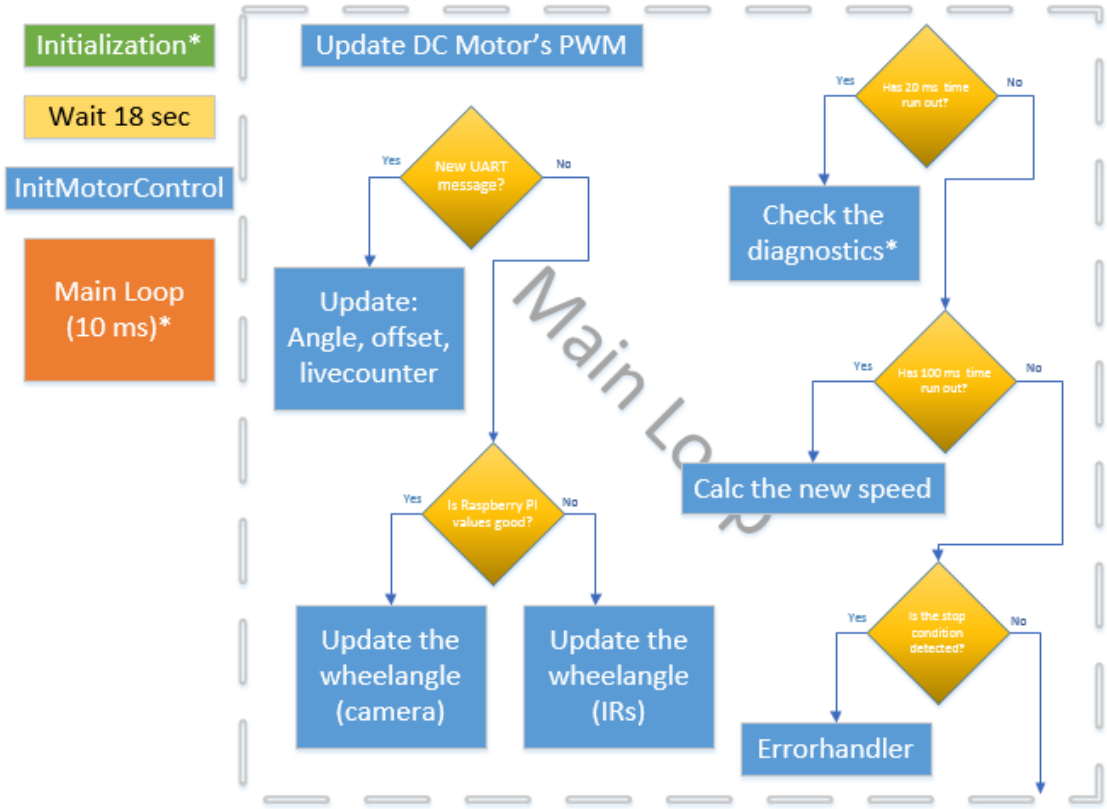


5.2. ábra: STM32F429 discovery board [38]

Az 5.3. ábra magyarázza, hogyan is működik az STM32-es szoftver a tervezett hardveren. Az inicializálást követően, amely során megvárjuk, míg a tápfeszültségek stabilizálódnak, illetve a szükséges perifériákat bekapcsoljuk. Itt az első és egyben a legfontosabb lépés, hogy az autó ne induljon el, amíg nem telik le a 18 másodperces időzítés, mivel ennyi idő alatt indul el a Raspberry PI 3 szoftvere. Ezt követően szükséges még a motorvezérlő IC paramétereinek beállítása, amely végeztével már a főciklusban történik a cirkuláció.

A főciklus 10 ms-os periódusideje elegendő arra, hogy megvizsgáljuk, hogy mi változott a robotautó környezetében. Az RPI-től kaptunk új UART üzenetet vagy sem, mert ha igen, akkor a képfeldolgozás eredményét el kell mentenünk, a vonal szögét, a vonal ofszetjét, illetve egy számláló értéket, hogy milyen régi az üzenet. Körülbelül 300 ms-onként kapunk új információt, ezért a számláló kezdeti értéke 60, tehát ha 600 ms

alatt nem jön új érték, akkor eldobjuk a korábbi mintákat és áttérünk az IR szenzorokra. Amint eldöntöttük, hogy melyik vonalkövetést használjuk, kiszámoljuk az aktuális kormányszöget, amelyet 20 ms-onként frissítünk.



5.3. ábra: STM32 inicializálása és mai függvénye

A sebesség értéke nem változik olyan gyakran, mint a kormányszög, ezért hosszabb ideig tart a mérés, hogy pontosabban tudjuk meghatározni az aktuális eredményt. 200 ms-onként újra számoljuk az aktuális sebességet. Ha az autó egymás után (az eredeti vonalra merőlegesen) három vonalszélességű fekete szakaszt észlel, akkor megállítja a kocsit a hibakezelő függvénnyel (ErrorHandler()).

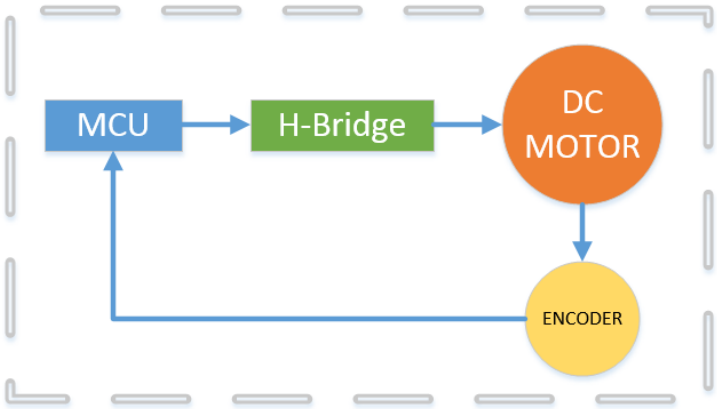
Az utóbbi függvény kikapcsolja a perifériákat (például a motor vezérlő IC-t), majd biztonsági állapotba helyezi az autót. Ezek után minimális lesz az STM32-es panelon az áramfogyasztás, hiszen lehet, hogy azért kellett megállítani a robotautót, mert túl alacsony volt az akkumulátor feszültség. Az 5.3. ábra némelyik függvényénél látható egy csillag, amely azt jelenti, hogy ha nem megfelelő értékkel tér vissza, akkor meghívja az ErrorHandler-t.

5.2.2 Sebesség szabályozás implementálása

A legegyszerűbb P sebesség szabályozót valósítottam meg, amelynek így lesz egy minimális maradó hibája (például 1 $\frac{m}{s}$ helyett 1,05 $\frac{m}{s}$ lesz a sebesség), de ez nem kritikus, hiszen az autónak egy konstans tempóban kell haladnia.

5.2.2.1 DC Motorvezérlés PWM-el

A robotautó motorvezérlőjének meg kell mondani, hogy melyik irányba menjen (előre vagy hátra), ezután pedig egy PWM jelet kell a bemenetére kapcsolni és már el is indult a kisautó. Amint arról korábban szó esett a PWM értékek nem lehetnek tetszőlegesen kicsik vagy nagyok a kapcsolási idők miatt, ezért szoftveresen egy minimum korlátot kell beiktatni.



5.4. ábra: DC motor kontrol diagramja

5.2.2.2 Enkóder illesztése

Az enkódert a motor tengelyére rögzíthetjük, ahol körülbelül egy 10,5-ös áttétel van a motor és kerekek között. Egy aluláteresztő szűrőn a jelet a 16 bites Timer 3-mas bementére köthetjük és számoltathatjuk vele a 0-1 átmeneteket egészen 200 ms-ig. Az áttétel ,a frekvencia és a felbontásunk szorzatai fogják megadni a maximális bemeneti frekvenciát, ami körülbelül 600 Hz. A számláló 200 ms alatt nem tud túlcsordulni, hiszen a maximális értéke körülbelül 300. Észrevehető, hogy a sebességmérés pontossága függ az időtől és függ a sebességtől, hiszen minél több ideig mérünk, annál pontosabb lesz a mérésünk. (Ha feltételezzük, hogy konstans a sebesség, mert egyébként meghamisítja a mérést!). Minél nagyobb a sebességünk, annál kisebb lesz a hibánk, ha a sebességmérés ideje állandó. A 14-es, 15-ös, 16-os egyenletekbe behelyettesítve az aktuális értékeket, melyek eredményeit az 5.1. táblázat reprezentálja.

Induljunk ki az autó tempójából, hogy mennyit fordulnak a kerekek az adott sebességen, melyet már korábban kiszámoltunk (lásd 4.8-as egyenlet). Az enkóder értékek meghatározásához az utóbbi eredményeket be kell szorozni néhány változóval, melyeket az 5.1-es egyenlet is mutat. Az a, az áttétel, a felbontás q (6-6 fekete fehér körcikk), míg az EVA_{2x} a kétszeres kiértékelést jelenti. Az 5.2-es egyenletben felhasználjuk a korábbi eredményeket a sebességszámoláshoz, ahol a $T_1 - T_2$ a mérés ideje (200 ms), míg a $WheelRatio$ egy konstans arányossági állandó, melyet a sebesség és a fordulatszám határoz meg $(0{,}0395)$[39]. Ezt követően meg kiszámoljuk az abszolút sebességhibákat az 5.3-mas egyenlet alapján, ahol V lenti táblázat pontos sebességértékei.

$$EncoderValue = \frac{V_{Act}}{2\pi r} aqEVA_{2x} \tag{5.1}$$

$$V_{Calc} = \frac{s}{t} = \frac{EncoderValue}{T_1 - T_2} WheelRatio \tag{5.2}$$

$$V_{Abs} = |V_{Calc} - V\,| \tag{5.3}$$

	Enkóderértékek (2x-es kiértékeléssel)				
Mérési idők [ms]	1 m/s	2 m/s	3 m/s	4 m/s	5 m/s
10	8,02	16,04	24,06	32,09	40,11
100	80,21	160,43	240,64	320,86	401,07
1000	802,14	1604,28	2406,42	3208,56	4010,70
	Kalkulált sebesség [m/s]				
10	0,87	1,87	2,87	3,86	4,86
100	0,98	1,98	2,98	3,98	4,99
1000	1,00	2,00	3,00	4,00	5,00
	Abszolút sebesség hiba [m/s]				
10	0,13	0,13	0,13	0,14	0,14
100	0,02	0,02	0,02	0,02	0,01
1000	0,00	0,00	0,00	0,00	0,00
	Relatív hiba [%]				
10	14,59	6,95	4,63	3,50	2,84
100	1,54	0,90	0,69	0,58	0,27
1000	0,14	0,08	0,06	0,05	0,04

5.1. táblázat: A sebesség mérés hibái

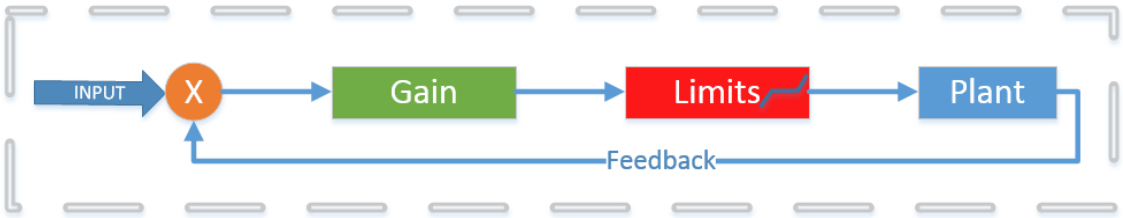
A fenti táblázat összehasonlítja, a kívánt sebességeket a valós sebeségekkel, illetve a relatív hibát is mellékeli. Az enkóder számláló értékei alacsony tempónál és mérési időnél a legnagyobb a mérési hibát okozzák (15%), amely érthető is, mivel a

[39] Szoftveres mérés eredménye

kvantálási hiba itt még jelentős. A 10 ms-os mérési időnél a sebesség fokozásával látszik, hogy a hiba csökken méghozzá körülbelül az ötödére, ami még mindig soknak mondható. Ha 10 ms-ról 100 ms-os értékre váltunk, akkor a relatív hiba az első mérési pontot kivéve 1% alatti, ami már elfogadható pontosság lesz, hiszen maga a P szabályozó sem fogja precízebben kiszabályozni a kívánt értéket. Így felesleges hosszabb ideig mérni. Alacsony tempó esetén „több ideje van” a vonalkövető szabályozónak korrigálni, mint a magasabb értékeknél, ezért sem jelent problémát a magasabb hibafaktor.

5.2.2.3 Egyszerű P szabályzó

A szabályozó bemenete a kívánt sebességérték, a visszacsatolás pedig az aktuális mért sebesség, ahogy az 5.5. ábra mutatja. A gain értékét kezdetben 100-as értékre válasszuk, mert a szabályozó a PWM kitöltési tényezőjét változtatja, melynek értékei a 720-tól 11248-ig terjed. Amint arról korábban is szó volt ezek az értékek 200 ms-onként frissülnek, melynek kezdeti értéke mindig 1800.



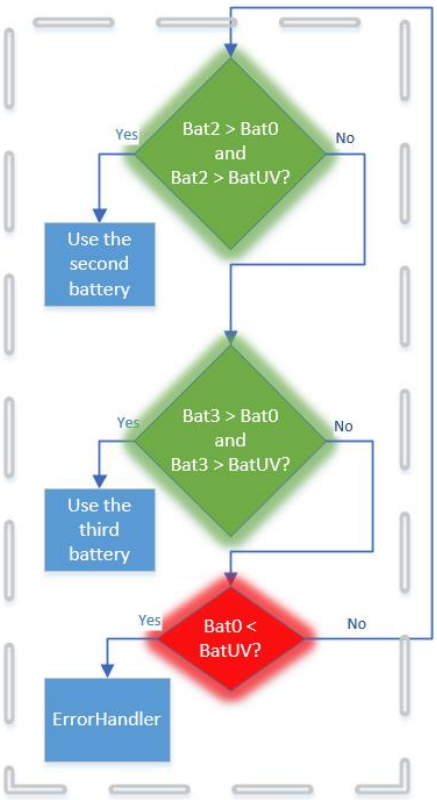
5.5. ábra: A robotautó sebesség szabályozója

Tételezzük fel, hogy a kívánt sebesség 1 $\frac{m}{s}$, és azt, hogy az első mérési pontnál már 0,1 $\frac{m}{s}$-os ez az érték. Ekkor az abszolút hiba 0,9 $\frac{m}{s}$ lesz, mely a számlálót a 2090-es értékre állítja. A tervezési fázisban minimális mérési ismereteim voltak az adott DC motorról, így felosztottam egyenlő tartományokra a PWM értékeit, ahol 2105-nek felelt meg az 1 $\frac{m}{s}$-os váltás. A szabályozó beállási ideje így körülbelül 1 másodperc, melyet az autó teszteléskor tovább lehet gyorsítani.

5.2.3 Akkumulátor menedzsment

Amint arról a 4.1.7-es fejezeten is szó volt, három akkumulátort tartalmaz a rendszer. Mindig az első áramforrást használja fel a bekapcsolásra (Battery_1), majd az inicializálást követően minden egyes periódusban megvizsgálja, hogy van-e nagyobb feszültségű telep. Az 5.6. ábra azt az esetet mutatja, amikor az első akkumulátorról megpróbál átkapcsolni a másikra, úgy hogy a kettő között van egy előre meghatározott

küszöbszint, amely 0,5 V. Az első feltétel teljesülése után meg kell nézni, hogy az adott feszültségszint nincs-e a minimum feszültség alatt (6,4 V), különben nem használhatjuk. Ha a második feszültséggenerátorra nem tudunk átkapcsolni, akkor még megnézhetjük a harmadik paramétereit, majd utána döntünk. Ha egyik feltétel sem igaz, az azért lehetséges, mert nincs hozzá csatlakoztatva több egység, vagy éppen azok feszültségei alacsonyabbak. Utolsó lépésként ellenőrizni kell, hogy az aktuális feszültségszint magasabb-e a minimumszinttől.



5.6. ábra: Akkumulátor menedzsment blokkdiagramja

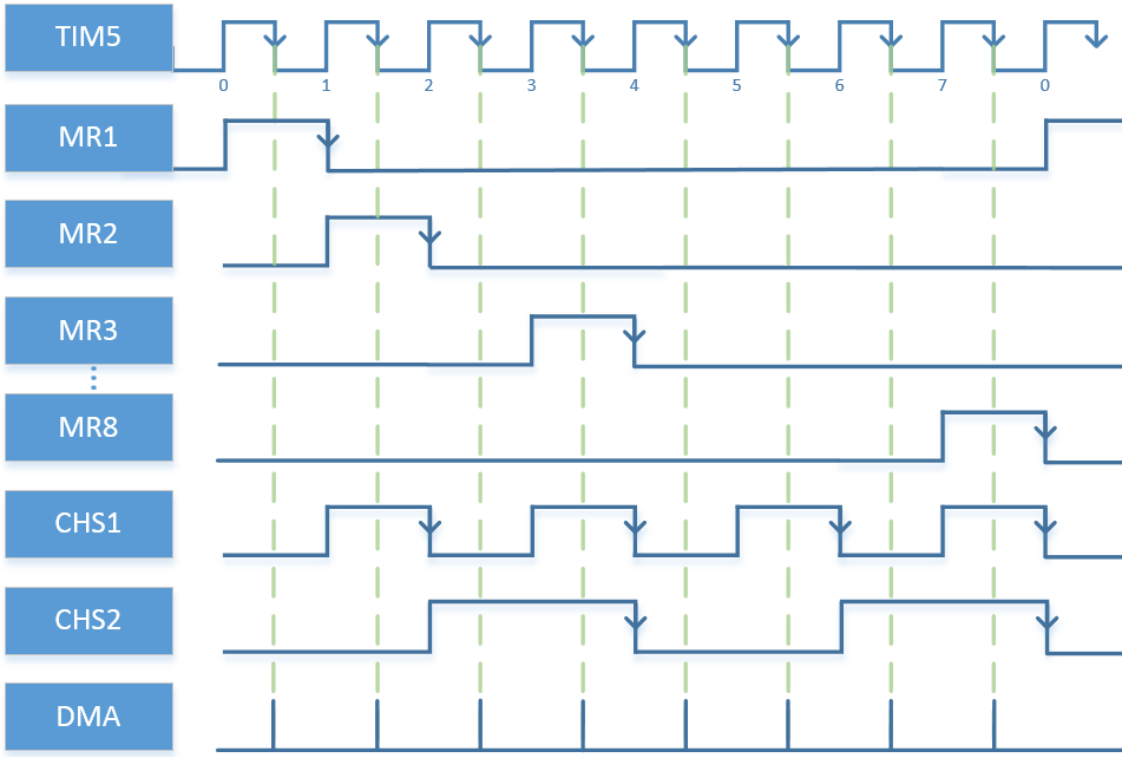
A blokkdiagramm csak azt az egy esetet mutatja be, amikor az első akkumulátorról próbálunk átkapcsolni a másodikra vagy a harmadikra. A változásról a többi esetben is e logika alapján döntünk. A 0,5 V-os küszöbszint elegendő, hogy az átkapcsolás után ne álljunk vissza a korábbira, hiszen a terhelés megszűnésekor 0,1-0,2 Volttal megemelkedik a „régi” akkumulátor kapocsfeszültsége, míg az újé csökken. Ezen kikötés nélkül elképzelhető, hogy a telepek folyamatosan „cserélődnének”, úgymond egy oszcilláció alakulna ki.

5.2.4 Vonalkövetés állapot visszacsatolással

5.2.4.1 Analóg szenzorok jelfeldolgozása

A 48 darab IR vonalszenzor értékét, multiplexálva olvassuk be a DMA 1 segítségével. A kocsinak minden sebességen detektálni kell a vonalat, mely a maximális 5 $\frac{m}{s}$-os érték elérésekor kritikussá válhat. Ha 800 µs alatt ki tudjuk elemezni a szenzorok értékeit, ami a maximális sebességnél 4 mm-es haladást jelent, akkor 4,75-szörös túlmintavételezést sikerült megvalósítani. Értelemszerűen ez alacsonyabb sebességeknél ez a sokszorosára nő.

A mátrixvezérléshez egy periodikus időzítőre van szükség (TIM5), amely gondoskodik a megfelelő GPIO-k szekvenciális kapcsolásáról.

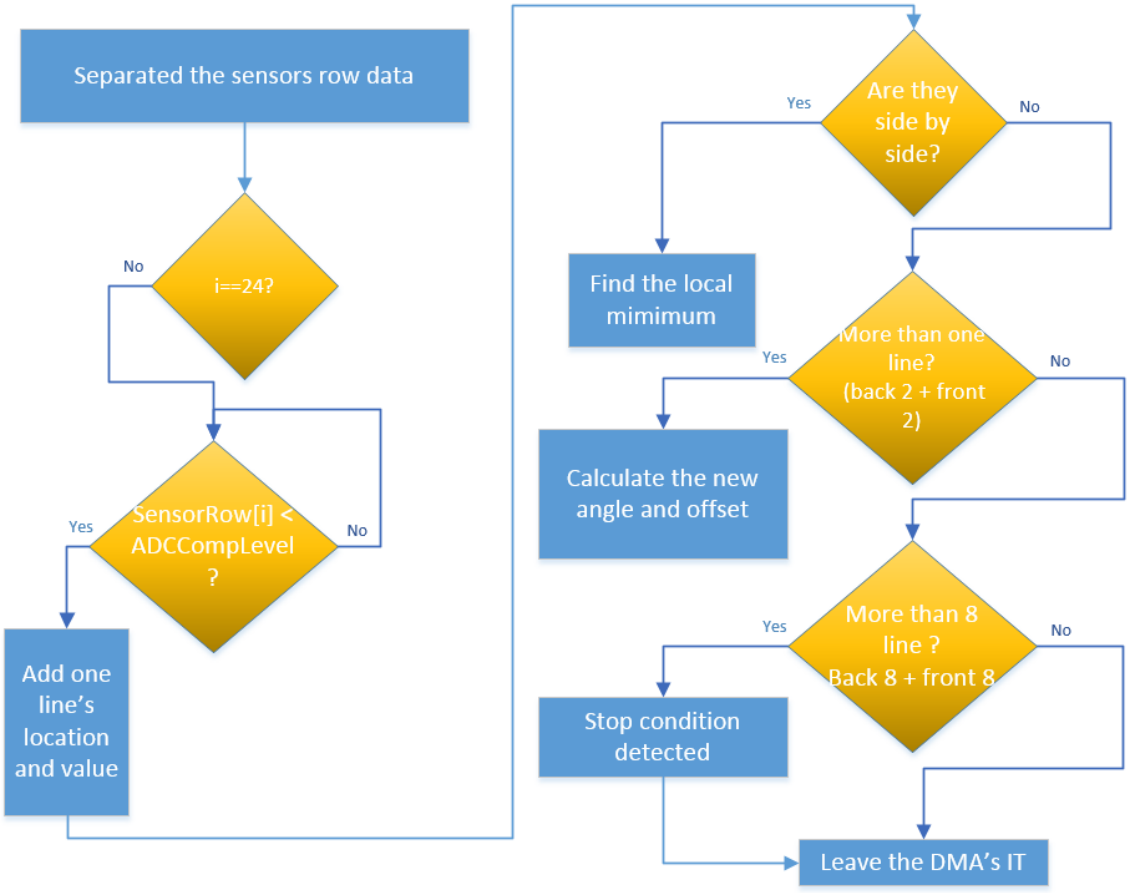


5.7. ábra: DMA interruptjai a szenzorsorok beolvasása közben

Az 5.7. ábra illusztrálja, hogy a TIM5 felfutó élekor bekapcsolja az aktuális szenzorokat (3-3-mat), majd a TIM5 lefutó élekor mintavételezünk a vonalszenzorokat, elkerülve ezzel a beállási idők ráhatását. Ha az időzítő frekvenciája 10 kHz, akkor egy teljes szenzorsor beolvasási ideje 800 µs.

A TIM5 lefutó éle triggereli a DMA-t, mely ilyenkor az ADC1-et és ADC2-öt értesíti, hogy hajtsanak végre 3-3 konverziót, az adott csatornákon. A két ADC modult

egyszerre indítja el, vagyis néhány ps-os hibát leszámítva azonos időben fejezik be a mérést, melynek eredményét egy közös regiszterben tárolják. Ez egy 32 bites regiszter, melyből az ADC felbontása miatt csak a felső és alsó 12 bit van kihasználva. Az ADC jelez a DMA-nak, hogy befejezte a konverziót és az adatokat cirkulárisan egy előre meghatározott memóriacímre töltse be. A 24 darab konverzió befejezése után a DMA egy megszakítást küld a processzornak, ahol a nyers adatértékekből kiszámolja a vonal ofszetjét, illetve szögét.



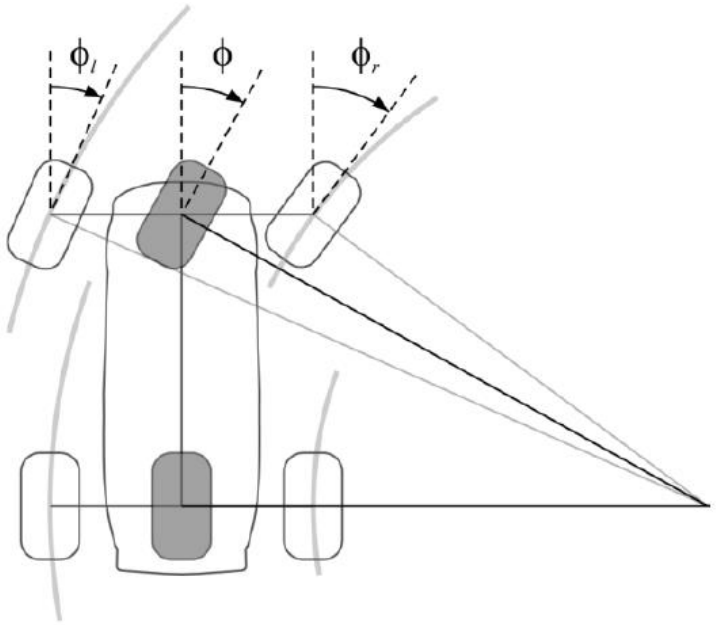
5.8. ábra: DMA megszakításban a szenzoradatok kiszámolása

A fehér vonalnál az ADC értéke körülbelül 3800 (> 3,06 V), míg a fekete esetében ez 500 (< 0,4 V) alatti, ami robosztus detektálást tesz lehetővé. Az 5.8. ábra mutatja, hogy először két részre bontjuk fel a mérési eredményeket, majd egy komparálási szinttel meghatározzuk, hogy hány darab vonalat érzékelt, illetve, hogy mely pozíciókban. A következő for ciklusban megnézzük, hogy mely pozíciók voltak egymás szomszédjai, hiszen egy vonalérzékelésnél legalább kettő szenzornak kell alacsony értéket reprezentálni (19 mm, 7 mm). A korábbi információkat felhasználva nem a teljes szenzorsort olvassuk végig, hanem csak a régebbi vonal közvetlen környezetét (például

5-5 szenzort). Ezután kiszámoljuk az új vonal paramétereit, amiket majd a szabályozó felhasznál. Végső lépésként, még meg kell vizsgálni a megállási feltételt is, hogy az első és a hátsó szenzorsorok 8-nál több fekete vonalat mértek-e vagy sem. Ha igen, akkor a következő főciklusban megállítjuk az autót az ErrorHandler segítségével. A vonal és az autó szögét egy arcos tangenssel számoljuk ki °-ban, míg az ofszetjet mm-ben adjuk meg a szabályozónak.

5.2.4.2 Szabályozó választása [39]

A robotautó Ackermann-kormányzással rendelkezik, ami azt jelenti, hogy a kormányzott kerekek kanyarodáskor eltérő szögeket zárnak be, hogy a kerekek csúszás nélkül és koncentrikusan kanyarodhassanak. Kinematikai szempontból használhatjuk az úgynevezett kerékpár modellt, vagyis a kétkerekű helyettesítőképét, amelyet az 5.9. ábra illusztrál.



5.9. ábra: A kocsi Ackerman-modellje

A vonalérzékelésnél két dolgot kell figyelembe venni: az egyik, hogy a kocsinak az orientációja nem lesz mindig párhuzamos a vezetővonallal (δ), a másik, hogyha párhuzamos, akkor lesz egy ofszet a kettő között (p).

Állapot-visszacsatolásos szabályozásnál a beavatkozó jelre (kormányszög), tekinthetünk úgy is, mint az állapotokból képzett visszacsatolás, ahol δ állapot visszacsatolási együtthatója nulla. Ezért könnyedén felhasználhatjuk a fenti mérhető jeleket kétsoros szenzorsor esetében, mely az 5.4-es beavatkozó egyenletet adja:

$$u(t) = 0 - k_\delta \delta(\mathrm{t}) - k_p p(t) \tag{5.4}$$

Az állapot-visszacsatolásnál a zérusokat nem tudjuk megválasztani, azonban a pólusokat annál inkább. Felírva az átviteli függvényt szemléletesebben látszik, ahol L a kocsi első és hátsó tengelye között mért távolsága (5.5):

$$T_k(s) = \frac{-vs - \frac{v^2}{L}}{s^2 + s\left(-\frac{k_\delta v}{L} - vk_p\right) - \frac{v^2}{L}k_p} \tag{5.5}$$

A pólusok elhelyezése függ a tempótól, ami azt jelenti, hogy az imaginárius tengelytől a sebességnövelés hatására, egyre jobban eltávolodnak. Az 5.6-os és az 5.7-es egyenletekből is ez észrevehető, melyek alapján megválaszthatók a beavatkozó értékek:

$$k_p = -\frac{L}{v^2} s_1 s_2 \tag{5.6}$$

$$k_\delta = \frac{L}{v}((s_1 + s_2) - vk_p) \tag{5.7}$$

Az előnyei a P vagy a PID szabályozóhoz képest, hogy nincs a rendszerben telítődés, nincs memóriája, és nincs statikus hibája sem. Ezen felül még a számítási teljesítményigénye is alacsony, hiszen csak két szorzásból és egy összeadásból áll.

5.2.4.3 Kormánykerekek vezérlése

Amint arról már korábban is szó volt, a kormányszervót egy 50 Hz-es PWM kitöltési tényezője alapján vezéreljük, amelyet a főciklus minden második periódusban (azaz 20 ms-onként) felülíródik. A mechanikai korlátok miatt, egy felső és alsó szaturációt kellett létrehozni, ami körülbelül ±25 °-ra maximalizálódott.

5.2.5 Diagnosztikák

A robotautónál fontos, hogy legalább néhány (körülbelül 8-10) diagnosztikával felszereljük, melyek közül a legkritikusabb talán a távolságérzékelés. Ha lemerül az akkumulátor, az nem akkora hiba, minthogy az autó 5 $\frac{m}{s}$ -mal a falnak ütközik.

A diagnosztikák analóg részét szintén az ADC és a DMA segítségével valósíthatjuk meg a leghatékonyabban, a korábbihoz képest annyi különbséggel, hogy most 1 ms-onként frissülnek az értékek, illetve ennek a taszknak a prioritása magasabb[40].

[40] Itt a magasabb prioritást az alacsonyabb megszakíthatja

A TIM4 felfutó élére triggerel, melynek központi órajele ugyan onnan osztódik le, mint a TIM5. Így a két időzítő szinkronban van egymáshoz képest. Az analóg értékek zaj mentesítésére, illetve átlagolására, egy 25 mintából álló digitális exponenciális szűrőt valósítottam meg. A korábbi értékeket nagyobb súllyal, míg a régebbeiket kisebb prioritással veszi figyelembe, ami gyakorlatilag úgy működik, mint egy analóg R-C szűrő.

A kontroller a diagnosztikákat, a főciklus minden második ütemében ellenőrzi, pontosan akkor, amikor nem a kormányszervó PWM értékeit frissíti. Így elkerülhető, hogy átlepjük a 10 ms-os főciklus időt.

5.2.5.1 Távolságérzékelő szenzor(ok)

Egy szoftveres komparátort létrehozva, talán a lehető legegyszerűbb módon oldhatjuk meg ezt a részfeladatot, elkerülve a bonyolultabb szoftveres hibákat. Tételezzük fel, hogy az elindulás pillanatában, a kocsi 20 cm-es körzetében nincs akadály. Ellenkező esetben a diagnosztika nem fog működni a második fejezetben említett ok miatt. Ha a visszamért szenzor feszültségértéke magasabb, mint a 1,5 V, akkor 40 cm-nél közelebb van az objektum. Ekkor a hibakezelő függvény megállítja az autót, hogy elkerülje az ütközést. 5 $\frac{m}{s}$ -mos sebességnél ez a távolság kevésnek bizonyulhat, de a tesztek legtöbbször zárt helységben történnek, ahol a tempó maximum 0,5-1 $\frac{m}{s}$ közötti. Ha magasabbra állítjuk a küszöbértéket, akkor kanyarodáskor a szenzor előre néz, míg az autó már kanyarodik, így a kijelölt detektálási sugárban nem lehet akadály, vagy a kanyar közepén megáll a robotautó. Egy kisebb helységben, ezt szinte lehetetlen elkerülni.

5.2.5.2 A DC motoráramának korlátozása

Ha az előző védelem valamilyen oknál fogva nem óvná meg a kocsit, akkor nekiütközne egy nagyobb akadálynak, aminek következtében a sebesség nullára csökkenne. Ennek folytán a sebességszabályozó rövid időn belül a maximális impulzusszélességgel vezérelné a DC motort. Az utóbbi egység fizikai modelljében ekkor a feszültséggenerátor értéke 0 V, vagyis csak az armatúra és a vezeték ellenállása korlátozza az áramot, ami akár megegyezhet az indulási árammal is (100 Amper). Ennek elkerülése végett, ha a motorban nagyobb áram folyik, mint 40 Amper, akkor a kisautó biztonsági állapotba kerül, az ErrorHandler függvény által.

5.2.5.3 Tápfeszültségek mérése

A tápfeszültségek közül a 6 V-os kivételével az mindegyiket vissza tudjuk mérni, melyeknél szintén egy digitális komparátort valósítottam meg, 20%-os toleranciával. Ha a 3,3 V-os feszültség elérte a 2,64 V-os értéket, akkor az ErrorHandler függvényt használva kikapcsolja a perifériákat. Összesen 6 ilyen feszültség diagnosztikával rendelkezik a robotautó, amik a következők: 3,3 V, 5 V, Bat1, Bat2, Bat3, BatAct. Ha az első akkumulátorról üzemel a rendszer, akkor a Bat1 feszültsége 0 V körüli, míg a BatAct mérési értékei megegyeznek a valós telepfeszültséggel. Az akkumulátor menedzsment nem teszi lehetővé, hogy az éppen használt telep feszültségét visszamérjük (lásd 4.10. ábra). Az asztali tesztelésekkor nem egyszer futottam bele abba a hibába, hogy lemerült az akkumulátor!

5.2.5.4 Üzemidő maximalizálása

Tegyük fel, hogy valamilyen okból kifolyólag felborul a szabályzási kör, vagy csak éppen valamilyen hibája lesz a rendszernek, mely esetben szeretnénk, ha épségben megállna a robotautó. Ilyen hiba, ha az enkóder egyik vezetéke leszakad és a kisautó azt érzékeli, hogy a sebessége 0 $\frac{m}{s}$, ami rövidesen a maximális sebességet eredményezi. Olyan rendellenesség is előfordulhat, hogy a motorvezérlő IC tönkremegy, melynek következménye lehet, hogy az autó összetörik.

Ezeknek a hibáknak az esetleges kivédésre a futási időt maximalizálhatjuk, mely a főciklus előtt könnyedén beállítható. Sajnos ez a megoldás nem nyújt 100%-os védelmet, ha a fentebb írt diagnosztikák nem érzékelik a hibát, de így egy kicsivel nagyobb esélyünk van.

A három különböző szoftver verziónál, más-más értékeket állítottam be. Az asztali verzióban végtelent, ahol a robotautó nem vezérelte sem a DC motort, sem kormányszervót. A futási típusnál 15 másodperces alapértéket írtam a kódba, ahol a teljes működés megvalósult.

5.2.6 Mikrokontrollerek közötti adatátvitel

Az STM32-es mikrokontroller elsődlegesen a Raspberry PI-al kommunikál, de ne feledkezzünk meg a kis PIC-ről sem. Itt gyakorlatilag GPIO-n keresztül informálódik, ahol az adatáramlás sávszélessége minimális, hiszen itt csak két állapot lehet az adott vonalon.

A RPI esetében az adatfogadást az UART8 és a DMA kooperációja valósítja meg. A kettejük közötti sebességet már korábban meghatároztuk 1,8 Mbps-os értékkel. Amikor a Raspberry PI üzenetet küld az STM32-nek, akkor az UART8 fogadja azt, majd amikor végzett egy csomag vételével, akkor a DMA 2-nek küld egy megszakítást. Az előre beállított bufferbe másolja bele az adatot, majd megnöveli a számlálóját. Egészen addig folyik ez a másolási folyamat, ameddig el nem éri a beállított számláló értékét (17-et). A DMA 2-t cirkulárisan programozzuk fel, hogy az üzenetek ne csússzannak el a tömbben. Az adatok kiolvasásakor a cirkularitás miatt felül íródhatnak az adatok, de ennek elkerülése érdekében létrehozhatunk egy változót, ami garantálja a jelenség eltünését. Az üzenetek formáját az 5.10. ábra illusztrálja:

5.10. ábra: UART csomagformátuma

Nézzünk egy példát az egyik üzenetre: „1:+45.6:100:+:050;”. A csomag a kezdő bájtja az „***1***”, amíg a záróé „**;**”, amivel csomaghatárokat hozunk létre. Az egyes információszeletek között használhatunk egy-egy általunk megszavazott elválasztó jelet: „:”. A többi érték magától értetődő, melynek értelmezését az olvasóra bízom.

A kisautó az UART Rx, illetve Tx pin-jein keresztül a DMA 2 miatt, nem tudott egy időben küldeni és fogadni. Korábban az UART Tx részét arra használtam fel, hogy a kocsi állapotáról, változóiról küldjön tájékoztatást, melyet szintén a DMA 2 végzett. Elsősorban PC-s diagnosztikára használtam fel, mint például, hogy hogyan változtak a szögértékek a kocsi mozgása közben.

5.3 Raspberry PI 3 szoftvere

A Raspberry PI 3 képfeldolgozó szoftverét elsőként Pythonban kezdtem el programozni, hogy megismerkedjek a környezetével, illetve, hogy minden működik-e rajta (GPIO-k, kommunikáció, kamera).

5.3.1 Fejlesztői környezet

Először, mint minden új hardveren, próbáljunk meg egy demóprogramot létrehozni „LED-re” vagy GPIO-ra, amelyeket Pythonba, illetve C++-ban is vizsgáljunk meg. Az utóbbi sokszorta gyorsabb (100x), mint a Python, ahol a körülbelül 60-70 kHz-

et sikerült elérni, míg C++ esetében akár, 22 MHz-et is, attól függően, hogy milyen könyvtárakat használunk [40] [41].

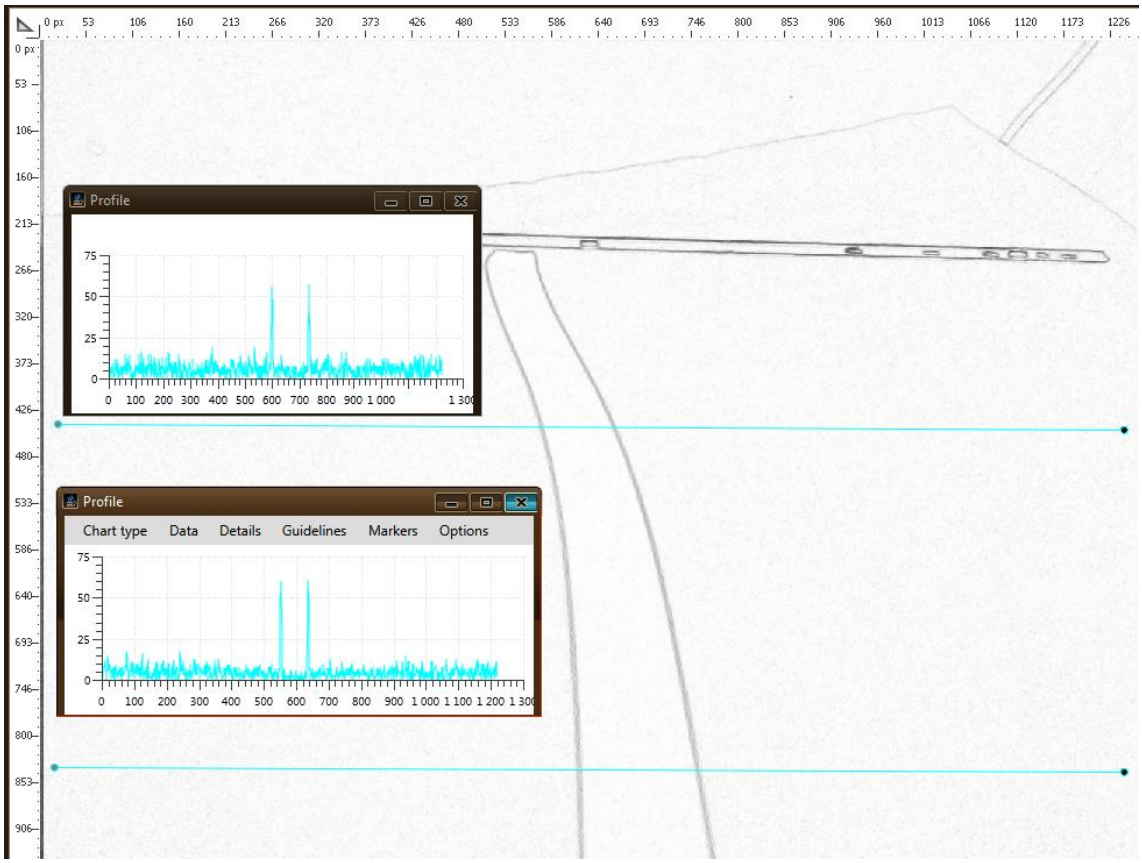
Az Eclipse ingyenes fejlesztői szoftver sajnos itt sem működött kielégítően, ezért a NetBeans C++ változatát használtam, amelyben beépített SVN kliens segítette a szoftverkövetést. Ez főleg azért volt hasznos, mert így a saját PC-men is láthattam a fejlesztést, illetve ha az SD kártyának valamilyen baja lett volna, akkor bármikor visszaállíthattam volna a legutóbbi változatra. A szükséges perifériakönyvtárak beimportálása után már kezdődhetett is a szoftverfejlesztés (OpenCV, kamera, BCM2837).

5.3.2 Kamera képének feldolgozása

Amint arról már 2.3.1.2-es alfejezet is szó volt, egy monokróm képet kell készíteni majd éleket kutatni. A mechanikai kialakítás miatt, a kamera képét 90 °-al el kellett forgatni, majd a Sobel operátor segítségével úgy átalakítani, hogy gradienseket tudjunk keresni.

5.3.2.1 Sobel operátor

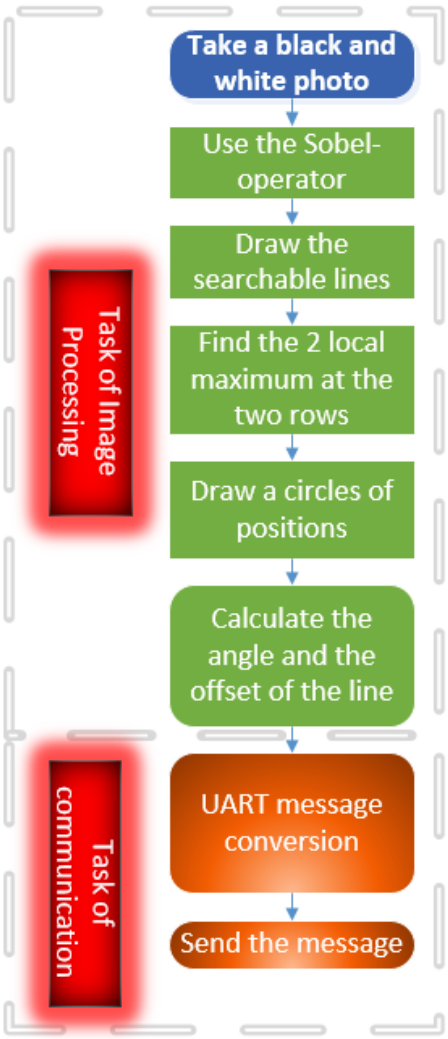
A második fejezetben már megvizsgáltuk az algoritmus működéséről, így most azt nézzük meg, hogyan keressük meg a fekete vonal széleit. Nézzük meg az 5.11. ábrát, ami invertált, hogy szemléletesebb legyen a jelentése. A követendő vonal szélei jól elkülöníthetők, melyeket úgy is meg tudunk találni, ha csak két sort kiválasztunk a képből, amiket kékkel jelölt az ábra. Ezeket végig böngészve, felfedezhetjük a két legnagyobb értéket, amelyeket a két kis grafikon illusztrál. Először megtaláljuk a legnagyobb értéket, elmentjük a pozícióját, majd azt töröljük a mátrixból, hogy a második legnagyobb pixelre is ráakadjunk. Ezt elég két soron végrehajtani, mivel nem kapunk annyival több információt a vonalról, mint amennyi számítási teljesítménybe kerül ez a folyamat. Gondoljunk csak bele, hogy jelen esetben 1280 pixel értékét kell megvizsgálni, összesen kétszer, míg a teljes kép esetében ez 360 szorosára növekedne. Ha ezt százalékban szeretnénk kifejezni, akkor 0,3%-át használjuk fel a képnek, ami meglepő számnak tűnhet. (Ha beleszámolnánk, hogy a kép eredetileg három színmélységgel is rendelkezett, akkor ez a szám, a töredékére csökkenne.)



5.11. ábra: Sobel operátor után a kép pixel értékei

A pozíciók ismeretében kirajzolhatunk egy-egy fehér vonalat a képre, hogy lássuk az aktuális értéket, vagyis, hogy ténylegesen megtalálta-e a vonal széleit. Ezek után már csak két egyszerű középértéket kellett számolni, továbbá ezen eredmények segítségével meghatározni a vonal szögét és ofszetjét. Az előbbit, mint ahogy az IR szenzoroknál, egy inverz tangens kalkulációval alkothatjuk meg, míg az utóbbit, az alsó kiválasztott vonalhoz képest állítsuk be abszolút középre, de ezt majd kocsi kalibrációjánál minden valószínűséggel módosítani kell.

A 5.12. ábra lépésről lépésre mutatja a képfeldolgozást, illetve a kommunikációt, de a GPIO-k vezérlésétől most tekintsünk el, melyeket amúgy is csak mérésre használunk.



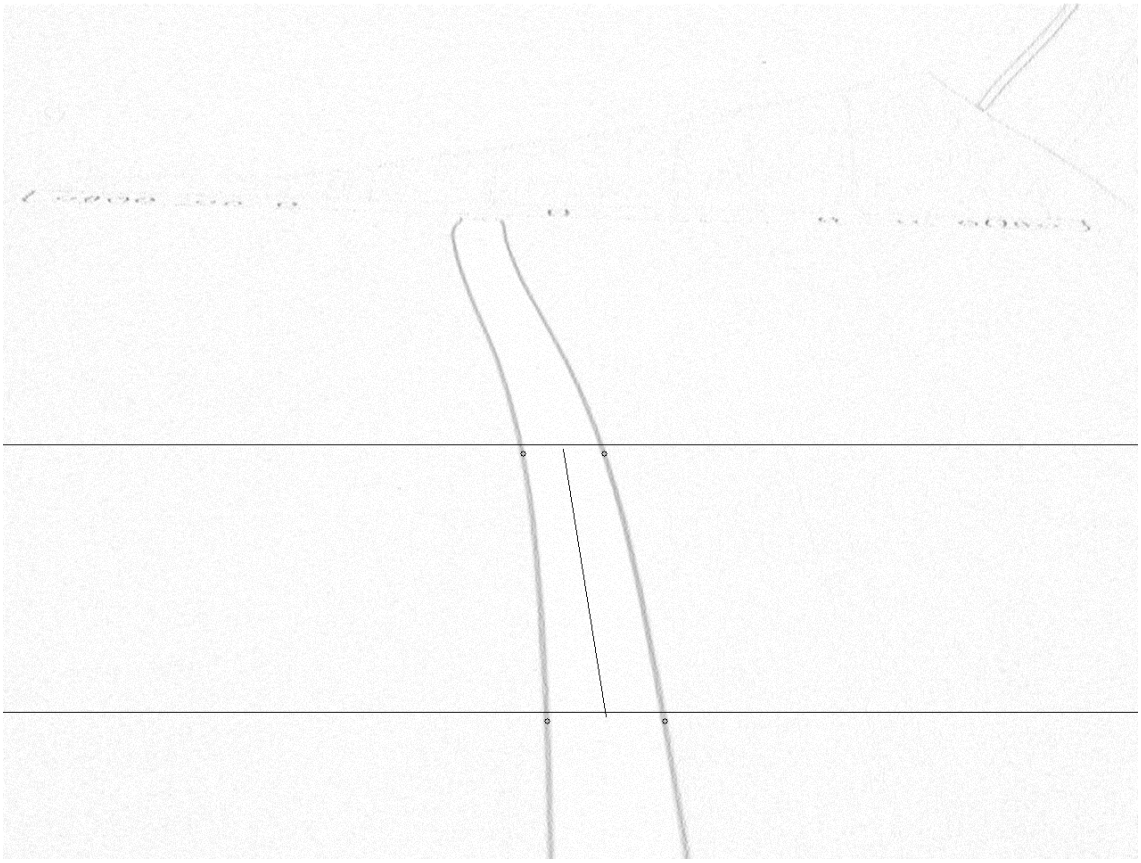
5.12. ábra: Raspberry PI részletesebb szoftvere diagrammja

Sajnos az OpenCV vonal operátora csak korlátozott mértékben működött, amellyel talán gyorsabbá lehetett volna tenni az élek keresését (Find the 2 local..).

5.3.2.2 Vonalérzékelés

A Raspberry PI 3 vonalérzékelési algoritmus lefutása után, a kirajzoltatott képet a 5.13. ábra reprezentálja. A két keresztbemenő fekete sor a fentebb bemutatott, algoritmus eredménye, melyek nem a 2-2 kis körön vannak, hanem fentebb. Ha pont a körökre rajzoltatjuk ki az egyeneseket, akkor felülírjük a mátrixnak intenzitás értékeit. Ekkor a körök a bal vagy a kép jobb szélén helyezkedtek volna el, de ahogy az ábra is mutatja, mind a négy kör a 19 mm-es vonal szélén található. Az alsó sornál ez a távolság

körülbelül 142 pixel, mely azt eredményezi, hogy 1 mm körülbelül 7 pixelnek felel meg, ami jóval pontosabb, mint az IR-es vonalszenzor esetében.



5.13. ábra: Raspberry PI 3 vonalinformáció a képfeldolgozás követően

A fentebbi képen a vonal szöge körül 8 °-os, illetve az ofszetje 4 mm. Az asztalon, az autóforgatásakor az ofszet értékeit nagymértékben (±30 mm), de a vonal szögei csak kismértékben (±0,5 °) változtak. Utóbbinak azért örülünk, mert az esetleges útegyenetlenségnél sokkal toleránsabb lehet, mint az IR szenzor. A kalkulált adatokon egy opcionális ablakozott átlagoló alálható, melyet majd tesztelés során eldöntünk, hogy kell-e vagy érdemes-e használni. Ezt külön nem jelöli az 5.12. ábra.

A képfeldolgozási algoritmus taszkját a processzor körülbelül 40%-ban foglalta le, ami lehetőséget ad arra, hogy még egy másik kamerával kiegészítve megvalósíthassuk a képfeldolgozást.

6 A modellautó élesztése

A szoftver kódolás közben már elkezdtem összerakni a hardver azon részeit, amelyeket szerettem volna mihamarabb letesztelni. Lépésenként haladtam, hogy minél biztosabb legyen az adott egység lefedettsége, tesztelése. A rendszer integrációja során néhány kisebb nagyobb módosításra volt szükség, esetenként hosszabb debuggolási idővel, melyek átfogóbb ismereteket igényeltek.

6.1 Alkatrészek beforrasztása és programozása

6.1.1 Komponensek forrasztása

Az alkatrészek kézi beültetésénél két lehetőség közül választhatunk, melyek az alábbiak:

- Először az IC-ket, majd a passzív komponenseket ültetjük be, végül bekapcsoljuk az áramkörünket
- Lépésről lépésre haladva, a nagyobb blokkokat (pl. tápegység, IR szenzorok) megpróbálhatjuk priorizálva beforrasztani.

Döntsünk az utóbbi mellett, amivel egy kis időt veszíthetünk, de ha rendszerszinten gondolkozunk, akkor ez a befektetés könnyedén megtérülhet[41]. Elsőként az akkumulátor menedzsmentnél kezdjük a themal-pad-es alkatrészekkel, majd a PIC-es kontrollerrel. Az első bekapcsolást követően működött az áramkör, de mint majd a későbbiekben látjuk, egy tervezési hiba került a rendszerbe, ami a szoftveres teszteléseken soha nem mutatkozott.

A robotautót vezérlő tápegység komponenseit forraszuk be, ahol a bekapcsolás után mindegyik feszültség maradéktalanul előállt. Ezután az MCU-t, majd a szenzorsorokat ültessük be, végül az áramkör maradék részeit. A beültetés közben az alkatrészek sorrendje néhol megköti a kezünket, mert a nagyobb komponensektől nem marad hely a forrasztópákának, így ha lehet, akkor mindig a kisebbekkel kezdjük, majd haladjunk a nagyobb méretű alkatrészek felé. Ha

[41] Sokszor a hibák keresésével több időt töltünk, mint magával az implementációval

esetleg szeretnénk cserélni ezeket az alkatrészeket, akkor lehet, hogy teljesen hozzáférhetetlenek lesznek az előbb említett okok miatt.

6.1.2 Mikrokontrollerek felprogramozása

Elsőként a PIC-es kontrollerre töltsük fel a programot, melynek flash-elését sikerült elsőre megvalósítani. Az STM32-es MCU-nál már nem voltunk ilyen szerencsések, mert a reset vezetéket a tervező nem vezette ki az SWD csatlakozóra. Ezt a hiányt pótolva, szintén képesek leszünk felprogramozni az adott kontrollert. Az előbbi MCU programozását, csak a rendszer tápfeszültsége nélkül lehet elvégezni, a másikat viszont pont ellentétesen, mivel a 3,3 V-os debugger tápfeszültséget nem használjuk fel. Gondoljunk csak bele, flash-elés után elindítjuk a programot, majd az MCU és a 3,3 V-os perifériáit (pl. IR szenzorok) is bekapcsoljuk. Ezt követően, szerencsésebb esetben újra indítják az MCU-t. Rosszabb esetben, tönkremegy a debugger, mert túllépjük a maximális megengedett áramfogyasztást. Leválaszthattuk volna a perifériákat egy 0 Ω-os ellenállással vagy jumperrel, de mivel az autót szeretnénk tesztelni, nem pedig az MCU-t, ezért ez egy logikusabb megoldás.

6.2 A rendszer integrációja

A PIC és az STM32-es MCU-k egy egységnek tekinthetők, a másik pedig a Raspberry PI. Ideális esetben a két rész szoftveres módosítás nélkül kommunikált volna, de sajnos az integráció előhozta azokat a problémákat melyekre nem, vagy csak részben gondoltunk. A két egység külön-külön jól működött, de az összeszerelést követően már felfedezhetünk némi hibát.

Az UART jelei csak egy soros 100 Ω-os ellenállást, illetve TVS diódákat tartalmaztak. A bekapcsolást követően az előbbi tönkrement, mivel a Raspberry PI majdnem 20 másodperc múlva inicializálja azokat a GPIO-kat, így az az ellenállás a multiméter szerint „megnyúlt”100 kΩ környékére. Az SPI-os összeköttetést megnézve nem maradt le a kapcsolási rajzról a 10 kΩ-os és a 100 pF-os R-C kör, amely pont ezt jelenséget küszöböli ki. A komponensek javítását követően az UART még mindig nem

produkált hibátlan működést, mert a RPI központi frissítése után, az UART perifériájának a rendszerórajelét felülírta, amelyet még be kellett álltani[42] .

Az STM32-es MCU-val elsőkörben az asztalon felfordítva teszteltem a motorvezérlést, illetve a P sebességszabályozót. Az előbbit az oszcilloszkóp segítségével validáltam, míg a P szabályozónál beállítottam egy előre meghatározott alacsony sebességet (pl. 0,2 $\frac{m}{s}$), majd azt megdupláztam (0,4 $\frac{m}{s}$). Az állandósult állapotot megvárva, a debugger segítségével megállítottam az autó programját abban a függvényben, ahol ki tudom olvasni az enkóder értékét. Ha megkétszereztem a sebességet, akkor a számláló értéke is megduplázódott.

A kormányzás előtt sokkal nagyobb prioritást tulajdonítottam az elülső távolságérzékelő szenzornak. A fenti esethez képest most nem kellett a motort vezérelni, elég volt azt figyelni, hogy a kontroller mikor kapcsolja be a kék LED-et, ha egy A4-es papírt közelítünk hozzá. A pontos távolságot ekkor még nem határoztam meg, csak a közelítő értéket, ami 35 cm.

A kormányszervónál némi szoftveres korrekcióra volt szükség, ugyanis az értékeket ±30 °-ra limitáltuk, melyet a mechanika nem tette lehetővé, amikor jobbra fordultak az első kerekek. Néhány mérés után ±25 °-ra maximalizáltam az értékeket. Emlékezzünk vissza, hogy eredetileg az előző értéknek kellett volna a maximumnak lenni, amely most az elméleti maximális kanyarodási szög, a 36 ° helyett.

A Raspberry PI szoftvere elküldte azokat az adatcsomagokat, amiket a saját kijelzőjén megjelenített, így biztos lehetünk abban, hogy a kommunikáció jól működik. Ez után ellenőriztem, hogy ha egy fehér A4-es papírra ragasztok egy fekete szigetelőszalagot, akkor mennyire fordítja el a kormányt. Szerencsére itt már nem volt gondom és a papír orientációját változtatva követte a vonalat.

6.3 Tervezési hibák

Egy ilyen bonyolultságú rendszernél szinte lehetetlen elkerülni, hogy hiba nélkül valósuljon meg.

[42] Raspberry PI 3 turbó üzemmódja, melyet csak hűtőbordával érdemes használni!

6.3.1 Kisebb hardveres hibák

Kezdjük a vonalszenzoroknál, amelyből majdnem a lehető legtöbb van a panelen (48 darab). Az előtétellenállást méreteztem el, mert 100 Ω helyett 33 Ω-ra lett volna szükséges, hogy 20 mA-es áramerősséggel vezéreljük az IR szenzorokat.

A következő kisebb hiba, hogy az 5 V-ot, illetve a 6 V-ot nem tudjuk felváltva visszamérni, mert a P csatornás FET Source, a 6 V-ra volt kötve, míg a gate-je, az MCU lábára, ami csak 0 és 3V3-as tartományban üzemel. Ez azt jelenti, hogy az a FET folyamatosan be van kapcsolva. Ezt a problémát úgy oldottam meg, hogy a 6 V-os tápfeszültséget a továbbiakban nem méri az MCU, mivel a 6 V-os kormányszervót az akkumulátorra kötöttem. Ennek egyik oka, hogy hirtelen (nagy) irányváltáskor a pufferkondentátor nem tudott elég energiát eltárolni, hogy áthidalja ezt a tranzienst. Sajnos nem volt időm kísérletezni a pólus-zérus elrendezésekkel, ezért az akkumulátorra csatlakoztattam. A hátránya, hogy nem tudjuk diagnosztizálni a kormányszervó meghibásodását.

Az STM32-es bekapcsolása után, de még az inicializálási rész előtt a LED-ek kapcsoló FET-jei a parazitahatások miatt feltöltődhetnek, melyek ilyenkor elkezdenek világítani. Hasonlóképpen, mint a többi ilyen kapcsolási részben, egy párhuzamos lehúzó ellenállást kell tenni a gate-source pin-jeire, ami kiküszöböli ezt a jelenséget.

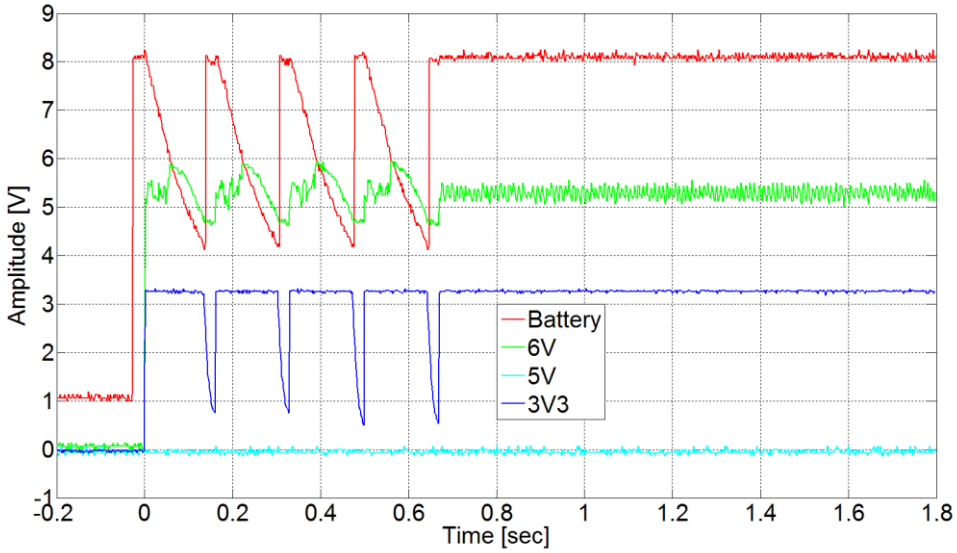
A Raspberry PI tápfeszültségét eredetileg 5 V-osra terveztük, de az USB-s vezeték nem megfelelő minősége miatt, 1 A-es áramtüskék esetén körülbelül 700-800 mV esett rajta, ami a kis számítógép alacsony feszültség flag-jét (under voltage) bebillentette a monitoron. Ennek elkerülése érdekében megemeltem a feszültséget 5,5 V-ra, amely a maximális megengedhető feszültség a panelen[43]. Választhattam volna azt a megoldást is, hogy készítek egy jobb minőségű vezetéket, de így csak egy ellenállást kellet kicserélnem és ezt kevesebb befektetésnek ítéltem meg.

[43] A távolságmérő szenzorok, a multiplexer IC-k maximális megengedhető feszültsége 5,5 V

6.3.2 Érdekesebb hardveres hibák

6.3.2.1 Tápfeszültség újraindulásának véletlen oka

A rendszer működése közben azt tapasztaltam, hogy egyik pillanatról a másikra leállt a robotautó, ahol néhány esetben újraindult, néhány esetben viszont mozdulatlanul ált. A diagnosztikák sem jelezték, hogy baj van, pedig egy LED-nek mindenféleképpen világítani kell, amikor a hibakezelőbe függvény a végére ér. Ez csak arra utalhat, hogy az akkumulátor menedzsment egyik pillanatról a másikra teljesen kikapcsolt.

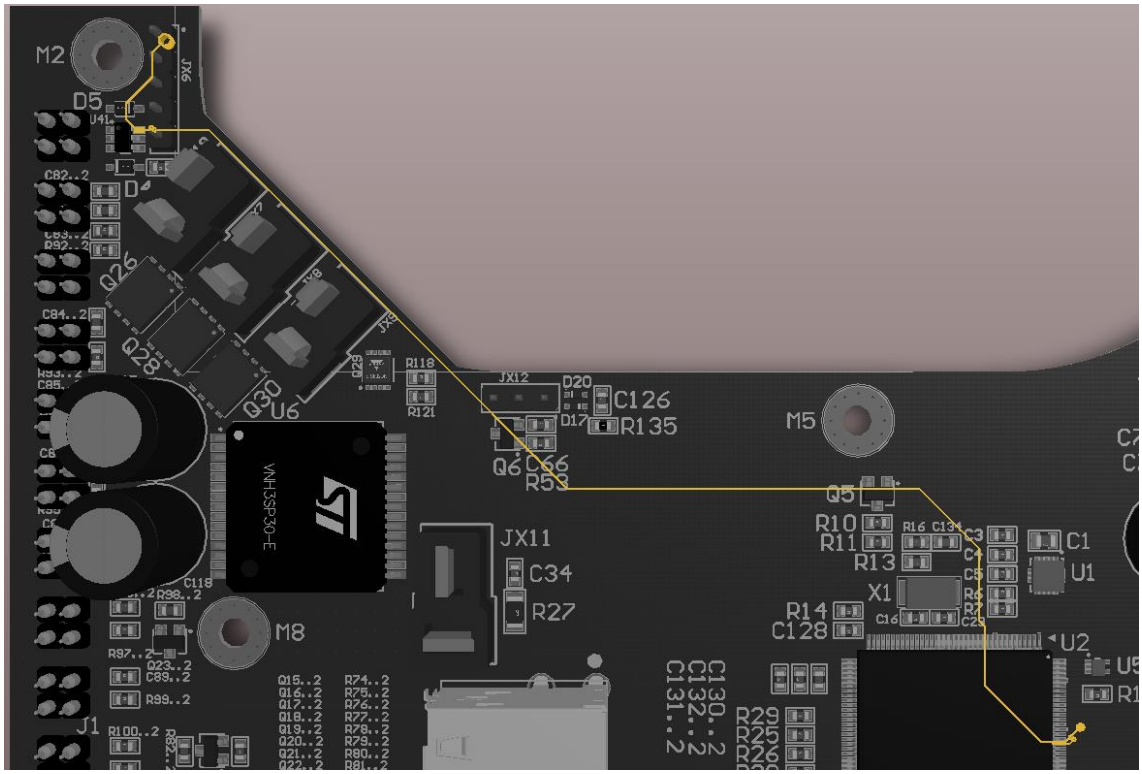


6.1. ábra: A 6 V-os tápfeszültség beszakadása

Újra csatlakoztatva az akkumulátort, minden hiba nélkül elindult a rendszer, majd egy másik alkalommal újra előfordult ez a jelenség. Többször mértem az akkumulátorokat, illetve a többi tápfeszültséget, de az asztalon oszcilloszkóppal egyszer sem jött tudtam reprodukálni a hibát. Amikor először elindítottam az autót, akkor a motor vezérlő IC meghibásodása miatt, maximális sebességgel a falnak csapódott. Ekkor a NYÁK széléből letört egy kis darab, illetve az egyik szenzor is, de nem tulajdonítottam neki nagy jelentőséget, hiszen működött tovább. Legtöbbször a lakásban jött elő ez a jelenség, ahol a „pálya” felülete nem volt egyenletesen, mert járólapok között fuga található. Amikor a robotautó áthaladt ezeken a „mélyedéseken”, akkor előfordult, hogy megállt. Ebből arra következtethetünk, hogy a NYÁK-ban, az egyik tápvezeték delaminálodott, de ez kívülről nem látszódik, csak röntgennel tudnánk igazolni. Analizáltam a tápfeszültség vezetékeinek a vastagságát, majd a legvékonyabbat átforrasztva, megszűnt ez a probléma.

6.3.2.2 Tápfeszültség újraindulása tervezési hiba miatt

A másik hasonló probléma viszont a nem megfelelő tervezésre vezethető vissza. A rendszer bekapcsolásakor többször azt láthatjuk, hogy sorozatosan újraindul az STM32-es MCU, amiből arra következtethetünk, hogy túl hamar akarja bekapcsolni az IR szenzorokat és a többi fogyasztót, de a pufferkondenzátorokban nincs elég tartalék, hogy fedezzék ezt a tranziens terhelést. Ha egy szoftveres késleltetést konstruálunk, akkor azt fogjuk látni, hogy ez nem oldja meg a kellemetlenségeket. Ha viszont a 6 V-os feszültség 0 Ω-os ellenállását kiforrasztjuk, akkor megszűnik a jelenség. Gondolhatnánk, hogy túl nagy a terhelés a 6 V-os kimeneten, mert akkor még innen üzemelt a kormányszervó. Kössünk át az előbbi egységet az akkumulátorra, mellyel elviekben megoldható ez a probléma. Sajnos egyszer-egyszer még előfordult a jelenség, így újra végig kell gondolnunk, hogy mi okozhatja a hibát. Ha az áramkör helyébe képzeljük magunkat, illetve megnyitjuk a NYÁK tervet, akkor szembe tűnhet, hogy a PIC-es MCU bemenetén nincs szűrőkondenzátor, amely CMOS bemenetű, tehát nagyon érzékeny a zajokra.



6.2. ábra: PIC-es szűrőkondenzátor hiánya

Nézzük meg az 6.2. ábrán a PCB egy részét, ahol ezt az összeköttetést kiemeli az ábra sárgával. A kis rézvezetéket a teljesítményelektronika közelében helyezkedik el,

amelyre biztosan rásugároz ez a rész. A kettő között van egy csatolás, illetve ez a hosszú vezeték úgy viselkedik, mint egy induktivitás, ami tovább gerjeszti ez a folyamatot. Ha egy kis értékű kondenzátort (pl. 1 nF) forrasztunk a PIC-es MCU közvetlen bemenetére, akkor véglegesen megoldhatjuk a tápfeszültség újraindulásának az okát.

6.3.3 Szoftveres módosítások

Alapvetően elmondható, hogy egy-egy hardveres módosítás után a szoftvert is korrigálni kell. Nem túl jelentős szoftvereshibák fordultak elő, úgymint az akkumulátor feszültségek visszamérése, vagy éppen az 5 V-os feszültségnél az egyik arányszámot úgy módosítani, hogy 5,5 V legyen az új nominális érték.

Az idő előrehaladtával egyre többet lehetett tesztelni a parkolókban, vagy adott esetben a lakásban. Ekkor különböző szoftververziók futottak a robotautón, melyeknél más-más szempontokat kellett figyelembe venni, mint például a maximális futási idő.

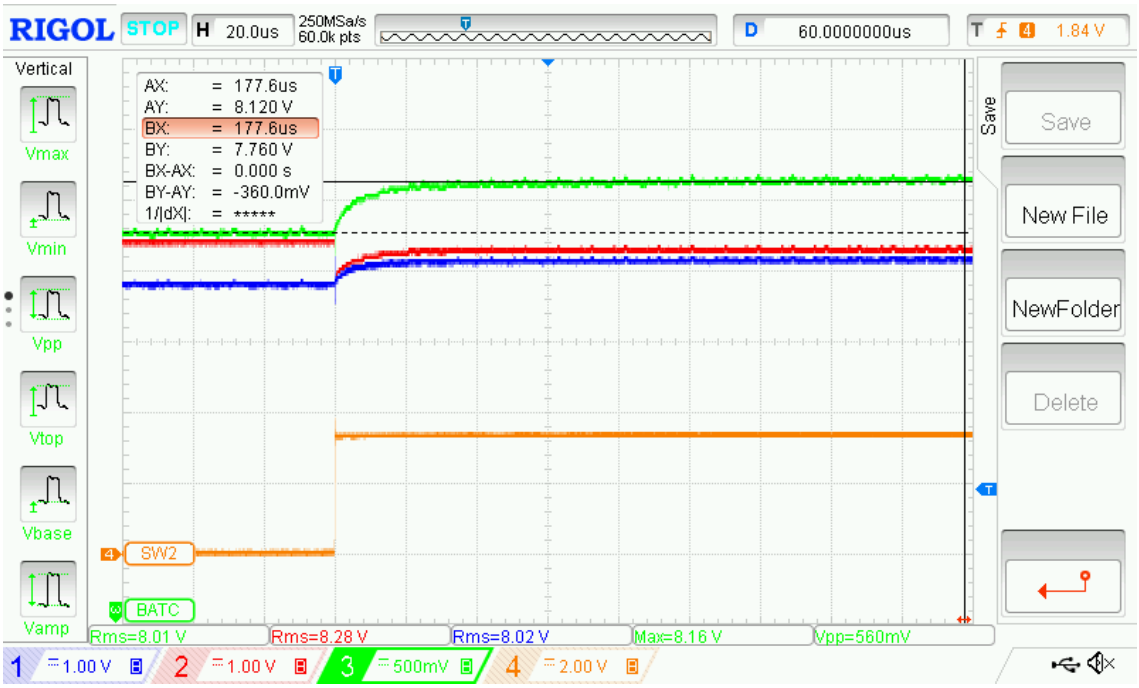
Hasonló verziókat hozhatunk létre a Raspberry PI-nél is, hogy validáljuk az általa számolt eredményeket. Míg a RUN verziónál fő szempont lehet, hogy csak három dolgot írjon ki a konzolra, amit esetenként leolvashatunk a kijelzőről (asztali monitor), míg a DESKTOP verziónál az adott végeredményt kirajzolhatja a képernyőre, illetve a konzolon továbbra is megjelenik az eddig információcsomag.

7 A rendszer tesztelése

A robotautó tesztelése az asztali méréseken túl a pályára is kiterjed, hiszen itt derül ki igazán, hogy mennyire tudja követni a vonalat, illetve mekkora a maximális sebessége amivel képes bevenni a kanyarokat. A tesztek során inkább a vonalkövetésre fókuszáltam, mint az akkumulátor időre, illetve, hogy mennyire reális a beállított sebesség.

7.1 Általános rendszerteszt

Az elsődleges kis tesztek után, amelyeket az előző fejezetben említettem, nézzük meg a rendszert általánosságban. Kezdjük rögtön az akkumulátor menedzsmenttel, ahol az első feszültségforrásról átkapcsolunk a másodikra.



7.1. ábra: Akkumulátor menedzsment tesztelése

A 7.1. ábra oszcilloszkóp képe négy csatornát mutat be, ahol a BAT1 (piros), a BAT2 (kék), illetve és a BATC (zöld) ofszetje azonos, azaz 7 V. Az első telep feszültsége körülbelül 7,8 V, amíg második feszültsége 8,4 V, ami teljesíteni korábban ismertetett kapcsolási feltételeket. Az SW2-es probe felfutó élére triggereljünk, ami a második akkumulátort kapcsolja a rendszerez. Az ábrán jól kivehető, hogy a BATC feszültsége a kapcsolás utána megemelkedik 560 mV-tal, míg a BAT2-es feszültség egy pillanatra „beszakad” a hirtelen terhelés miatt. Az is észrevehető, hogy a váltás pillanatában

kevesebb, mint egy 1 µs ideig összenyitnak a kapcsoló FET-ek, de ez egyálalán nem okoz gondot. A BAT1 és a BAT2 között a FET body-diódája kinyit, mert a BAT2 feszültsége nagyobb, ezért néhány mA-rel elkezdi tölteni, az átkapcsolást követően. Ebben az esetben szerencsés, hogy nem Schottky dióda található a kapcsolási rajzban, mert ha közel egységesen vannak feltöltve a telepek, akkor egyenletesen fognak merülni (pl. 7V8, 8 V, 8V2)

A rendszer üzemidejét körülményes meghatározni, mert ezt több tényező is jelentősen befolyásolja, de mégis határozzunk meg egy hozzávetőleges értéket. Az autó eredeti akkumulátor kapacitása 3000 mAh, amit kiegészítettünk még kettő darab 5000 mAh-ás Litium-Polimer teleppel, melyek körülbelül 0,5 kg-mal növelték meg az autó tömegét (2685 gramm). A teljesítmény súly aránya így is 24,2 kg/LE[44]. Csak összehasonlításképpen a mai Forma-1-es autókkal, ahol 0,7 körüli ez az érték.

Battery: 10800 mA		**5V: 863,12 mA**		**3V3: 192 mA**	
Kormányszervó[45] [9]	500 mA	Raspberry PI [17]	830 mA	Gyorsulásérzékelő [21]	0 mA
DC motor	10 A	Multiplexerek [20]	0,12 mA	Giroszkóp [25]	0 mA
		USB [18,19]	0 mA	Vonalszenzorok [10]	120 mA
		Távolságérzékelő [14]	33 mA	Enkóder [10]	20 mA
				STM32F4 [24]	52 mA

7.1. táblázat: A rendszer alegységeinek áramfogyasztásai

Átlagos értékekkel számolva, a korábbi 4.1-es táblázatot felhasználva, módosítsuk az értékeinket, ahogy az 7.1. táblázat illusztrálja. A DC motor áramát egy lakatfogó segítségével mértem meg, úgy, hogy az autó állandó sebességgel ($1\ \frac{m}{s}$) haladt, miközben leolvastam róla az áramértékeket. Ezzel egyrész validáltam a motor árammérését is, mert UART-on keresztül elküldte a PC-nek az áramértékeket, a sebességet és más egyéb információkat is. A 0 Ω-os ellenállásokat kiforrasztva megmérhetjük egy digitális multiméterrel az áramfelvételét az 5 V-os, illetve a 3,3 V-os részeknek. Az átlagos teljesítményfelvételt így a 7.1-es egyenlet írja le, ahol $\eta = 0{,}9$ faktor esetén ez az eredmény 83,74 W.

[44] 100 W-os DC motorral számolva

[45] A kormányszervó áramértéke csak közelítő becslés

$$P_{Avg} = U_{Bat}\, I_{Bat} + \frac{U_{5V}I_{5V}+U_{3V3}I_{3V3}}{\eta} \quad (7.1)$$

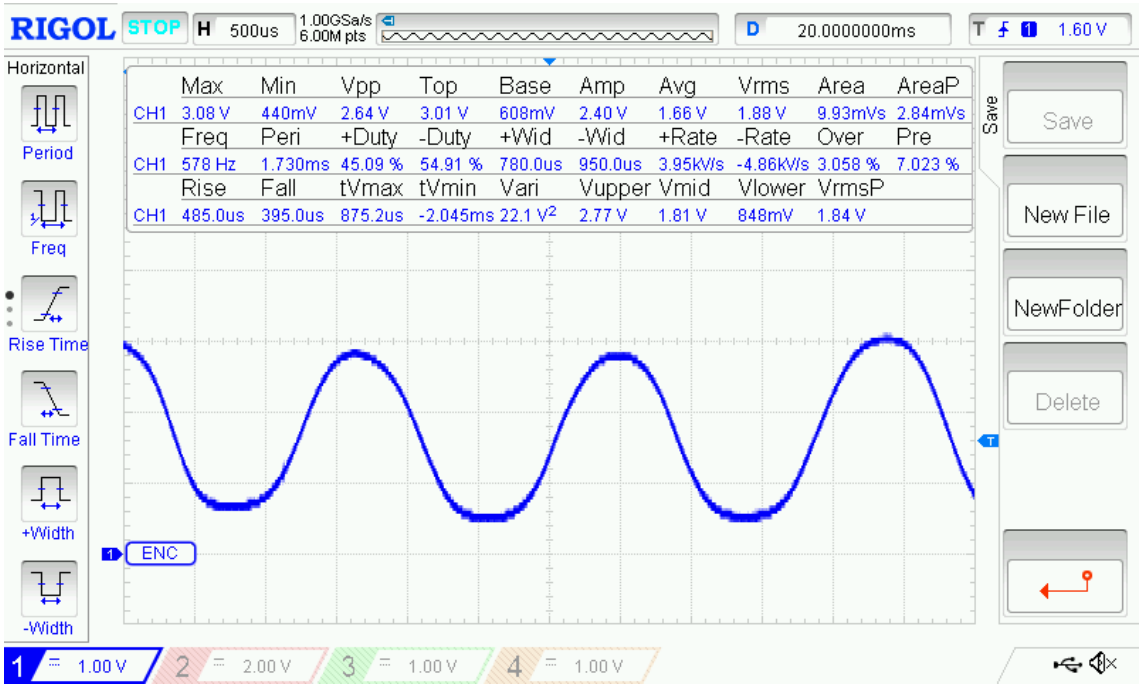
Ha mindhárom akkumulátorból a maximális kapacitás csak 90%-át használjuk fel, akkor a következő egyenletekkel (7.2, 7.3) kiszámolható az átlagos üzemidő, ahol q 2, hiszen ennyi 5000 mAh-ás akkumulátortelepünk van ($W_{90\%} = 84{,}24\ Wh$)

$$W_{90\%} = 0{,}9[(U_{5000}C_{5000}q) + (U_{3000}C_{3000})] \quad (7.2)$$

$$T_{Avg} = \frac{W_{90\%}}{P_{Avg}} \quad (7.3)$$

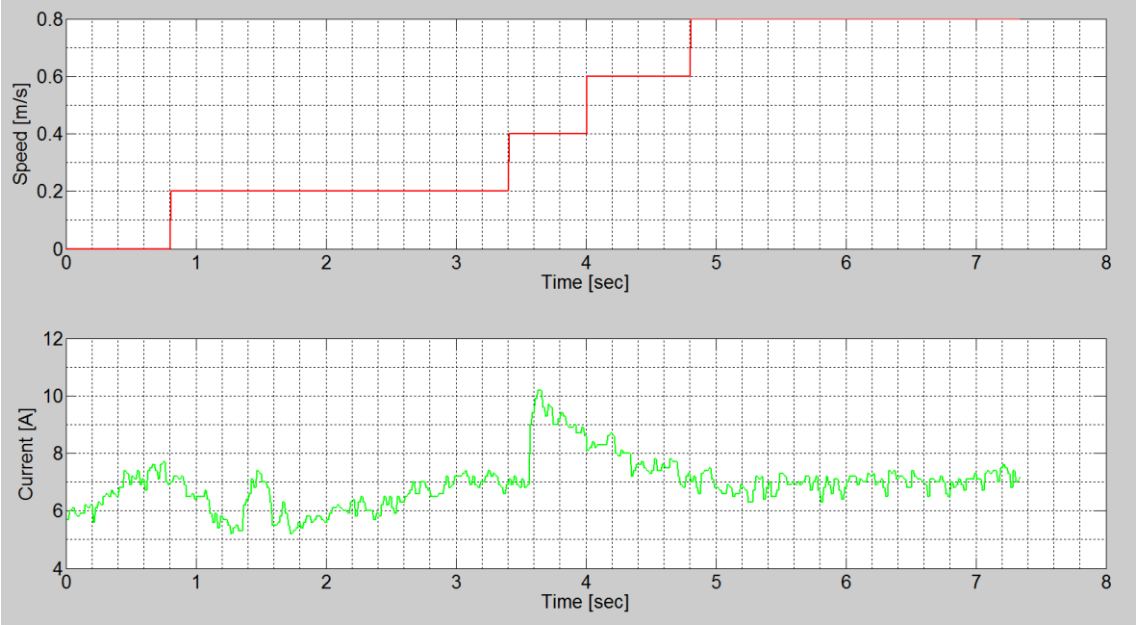
A rendszer tehát körülbelül 1 óráig képes működni, ami persze egy erős becslés, hiszen ezt nagymértékben befolyásolja az autó sebessége, a gyorsítások és lassítások sorozata.

A robotautó sebesség teszteléséhez, egy parkolóban kijelöltem egy 10 m-es szakaszt, mely a gyorsítási és lassítási periódusokat nem tartalmazta. A validáláshoz egy egyszerű stoppert és mindenki által ismert sebességképletet használtam. Az előre beállított 0,5 $\frac{m}{s}$ –os sebesség, helyett a dupláját mértem. A hiba megkereséséhez megmértem a maximális fordulatszámhoz tartozó enkóder jeleit, illetve a hozzá tartozó enkóder értékeket.



7.2. ábra: Maximális fordulatszámnál a sebességmérő enkóder jelalakja

Amint a 7.2. ábráról leolvasható, a maximális frekvencia 578 Hz, amit a kocsi szabadonfutó állapotában mérthetünk meg. A felfutási és lefutási idők természetesen a fordulatszám csökkenésével arányosan redukálódig. Ehhez a frekvenciához 318-319-es enkóder értékek tartoztak, miközben a timer mérési ideje továbbra is 200 ms volt, így csak a WheelRatio arányosságot kellett megduplázni, hogy a valósággal megegyezzen a sebességértéke.



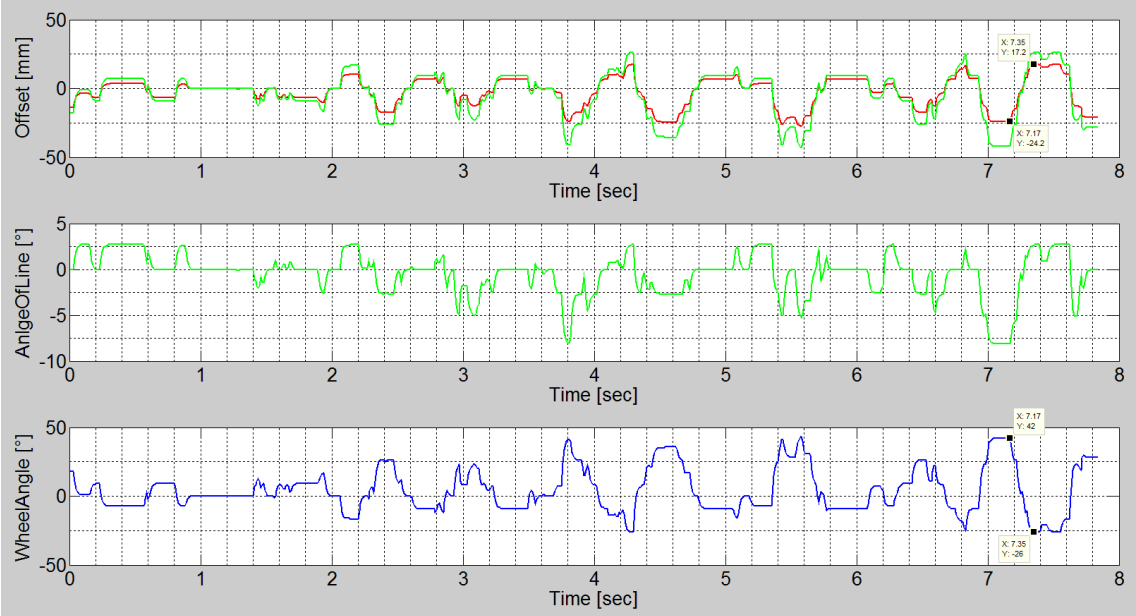
7.3. ábra: A robotautó sebessége vs áramfelvétele

Amint arról már korábban is szó volt, a P sebességszabályozó erősítését kezdetben 100-as értékre állítottuk be, amelyet 7.3. ábra tanulmányozását követően meg kell, hogy változtassunk. A sebességmérésből ekkor még hiányzott a kettes szorzófaktor, amelyet az ábrázoláskor kiegészítettem. A sebesség grafikonja pontosan ezért ilyen lépcsőzetes, másrészről a felbontása egy tizedes jegyik terjed, tehát kerekítési hibák is vannak benne. Ha megnézzük, hogy 0,2 $\frac{m}{s}$-os sebesség változáshoz körülbelül 0,6 másodpercre van szüksége a rendszernek, akkor a maximális gyorsulás értéke 0,33 $\frac{m}{s^2}$. Ez egy kicsit lassú, ezért nyugodtan megnövelhetjük az erősítés értékét 200-ra, vagy akár 300-ra is, mert még a sebesség beállásánál egyáltalán nem látunk túllövést. A motor árama a hatodik másodperctől kezdve alig változik, olyan 7 A körüli értékre áll be. A csúcsértéke picivel több, mint 10 A, amelyet a maximális gyorsuláskor ért el a robotautó. A P változtatásával ez az érték is növekedni fog.

A vonalkövetést elsőkörben csak az ofszet alapján irányította a kocsi, mely 0,25 $\frac{m}{s}$-os érték mellett soha nem csúszott le a vonalról. A k_p értéke ekkor -1,34 volt, amelyet a sebesség négyzetesfüggvényében csökkentettem. Sajnos nem tudtam letesztelni a teljes működési tartományt, így csak hozzávetőleges adataim vannak maximális sebességről vagy éppen a kanyarodási sugárról. A kocsi a tesztek során nem lépte át a 3 $\frac{m}{s}$-os sebességet. A kanyarodási sugara körülbelül 2-2,5 méter.

Amint arról márt korábban is szó esett, az autót menet közben össze lehet kötni egy PC-vel és az adatokat időbélyegesen lehet rögzíteni. Az információcsomagot a DMA 10 ms-onként elküldte a külvilágnak, melyek a következőket tartalmazták:

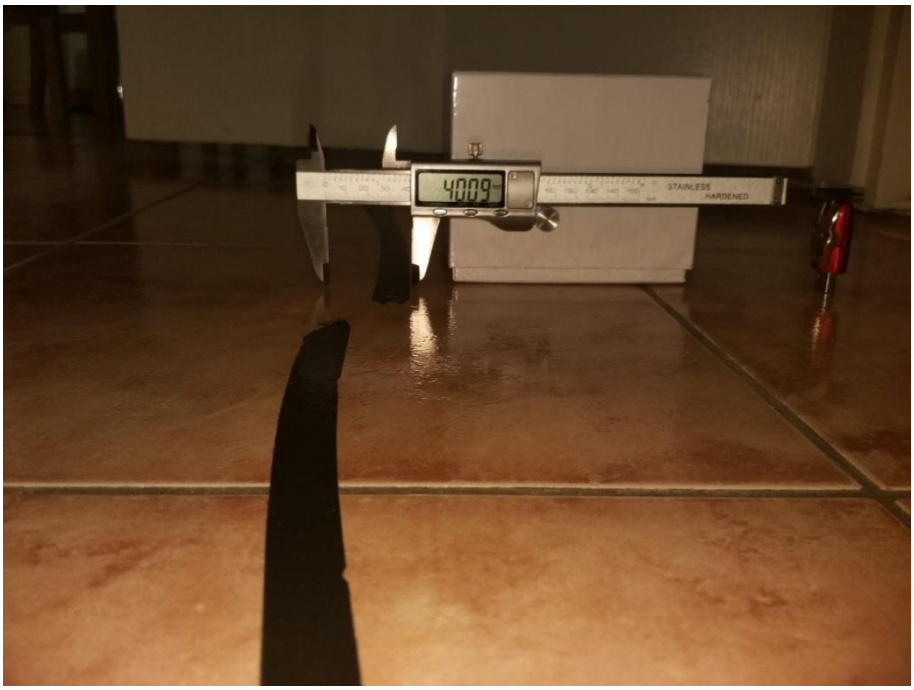
- Vonalinformációk (ofszet, szög)
- Kormánykerék szöge
- A kocsi sebességértéke
- A DC motor árama
- Feszültségek értékek (akkumulátorok, 5 V, 3V3)
- A távolságszenzorok értékei



7.4. ábra: A robotautó ofszet, szög és kormányszög értékei

Ezeknek az információknak a birtokában ki lehet rajzoltani egy olyan görbét, ahol az állapotvisszacsatolásos szabályozót nem csak szemmel, hanem grafikonon is

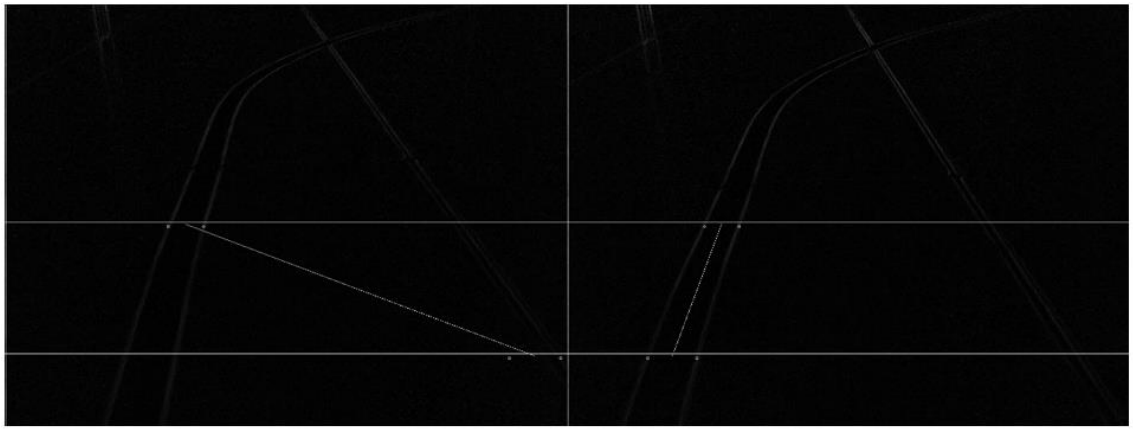
elemezzük. Vizsgáljuk meg a 7.4. ábrát, ahol az idő függvényében analizálhatjuk a vonalkövetést az IR szenzorok alapján. Az ofszet értékét mm-ben olvashatjuk le, melynek értéke nem lehet nagyobb, mint 84 mm, hiszen ekkor a legszélső optikai egység érzékeli a vonalat. A görbét megnézve látszik, hogy maximális értéke 24,2 mm, ami azt jelenti, hogy a harmadik vagy negyedik szenzor érzékeli a vonalat. Egyértelműen kivehető az ábráról, hogy a kis robotautó nem csúszik le a vonalról, sőt inkább maximálisan párhuzamosan halad vele. Nézzük meg az érzékelendő vonal orientációját, mely alig néhányfokos szögben változik. A hetedik másodperc körül a vonalban van egy szakadás, ahol a két a távolságot a 7.5. ábra validálja. Ha ekkor megnézzük a kormányszervó értékeit, akkor azt látjuk, hogy telítődik, hiszen 42 °-os szögben nem tud elfordulni, mert korábban 25 °-ra maximalizáltuk. Az ábráról leolvasható, hogy 0,45 másodperc alatt 50 °-ot fordulnak a kerekek, tehát a szervó beállási ideje terheléssel együtt körülbelül $\frac{0{,}54\ s}{60\ °}$.



7.5. ábra: A követendő vonalszakadása

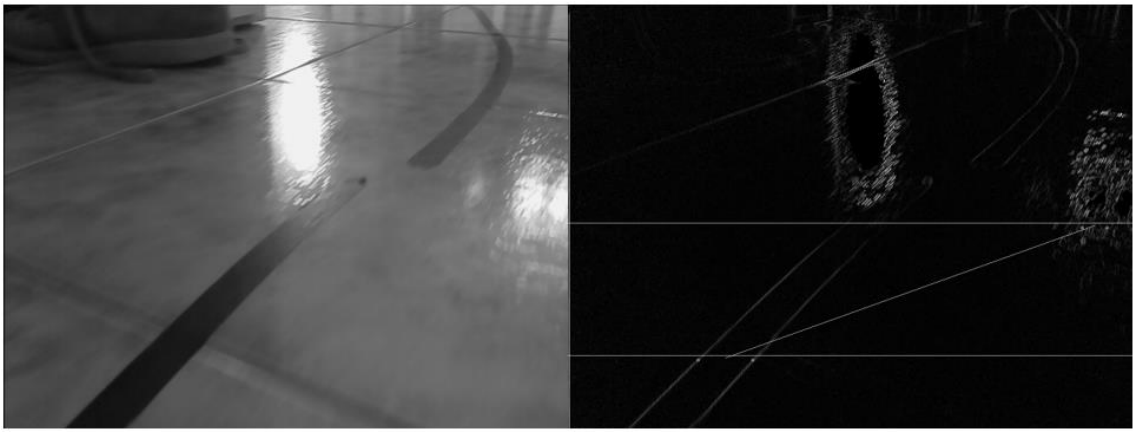
A 7.5. ábrán tolómérő körülbelül 40 mm-et mutat, ami a két vonal közötti különbség. Nagyon közeli értéket tudunk leolvasni a 7.4. ábráról a hetedik másodperc környékén, ahol a robotautó 41,4 mm-t „ugrik” a szenzorok szerint. Az előző ábrából, még a robotautó sebességét is meg lehet becsülni, hiszen tudjuk, hogy 0,18 másodperc alatt 41,4 mm-et változik, ami 0,23 $\frac{m}{s}$-nak felel meg, miközben a valós sebesség 0,25 $\frac{m}{s}$ volt.

Tanulmányozzuk a vonalkövetést a képfeldolgozó egységgel. Amint arról már korábban szó volt, a robotautó álló helyzetében már követte a vonalat. Nézzük meg, hogy az általa látott képeken miket állapíthatunk meg, ha vetünk egy pillantást a következő képekre, ahol két közvetlen egymás utáni felvételt láthatunk.



7.6. ábra: A hibás és a helyes vonaldetektálás

A 7.6. ábra bal oldala a hibás esetet mutatja be, amikor a fugának az intenzitása nagyobb, mint az érzékelendő vonal körül. Ez lehet a megvilágosítás miatt, de fókusz problémára is visszavezethető. Az ábra másik felén már a helyes a végeredmény. A jobb oldali fúga vonalán végig haladva, megláthatunk egy körszerű kiszélesedést, amelyet feltételezhetünk referenciapontoknak. A jobb oldali kép később készült, tehát hozzánk közelebb van ez a pont, mint az ábra baloldalán.



7.7. ábra: A fényviszonyok megváltozása a vonalkövetésre

A 7.7. ábrán a felületen visszaverődő fényeket láthatjuk, melyek nem kívánt módon befolyásolják a vonalkövetést. Megoldás lehet a problémára, ha csak a vonalkörnyezetét böngésszük végig, illetve a kettő helyett három sort használunk a

mátrixból, majd többségi szavazással döntünk, hogy melyek helyesek. Az eredeti nyers kép már alacsonyabb sebességeknél is veszít az élességéből. Ezt kiküszöbölhetjük ha igénybe besszük az élesítő algoritmusokat.

A kamera által megvalósított vonalkövetés konverziós idejét analizáljuk. Korábban szó volt arról, hogy a GPIO-k felhasználásával lehet mérni a kép készítése és az elküldött információ között az eltelt időt. A beépített ciklusszámláló értékét 10-re állítva az oszcilloszkópról 1224[46] ms-ot olvashatunk le, ami azt jelenti, hogy egy kép átlagos konverziós ideje 122 ms. A rendszertervezés elején ezt az értéket 100 ms-ra becsültük.

7.2 Vonalérzékelések összehasonlítása

Amint korábban már írtam, a kis robotautó redundáns vonalkövetére képes, melyek működésükről, algoritmusairól a korábbi fejezetekben szó volt. Nézzük, meg melyik megoldásnak milyen előnyei, illetve hátrányai lehetnek, amelyben a 7.2. táblázat a segítségünkre lehet.

A kamera nagy felbontása miatt, egyértelműen kijelenthető, hogy sokkal pontosabban lehet percepciónálni a vonalat, mert míg az optikai szenzoroknál 1 „pixel” értéke 7 mm, addig a másik megoldásnál ez az érték az 50-szerese. A kérdés az, hogy valóban szükség van-e ilyen pontos felbontásra? A választ nem triviális eldönteni, hiszen a korábbi IR szenzoros vonalkövetésen láttuk, hogy alacsony sebességeknél teljesen párhuzamosan követi a vonalat. A gond akkor jelentkezhet, ha fokozzuk a sebességet, mert ilyenkor a kamerával úgymond jósolni tudunk az IR-es megoldáshoz képest. Például, hogy a vonalban van egy szakadás, amelyet nagy sebességnél nehéz kikerülni, de ha kellően előre tudjuk megtervezni a robotautó pályáját, akkor könnyedén teljesíthetjük ezt az akadályt.

A kamera szoftvere majdnem 200-szor lassabb[47], mint az IR szenzoroké, de ez nem probléma, hiszen sokkal több időnk van „gondolkozni”. Természetesen, ha egy nagyobb kontrollert, vagy optimálisabb szoftvert használnánk, akkor ez az érték tovább csökkenhetne.

---

[46] Lásd függelék [4]

[47] Amint a következő fejezetben látjuk, a képfeldolgozást tovább lehet gyorsítani (talán 50 ms-ra)

	IR szenzorok	Kamera
Pontosság (7 mm)	1 „pixel”	52,3 pixel
Gyorsaság	800 µs	150 ms
Áramfelvétel	120 mA	240 mA
Számítási teljesítmény	alacsony	magas
Elhelyezkedés	5 – 10 mm	5 – 10 cm
Helyigény	154x168x6	20x25x9 [42]
Éjszakai üzemmód	Igen	Csak NOIR esetében
Működési hőmérséklet	-40 °C -> + 85 °C	-40 °C -> + 125 °C[48]
Ár	0,537 x 48 =25,8 $	19,95[49] $

7.2. táblázat: A vonalszenzorok összehasonlítása

Az áramfelvételt tekintve szintén a CNY70-es megoldás a preferált, de ugyanazzal a módszerrel, mint az IR szenzorok nem minden pillanatban használnánk a kamerát.

A kamera egyik legnagyobb előnye, hogy szinte bárhova el lehet helyezni az autón, hiszen kevés helyet foglal a maga 5 cm^2-ével. A másik megoldás –mint ismert- nagy a felület igénye (259 cm^2), illetve csak a kisautó aljára lehet rögzíteni.

Végső soron megvizsgálva a megoldások árait[50], amelyek csak az érzékelőket foglalja magába, akkor látszik, hogy a kamera még mindig olcsóbb, mint az IR szenzorok.

Összefoglalva a vonalkövetést az IR szenzorokkal is meg lehet oldani a feladatot, de a kamerás megoldásban jóval több potenciál van, hiszen egyetlen hátránya, hogy lassabb, amin egy szoftveres optimalizáció könnyedén segíthet. A 7.2. táblázat szinte összes sora egyértelműen a kamera mellett teszi le a voksát, nem is beszélve a továbbfejlesztési lehetőségekről.

[48] Stabil kép csak ezen a tartományon belül: 0 °C -> + 50 °C [42]

[49] Noir esetében ez az érték 29,95 $

[50] Az árakat a Digi-Key Electronics weboldal által kalkuláltattam

8 További fejlesztési lehetőségek

A kocsiban még renget kiaknázatlan lehetőség van, mind hardveresen és mind szoftveresen, de akár még a mechanikát is ide vehetjük. Amint ismert, a kocsi teljes sebességtartományát nem használjuk ki, ami lehetőséget ad arra, hogy kisebb (simább felületű) kerekekkel szereljük fel az autót, amivel sokszorta megnő a tapadás, a gyorsulási értékek, illetve futófelület miatt, sokkal egyenletesebb lesz a z irányú mozgás.

8.1 Hardveres lehetőségek

8.1.1 A kormányszervó 6V-ról üzemeltetése

A kis szervót kénytelen voltam az akkumulátorról üzemeltetni, a korábban ismertet okokból kifolyólag. Összehasonlítottam milyen előnyei, hátrányai lennének akkor, ha a 6 V-os tápfeszültséget terhelné a kormányszervó.

- A hatásfok legalább 10%-al csökkenne, de sokkal könnyebben lehetne érzékelni az esetleges tápfeszültség hibákat, vagy szervó a lekapcsolását vezérelni.
- Akkumulátoros működés esetén, szükség van egy árammérőre, illetve egy kapcsolóelemre, a rövidzár, vagy a túlzott áramfelvétel miatt. Egyrészt sokkal több alkatrészt tartalmazna, másrészt szoftveres diagnosztikák is kellenének a hibátlan működéshez. A legegyszerűbb megoldás is szóba jöhet, ami nem más, mint egy olvadóbiztosíték, amelyet hiba után cserélni kell.

8.1.2 A tápegységek lekapcsolása

A rendszer tápfeszültségeit a megalkotott rendszerben sajnos nem lehet egyesével kapcsolni, amely főleg akkor jönne jól, ha a kocsi biztonsági állapotba kerül. Tegyük fel, hogy a kocsi akadályt érzékelt, ezért lekapcsolja a perifériáit, de a tápfeszültségeket nem tudja kikapcsolni. A RPI erről a lekapcsolásról mit sem sejtve dolgozik tovább, ahelyett, hogy csökkentené a rendszer áramfogyasztását. Sajnos ebben a verzióban maximum azt lehet megvalósítani, hogy a két nagy MCU-t összekötni egy GPIO-n keresztül, melyet a Raspberry PI figyelne. Ha egy előre meghatározott idő alatt nem érkezik GPIO

állapotváltozás, akkor kapcsolja ki magát. Az áramfogyasztása ekkor sem lesz 0 mA, de máris intelligensebben kapcsoltunk le.

Ez a megoldás nem alkalmazható akkor, ha valami oknál fogva a képfeldolgozós MCU tönkremenne. Ekkor az egész rendszert magával rántaná, hiszen az 5 V-on túl nagy lenne az áramfelvétel, ezért a többi egységet is lekapcsolná, majd újraindítaná. Ennek elkerülése végett, az STM32-es MCU-t ki lehetne jelölni fő processzornak, amelynek GPIO-i össze lennének kötve az 5 V-os, 6 V-os tápfeszültséget előállító IC engedélyező lábaival. Így elérhetővé válna a tápfeszültségek kapcsolgatása.

Ha az STM32-es MCU lenne a vezér a tápfeszültségek kapcsolása szempontjából, akkor a kis PIC-et fel kellene úgy programozni, hogy watchdog-ként is működhessen. Az utóbbi ellenőrizné az előbbi működését, mely hiba esetén lekapcsolná az akkumulátor menedzsment segítségével a FET-eket. Így teljesen áramtalanítaná a rendszert, de a diagnosztikákat nem jelezné. Talán egy LED-et lehetne tenni a BAT_1_SW-re, (lásd 4.10. ábra) ha az előbbi eset következik be. Ezzel el lehetne érni, hogyha alacsonyak az akkumulátor feszültségek, akkor mindössze néhány μA-t fogyasztanánk a telepből, ami elhanyagolható.

8.2 Szoftveres potenciál

A DC motor sebességszabályozóját elsőkörben a motort indentifikációjával lehetne tovább fejleszteni, második körben viszont egy PID a szabályzóval. Lehetőség nyílhatna arra is, hogy a meglévő enkódereket, a sebesség pontosabb számításához használnánk fel. A korábban említett kipörgésgátlót, illetve a kerekek blokkolását is lehetne szoftveresen fejleszteni a mozgást érzékelő szenzorokkal (giroszkóp, gyorsulásérzékelő).

A távolságérzékelő szenzorok fogyasztását is lehetne optimalizálni, hiszen nem kell mindegyik ms-ban megmérni, amikor 39 ms-os periódusidővel frissülnek az értékei. Talán elég lenne 300-400 ms-onként bekapcsolni, majd megmérni az analóg jeleket. Ezzel némileg javulna a rendszer hatásfoka.

Érdemes lenne elgondolkodni azon, hogy milyen elemek kellenek még a kocsi 100%-os redundanciájához, mert véleményem szerint nem túl sok hardveres módosítást igényelne.

Talán érdemes lenne utánanézni, hogy szükség van-e egy hőmodellre a motor vezérlő IC-hez, hogy „bármilyen” hőmérséklettartományban elindulhasson a kisautó.

A kommunikáció formáját a két MCU között tovább lehetne fejlesztve CRC-vel, illetve másfajta csomagküldési formával. A jelenlegi UART-os kommunikációt ki lehetne egészíteni az USB 2.0-ás adatküldéssel is, hogy gyorsabb lehessen az adatátvitel.

Az STM32-es szoftverét úgy is meg lehetne írni, hogy a mérések vagy tesztek idejére a Raspberry PI-nek küldené a diagnosztikai adatokat, amelyeket időbélyegesen tárolná, hogy a későbbiekben nem csak a hiba, de maga az útvonal is visszakövethető legyen.

A képfeldolgozásnál különböző éldetekáló algoritmusokat ki lehetne próbálni, illetve csak a vonal közvetlen közelében nézni meg a pixelértékeket, amivel véleményem szerint a 150 ms-os képfeldolgozási idő 50-60%-al csökkenne. A Sobel operátor zajérzékeny, de érdemes lenne egy zajszűrőt (pl. 3x3-mas Medián szűrő) alkalmazni. Ha ehelyett küszöbérték algoritmusokat használunk, akkor nem feltétlen van szükség. Ugyan ennél a résznél maradva, az akadályokat is felismerhetné a kamera segítségével, melynek frissítési ideje akár elérhetné az 1 másodpercet is. A kamera áramfelvételét is lehetne csökkenteni, ha a képfeldolgozó szoftver elején kikapcsoljuk, majd a bekapcsolási idejét levonva, úgy kapcsolnánk vissza, hogy a pontosan a kép készítésékor már a rendelkezésünkre álljon. A kamerát le lehetne cserélni egy Rasperry PI NOIR eszközre, amely éjszaka is képessé tenné a robotautót percepciókra. A Raspberry PI négy processzormagja közül csak egyen fut képfeldolgozás szoftver, amelyet ki lehetne terjeszteni négyre. A telemetria adatokat elküldhetné Bluetooth-on vagy akár Wi-fi-n is, attól függően, hogy mennyire van közel a telefonunk vagy laptopunk, de akár rögzíthetné is egy fájlba, hiszen az adattárolója 32 GB-os.

9 Összefoglalás

A rendszer a valós idejű jelfeldolgozás keretében képes a laterális, illetve a longitudinális mozgásokra, miközben figyelembe veszi a környezeti változásokat. A nagy teljesítményű mikrokontroller, a nyílt forrású valós idejű képfeldolgozó rendszer (Open CV) segítségével kiszámítja az útvonalat, majd továbbítja a másik vezérlőegységnek, amely ennek függvényében irányítja az autót. A kamera képének minőségét a modellautó környezte épp úgy befolyásolja, mint a sebessége, amelyet sok fejlesztési lehetőséget tartogat magában.

A robotautón végzett tesztek, mérések alapján megállapítható, hogy a rendszer megfelelt a diplomaterv kiírásában definiált követelményeknek, illetve néhány ponton túl is teljesítette azokat. Az akkumulátor menedzsment segítségével például az eredeti üzemidő a többszörösére emelkedett, valamint redundáns vonalkövetést valósít meg a tervezett eszköz. A robotautó tervezése, mérése közben egyértelművé vált, hogy a képfeldolgozás igényli a legnagyobb számítási kapacitást, amely a hatalmas adatmennyiséget (211 Mbps) mindössze néhány tíz bitté (1,2 kbps) konvertálja át. A mérések, a hibakeresések nem voltak triviálisak a modellautó mozgása miatt. A kormányszervó szögének túl gyakori és túl nagy változtatása hatalmas terhelőáramokat generál az adott tápegységen, melyek működését, illetve vezérlését jól végig kell gondolni a tervezés során. A diagnosztikák implementálása és kijelzése kiemelkedő szerepet tölt be egy ilyen bonyolultságú beágyazott rendszerben. Ezen kívül esszenciális, hogy a rendszertervhez, egy jól megszabott időtervet konstruáljunk, mely szignifikánsan befolyásolja a fejlesztést.

Ábrajegyzék

Táblázatjegyzék

Irodalomjegyzék

[1] BME AUT: RobonAUT 2016
http://www.robonaut.hu/sites/default/files/robonaut_2016_elozetes.pdf
(revision: 16.11.2015)

[2] Allegro: Automotive Full Bridge MOSFET Driver
http://pdf1.alldatasheet.com/datasheet-pdf/view/239890/ALLEGRO/A3941.html
(revision: Rev. 5)

[3] ST: Automotive fully integrated H-bridge motor driver
http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00234623.pdf
(revision: Rev. 9, 09.2013)

[4] Infineon: AUIRF2804 datasheet
http://www.irf.com/product-info/datasheets/data/auirf2804.pdf
(revision: 30.09.2015)

[5] WPI: Novel Interfaces for Interactive Environments
http://web.cs.wpi.edu/~gogo/courses/imgd3100_2013a/slides/imgd3100_08_Physical_Feedback.pdf
(revision: 16.11.2015)

[6] Futaba: Digital FET Servos
http://www.futabarc.com/servos/digitalservos.pdf
(revision: 16.11.2015)

[7] Drive Technologies
http://www.compumotor.com/catalog/catalogA/A31-A33.pdf
(revision: 16.11.2015)

[8] Pololu: Servo control interface in detail
https://www.pololu.com/blog/17/servo-control-interface-in-detail
(revision: 09.01.2011)

[9] GS-D9257 datasheet
https://www.pololu.com/file/0J572/GS-D9257.pdf
(revision: 16.11.2015)

[10] Vishay:Reflective Optical Sensor with Transistor Output
http://www.vishay.com/docs/83751/cny70.pdf
(revision: 1.8, 30.07.12)

[11] ROHM: RPR-220 datasheet
http://rohmfs.rohm.com/en/products/databook/datasheet/opto/optical_sensor/photosensor/rpr-220.pdf
(revision: Rev 1.1)

[12] TT OPTEK: Reflective Object Sensor Type OPB733TR
http://optekinc.com/datasheets/OPB733TR.pdf
(revision: Issue A.1 3/09)

[13] Vishay: TCRT5000 datasheet
http://www.vishay.com/docs/83760/tcrt5000.pdf
(revision: Rev 1.7)

[14] Sharp: GP2Y0A02YK datasheet
http://www.sharpsme.com/download/GP2Y0A02YK-DATA-SHEETPDF
(revision: 17.11.2015)

[15] PING))) Ultrasonic Distance Sensor (#28015)
https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf
(revision: v2.0 02.04.2013)

[16] OpenCV
http://opencv.org/
(revision: 18.11.2015)

[17] Raspberry PI faqs
https://www.raspberrypi.org/help/faqs/
(revision: 25.04.2016)

[18] Microchip: USB3343 datasheet
http://ww1.microchip.com/downloads/en/DeviceDoc/334x.pdf
(revision: Rev 1.2 02.08.13)

[19] TI: TPD4S014 datasheet
http://www.ti.com/lit/ds/symlink/tpd4s014.pdf
(reviseion: 09.2015)

[20] TI: SN74LV4051 datasheet
http://www.ti.com/lit/ds/symlink/sn74lv4051a.pdf
(revision: 04.2005)

[21] Freescale Semiconductor: MMA8451Q dataheet
http://www.freescale.com/files/sensors/doc/data_sheet/MMA8451Q.pdf
(revision: 9, 11.2014)

[22] Microchip: PIC10(L)F320/322 datasheet
http://ww1.microchip.com/downloads/en/DeviceDoc/40001585B.pdf
(revision: B 02.2014)

[23] TI: TPS65251 datasheet
http://www.ti.com.cn/cn/lit/ds/symlink/tps65251.pdf
(revision: 12.2014)

[24] ST: STM32F429ZI datasheet
http://www.st.com/st-web-

ui/static/active/en/resource/technical/document/datasheet/DM00071990.pdf
(revision: Rev 4)

[25] ST: L3GD20H datasheet
http://www.st.com/web/en/resource/technical/document/application_note/DM00119037.pdf
(revision: 04.2015)

[26] Panasonic: EEUFR1A472S datasheet
http://www.farnell.com/datasheets/1897635.pdf
(revision:

[27] The Effect of Forced Air Cooling on Heat Sink Thermal Ratings
http://www.crydom.com/en/tech/hs_wp_fa.pdf
(revision: 21.11.2015)

[28] Eurocircuits: PCB design classification overview
http://www.eurocircuits.hu/images/stories/ec13/ec-classification-english-3-2013-v3-final.pdf
(revision: 18.12.2014)

[29] YUV vs RGB – Choosing a Color Space for Human-Machine Interaction
https://fedcsis.org/proceedings/2014/pliks/206.pdf
(revison: 2014, ACSIS, Vol. 3)

[30] Littelfuse: Selecting an Appropriate ESD Device
http://www.digikey.com/Web%20Export/Supplier%20Content/Littelfuse_18/PDF/LF_SelectingESDDevice.pdf?redirected=1
(revision: 2009)

[31] ST: UM1467 User manual
http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00037368.pdf
(revision: Rev 1)

[32] Performance Analysis of Canny and Sobel Edge Detection Algorithms in Image Mining
http://www.ijircce.com/upload/2013/october/13Performance.pdf
(revision: Vol. 1, Issue 8, 10.2013)

[33] PERFORMANCE ANALYSIS OF SOBEL EDGE FILTER ON HETEROGENEOUS SYSTEM USING OPENCL
http://esatjournals.net/ijret/2014v03/i15/IJRET20140315011.pdf
(revision: Volume: 03 Special Issue: 03 | 05.2014)

[34] Performance Analysis of Different Edge Detection Techniques for Image Segmentation
http://www.indjst.org/index.php/indjst/article/viewFile/72946/56716
(revision: Vol 8(14) | July 2015)

[35] Raspberry Pi 2 vs Banana Pi vs x86 vs x64 unRAR PAR2 Benchmarks
http://www.htpcguides.com/raspberry-pi-2-vs-banana-pi-vs-x86-vs-x64-unrar-par2-benchmarks/
(revision: 18.04.2016)

[36] DC Motor control Reversing
http://www.crydom.com/en/tech/newsletters/jan%202013%20-%20solid%20statements%20-%20dc%20motor%20reversing.pdf
(revision: 01.2013)

[37] Direct Current Motors
http://web.alfredstate.edu/albaflr/Spring08/ELET4143/LEC%5CChp5-DC-motors.pdf
(revision: Spring 2008)

[38] Discovery kit for STM32F429/439 lines - with STM32F429ZIT6 MCU
https://www.element14.com/community/docs/DOC-67574/l/discovery-kit-for-stm32f429439-lines-with-stm32f429zit6-mcu
(revision: 12.05.2016)

[39] Kiss Domokos, Kolumbán Sándor:Vonalkövető autó rendszermodellje és szabályozása
BME Automatizálási és Alkalmazott Informatika Tanszék
(revision: 10.2014)

[40] SpazzOut for Python vs C on Raspberry PI B+
http://spazztech.net/python-vs-c-on-rpi.html
(revision: 05.2016.)

[41] Benchmarking Raspberry PI GPIO speed
http://codeandlife.com/2012/07/03/benchmarking-raspberry-pi-gpio-speed/
(revision: 15.02.2015)

[42] OV5647 datasheet
http://www.seeedstudio.com/wiki/images/3/3c/Ov5647_full.pdf
(revision: 11.03.2009)

[43] NEC: ISSP/90 nm ISSP90 - Standard Family
http://www2.renesas.eu/_pdf/A16971EU3V0PBE1.PDF
(revision: 05.2016)

[44] Adrian Ford and Alan Roberts: Colour Space Conversions
http://www.poynton.com/PDFs/coloureq.pdf
(revision: 11.08.1998)

[45] Daiel Kim: Sobel Operator and Canny Edge Detector ECE 480 Fall 2013 Team 4
http://www.egr.msu.edu/classes/ece480/capstone/fall13/group04/docs/danapp.pdf
(revision: 2013)

[46] Stefano Mattoccia: Stereo Vision: Algorithms and Applications
http://vision.deis.unibo.it/~smatt/Seminars/StereoVision.pdf
(revision: 01.2015)

[47] K. Murawski: Method of measuring the distance to an object based on one shot obtained from a motionless camera with fixed-focus lens
http://przyrbwn.icm.edu.pl/APP/PDF/127/a127z6p04.pdf
(revision: 12.05.2014)

Függelék

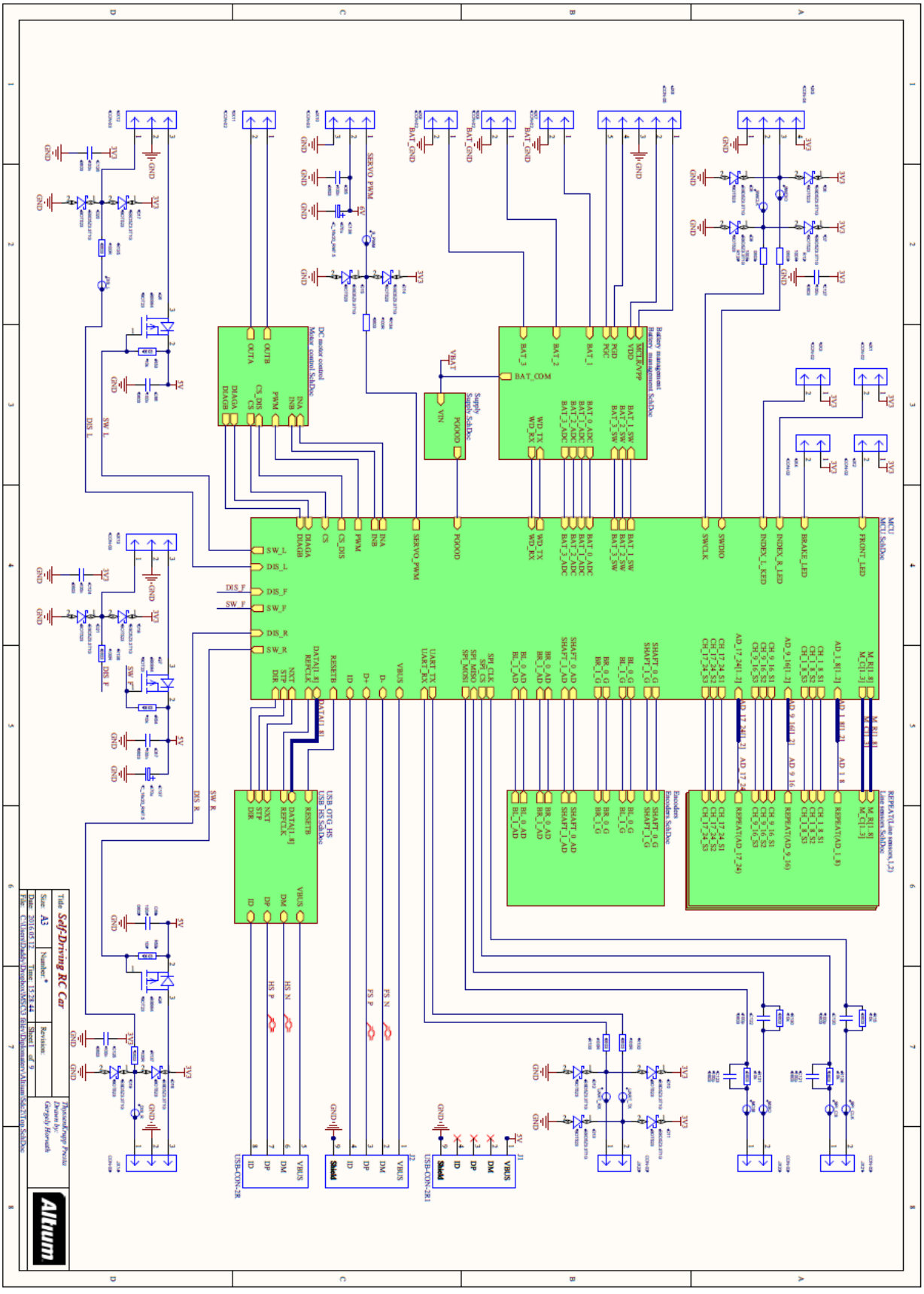
[1] Kapcsolási rajz

[2] Nyomtatott áramkör panel terve

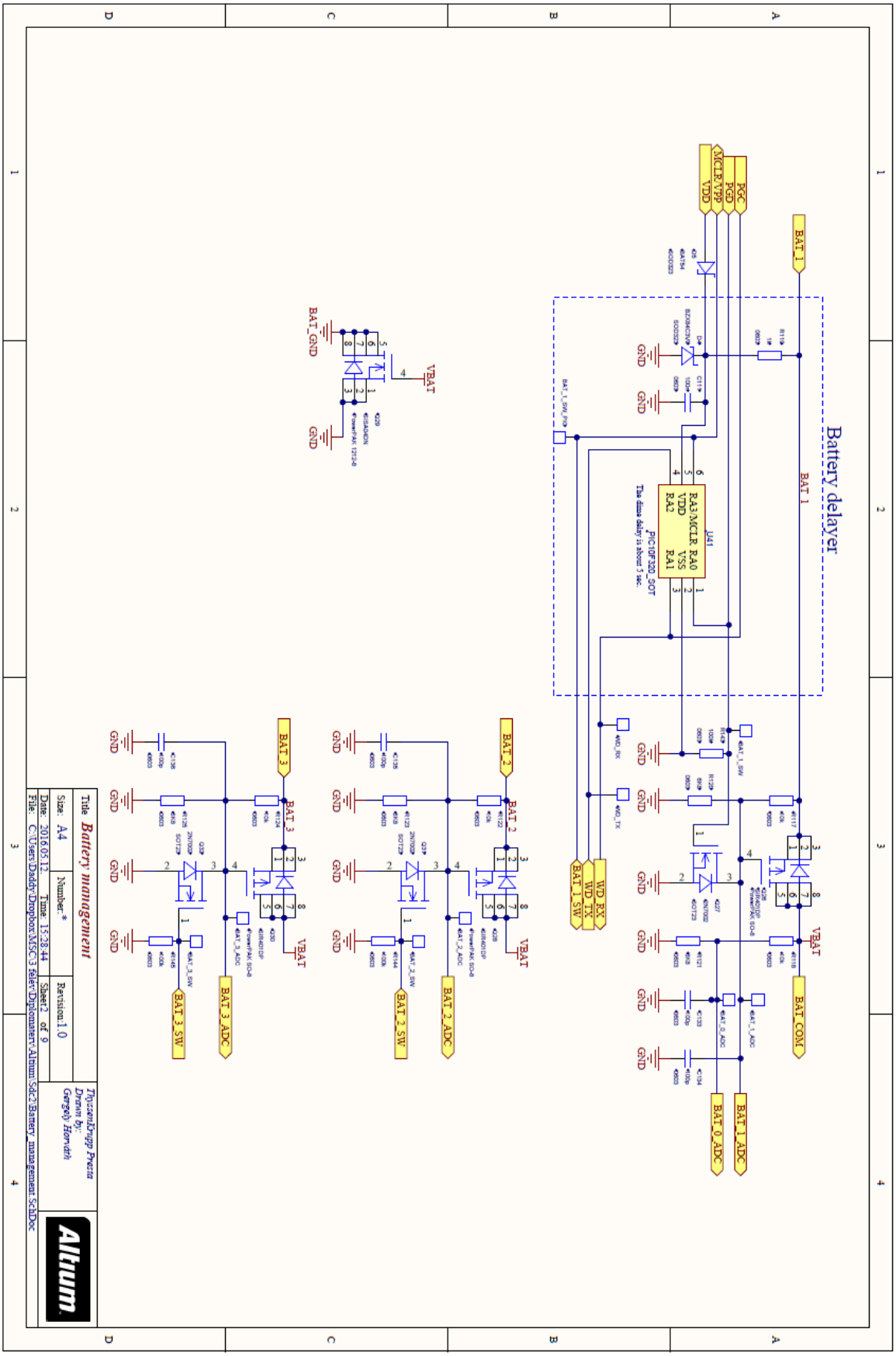
[3] A robotautó a valóságban

[4] Kamera konverziós ideje oszcilloszkóppal

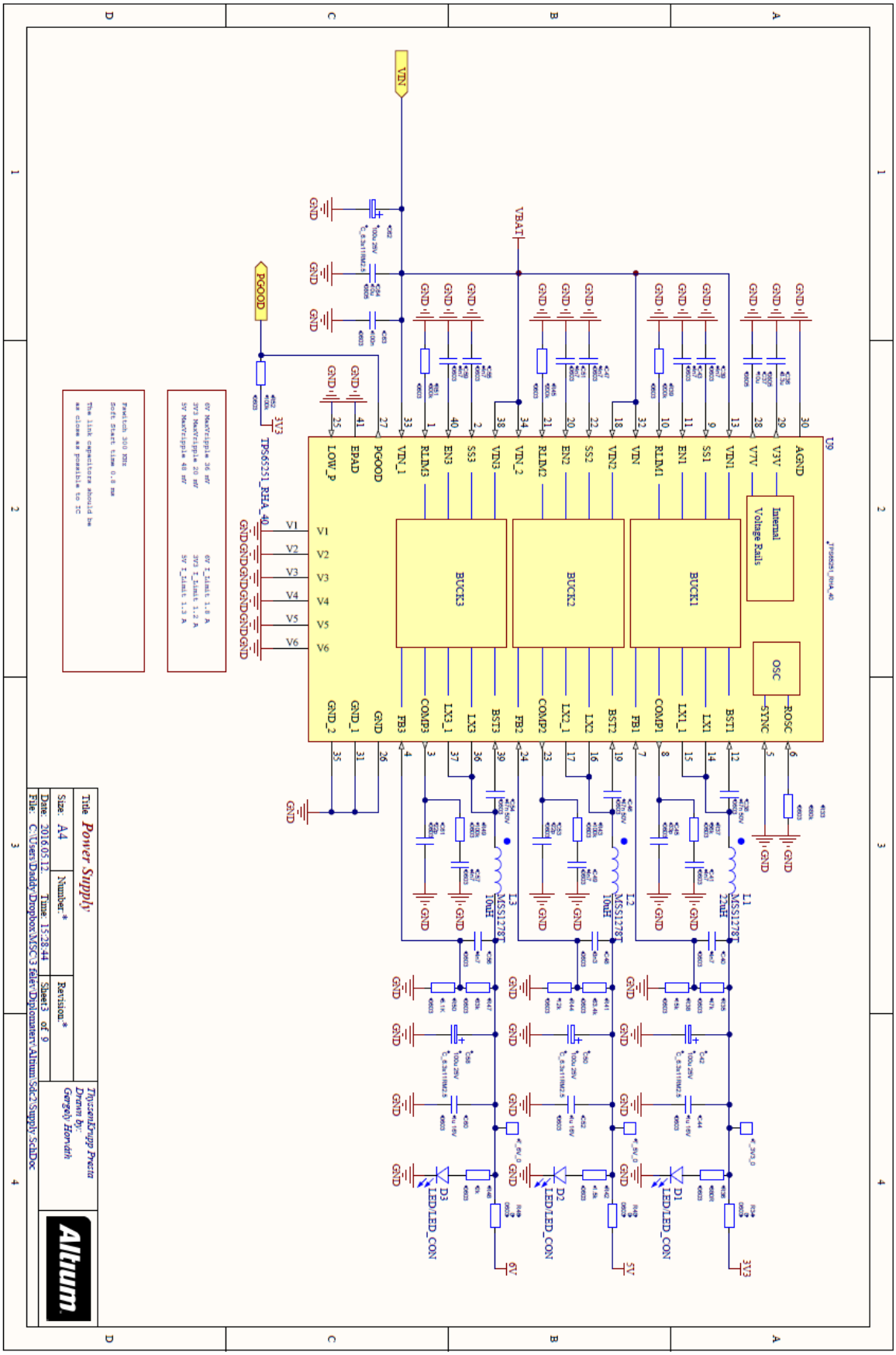
[1] Kapcsolási rajz



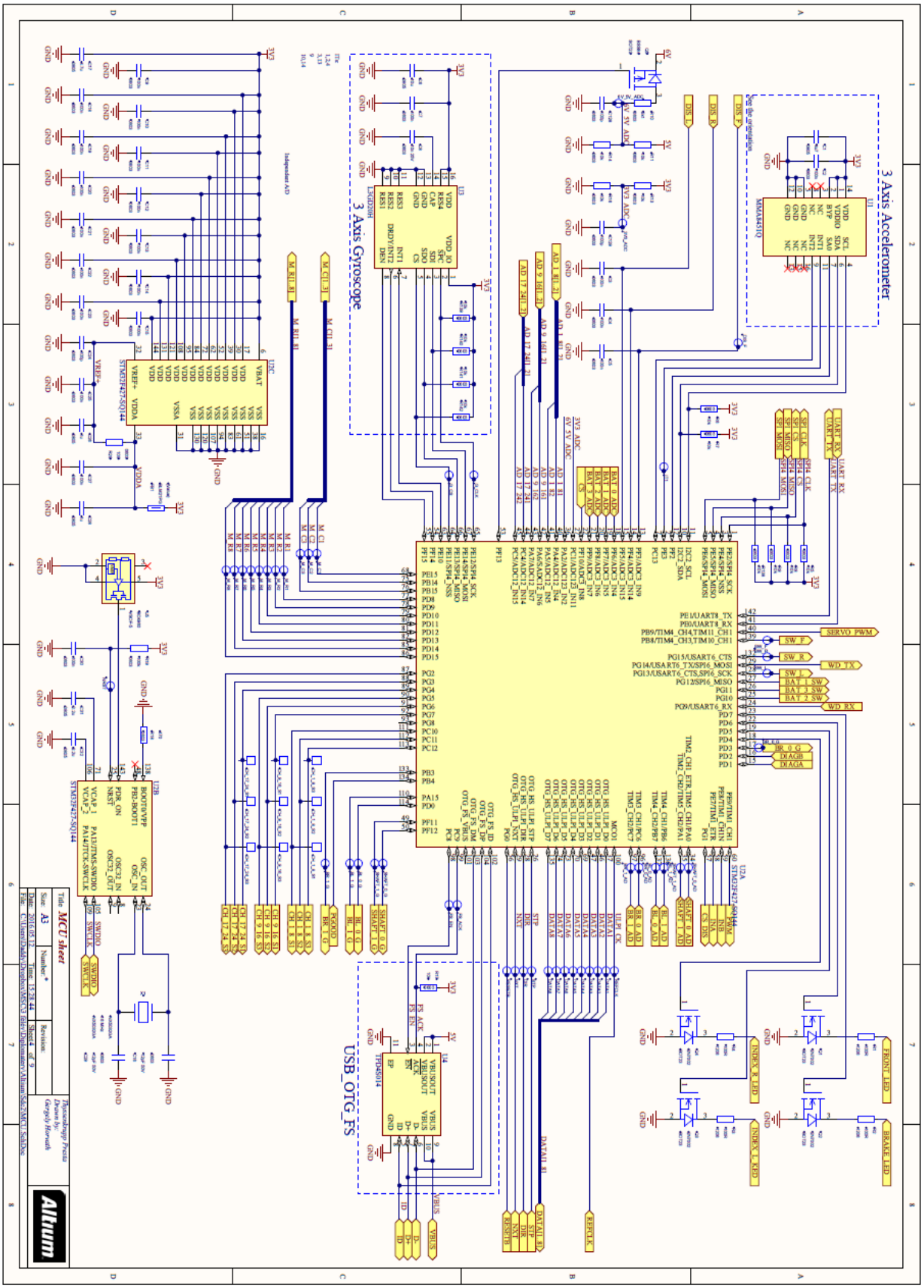
1. (Függelék)ábra: The block sheet



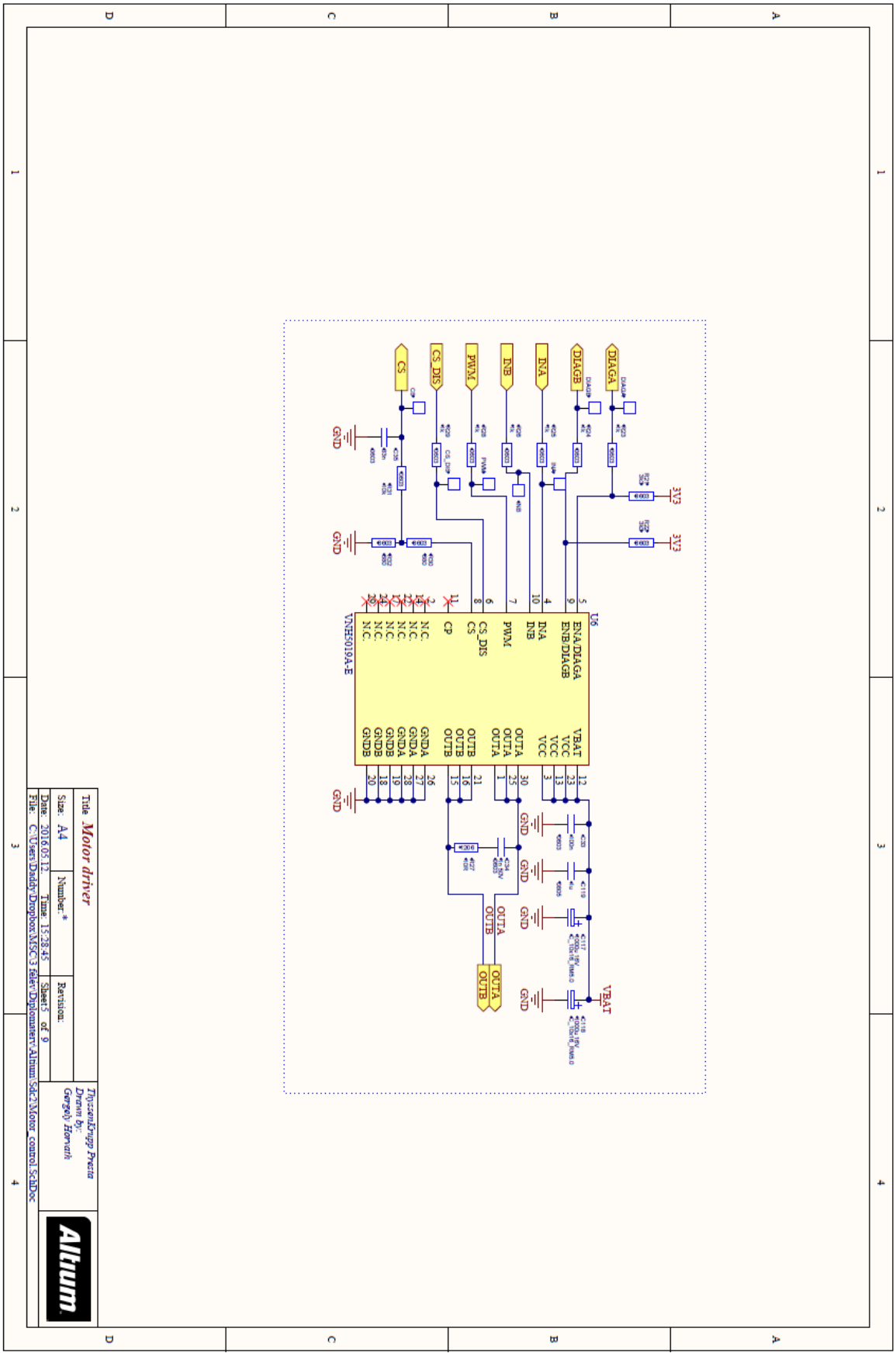
2. (Függelék)ábra: Battery management



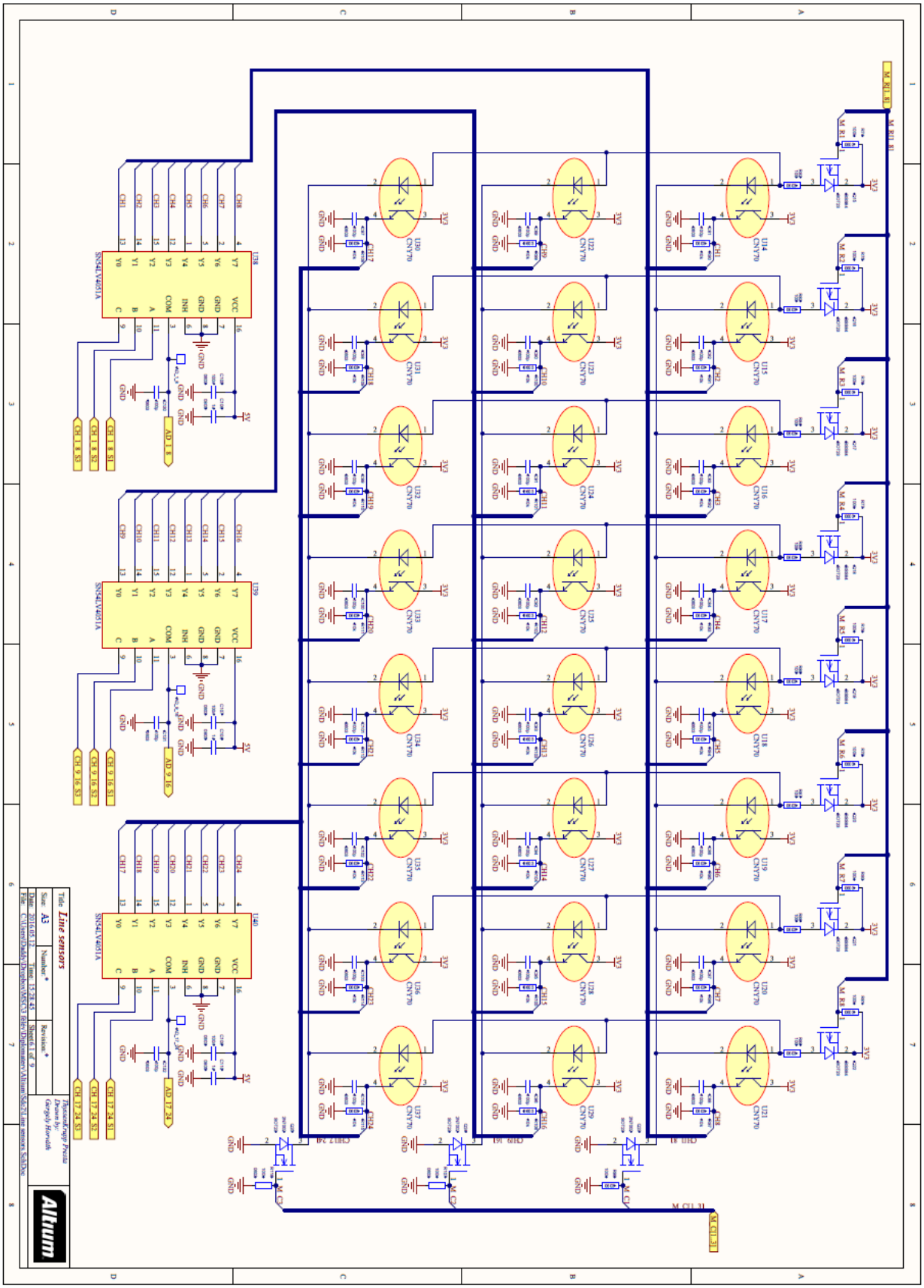
3. (Függelék)ábra: Power Supply



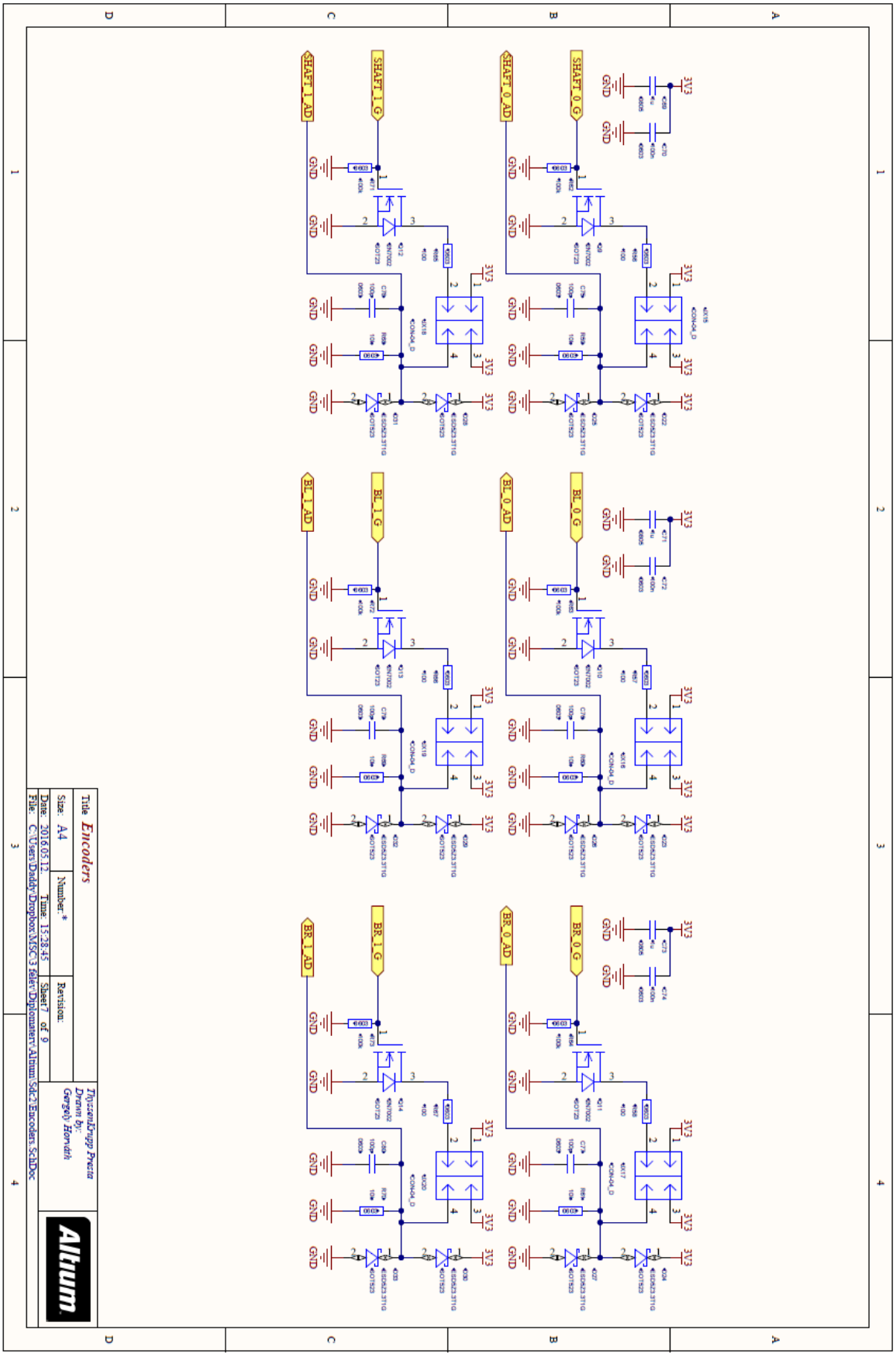
4. (Függelék)ábra: MCU sheet



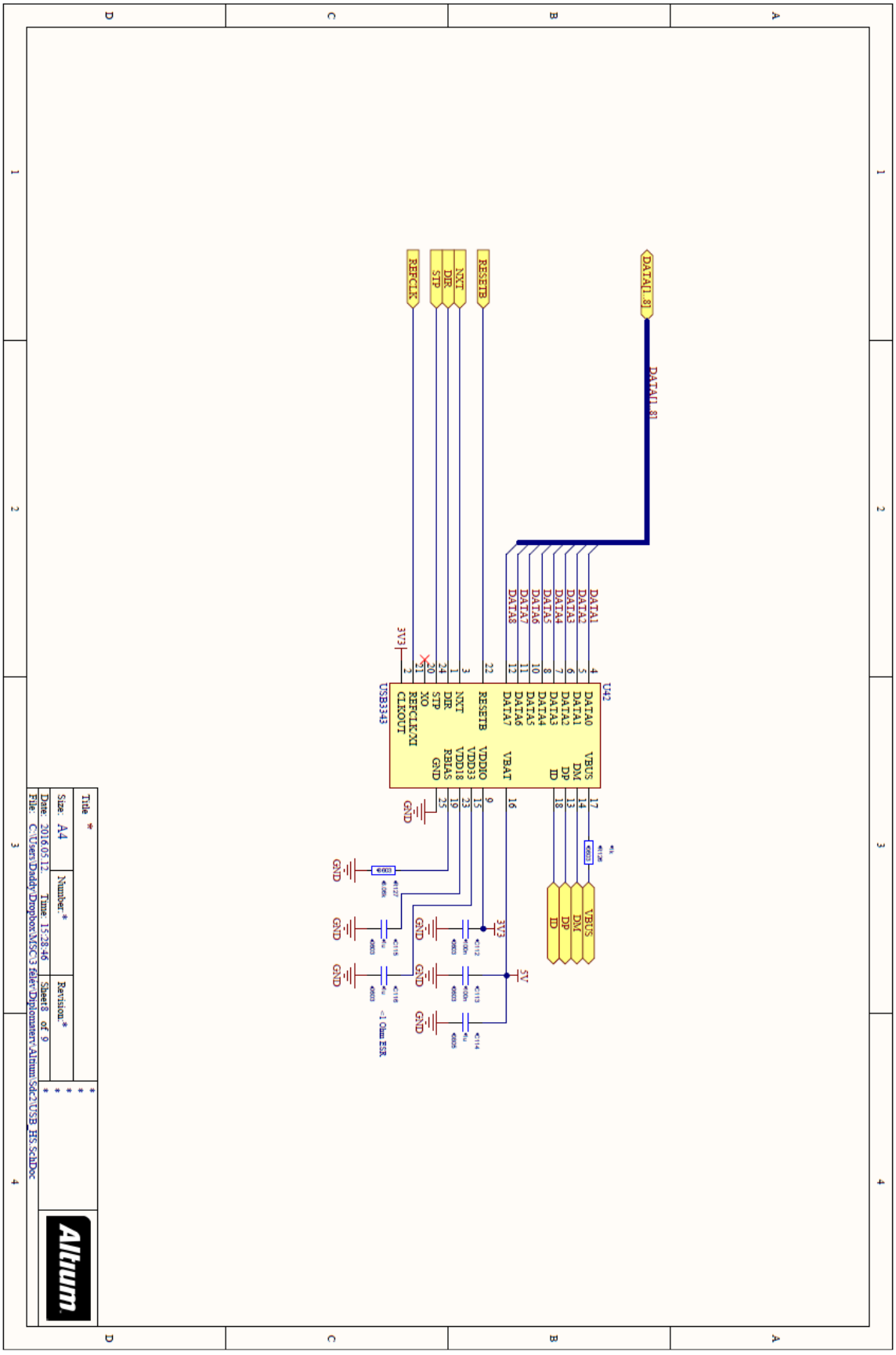
5. (Függelék)ábra: Motor Driver



6. (Függelék)ábra: Line Sensors

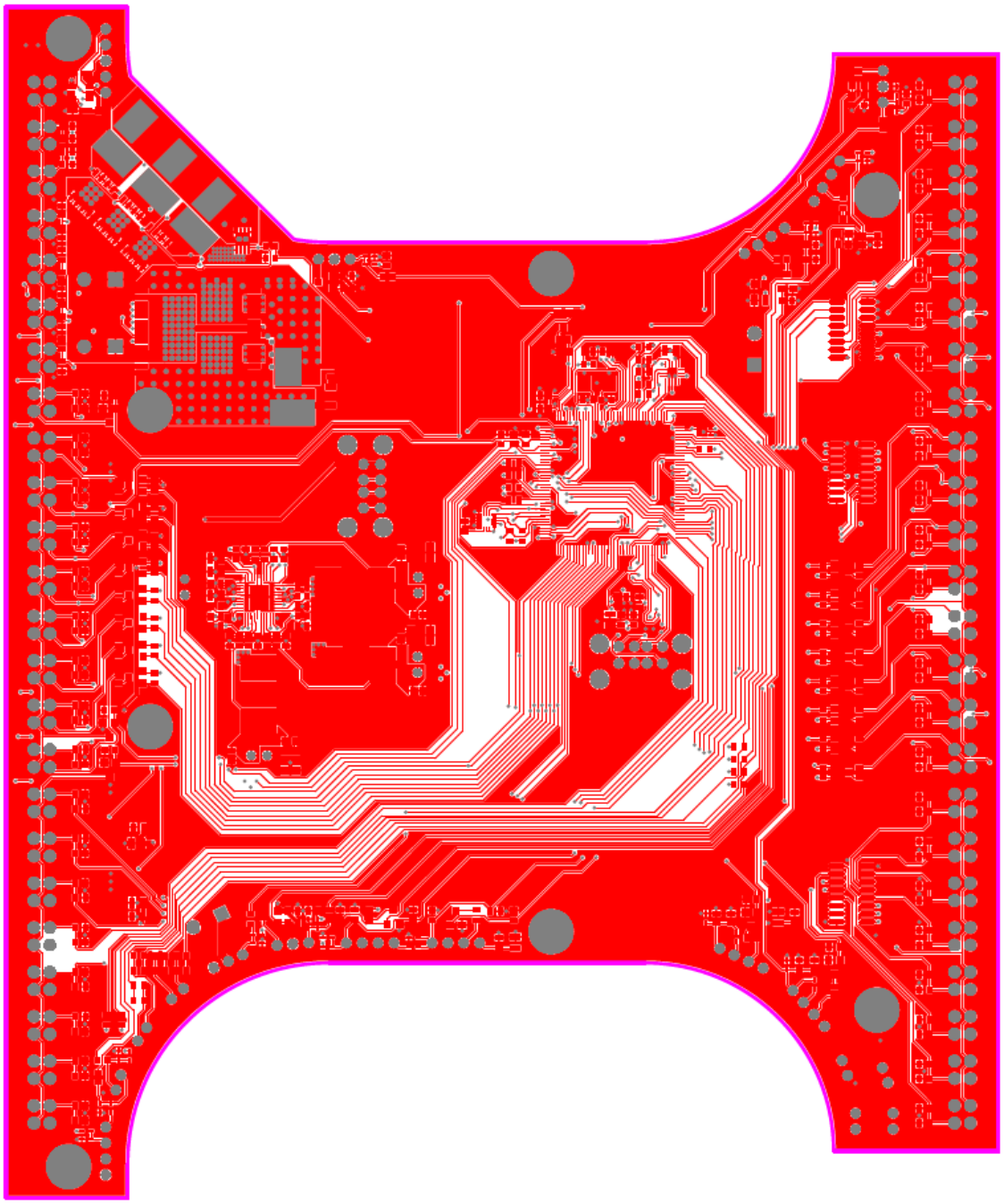


7. (Függelék)ábra: Encoders

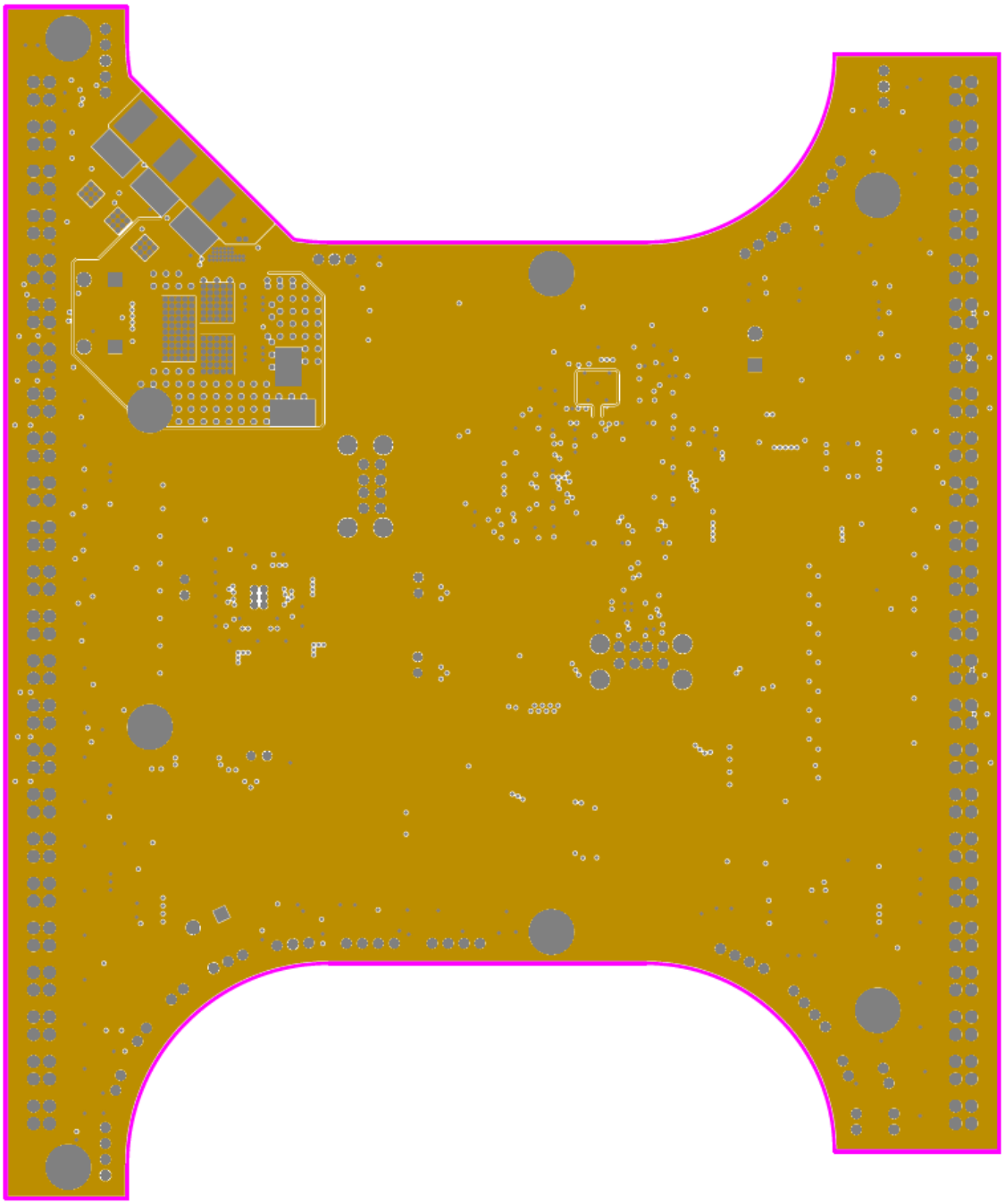


8. (Függelék)ábra: USB 2.0 transcaiver

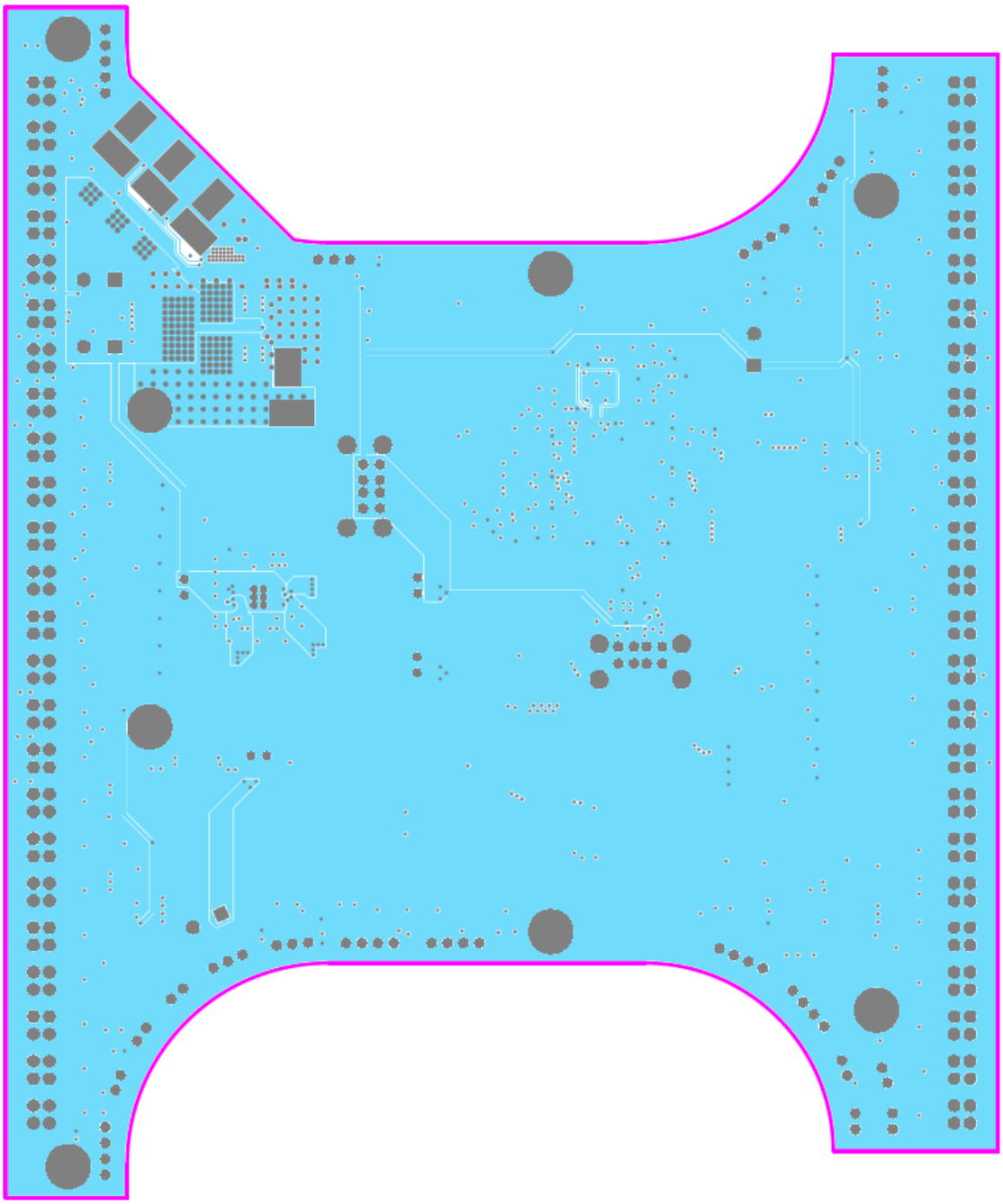
[2] Nyomtatott áramkör panel terve



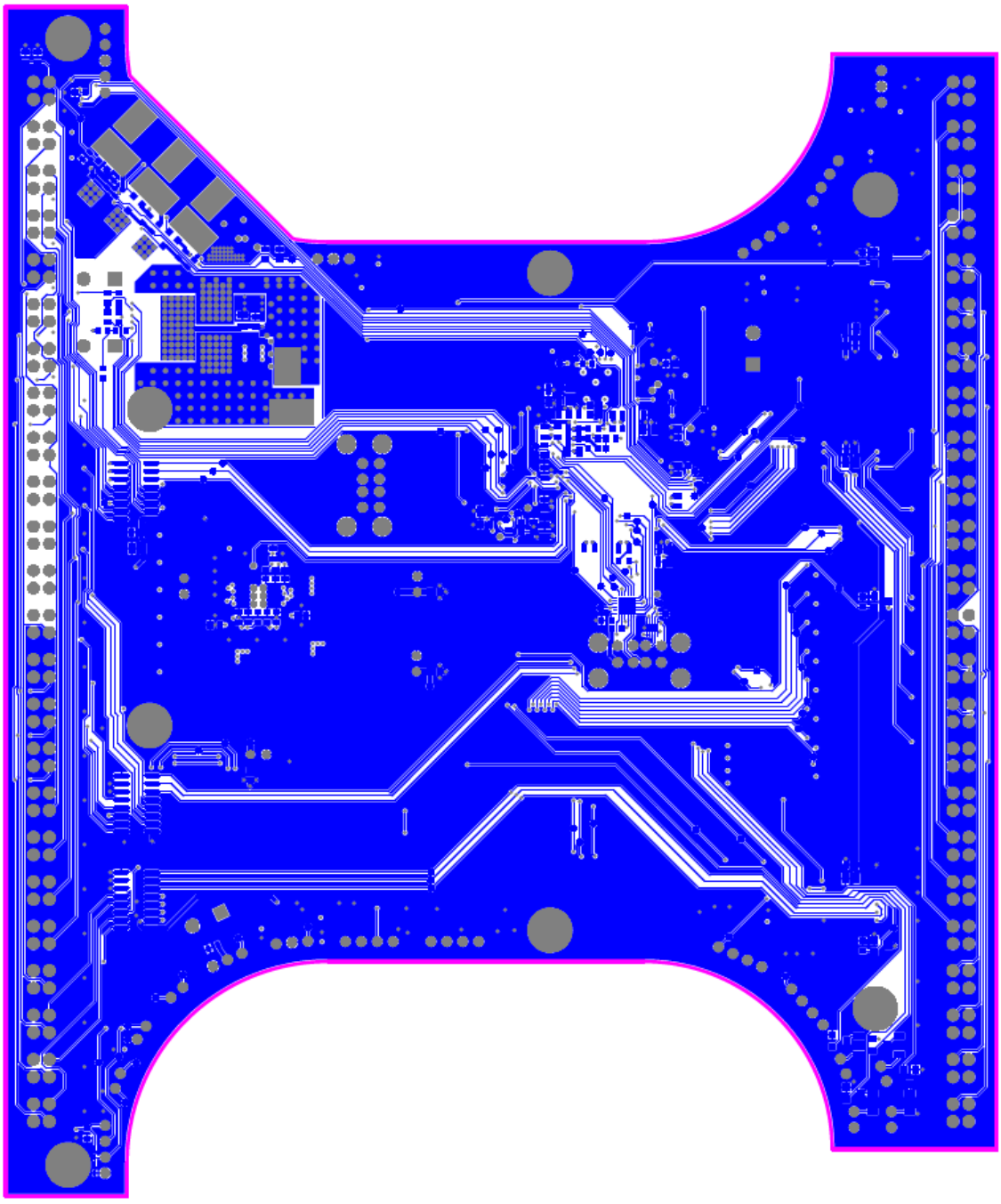
9. (Függelék) ábra: A NYÁK TOP oldali rajzrétege



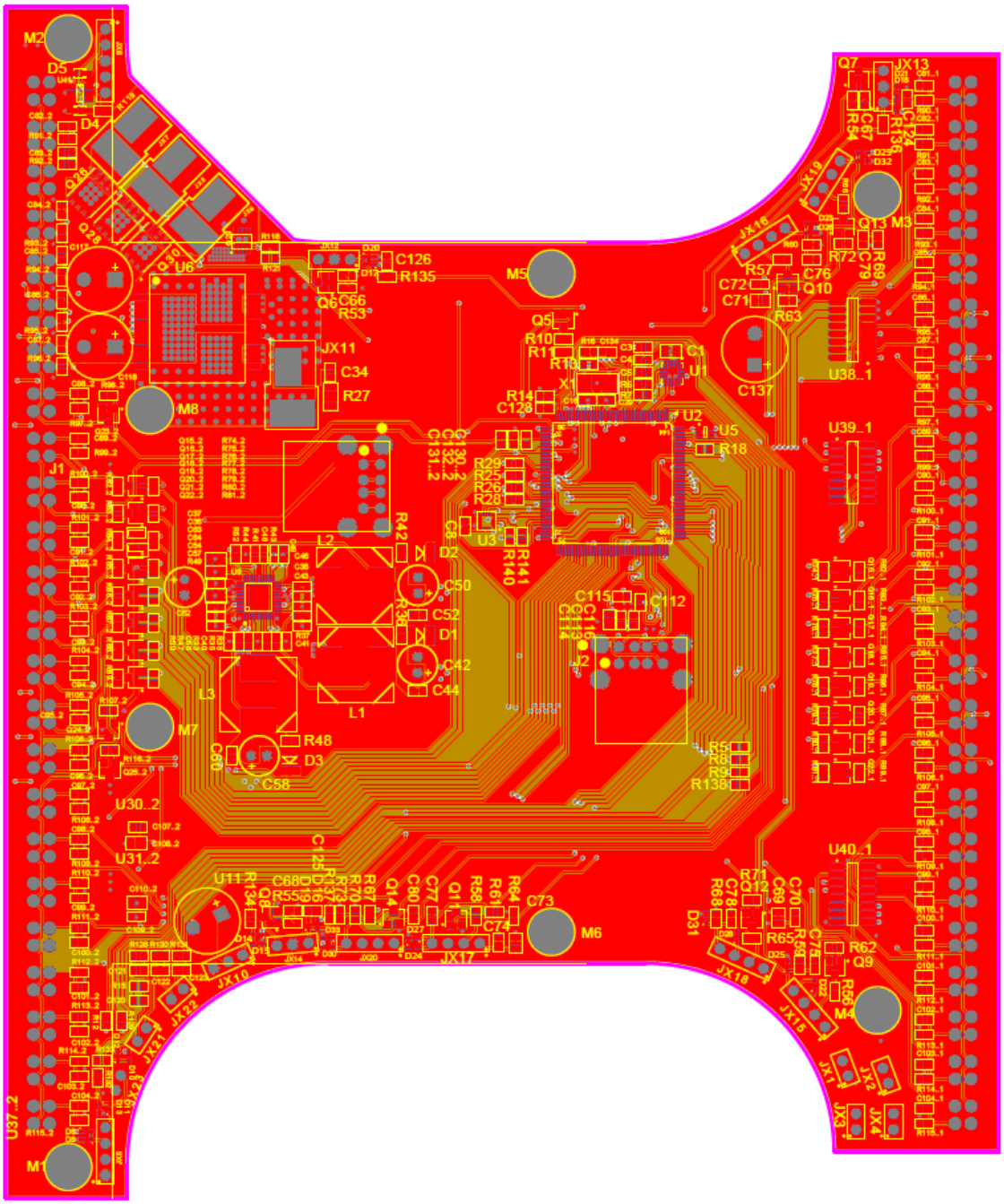
10. (Függelék)ábra: A NYÁK MID1 oldali rajzrétege



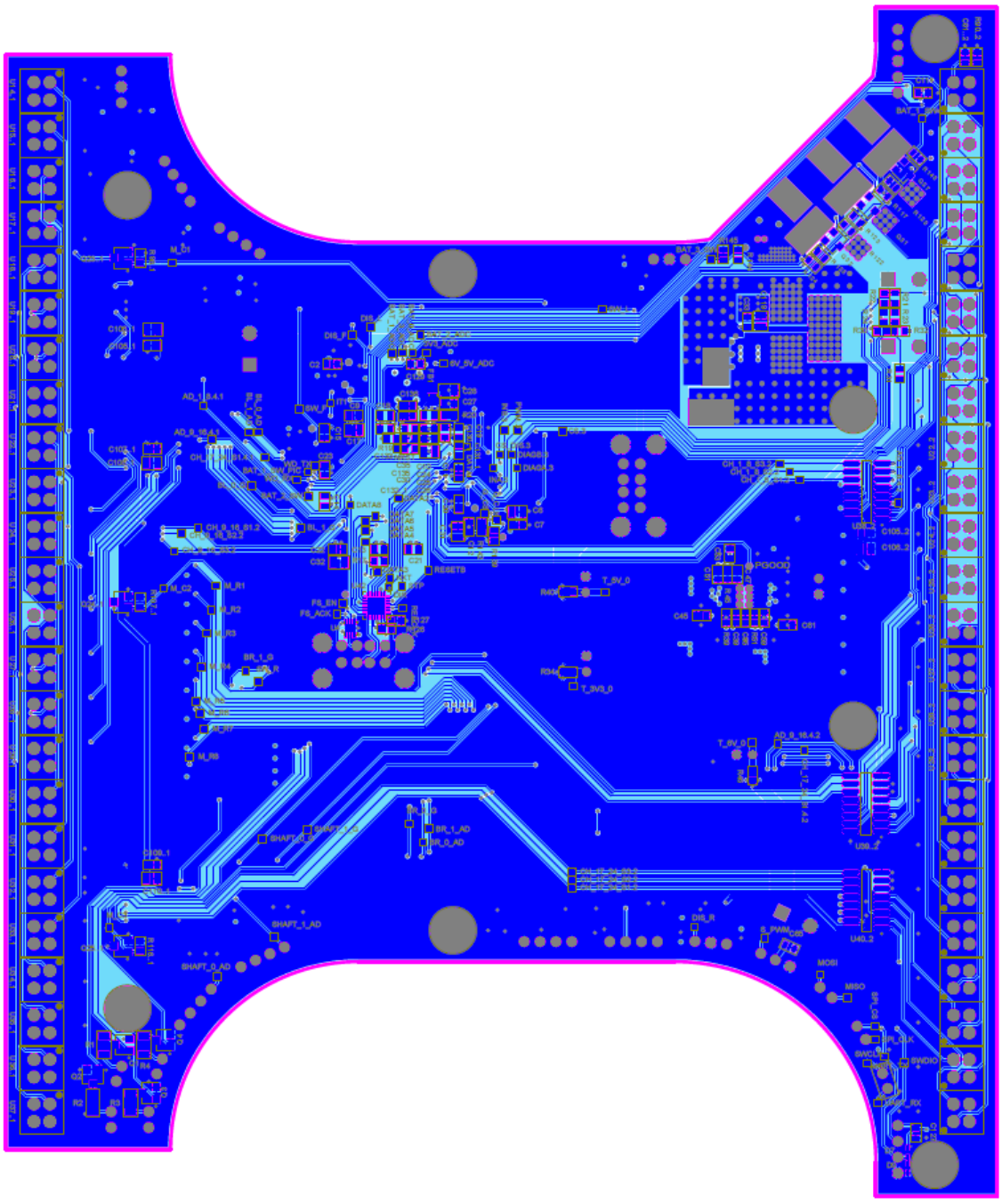
11. (Függelék)ábra: A NYÁK MID2 oldali rajzrétege



12. (Függelék)ábra: A NYÁK BOTTOM oldali rajzrétege



13. (Függelék)ábra: A NYÁK TOP- és szitarétege

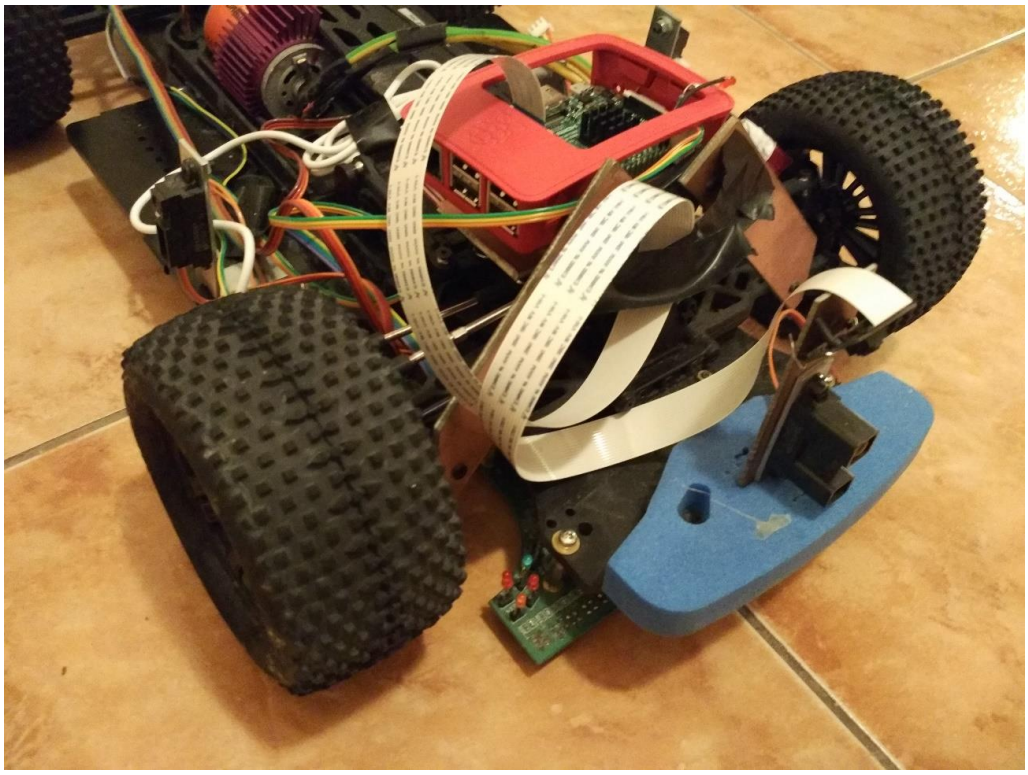


14. (Függelék)ábra: A NYÁK BOTTOM- és szitarétege

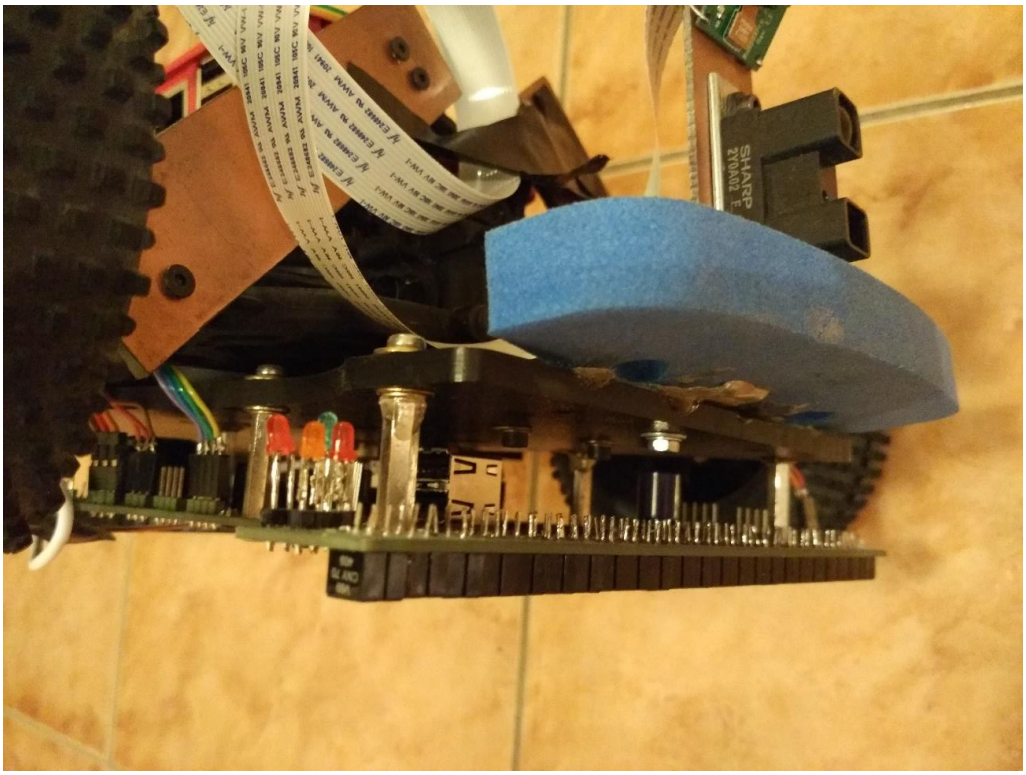
[3] A robotautó a valóságban

15. (Függelék)ábra: A robotautó szemből

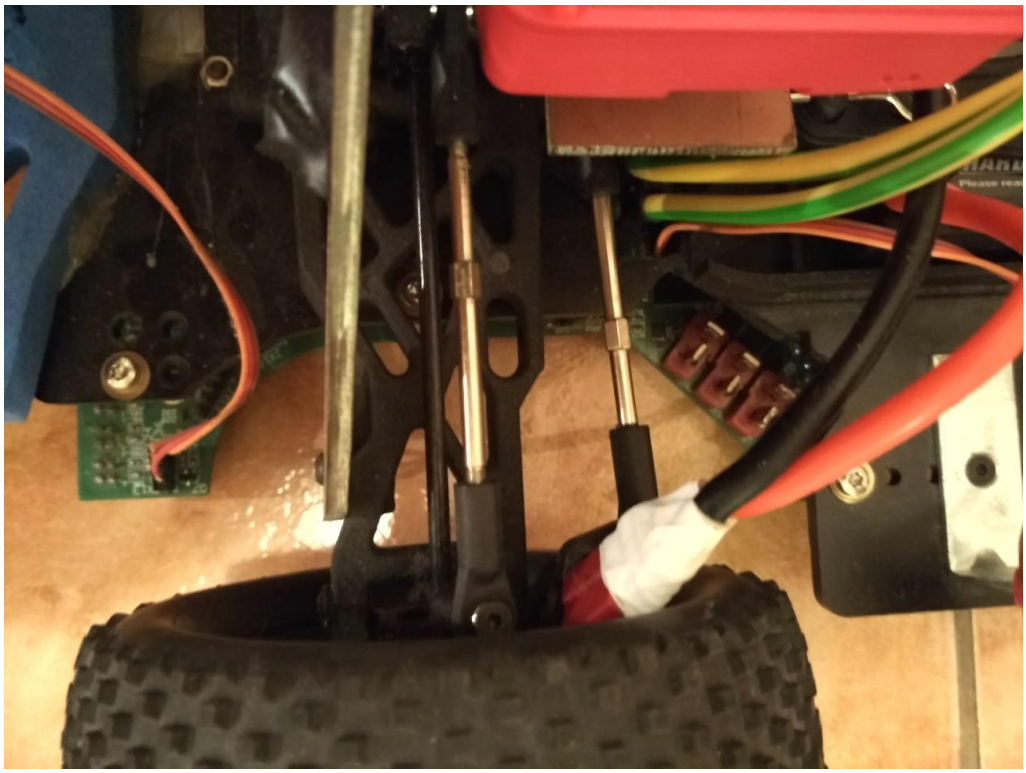
16. (Függelék)ábra: A modellautó felülnézetből



17. (Függelék)ábra: Burkolat nélkül

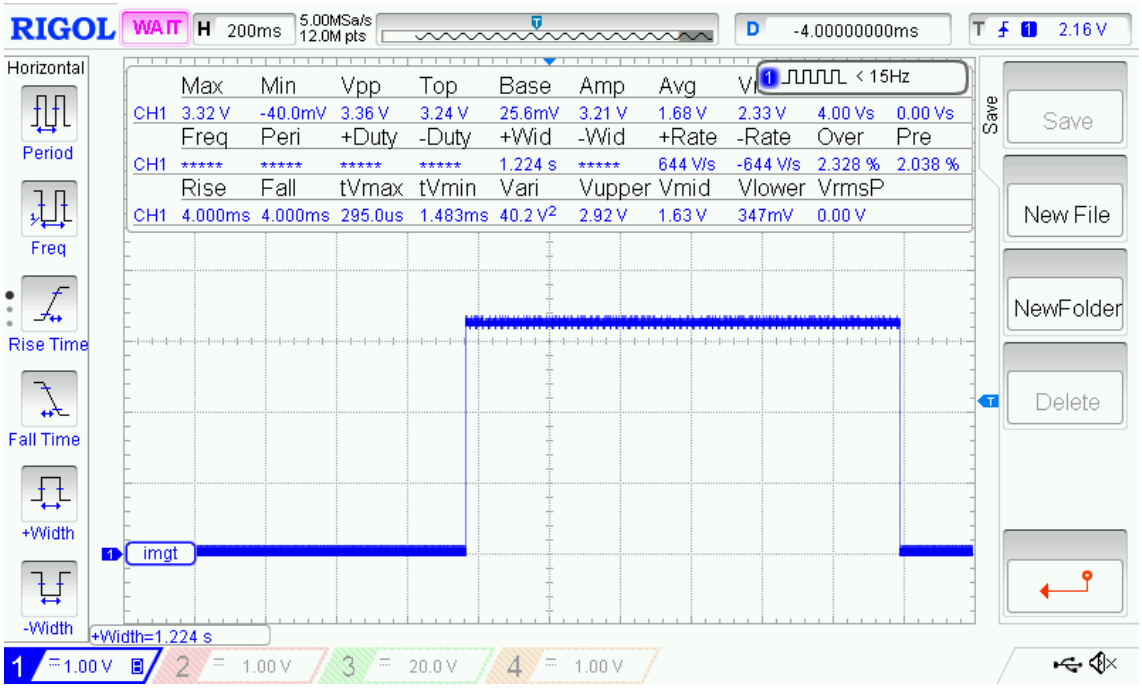


18. (Függelék)ábra: Oldalnézetből burkolt nélkül



19. (Függelék)ábra: Az akkumulátor csatlakozók helyei

[4] Kamera konverziós ideje oszcilloszkóppal



20. (Függelék) ábra: A kamera konverziós idejének ábrája